CLIPasso: Semantically-Aware Object Sketching

Yael Vinker[2,1] Ehsan Pajouheshgar[1] Jessica Y. Bo[1] Roman Christian Bachmann[1]
Amit Haim Bermano[2] Daniel Cohen-Or[2] Amir Zamir[1] Ariel Shamir[3]

[1]Swiss Federal Institute of Technology (EPFL) [2]Tel-Aviv University [3]Reichman University

https://clipasso.github.io/clipasso/

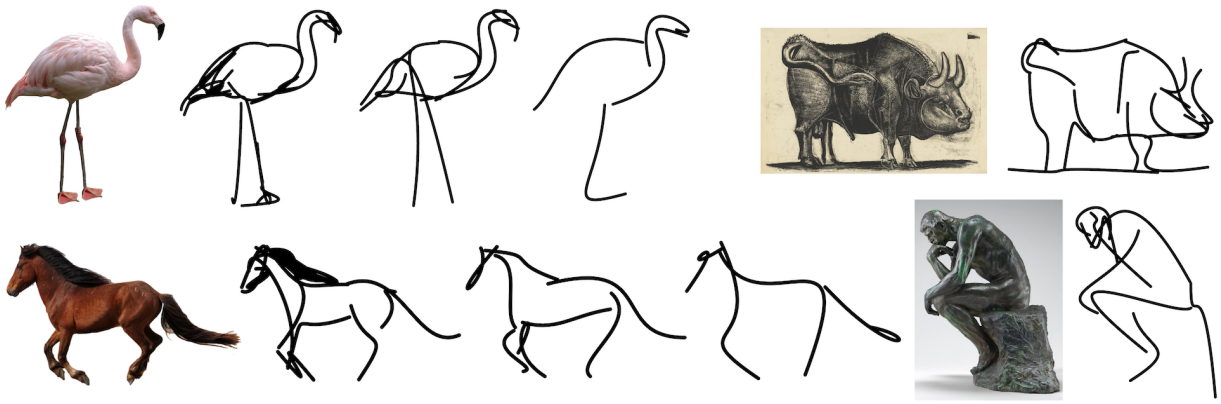


Figure 1. Our work converts an image of an object to a sketch, allowing for varying levels of abstraction, while preserving its key visual features. Even with a very minimal representation (the rightmost flamingo and horse are drawn with only a few strokes), one can recognize both the semantics and the structure of the subject depicted.

Abstract

Abstraction is at the heart of sketching due to the simple and minimal nature of line drawings. Abstraction entails identifying the essential visual properties of an object or scene, which requires semantic understanding and prior knowledge of high-level concepts. Abstract depictions are therefore challenging for artists, and even more so for machines. We present CLIPasso, an object sketching method that can achieve different levels of abstraction, guided by geometric and semantic simplifications. While sketch generation methods often rely on explicit sketch datasets for training, we utilize the remarkable ability of CLIP (Contrastive-Language-Image-Pretraining) to distill semantic concepts from sketches and images alike. We define a sketch as a set of Bézier curves and use a differentiable rasterizer to optimize the parameters of the curves directly with respect to a CLIP-based perceptual loss. The abstraction degree is controlled by varying the number of strokes. The generated sketches demonstrate multiple levels of abstraction while maintaining recognizability, underlying structure, and essential visual components of the subject drawn.

1. Introduction

Free-hand sketching is a valuable visual tool for expressing ideas, concepts, and actions [10, 15, 18, 43]. As sketches consist of only strokes, and often only a limited number of strokes, the process of *abstraction* is central to sketching. An artist must make representational decisions to choose key visual features of the subject drawn to capture the relevant information she wishes to express, while omitting (many) others [6]. For example, in the famous "Le Taureau" series (Figure 2), Picasso depicts the progressive abstraction of a bull. In this series of lithographs, the artist transforms a bull from a concrete, fully rendered, anatomical drawing, into a sketch composition of a few lines that still manages to capture the essence of the bull.

In this paper, we pose the question — can computer renderings imitate such a process of sketching abstraction, converting a photograph from a concrete depiction to an abstract one?

Today, machines can render realistic sketches simply by applying mathematical and geometric operations to an input photograph [5, 42]. However, creating abstractions is more difficult for machines to achieve. The abstraction process suggests that the artist selects visual features that capture

Figure 2. "Le Taureau" by Picasso — note how the abstraction process is achieved by gradually *removing* elements while the bull's essence is preserved.

the underlying structure and semantic meaning of the object or scene, to produce a minimal, yet descriptive rendering. This demands semantic understanding of the subject, which is more complex than applying simple geometric operations to the image. To fill this semantic gap, we use CLIP [34], a neural network trained on various styles of images paired with text. CLIP is exceptional at encoding the semantic meaning of visual depictions, regardless of their style [14].

Previous works that attempt replicating human-like sketching often use sketch datasets of the desired level of abstraction to guide the form and style of the generated sketch [3, 25, 30]. While this data-driven approach can certainly imitate the final rendering of human artwork, it requires the existence and availability of relevant datasets, and it restricts the output style to match this data. In contrast, we present an optimization-based photo-to-sketch generation technique that achieves different levels of abstraction without requiring an explicit sketch dataset.

Our method uses the CLIP image encoder to guide the process of converting a photograph to an abstract sketch. CLIP encoding provides the semantic understanding of the concept depicted, while the photograph itself provides the geometric grounding of the sketch to the concrete subject.

Our sketches are defined using a set of thin, black strokes (Bézier curves) placed on a white background, and the level of abstraction is dictated by the number of strokes used. Given the target image to be drawn, we use a differentiable rasterizer [24] to directly optimize the strokes' parameters (control points positions) with respect to a CLIP-based loss. We combine the final and intermediate activations of a pretrained CLIP model to achieve both geometric and semantic simplifications. For improved robustness, we propose a saliency-guided initialization process, based on the local attention maps of a pretrained vision transformer model.

The resulting sketches demonstrate a combination of the semantic and visual features that capture the essence of the input object, while still being minimal and providing good category and instance level object recognition clues.

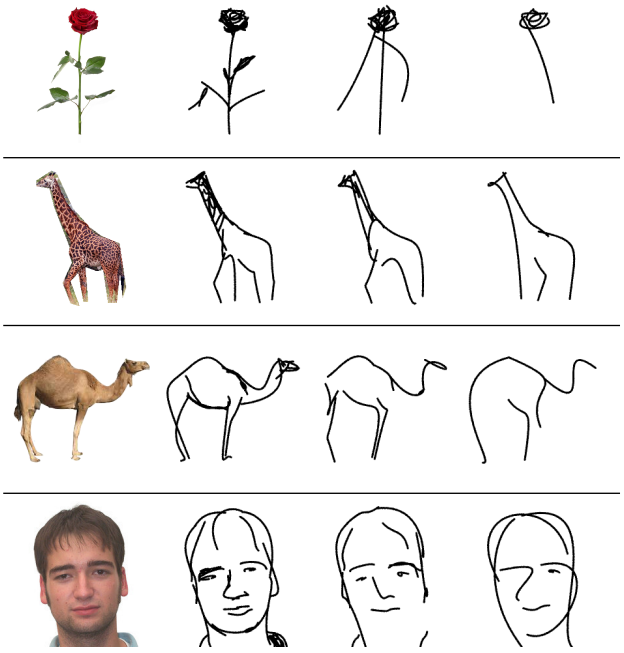


Figure 3. Different levels of abstraction produced by our method. Left to right: input images and increased level of abstraction, achieved by reducing the number of strokes used by half each column.

2. Related Work

Unlike edge-map extraction methods [5, 42] which are purely based on geometry, free-hand sketch generation aims to produce sketches that are abstract in terms of structure and semantic interpretation so as to mimic a human-like style. This high-level goal varies among different works, as there are many styles and levels of abstraction that can be produced. Consequently, existing works tend to choose the desired output style based on a given dataset: from highly abstract — guided only by a category-based text prompt [16], to more concrete [1], which is guided by contour detection. Figure 4 illustrates this spectrum. While methods that rely on sketch datasets are limited to the abstraction levels present, our method is optimization based. Hence, it is capable of producing multiple levels of abstraction without relying on the existence of suitable sketch datasets or requiring a lengthy new training phase.

We provide a brief review of existing relevant photo-sketch synthesis works, which all rely on sketch-specific datasets. Table 1 summarizes the high-level characteristics that differentiate these methods.

Photo-Sketch Synthesis Early methods learn explicit models to synthesize facial sketches [3, 8]. To generalise to categories beyond faces, Li et al. [25] learn a deformable stroke model based on perceptual grouping.

In the deep learning era, it is intuitive to think of photo-

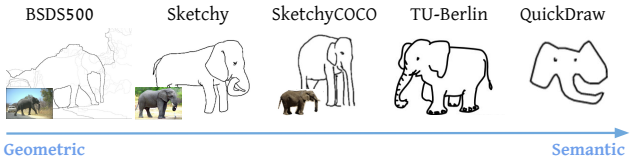


Figure 4. Variations in style and abstraction among sketch datasets — examples are arranged from left to right by the degree of abstraction: from edge-based to category-based sketches. For datasets that have image references, the input image is placed alongside the sketch; otherwise, the input is just the category label (e.g., "elephant").

Table 1. Comparison of sketch synthesis algorithms. **(A)** Is not restricted to categories from training dataset, **(B)** Can produce different levels of abstractions, **(C)** Is not limited to abstractions in the dataset, **(D)** Can produce vector sketches, **(E)** Can produce a sequential sketch **(F)** Is not directly relying on the edge map.

Method	A	B	C	D	E	F
Berger et al. [3]	✗	✓	✗	✗	✓	✗
Li et al. [25]	✗	✗	✗	✓	✗	✗
Muhammad et al. [30]	✗	✓	✓	✓	✓	✗
Song et al. [38]	✗	✗	✗	✓	✓	✓
Li et al. [23]	✓	✗	✗	✗	✗	✓
Kampelmühler and Pinz [21]	✗	✗	✗	✗	✗	✓
Qi and Su et al. [32]	✗	✓	✓	✓	✓	✗
Ours	✓	✓	✓	✓	✗	✓

sketch generation as a domain translation task. However, the highly sparse and abstract nature of sketches introduces challenges for trivial methods [20, 41] to adhere to the sketch domain, and therefore sketch-specific adjustments must be made.

Song et al. [38] propose a hybrid supervised-unsupervised multi-task learning approach with a shortcut cycle consistency constraint. Li et al. [23] present a learning-based contour generation algorithm to resolve the diversity of the human drawings in the dataset. Kampelmuhler and Pinz [21] propose an encoder-decoder architecture, where the loss is guided by a pretrained sketch classifier network. Qi and Su et al. [32] propose a lattice representation for sketches, employing LSTM and graph models to generate a vector sketch from points sampled from the edge map. The density of points determines the abstraction level of the sketch.

Vector Graphics There is a substantial literature on stroke-based rendering, contour visualization, and feature line rendering, summarized in the surveys by Hertzmann [17], and by Bénard and Hertzmann [2]. Vector representations are widely used for a variety of sketching tasks and applications, employing a number of deep learning models including RNN [16], BERT [26], Transformers [4, 35], CNNs [9] GANs [40] and reinforcement learning algorithms [12,27,46]. The recent development of differentiable rendering algorithms [24,28,45] makes it possible to manipulate or synthesize vector content by using raster-based loss functions. We use the method of Li et al. [24], as it can handle a wide range of curves and strokes, including Bézier curves.

Sketches Abstraction Only two previous works propose a unified model to produce sketches of a given image at different levels of abstraction. Berger et al. [3] collected a dataset of portraits drawn by professional artists at different levels of abstraction. For each artist, a library of strokes is created indexed by shape, curvature, and length, and these are used to replace curves extracted from the image edge map. Their method is limited to portraits and requires a new dataset for each level of abstraction.

Muhammad et al. [30] propose a stroke-level sketch abstraction model. A reinforcement learning agent is trained to select which strokes can be removed from an edge map representation of the input image without affecting its recognizability. The recognition signal is provided by a sketch classifier trained on 9 classes from the QuickDraw dataset [16], and hence to operate on new classes, a fine-tuning stage is required.

CLIP-based Image Abstraction CLIP [34] is a neural network trained on 400 million image-text pairs collected from the internet with the objective of creating a joint latent space using contrastive learning. Being trained on a wide variety of image domains along with lingual concepts, CLIP models are found to be very useful for a wide range of zero-shot tasks. The most relevant works within our context are by Frans et al. [11] (CLIPDraw), and Tian and Ha [39]. CLIPDraw optimizes a set of random Bezier curves to create a drawing that maximizes the CLIP similarity for a given text prompt. Likewise, we also use a differentiable rasterizer [24] and a CLIP-based loss. However, while CLIPDraw is purely text-driven, we allow control over the output appearance, conditioned on the input image. For this purpose, we introduce a new geometric loss term and a saliency-guided initialization procedure.

Tian and Ha [39] employ evolutionary algorithms combined with CLIP, to produce creative abstract concepts represented by colored triangles guided by text or shape. Their results are limited to either fully semantic (using CLIP's text encoder) or entirely geometric (using L2), whereas we are able to integrate both.

3. Method

We define a sketch as a set of n black strokes $\{s_1, ..s_n\}$ placed on a white background. We use a two-dimensional Bézier curve with four control points $s_i = \{p_i^j\}_{j=1}^4 = \{(x_i, y_i)^j\}_{j=1}^4$ to represent each stroke. For simplicity, we only optimize the position of control points and choose to

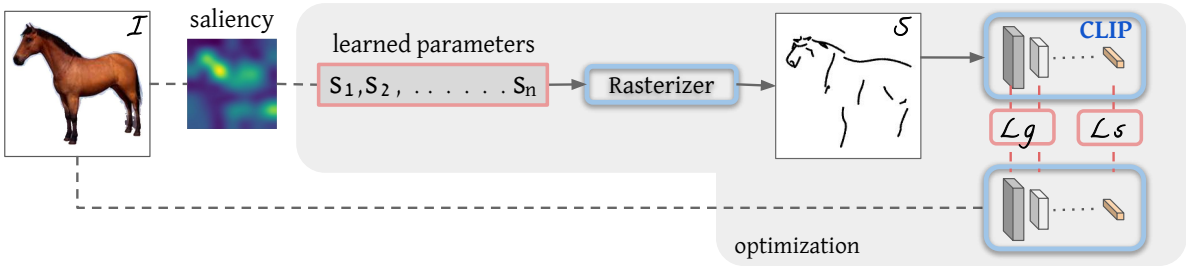


Figure 5. Method overview – Given a target image $\mathcal{I}$, and the number of strokes n, a saliency map is used as the distribution to sample the initial strokes locations $\{s_1, ..s_n\}$. A differentiable rasterizer $\mathcal{R}$ is used to create a resterized sketch $\mathcal{S}$. Both the sketch and the image are fed into a pretrained CLIP model to evaluate the geometric distance L_g and semantic distance L_s between the two. The loss is backpropagated through $\mathcal{R}$ to optimize the strokes parameters until convergence. The learned parameters and loss terms are highlighted in red, while the blue components are frozen during the entire optimization process, solid arrows are used to mark the backpropagation path.

keep the degree, width, and opacity of the strokes fixed. However, these parameters can later be used to achieve variations in style (see Figure 10a). The parameters of the strokes are fed to a differentiable rasterizer $\mathcal{R}$, which forms the rasterized sketch $\mathcal{S} = \mathcal{R}(\{p_1^j\}_{j=1}^4, ...\{p_n^j\}_{j=1}^4) = \mathcal{R}(s_1, ..s_n)$. As is often conventional [27, 30, 39], we vary the number of strokes n to create different levels of abstraction.

An overview of our method can be seen in Figure 5. Given a target image $\mathcal{I}$ of the desired subject, our goal is to synthesize the corresponding sketch $\mathcal{S}$ while maintaining both the semantic and geometric attributes of the subject. We begin by extracting the salient regions of the input image to define the initial locations of the strokes. Next, in each step of the optimization we feed the stroke parameters to a differentiable rasterizer $\mathcal{R}$ to produce the rasterized sketch. The resulting sketch, as well as the original image are then fed into CLIP to define a CLIP-based perceptual loss. We back-propagate the loss through the differentiable rasterizer and update the strokes' control points directly at each step until convergence of the loss function.

3.1. Loss Function

As sketches are highly sparse and abstract, pixel-wise metrics are not sufficient to measure the distance between a sketch and an image. Additionally, even though perceptual losses such as LPIPS [44] can encode semantic information from images, they may not be suitable to encode abstract sketches, as illustrated in Figure 6 (for further analysis, please refer to the supplementary material). One solution is to train task-specific encoders to learn a shared embedding space of images and sketches under which the distance between the two modalities can be computed [21, 38]. This approach depends on the availability of such datasets, and requires additional effort for training the models.

Instead, we utilize the pretrained image encoder model of CLIP, which was trained on various image modalities so that it can encode information from both natural images and

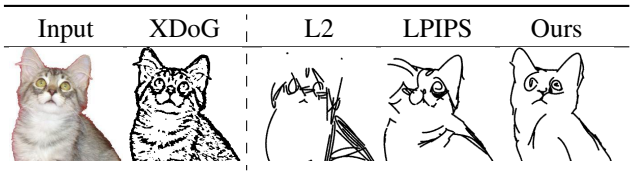


Figure 6. Loss functions comparison — we optimize the strokes by minimizing different losses: L2 loss simply encourages the filling of colored pixels, LPIPS is more semantically aware, but the resulting sketch is still close to the edge map (see the XDog edges for comparison). In contrast, our CLIP-based loss allows better semantic depiction while preserving the morphology of the subject.

sketches without the need for further training. CLIP encodes high-level semantic attributes in the last layer since it was trained on both images and text. We therefore define the distance between the embeddings of the sketch $CLIP(\mathcal{R}(\{s_i\}_{i=1}^n)$ and image $CLIP(\mathcal{I})$ as:

$$L_{semantic} = dist\big(CLIP(\mathcal{I}), CLIP(\mathcal{R}(\{s_i\}_{i=1}^n)\big), \quad (1)$$

where $dist(x, y) = 1 - \frac{x \cdot y}{||x|| \cdot ||y||}$ is the cosine distance. However, the final encoding of the network is agnostic to low-level spatial features such as pose and structure. To measure the geometric similarity between the image and the sketch, and consequently, allow some control over the appearance of the output, we compute the $L2$ distance between intermediate level activations of CLIP:

$$L_{geometric} = \sum_l \| \, CLIP_l(\mathcal{I}) - \, CLIP_l(\mathcal{R}(\{s_i\}_{i=1}^n)) \, \|_2^2, \quad (2)$$

where $CLIP_l$ is the $CLIP$ encoder activation at layer l. Specifically, we use layers 3 and 4 of the ResNet101 CLIP model. The final objective of the optimization is then defined as:

$$\min_{\{s_i\}_{i=1}^n} L_{geometry} + w_s \cdot L_{semantic}, \quad (3)$$

with $w_s = 0.1$. We analyze the contribution of different layers and weights, as well as the results of using different CLIP models in the supplementary material.

3.2. Optimization

Our goal is to optimize the set of parameters $\{s_i\}_{i=1}^n = \{\{p_i^j\}_{j=1}^4\}_{i=1}^n$ to define a sketch that closely resembles the target image $\mathcal{I}$ in terms of both geometry and semantics. At each step of the optimization, we use off-the-shelf gradient-based solving technique to compute the gradients of the loss with respect to the strokes' parameters $\{s_i\}_{i=1}^n$. We follow the same data augmentation scheme suggested in CLIPDraw [11] and augment both the sketch and the target image before feed-forwarding into CLIP. These augmentations prevent the generation of adversarial sketches, which minimize the objective but are not meaningful to humans. We repeat this process until convergence, when the optimization error does not change significantly (taking typically ~ 2000 iterations). Figure 7 illustrates the progression of the generated sketch as the optimization evolves.

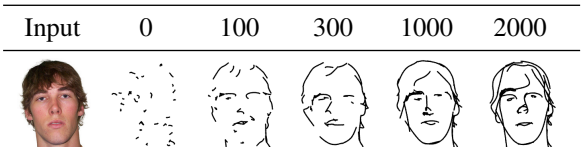


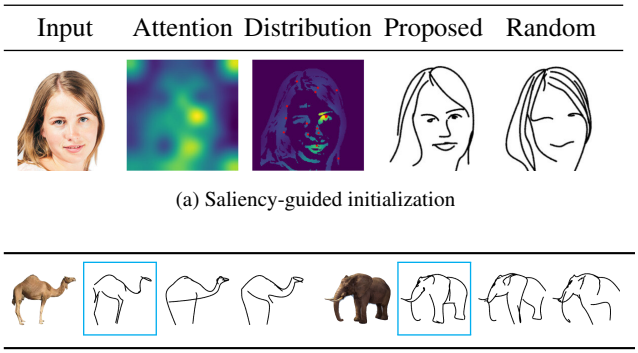
Figure 7. The sketch appearance throughout the optimization iterations.

3.3. Strokes Initialization

Our objective function is highly non-convex. Therefore, the optimization process is susceptible to the initialization (i.e., the initial location of the strokes). This is especially significant at higher levels of abstraction – where very few strokes must be wisely placed to emphasize semantic components. For example, in Figure 8a, the sketches in the last two columns were produced using the same number of strokes, however, in the "Random" initialization case, more strokes were devoted to the hair while the eyes, nose, and mouth are more salient and critical features of the face.

To improve convergence towards semantic depictions, we place the initial strokes based on the salient regions of the target image. We use the pretrained vision transformer [22] ViT-B/32 model of CLIP, that performs global context modeling using self-attention between patches of a given image to capture meaningful features. We use the recent transformer interpretability method by Chefer et al. [7] to extract a relevancy map from the self-attention heads, without any text supervision.

Our final distribution map is produced by multiplying the relevancy map with the edge map of the image extracted using XDoG [42], followed by a softmax normalization.



(a) Saliency-guided initialization

(b) Automatic selection procedure of the final sketch

Figure 8. Strokes Initialization. (a) Left to right: input, the saliency map produced from CLIP ViT activations, final distribution map (adjusted to adhere to image edges) with sampled initial stroke locations (in red), the sketch produced using the proposed initialization procedure, and the sketch produced when using random initialization. (b) Results of three different initializations with the same number of strokes. The sketches marked in blue produced the lowest loss value, and would thus be used as the final output.

XDoG is used to strengthen the morphological positioning of the strokes, motivated by the hypothesis that edges are effective in predicting where people draw lines [19].

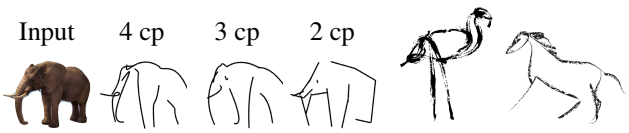
Figure 8a illustrates this procedure. It can be seen that our saliency-based initialization contributes significantly to the quality of the final sketch compared to random initialization.

This sampling-based approach also lends itself to providing variability in the results. In all our examples we typically use 3 initializations, and automatically choose the one that yields the lowest loss (see Figure 8b). We further analyze the initialization procedure and variability in the supplementary material.

4. Results

Section 4.1 provide qualitative evaluations. In section 4.2 we compare our method with existing image-to-sketch methods, which were all trained on sketch-specific datasets. In Section 4.3, we supply a quantitative evaluation of our method's ability to produce recognizable sketches testing both category and instance recognition. For images with

Figure 9. Sketches produced by our method for infrequent categories.



(a) Changing the degree of the curves

(b) Editing the brush style on SVGs

Figure 10. Changing sketch style. (a) From left to right are the results produced by our method when using Bézier curves with 4, 3, and 2 control points (cp), respectively. We can see how this affects the style of the output sketch. (b) Using Adobe Illustrator, horse — pencil feather, flamingo — dry brush.

background, we use an automatic method (U2-Net [33]) to mask out their background. We provide further analysis of our method, extra results, and extended comparison with other methods in the supplemental file.

4.1. Qualitative Evaluation

Our approach is different from conventional sketching methods in that it does not utilize a sketch dataset for training, rather it is optimized under the guidance of CLIP. Thus, our method is not limited to specific categories observed during training, as no category definition was introduced at any stage. This makes our method robust to various inputs, as shown in Figures 1 and 9.

In Figures 1 and 3 we demonstrate the ability of our method to produce sketches at different levels of abstraction. As the number of strokes decreases, the task of minimizing the loss becomes more challenging, forcing the strokes to capture the essence of the object. For example, in the abstraction process of the flamingo in Figure 1, the transition from 16 to 4 strokes led to the removal of details such as the eyes, feathers, and feet, while maintaining the important visual features such as the general pose, the neck and legs which are iconic characteristics of a flamingo.

Besides changing the number of strokes, different sketch styles can be achieved by varying the degree of the strokes (Figure 10a) or using a brush style on top of the vector strokes (Figure 10b).

4.2. Comparison with Existing Methods

Sketches with different levels of abstraction. Only a few works have attempted to sketch objects at different levels of abstraction. In Figure 11 we compare with Muhammad et al. [30] and Berger et al. [3]. The results by Muhammad et al. demonstrate four levels of abstraction on two simple inputs — a shoe and a chair (in the absence of their code, the results were taken directly from the paper). We produce sketches at four levels of abstraction using 32, 16, 8, and 4 strokes. The sketches by Muhammad et al. are coherent with the geometry of the image; but to achieve higher levels of abstraction, they only remove strokes from

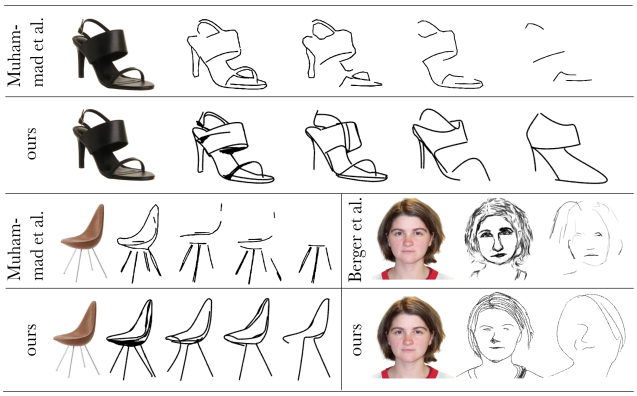


Figure 11. Levels of Abstraction Comparison — in the top and left part are comparisons to Muhammad et al. [30]. The leftmost column shows the input image, and the next four columns show different levels of abstraction. For the shoe and chair our results were produced using 32, 16, 8, and 4 curves (from left to right). In the right bottom part is a comparison to Berger et al. [3], we use 64 and 8 strokes to generate our sketches.

the generated sketch without changing the remaining ones. This can result in losing class-level recognizability at higher levels of abstraction (rightmost sketches). Such an approach is sub-optimal, since a better arrangement may be possible for fewer strokes. Our method successfully produces a recognizable rendition of the subject while preserving its geometry, even in the challenging 4-stroke case (rightmost sketch).

In the right bottom part of Figure 11 we compare with the method of Berger et al. [3]. Their results were provided by the authors and demonstrate two levels of abstraction generated based on the style of a particular artist. We use 64 and 8 strokes, respectively, to achieve two comparable levels of abstraction and place a pencil style on top of the generated sketch to better fit the artist's style. As can be seen, our approach is more geometrically coherent while still allowing abstraction. Their results fit better to a specific style, but can only work with faces and are limited to the dataset gathered.

Photo-Sketch Synthesis. In Figure 12 we present a comparison with the five works outlined in Table 1. The results by Kampelmühler and Pinz [21] (A), Li et al. [25] (B) and Li et al. [23] (C) were generated based on the authors' implementation and best practice. Due to the lack of a publicly available implementation of SketchLattice [32] (E), their results are taken directly from the paper. We present the sketches of Song et al. [38] (D) on shoe images, since their method only works with shoes and chairs.

Each of these methods define a specific objective which influences their dataset selection and final output style. Li et al. [23] (C) aim for boundary-like drawings, and indeed, geometric coherence is achieved with the input image, cap-

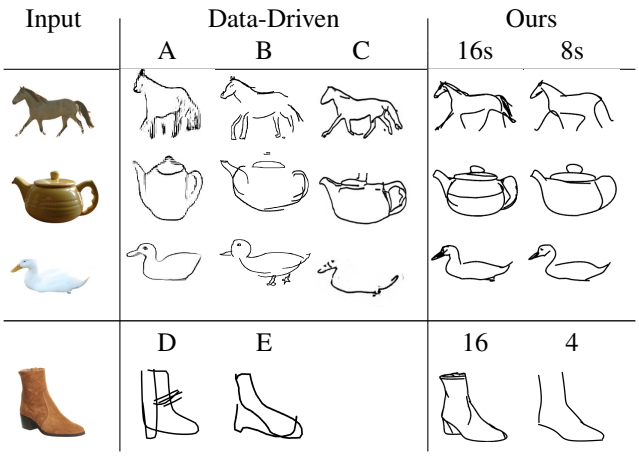


Figure 12. Comparison to Existing Image-to-Sketch Works — the leftmost column shows the input images. The methods presented are (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23] (D) Song et al. [38], (E) SketchLattice [32].

turing salient outlines. The other methods are designed to produce human-like sketches of non-experts, and indeed, the synthesized sketches exhibit a "doodle-like" style. Furthermore, the methods learn highly abstract concepts (such as highlighting the eyes) while maintaining some relation to the geometry of the input object.

Each method accomplishes their respective objective, but we wish to emphasize the particular benefits of our method. First, all of the above methods are sketch-data dependent, meaning they can only be used with the style and level of abstraction observed during training. With our framework, we can handle images of all categories and produce sketches of various levels of abstraction simply by changing the number of strokes. Although every method can be retrained with new datasets, this is neither convenient nor practical, and depends on the availability of such datasets. Second, while each method leans towards a more semantic or a more geometric sketching style, our method can provide both. For example, our method did not produce a perfect alignment of the legs of the horse, as in (C), but it captured the movement of the horse in a minimal way.

In Figure 13 we provide a comparison with CLIPDraw [11]. The text input for CLIPDraw is replaced with the target image. This was made possible since CLIP encodes both text and images to the same latent space. To provide a comparable visualization, we constrain the output primitives of CLIPDraw in the same manner as we defined our strokes. As can be seen, although the parts of the subject can be recognizable using CLIPDraw, since there is no geometric grounding to the image, the overall structure is destroyed. Further comparisons of CLIPDraw incorporating text and color can be found in the supplemental file.

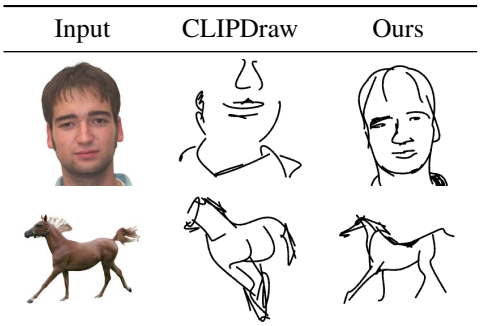


Figure 13. Comparison to CLIPDraw [11]. All sketches were produced using 16 strokes.

4.3. Quantitative Evaluation

We conduct a user study with 121 participants to assess both the category-level and instance-level recognizability of the sketches generated by our method at different levels of abstraction. Additionally, similar to previous image-to-sketch works [21, 30, 38], we also use pretrained classifier networks to evaluate the category-level recognizability of the sketches generated by our method.

User-Study We choose five popular animal classes from the SketchyCOCO dataset [13] and randomly sample five images per class. We synthesize sketches at four levels of abstraction for each image, with 4, 8, 16, and 32 strokes. We compare the recognition rates with the two recent photo-sketch synthesis methods by Kampelmühler and Pinz [21] and Li et al. [23]. The sketches were produced with one level of abstraction, suited to the capabilities of the methods.

Each participant was presented with randomly selected sketches of a single method. To examine category-level recognition, participants were asked to choose the correct category text description alongside four confound categories and the option 'None'. For the instance-level recognition experiment, the distractors are images from the same object category. Table 2 shows the average recognition rates attained from the user-study. Both the category-level and instance-level recognizability are inversely correlated with the level of abstraction. At four strokes, we can see that our sketches are hardly recognizable at the category level (36%), which illustrates a "breaking point" of our method. At eight strokes and above, both instance and class level rates are high. Li et al. [23] demonstrate full recognition at the instance level; this is understandable given that the sketches are contour-based and not abstract. At 16 and 32 strokes, we achieve comparable rates, and even with the high level of abstraction when using only eight strokes, we achieve 95% instance level recognizability. The sketches by Kampelmühler and Pinz are more abstract, which explains

Table 2. User study results – average recognition rates. (A) Kampelmühler and Pinz [21], (B) Li et al. [23].

	A	B	Ours 4s	Ours 8s	Ours 16s	Ours 32s
Category-Level	65% ±2%	96.9% ±0.7%	36% ±3%	87% ±2%	97.9% ±0.8%	99.3% ±0.5%
Instance-Level	65% ±2%	99.1% ±0.4%	72% ±3%	95% ±1%	96% ±1%	97% ±1%

Table 3. Top-1 and Top-3 sketch recognition accuracy computed with ResNet34 and CLIP ViT-B/32 on 200 sketches from 10 categories. (A) Kampelmühler and Pinz [21], (B) Li et al. [23]

Classifier		Human Sketches	A	B	Ours 16s	Ours 32s
ResNet34	Top1	98%	**67%**	61%	54%	63%
	Top3	99%	**82%**	78%	75%	77%
CLIP ViT-B/32	Top1	75%	49%	60%	**78%**	91%
	Top3	93%	65%	77%	**93%**	97%

their low accuracy at the instance and class level.

Sketch classifier Two different classifiers are used for evaluating the synthesized sketches; a ResNet34 classifier from Kampelmühler and Pinz [21] trained on the Sketchy-Database [37] with 125 categories, and a CLIP ViT-B/32 zero-shot classifier using text prompts defined as "A sketch of a(n) *class-name*". Note that this is not the CLIP model we use for training. Table 3 compares the sketch-classifier recognition accuracy of our sketches to that of Kampelmühler and Pinz [21] and Li et al. [23] based upon 200 randomly selected images of 10 categories from the SketchyCOCO dataset [13]. The recognition accuracy on human sketches from the SketchyCOCO dataset is also calculated as a baseline. The method by Kampelmühler and Pinz achieves the highest scores for ResNet34 classifier, possibly since they use the same model and dataset during training. Despite the distribution differences, our method still achieves good recognition rates under this classifier. With CLIP classifier, our method achieves very high accuracy levels of 78% with 16 strokes and 91% with 32 strokes. For more details and analysis please refer to the supplementary material.

5. Limitations

For images with background, our method's performance is reduced at higher abstraction levels. This limitation can be addressed by using an automatic mask. However, a potential development would be to include such a remedial term within the loss function. In addition, our sketches are not created sequentially and all strokes are optimized simultaneously, which differs from the conventional way of sketching. Furthermore, the number of strokes must be determined in advance to achieve the desired level of abstraction. Another possible extension could be to make this a learned parameter, as different images might require different numbers of strokes to reach similar levels of abstraction.

6. Conclusions

We presented a method for photo-sketch synthesis, producing sketches with different levels of abstraction, without the need to train on specific sketch datasets. Our method can generalize to various categories and cope with challenging levels of abstraction, while maintaining the semantic visual clues that allow for instance-level and class-level recognition.

References

[1] Pablo Arbeláez, Michael Maire, Charless Fowlkes, and Jitendra Malik. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(5):898–916, 2011. 2

[2] Pierre Bénard and Aaron Hertzmann. Line drawings from 3d models. *CoRR*, abs/1810.01175, 2018. 3

[3] Itamar Berger, Ariel Shamir, Moshe Mahler, Elizabeth Carter, and Jessica Hodgins. Style and abstraction in portrait sketching. *ACM Trans. Graph.*, 32(4), jul 2013. 2, 3, 6

[4] Ayan Kumar Bhunia, Ayan Das, Umar Riaz Muhammad, Yongxin Yang, Timothy M. Hospedales, Tao Xiang, Yulia Gryaditskaya, and Yi-Zhe Song. Pixelor: a competitive sketching ai agent. so you think you can sketch? *ACM Trans. Graph.*, 39:166:1–166:15, 2020. 3

[5] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, pages 679–698, 1986. 1, 2

[6] Rebecca Chamberlain and Johan Wagemans. The genesis of errors in drawing. *Neuroscience & Biobehavioral Reviews*, 65:195–207, 2016. 1

[7] Hila Chefer, Shir Gur, and Lior Wolf. Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 397–406, October 2021. 5

[8] Hong Chen, Ying-Qing Xu, Heung-Yeung Shum, Song-Chun Zhu, and Nan-Ning Zheng. Example-based facial sketch generation with non-parametric sampling. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pages 433–438 vol.2, 2001. 2

[9] Yajing Chen, Shikui Tu, Yuqi Yi, and Lei Xu. Sketch-pix2seq: a model to generate sketches of multiple categories. *ArXiv*, abs/1709.04121, 2017. 3

[10] Judith E. Fan, Daniel L. K. Yamins, and Nicholas B. Turk-Browne. Common object representations for visual produc-

tion and recognition. *Cognitive science*, 42 8:2670–2698, 2018. 1

[11] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. Clipdraw: Exploring text-to-drawing synthesis through language-image encoders. *CoRR*, abs/2106.14843, 2021. 3, 5, 7, 11, 23, 24, 25, 26, 27, 28

[12] Yaroslav Ganin, Tejas D. Kulkarni, Igor Babuschkin, S. M. Ali Eslami, and Oriol Vinyals. Synthesizing programs for images using reinforced adversarial learning. *ArXiv*, abs/1804.01118, 2018. 3

[13] Chengying Gao, Qi Liu, Qi Xu, Limin Wang, Jianzhuang Liu, and Changqing Zou. Sketchycoco: Image generation from freehand scene sketches. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5174–5183, 2020. 7, 8, 12, 33, 34

[14] Gabriel Goh, Nick Cammarata †, Chelsea Voss †, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. https://distill.pub/2021/multimodal-neurons. 2

[15] Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia C. Pont, Frédo Durand, and Adrien Bousseau. Opensketch: a richly-annotated dataset of product design sketches. *ACM Trans. Graph.*, 38:232:1–232:16, 2019. 1

[16] David Ha and Douglas Eck. A neural representation of sketch drawings. *CoRR*, abs/1704.03477, 2017. 2, 3

[17] A. Hertzmann. A survey of stroke-based rendering. *IEEE Computer Graphics and Applications*, 23(4):70–81, 2003. 3

[18] Aaron Hertzmann. Why do line drawings work? a realism hypothesis. *Perception*, 49:439 – 451, 2020. 1

[19] Aaron Hertzmann. The role of edges in line drawing perception. *Perception*, 50:266 – 275, 2021. 5

[20] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, 2017. 3

[21] Moritz Kampelmühler and Axel Pinz. Synthesizing human-like sketches from natural images using a conditional convolutional decoder. *CoRR*, abs/2003.07101, 2020. 3, 4, 6, 7, 8, 12, 22, 23, 24, 25, 26, 27

[22] Alexander Kolesnikov, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR '21*, 2021. 5

[23] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. Photo-sketching: Inferring contour drawings from images, 2019. 3, 6, 7, 8, 12, 22, 23, 24, 25, 26, 27, 28

[24] Tzu-Mao Li, Michal Lukáč, Gharbi Michaël, and Jonathan Ragan-Kelley. Differentiable vector graphics rasterization for editing and learning. *ACM Trans. Graph. (Proc. SIGGRAPH Asia)*, 39(6):193:1–193:15, 2020. 2, 3, 11

[25] Yi Li, Yi-Zhe Song, Timothy M. Hospedales, and Shaogang Gong. Free-hand sketch synthesis with deformable stroke models. *CoRR*, abs/1510.02644, 2015. 2, 3, 6, 7, 23, 24, 25, 26, 27, 28

[26] Hangyu Lin, Yanwei Fu, Yu-Gang Jiang, and X. Xue. Sketch-bert: Learning sketch bidirectional encoder representation from transformers by self-supervised learning of sketch gestalt. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6757–6766, 2020. 3

[27] John F. J. Mellor, Eunbyung Park, Yaroslav Ganin, Igor Babuschkin, Tejas Kulkarni, Dan Rosenbaum, Andy Ballard, Theophane Weber, Oriol Vinyals, and S. M. Ali Eslami. Unsupervised doodling and painting with improved SPIRAL. *CoRR*, abs/1910.01007, 2019. 3, 4

[28] Daniela Mihai and Jonathon S. Hare. Differentiable drawing and sketching. *ArXiv*, abs/2103.16194, 2021. 3

[29] Haoran Mo, Edgar Simo-Serra, Chengying Gao, Changqing Zou, and Ruomei Wang. General virtual sketching framework for vector line art. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2021)*, 40(4):51:1–51:14, 2021. 23, 28

[30] Umar Riaz Muhammad, Yongxin Yang, Yi-Zhe Song, Tao Xiang, and Timothy M. Hospedales. Learning deep sketch abstraction. *CoRR*, abs/1804.04804, 2018. 2, 3, 4, 6, 7

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 11

[32] Yonggang Qi, Guoyao Su, Pinaki Nath Chowdhury, Mingkang Li, and Yi-Zhe Song. Sketchlattice: Latticed representation for sketch manipulation. *ArXiv*, abs/2108.11636, 2021. 3, 6, 7, 23

[33] Xuebin Qin, Zichen Vincent Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaïane, and Martin Jägersand. U^2-net: Going deeper with nested u-structure for salient object detection. *Pattern Recognit.*, 106:107404, 2020. 6, 12

[34] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 2, 3, 11

[35] Leo Sampaio Ferraz Ribeiro, Tu Bui, John P. Collomosse, and Moacir Antonelli Ponti. Sketchformer: Transformer-based representation for sketched structure. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14141–14150, 2020. 3

[36] Paul L. Rosin, David Mould, Itamar Berger, John P. Collomosse, Yu-Kun Lai, Chuan Li, Hua Li, Ariel Shamir, Michael Wand, T. Wang, and Holger Winnemöller. Benchmarking non-photorealistic rendering of portraits. In *NPAR '17*, 2017. 12, 23, 28

[37] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. The sketchy database: Learning to retrieve badly

drawn bunnies. *ACM Trans. Graph.*, 35(4), jul 2016. 8, 11, 23

[38] Jifei Song, Kaiyue Pang, Yi-Zhe Song, Tao Xiang, and Timothy Hospedales. Learning to sketch with shortcut cycle consistency, 2018. 3, 4, 6, 7, 23

[39] Yingtao Tian and David Ha. Modern evolution strategies for creativity: Fitting concrete images and abstract concepts. *CoRR*, abs/2109.08857, 2021. 3, 4

[40] V Varshaneya, Sangeetha Balasubramanian, and Vineeth N. Balasubramanian. Teaching gans to sketch in vector format. *Proceedings of the Twelfth Indian Conference on Computer Vision, Graphics and Image Processing*, 2021. 3

[41] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CoRR*, abs/1711.11585, 2017. 3

[42] Holger Winnemöller, Jan Eric Kyprianidis, and Sven C. Olsen. Xdog: An extended difference-of-gaussians compendium including advanced image stylization. *Comput. Graph.*, 36:740–753, 2012. 1, 2, 5, 11

[43] Peng Xu, Timothy M. Hospedales, Qiyue Yin, Yi-Zhe Song, Tao Xiang, and Liang Wang. Deep learning for free-hand sketch: A survey and a toolbox, 2020. 1

[44] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 4, 21

[45] N. Zheng, Yf Jiang, and Ding jiang Huang. Strokenet: A neural painting environment. In *ICLR*, 2019. 3

[46] Tao Zhou, Chen Fang, Zhaowen Wang, Jimei Yang, Byungmoon Kim, Zhili Chen, Jonathan Brandt, and Demetri Terzopoulos. Learning to sketch with deep q networks and demonstrated strokes. *ArXiv*, abs/1810.05977, 2018. 3

CLIPasso: Semantically-Aware Object Sketching
Supplementary Material

Yael Vinker[2,1] Ehsan Pajouheshgar[1] Jessica Y. Bo[1] Roman Christian Bachmann[1]
Amit Haim Bermano[2] Daniel Cohen-Or[2] Amir Zamir[1] Ariel Shamir[3]

[1]Swiss Federal Institute of Technology (EPFL) [2]Tel-Aviv University [3]Reichman University

Table of Contents

A. Implementation Details

We implement our method in PyTorch [31] utilizing the "diffvg" differentiable rasterizer implementation by [24].

Initialization details: Using the ViT-32/B CLIP [34] model, we extract the salient regions of an image by averaging all of the attention heads of each self-attention layer, resulting in 12 attention maps. Then, we average these attention maps and take the attention between the final class embedding and all 49 patches to get our relevancy map. We multiply the relevancy map with the edge map extracted using XDoG [42], and then use this final attention map to sample the locations of the initial strokes. We first sample n positions for the first control point of each curve using the map and then randomly sample the next three control points of each Bezier curve within a small radius (0.05) of the first point.

Augmentation details: Following the implementation of CLIPDraw [11], we apply random affine augmentations to both the sketch and target images before passing them as inputs to CLIP. The transformations we use are RandomPerspective and RandomResizedCrop. These augmentations make the optimization less prone to adversarial samples and improve the quality of the generated sketch.

Optimization details: We use Adam optimizer with a learning rate set to 1. We evaluate the output sketch every 10 iterations. Evaluation is done by computing the loss without random augmentations. We repeat the optimization process until convergence (when the difference in loss between two successive evaluations is less than 0.00001), this typically takes around 2000 iterations. As noted in the paper, the final result is highly sensitive to the initialization, therefore, we run the process in parallel with three different seeds and select the final sketch which has the lowest evaluation loss. It takes 6 minutes to run 2000 iterations on a single Tesla V100 GPU, however, after 100 iterations it is already possible to get a recognizable sketch for most images. In the case of images with backgrounds, it is possible to use an automatic tool such as U2-Net for background removal, and then apply the proposed method.

B. Dealing With Background

As stated in the paper, our method is most suitable for input images that contain only the object to be drawn without background. Generally, methods for object sketching that can cope with background are trained in a supervised manner to overcome this challenge, i.e. the training dataset included pairs of image with background and the corresponding sketch without background [37]. As a result, these

models are encouraged to omit the background. The fact that we perform zero-shot generation implies that we do not rely on the characteristics of a particular sketches dataset to constrain the appearance of the output sketch. Consequently, our method includes consideration of the background. Moreover, the geometric loss term encodes relatively shallow features, making it less semantic than the final fully connected layer. This also contributes to the inclusion of background details in the final sketch results.

Even though the task is challenging, our model can still handle images with background, but with reduced performance.

In Figure 14 we demonstrate two use cases in which a background is present – (1) images that contain a salient object to be drawn (for example, the tree and horse in rows 1 and 2), and (2) images of natural landscapes in which the background plays an important role and cannot be ignored, such as the mountains and beach in columns 3 and 4. As our method is designed for object sketching, the solution for case number (1) is straightforward - we simply use an automatic tool (such as U2-Net [33]) for masking the salient object in the scene. On the rightmost column are the sketches produced after applying the mask to the input image. It can be seen that the generated sketches for images with a salient object are satisfactory. Regarding use case number (2), although our method is not designed to handle such input sources, we can see that it nonetheless produces pleasing results when applied with 32 and 16 strokes (columns 3, 4). The method, however, failed to produce reasonable drawings for higher levels of abstraction (8 strokes) as seen in column 5.

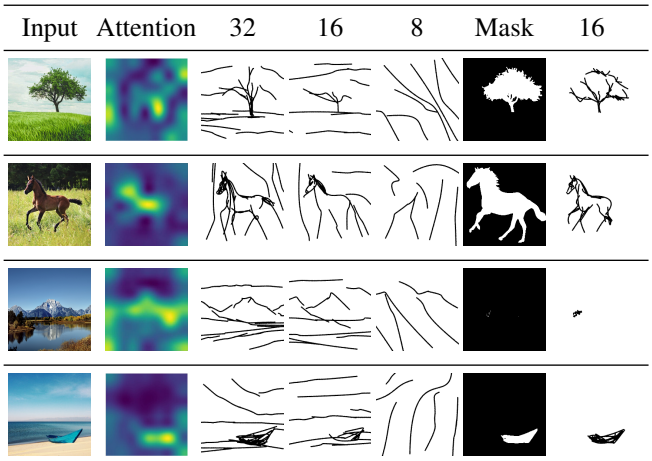


Figure 14. Background Analysis - the first column shows the input images with background that were used for our optimization, the second column shows the extracted attention map. Columns 3, 4, and 5 show the output sketches produced by our method, using 32, 16, and 8 strokes respectively. Column 6 shows the mask produced by pretrained U2-Net, and column 7 shows the results of our method when applied on the masked images.

C. User Study

This section provides additional details and analysis on the user study conducted on the sketches produced by our method.

We select five animal classes for the user study (cat, dog, elephant, giraffe, and horse), as animals typically have distinctive pictorial representations but also share visual features that would make class-level differentiation a non-trivial task. As was stated in the main paper, we use the SketchyCOCO dataset [13], and randomly sample 5 images for each class, see Figure 17 for reference. For the instance-level recognition test, we also include an additional class of human faces from the [36] dataset, and sample five portraits of men and five of women (see Figure 18). The 'face' samples were not included in the main paper's analyses. In total, we generate 100 sketches of animals and 40 sketches of human faces. We randomly split the sketches to 5 questionnaires, each questionnaire with 20 questions for the category-level recognition experiment (five classes at four levels of abstraction) and 28 questions for the instance-level recognition experiment (five animal classes, male faces, and female faces at four levels of abstraction). Figures 19 and 20 demonstrate the output sketches used in one of the questionnaires.

We surveyed 121 people in total. Out of 121, 60 people were assigned to evaluate our sketches, 38 to the sketches by Kampelmühler and Pinz [21], and 24 to the sketches by Li et al. [23]. Participants evaluating our sketches were allowed to randomly select the set of questions they received, with the ultimate split being 13 people receiving version one, 15 for version two, 8 for version three, 15 for version four, and 9 for version five. The weight of each response contributed equally to the overall accuracy computation, regardless of the version. It is possible that certain versions of the survey had higher difficulty than others, but the variations are not significant. In the following section we provide the detailed results of the user-study on our sketches.

C.1. Category-Level Recognition

Figure 15 shows the per-class recognition accuracy at each abstraction level. The color of the bar graph represents the number of strokes, for example blue represents 4-stroke sketches. In the main paper, we reported the accuracy averaged across all classes to be 36%. In the class-by-class breakdown, we observe that 'dog' and 'horse' classes received below average accuracies, while 'cat' and 'giraffe' had much better performance. One theory to support this is that cats and giraffes contain more unambiguous discriminative features in their pictorial representations — namely, triangular ears for the former, and a long neck for the latter. These are features that even amateur artists would capture

while sketching lightweight representations, and evidently they are well captured by our method through the saliency-based initialization and the CLIP-based loss.

Another interesting observation we made is in the confidence of the answers. The baseline performance of randomly guessing between the five animal classes would result in an accuracy of 20%. However, since the sixth choice is 'None', participants are able to defer their guess if they are uncertain of the answer. In highly abstract sketches with 4 strokes, 53% of the answers are 'None', but this fell to 7% in 8-stroke sketches, and virtually none in 16- and 32-stroke sketches, indicating that added stroke details reduced uncertainty significantly. If we exclude 'None' answers (as in, leaving only the confident guesses), then the recognition accuracy of 4-stroke sketches is improved from 36% to 76%. This shows that while high abstraction may introduce uncertainty, there is still enough semantic and/or geometric information to represent the object class for the study participants who were relatively confident in answering.

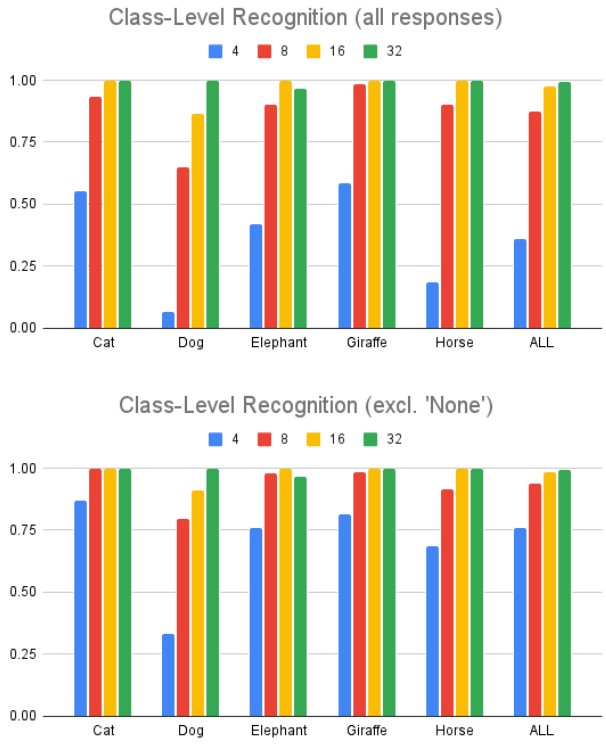


Figure 15. Class level recognizability user study results.

C.2. Instance-Level Recognition

Figure 16 shows the instance-level recognition for each class and abstraction level. Included in this experiment is the 'face' class, which depicts portrait-style human faces of various demographics[1]. As the goal is to identify the input image of a sketch given four additional confounder images of the same class, instance-level recognition relies more on the geometric feature matching between the sketch and the image. Semantic and class-level discriminative features are less important since these features would be shared across all options. Thus the instance-level recognizability is more determined by the geometric fidelity of the sketch to the original image.

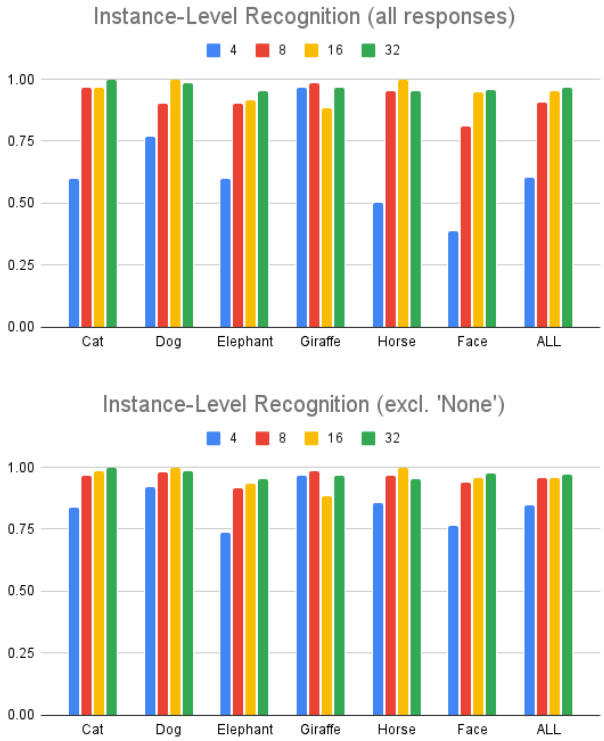


Figure 16. Instance level recognizability user study results.

The results show that in comparison to the class recognition test, the 4-stroke sketches yielded much better recognizability and less uncertainty. The reason is likely because geometric features are easy to match between the input image and the sketch. It should be noted that the variance across classes for 4-stroke sketches is quite high, from 39% on 'face' to 97% on 'giraffe', and the average accuracy across classes being 60%. The reason for the lower accuracy of the 'face' class is likely due to the less-distinctive poses of the subjects, as all faces had the similar orientations and sizes, so the participants had to rely more on the detailed features of the face for instance recognition. The proportion of 'None' answers is 40% for 4 strokes, and

[1]Notice that the face class was not included in the results reported in the main paper because one of the methods was only limited to the categories in the Sketchy Database

the accuracy discounting 'None' answers is 85%. The increase in overall recognizability from 4 strokes (60%) to 32 strokes (97%) is evident but less substantial than improvements seen in the class recognition task, likely due to visually similar poses of the subjects of the confounder images.

Figure 17. The randomly sampled animals images used for the user study.

Figure 18. The randomly sampled face images used for the user study.

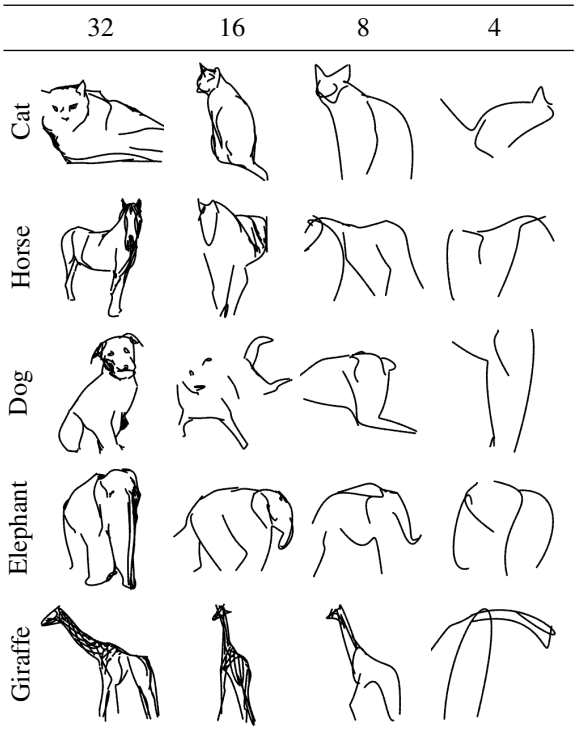


Figure 19. Sample animal sketches produced by our method, that were used in the user study.

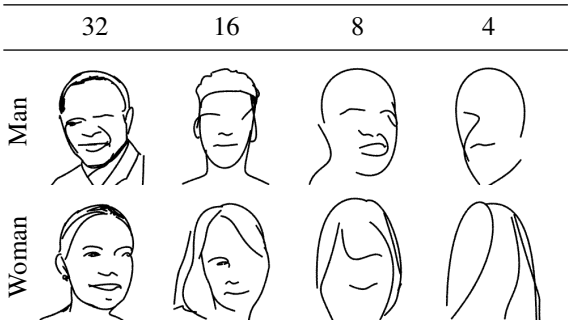


Figure 20. Sample face sketches produced by our method that were used in the user study.

D. Initialization Analysis

D.1. Initialization Ablation

This section provides a visual analysis demonstrating the sensitivity to initialization.

In Figure 21 we provide additional examples showing how multiple sketches of the same level of abstraction can be produced from a single input object by changing the random seed used. The red dots on each saliency map indicate the initial strokes' location sampled for three different random seeds. As was stated in the paper, since our method is optimization-based, and the objective function is highly non-convex, the strokes converge to different local minimas. Therefore, the initialization of the strokes has a significant impact on the final result of the optimization. In Figures 22, 24, and 26, we compare different approaches to initialization and demonstrate the synthesized sketches on images of cat, dog, and human face. In particular, we investigate the performance of the following stroke initialization techniques: (1) ViT[2] and XDoG, (2) only ViT, and (3) random initialization. All experiments were conducted using

[2]Here ViT refers to CLIP ViT-B/32 model.

16 strokes. The figures show that when using random initialization, strokes converge to a sub-optimal solution, often missing the semantic components of the object such as the eyes. We also provide the loss graphs for each image and initialization technique, in Figures 23, 25, and 27.

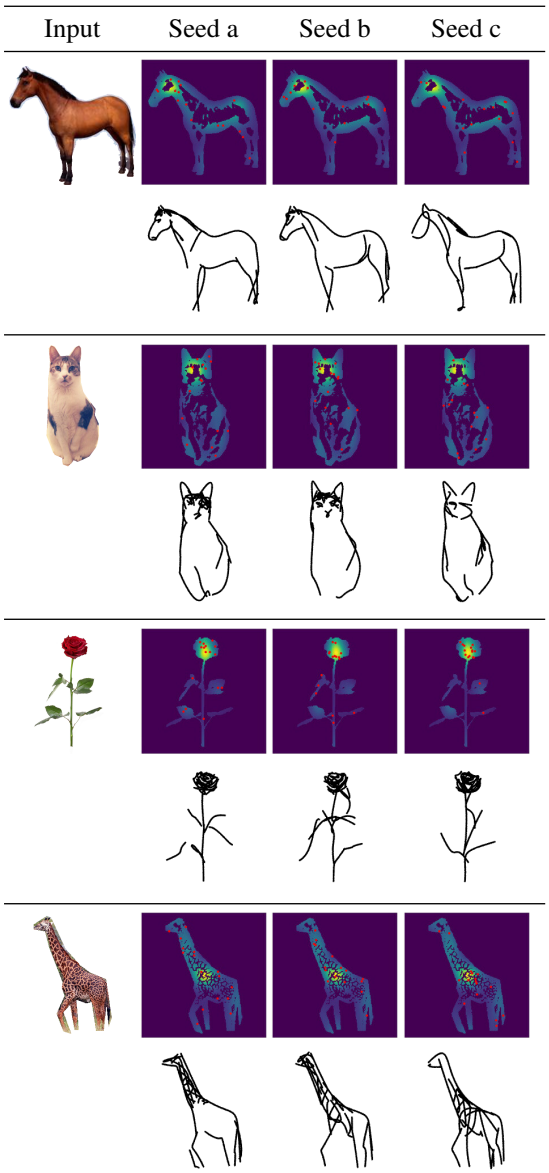


Figure 21. Different Seeds — sorted by final loss score from left to right (best to worst).

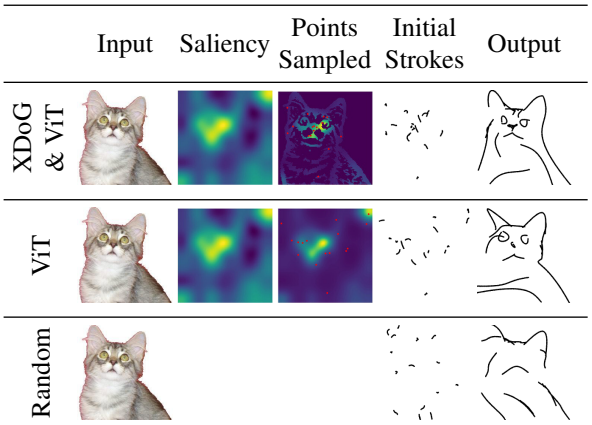


Figure 22. Comparison of different initialization approaches.

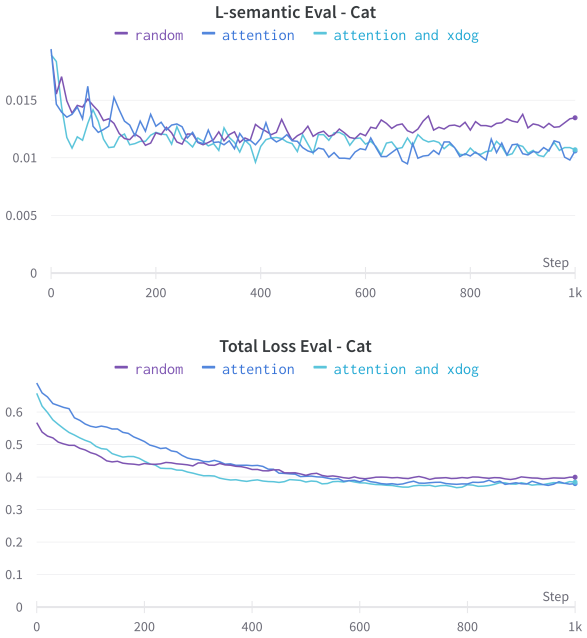


Figure 23. Convergence comparison with different initialization approaches.

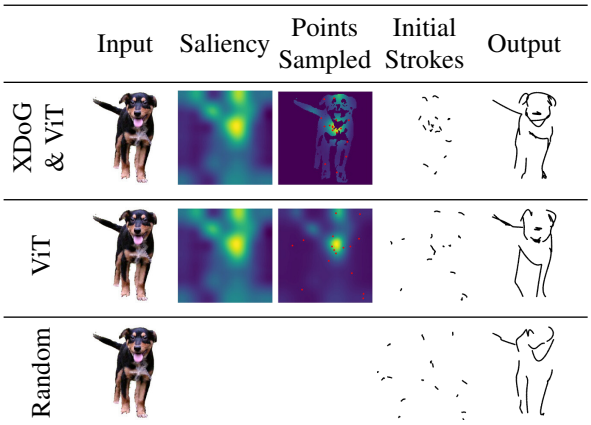


Figure 24. Comparison of different initialization approaches.

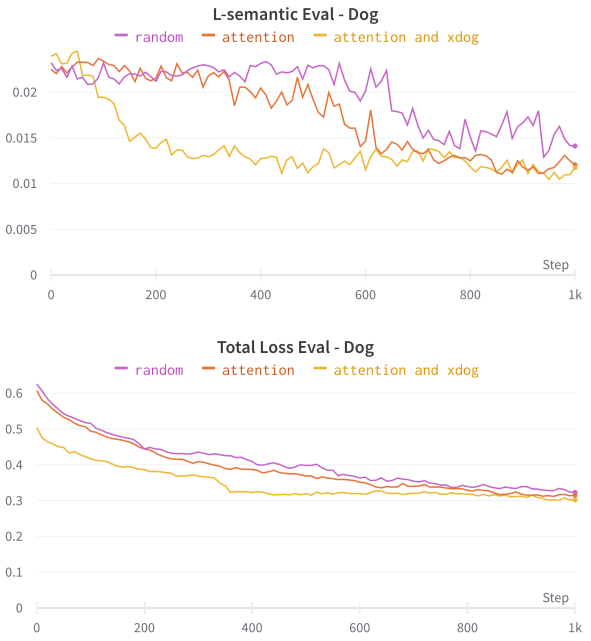


Figure 25. Convergence comparison with different initialization approaches.

D.2. CLIP Attention

The transformer encoder is comprised of alternate layers of self-attention blocks and MLP blocks. For saliency visualization, it is common to use the attention probabilities of each layer, and aggregate them to get the final relevancy map. In Figure 28, we visualize the attention maps of each layer in ViT-32/B architecture on several image classes. In order to produce the attention map for each layer, we take the average of 12 attention heads in that layer. The final relevancy map is simply the average of the attention maps of all layers. It can be seen that the average attention map highlights the salient regions of the input image, such as the head of the horse, cat, and dog, as well as the body.

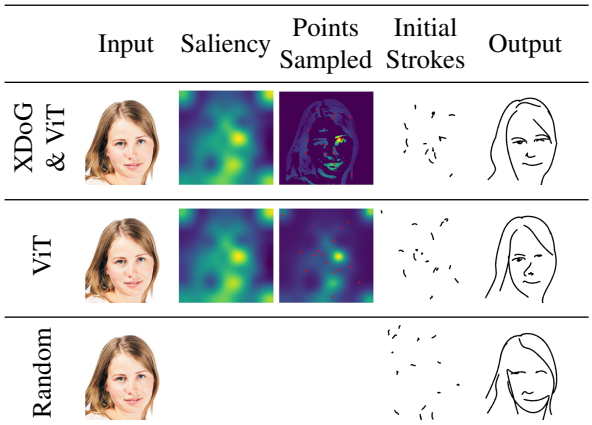


Figure 26. Comparison of different initialization approaches.

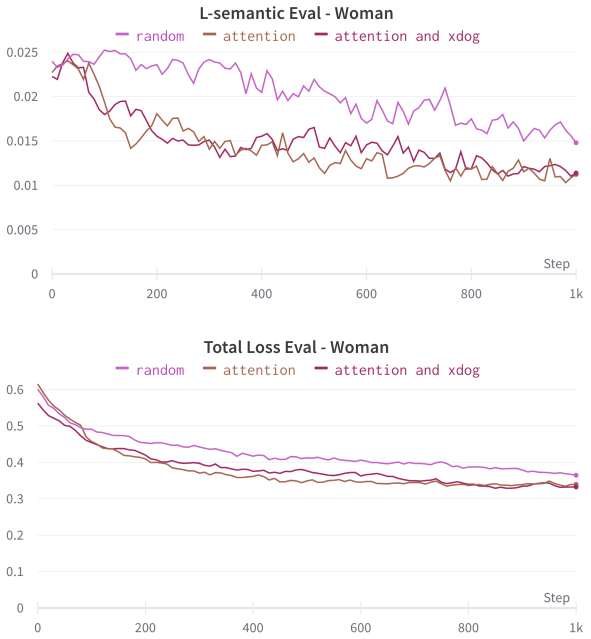


Figure 27. Convergence comparison with different initialization approaches.

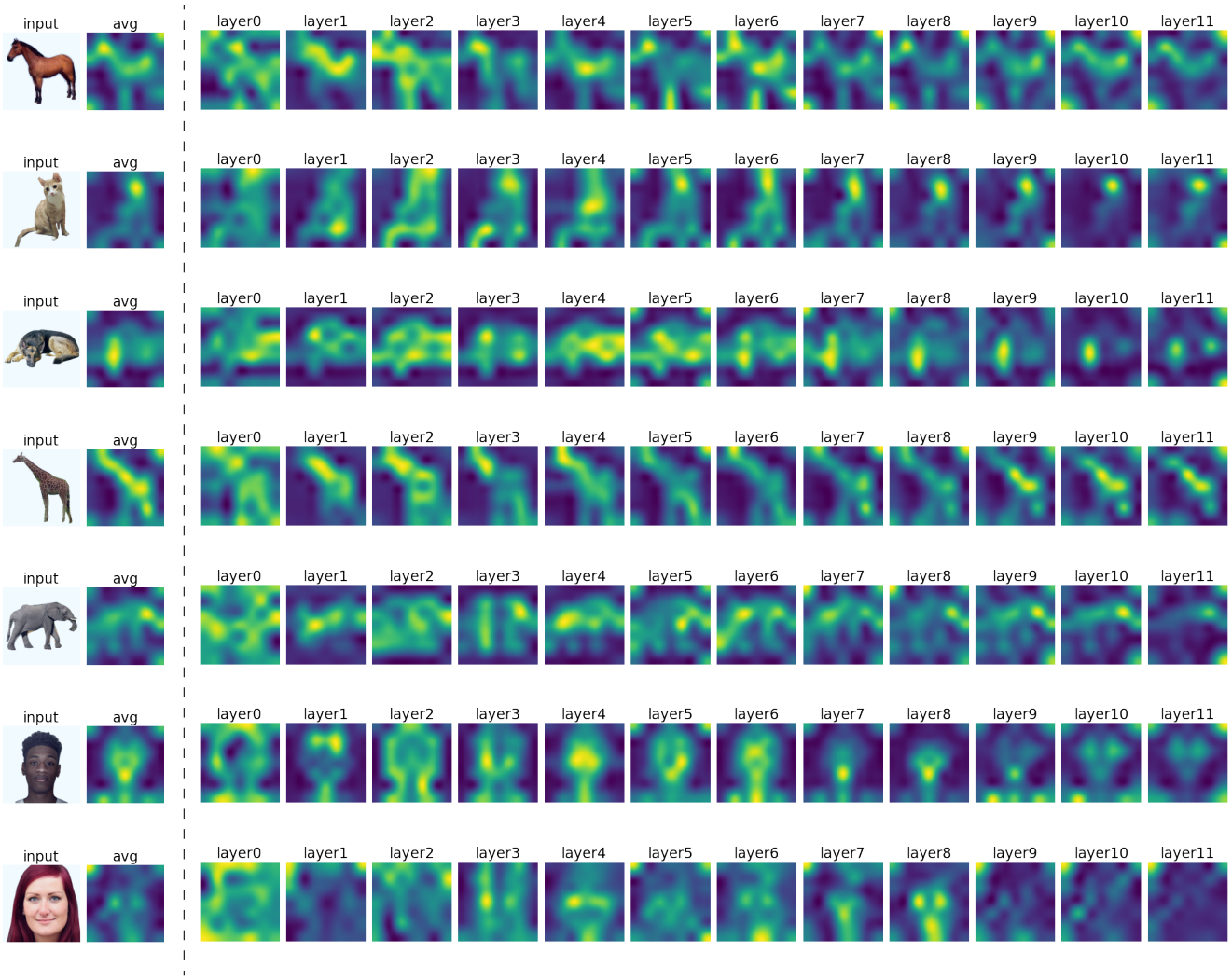


Figure 28. Attention heads of CLIP among all layers.

E. Loss Ablation and Analysis

As stated in the paper, we chose to define our semantic and geometric loss according to CLIP's ResNet101 architecture. In section E.1 we examine this choice by analyzing the results of optimization when different CLIP architectures and intermediate layers are chosen as the basis for the loss function. In section E.2 we provide an ablation study of the different components and weights in the chosen loss function. In section E.3 we further demonstrate the contribution of our loss compared to L2 and LPIPS.

It is important to note that throughout this analysis our goal is to select the objective which best encodes the visual representations of images for extracting a balanced combination of both geometric and semantic features. Enforcing too much geometrical fidelity would result in sketches being very close to simple contours. On the other hand, only focusing on the semantic features can produce uncoherent object depictions or reduce the instance-level recognizability of the synthesized sketches. With a lack of a good automatic metric for this objective, we use visual judgment to select the best configurations.

E.1. CLIP Layers and Architecture Analysis

The authors of CLIP published several pretrained models including ResNet50, ResNet101, and two additional vision transformers namely ViT-B/32 and ViT-B/16. In the following experiments, we used the same initialization method and general settings to obtain the sketches. We only change the layer or architecture used, and we apply the optimization with respect to the loss determined by this layer. We use the cosine similarity to compute the loss for the fully connected layer (final output) and the L2 distance for the loss determined by intermediate layers. In the comparisons, images from three different classes were examined - horse, cat, and human face.

Figure 29 shows the results obtained by using ResNet101 and ResNet50 as the base model. The layer1 for the ResNet architecture is defined as the input to the first Bottleneck block, and the other four layers are simply the output of each Bottleneck blocks in the network. Each column in the figure shows the resulting sketches when the loss is de-

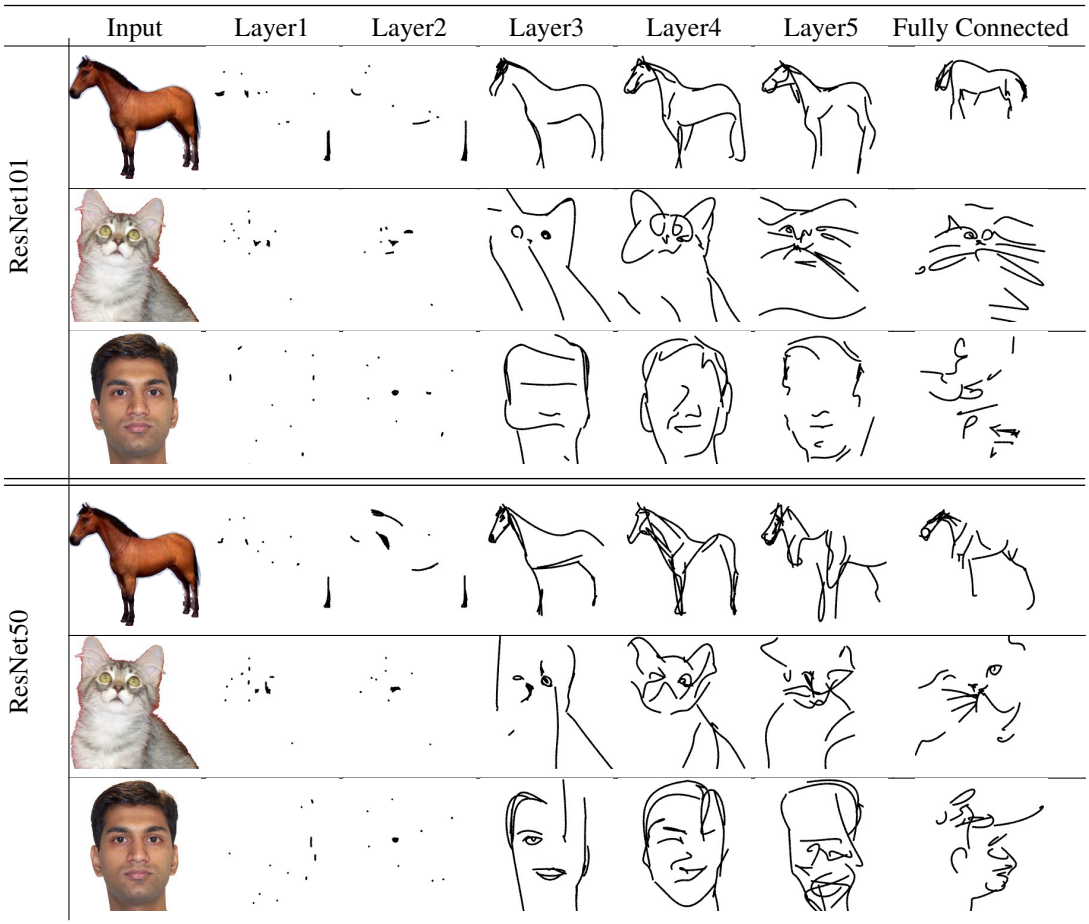


Figure 29. Synthesized sketches by optimizing w.r.t the different layers of RN101 and RN50 architectures

fined based on the activations of the respective layer. We can see that layers 1 and 2 produced non-recognizable results[3]. In contrast, layers 3, 4, and 5 result in reasonable sketches that combine both geometry and semantics, leaning towards more semantic fidelity as we go deeper. The rightmost column contains the results obtained by using only the fully connected layer. This is equivalent to using text as input. We can see that spatial coherence has disappeared completely, and that the model has produced some recognizable class-based features, such as whiskers for the cat. Overall, we observe that ResNet101 produced cleaner and more stable sketches than ResNet50. Layer 3 and 4 of the ResNet101 contain a desirable combination of globally coherent geometry and semantically recognizable features, which led us to use them for defining our geometric-fidelity loss $L_{geometric}$.

In Figures 30 and 31 we present the results of a similar experiment using the vision transformer architectures ViT-16/B and ViT-32/B. To perform the analysis, we utilized the 12 self-attention layers and the last fully connected layer. In the results, we observe that both ViT models encode a higher bias towards shape compared to the ResNet models, especially in layers 5 and lower. The strokes attained using ViT16 are observably messier compared to ViT32 and do not have well-defined start and end points. Comparing the ViT vs. CNN architectures, we favor the latter, and specifically ResNet-101 as it seems to strike a balance between geometric fidelity and semantics.

[3]This can be because of the small receptive field of the feature-maps at this layers.

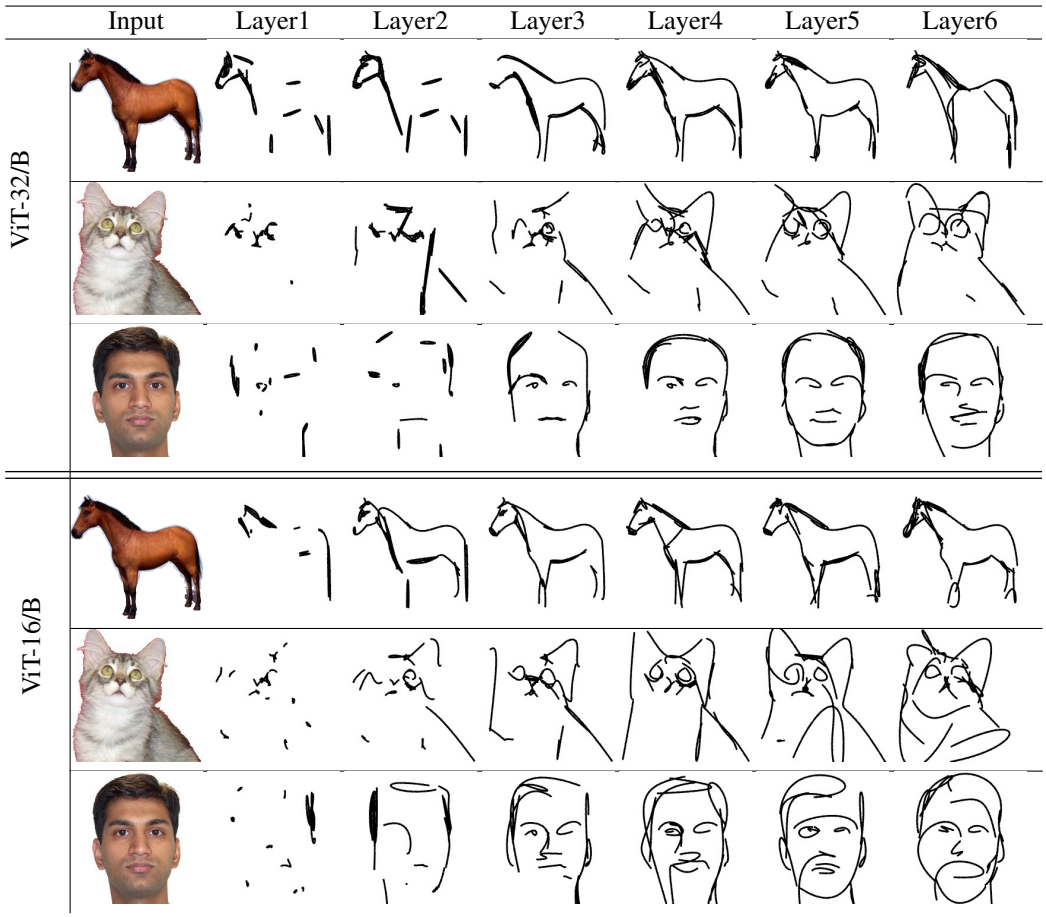


Figure 30. Synthesized sketches by optimizing w.r.t the different layers of ViT32 and ViT16 architectures

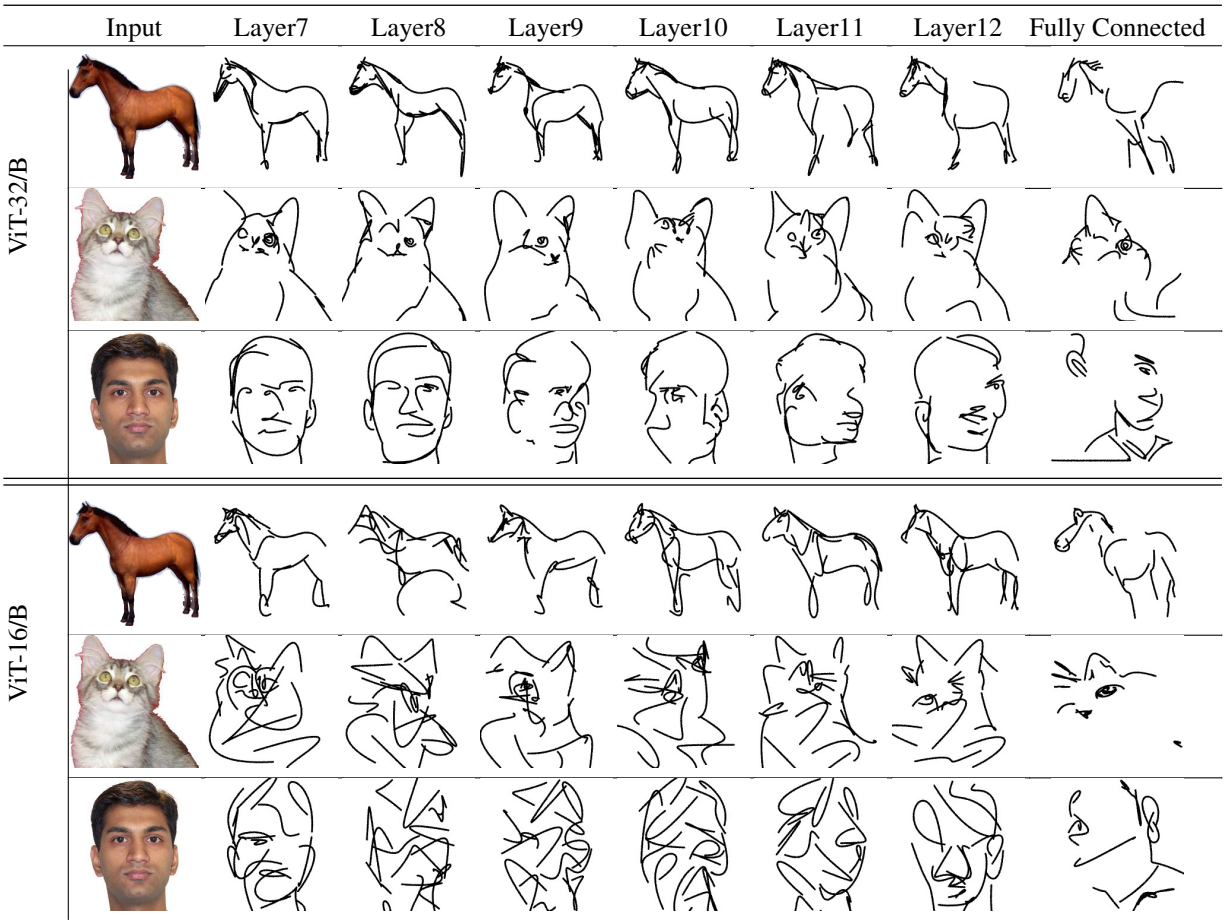


Figure 31. Synthesized sketches by optimizing w.r.t the different layers of ViT32 and ViT16 architectures

E.2. Loss Term Weights

Figure 32 illustrates the effect of excluding each term from the loss formula, the term excluded is listed on top of each column. Omitting $L_{semantic}$ (column L_s) results in sketches that fit well with the geometry, but may not be semantic enough, whereas when omitting $L_{geometric}$ (column L_g), we lose instance-level recognition. For animal subjects, the sketches' class is recognizable, but they do not fit the specific object's geometry well, while for the face image, they are unrecognizable. In columns 4 and 5, layers 3 and 2 have been omitted from $L_{geometric}$. It can be seen that when layer 3 is removed, the results become less semantic and more biased towards geometry, whereas the opposite happens when layer 2 is removed.

Figure 33 illustrates the impact of applying different weights to $L_{semantic}$ - ranging from 0 to 1 (left to right). When applied with weights of 0.5 and 1 (columns 4 and 5), the resulting sketches exhibit geometric changes that may be too extreme, however when we cancel $L_{semantic}$ completely (column 2) the results tend towards geometry. We

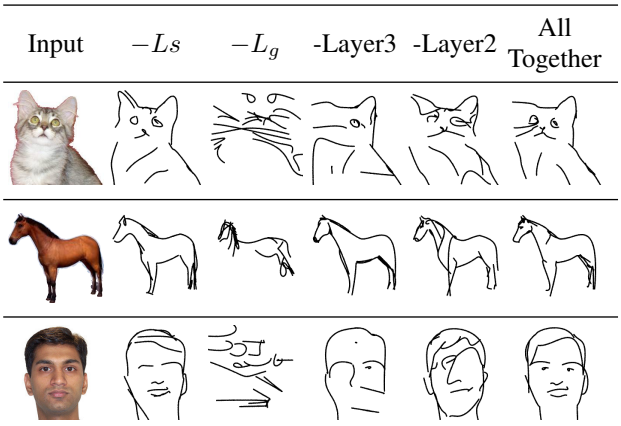


Figure 32. Excluding terms from the loss function - in the left column are the input images, and from left to right are the sketches produced when omitting - $L_{semantic}$, $L_{geometric}$, layer3 from $L_{geometric}$, and layer2 from $L_{geometric}$. In the rightmost column are the results of the proposed method when all parts are included.

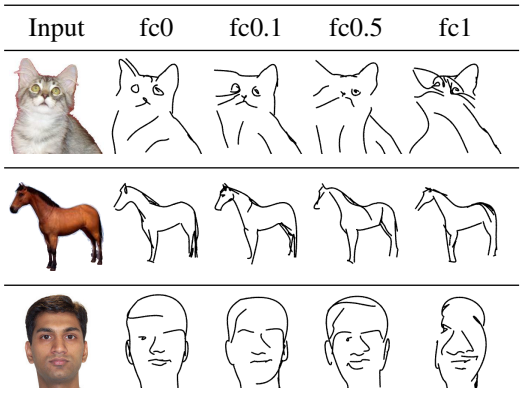


Figure 33. Weights intervals of semantic loss, columns 2 - 5 shows the optimization result when $L_{semantic}$ is weighted by 0, 0.1, 0.5, and 1 correspondingly.

find that weight 0.1 provides a good balance between geometry and semantics. For example, some whiskers are added to the cat while the general geometry is also maintained. Additionally, the geometry of the horse's head varies slightly from that in the input image, but the pose is correct.

E.3. Comparison to Other Loss Functions

Our proposed loss is a perceptual loss, providing a combination between geometry and semantics. In this section we compare the proposed loss to L2 and LPIPS [44]. While L2 is purely geometric in that it measures pixel-wise distance, LPIPS is considered more semantic, since it measures the distance between the intermediate activations of a pretrained vision network (trained for a classification task using ImageNet dataset and the VGG16 architecture). In Figure 34 we compare these losses with our loss on four images. Each result displayed in the figure was obtained through the same initialization method (saliency guided), with the only difference being the loss used for optimization. The XDoG edge map is presented as an edge-map baseline. It is evident that the results produced using L2 do not contain any semantic information. L2 encourage the optimization to "fill" the colored pixels in the input image. The sketches improved significantly when using LPIPS, the strokes follow the semantics of the image. We observe, however, that LPIPS is too geometric in comparison with our loss. The results produced by our method are shown in the last column. Improvements can be seen clearly, especially for the face, where semantic visual attributes are introduced to the sketch.

F. Quantitative Comparison

Figure 35 provides the per-class classification accuracies of the CLIP ViT-B/32 and ResNet34 classifiers on the sketches synthesized by different methods using images from SketchyCOCO dataset. Figure 36, 37 show sam-

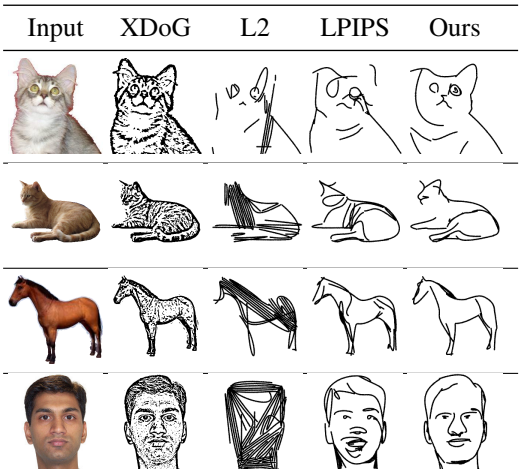


Figure 34. Replacing our CLIP-based geometric loss with LPIPS, and L2. The animal sketches are drawn using 16 strokes and the face sketch is drawn using 32 strokes.

ple sketches that are missclassified by the pretrained classifiers that we used for our quantitative evaluation, namely ResNet34, and CLIP ViT-B/32. Each column in the figure shows the sketches synthesized by the respective method, while the text in green showing the correct class and the text in red showing the prediction of the model.

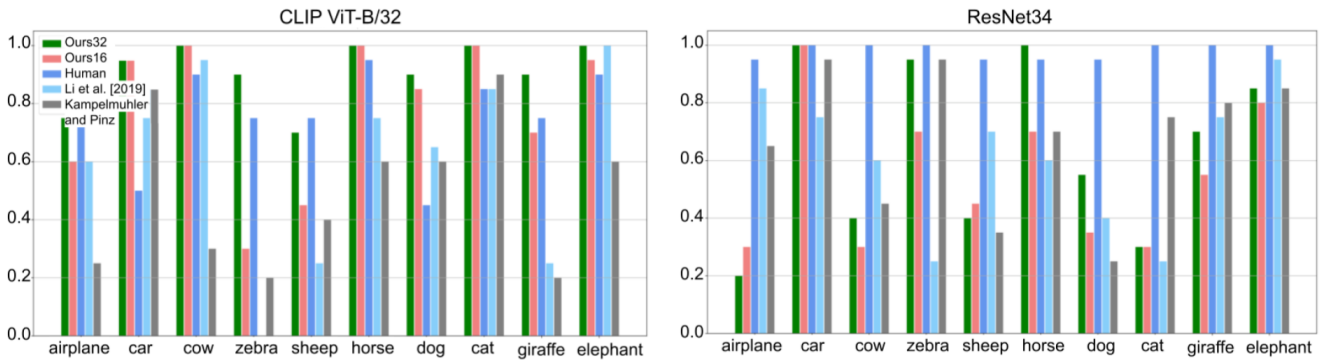


Figure 35. Per-class classification accuracy of the two classifiers we used in our quantitative analysis. The figure on top shows the results for CLIP ViT-B/32 and the figure below shows the results for ResNet34 classifier. Each color represent sketches by the corresponding sketch synthesis model.

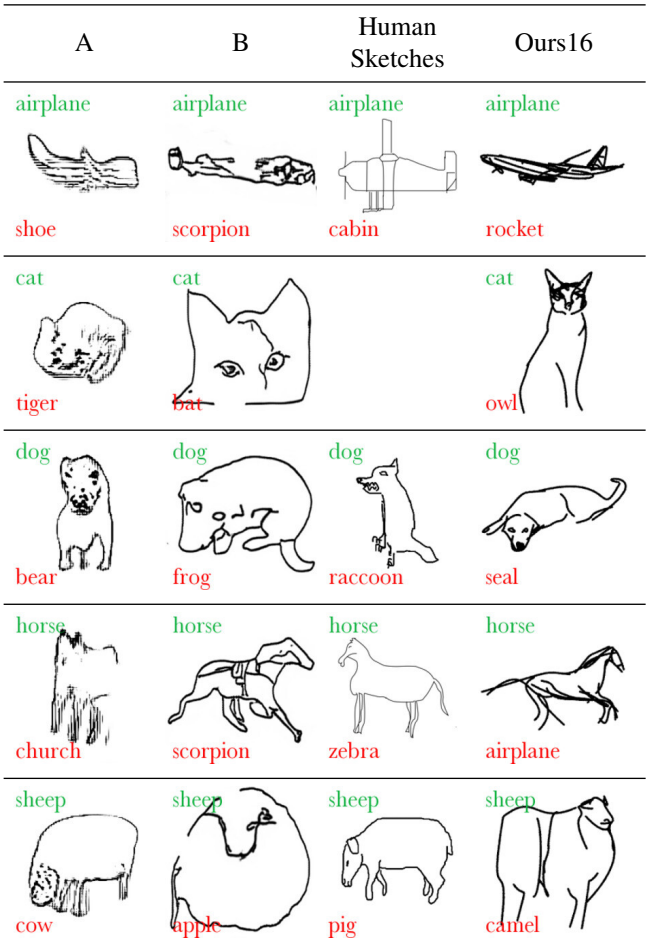


Figure 36. ResNet34 miss-classifications on the sketches generated by different methods (A) Kampelmühler and Pinz [21], (B) Li et al. [23]. The green text on each sketch shows to correct class and the red text shows the predicted class.

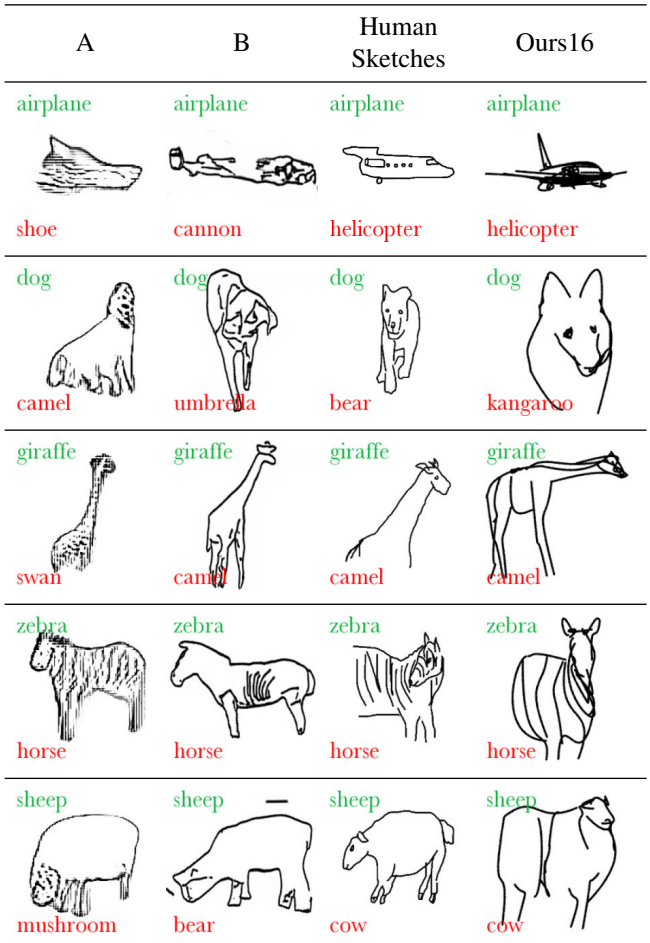


Figure 37. CLIP ViT-B/32 miss-classifications on the sketches generated by different methods (A) Kampelmühler and Pinz [21], (B) Li et al. [23]. The green text on each sketch shows to correct class and the red text show the predicted class.

G. Additional Qualitative Comparisons

In this section, we provide additional comparisons to competitor methods. In Figure 38 we show the results generated by CLIPDraw when applied with different settings. Based on the authors' standard practice, the colored drawings were generated with 256 color strokes. In the second column, the input to the optimization is the text "A sketch of a(n) *class-name*". Under these circumstances, it is not possible to control the geometry or appearance of the output drawing to produce a sketch that closely resembles an input image. In the third column, we present the output of CLIPDraw when the image shown in the first column is provided as input instead of text. Under these settings, the output drawing is still not consistent with the geometry of the input image. When using both text and image to guide the optimization of CLIPDraw (fourth column), the output drawing still lacks the geometric grounding. Results presented in column 5 are from the main paper, where we limit the output domain of CLIPDraw to sketch styles and utilize the input image to guide the optimization process. The sketches produced by our method are presented in the right column. Our method allows control over the visual appearance of the sketch, which is a valuable feature in a variety of art and design settings. Additionally, since we restrict the primitives to a small set of thin, black curves, we encourage the depiction of critical semantic and structural details from the input image.

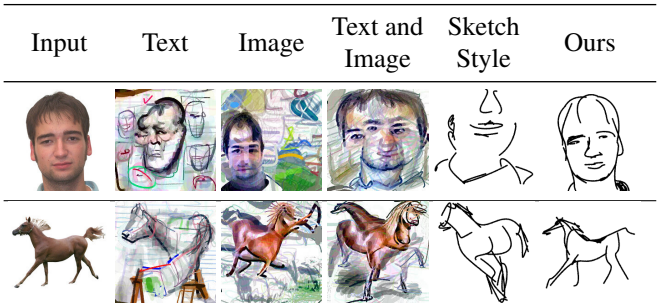


Figure 38. Comparison to CLIPDraw [11].

In Figures 39 and 40 we provide additional comparison to SketchLattice [32] and Song et al. [38] on shoes images.

In Figures 41, 42, 43, and 44 we provide additional comparison to (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11] (using the sketch style and image as an input). The input images used are from the SketchyDatabase [37] with four classes - teapot, horse, duck, and bicycle. The classes and images were chosen according to the publicly available results provided by Li et al. [25].

In Figure 45 we provide additional comparison on portrait-style human faces from the NPR benchmark [36].

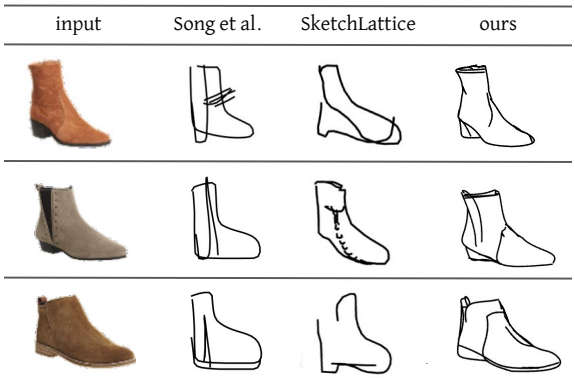


Figure 39. Comparison to Song et al. [38] and SketchLattice [32]

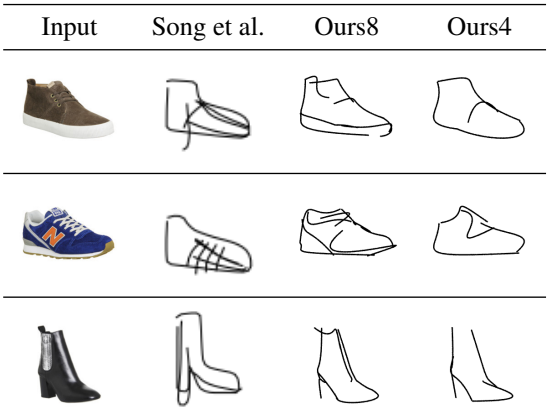


Figure 40. Comparison to Song et al.

This comparison was conducted using the methods of (A) Li et al. [25], (B) Li et al. [23] and (C) Mo et al. [29]. (C) is a recent method to produce vector line drawings from a rasterized input image. Photo-sketch generation is not the focus of this work; nonetheless, they demonstrate an application of converting a face image into a line drawing. The figures illustrate that the sketches produced by Mo et al. are mostly geometric and do not reflect the semantic characteristics of the input faces.

Figure 41. Comparison to competitor methods on images from the ”teapot” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

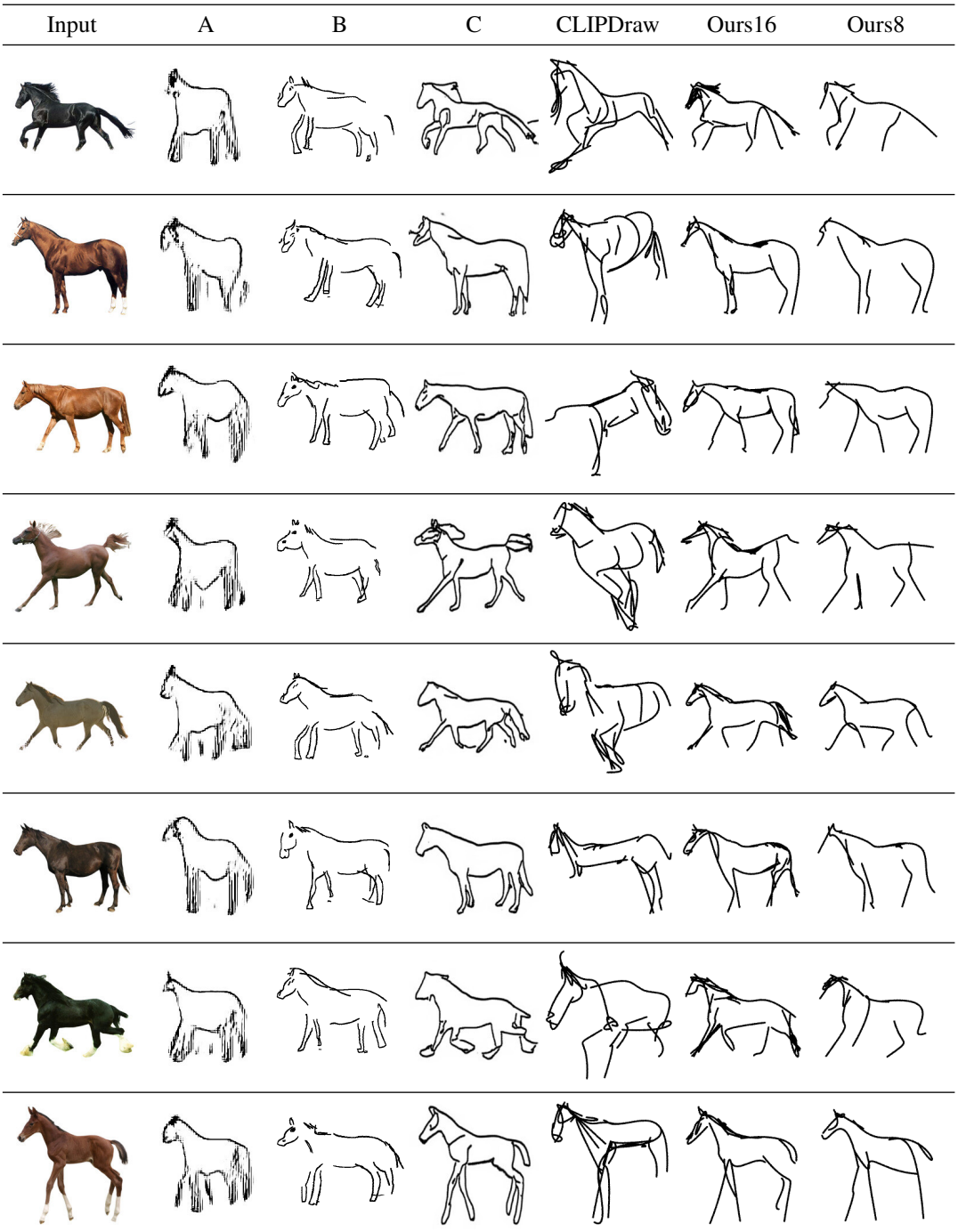


Figure 42. Comparison to competitor methods on images from the ”horse” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

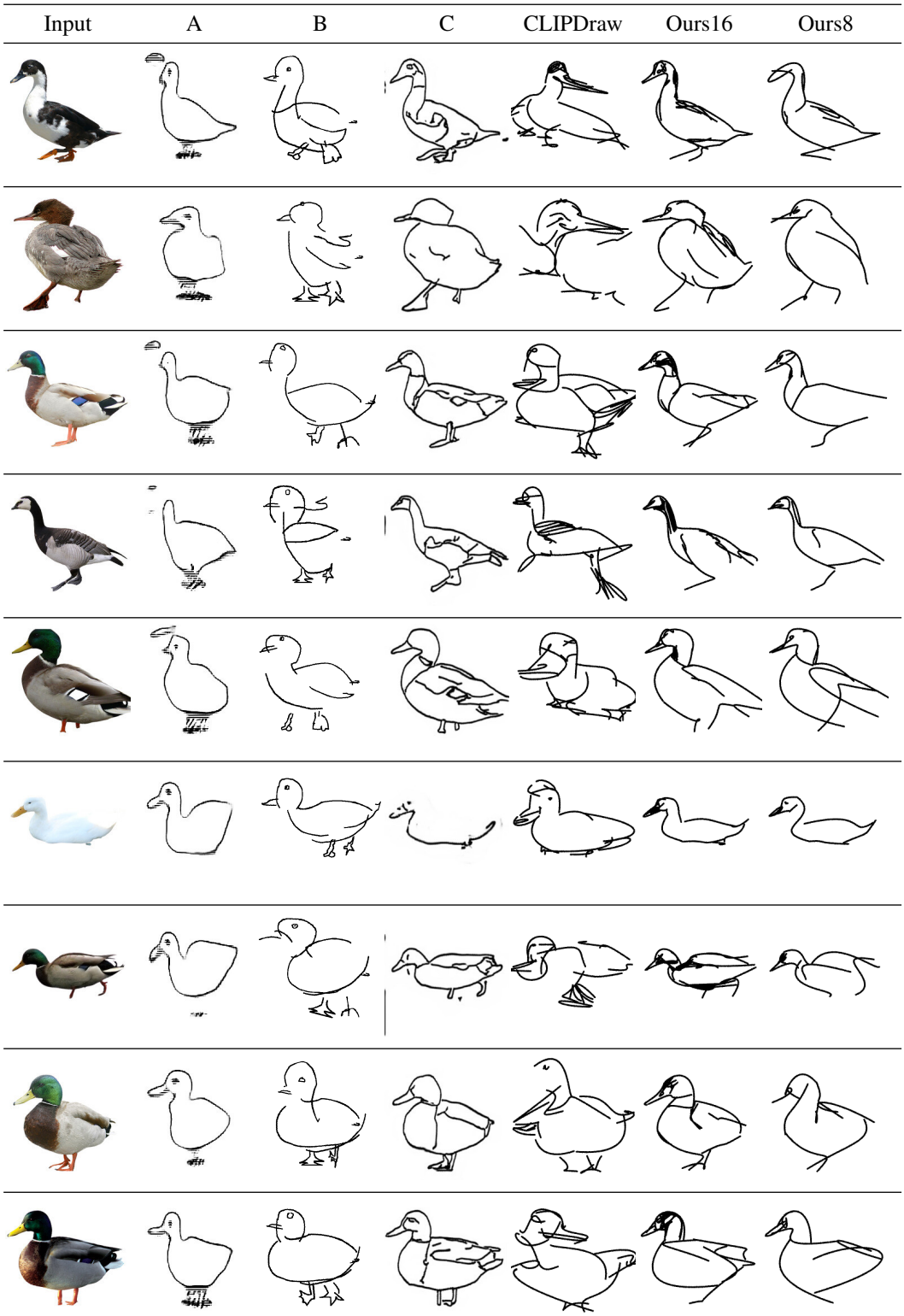


Figure 43. Comparison to competitor methods on images from the ”duck” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

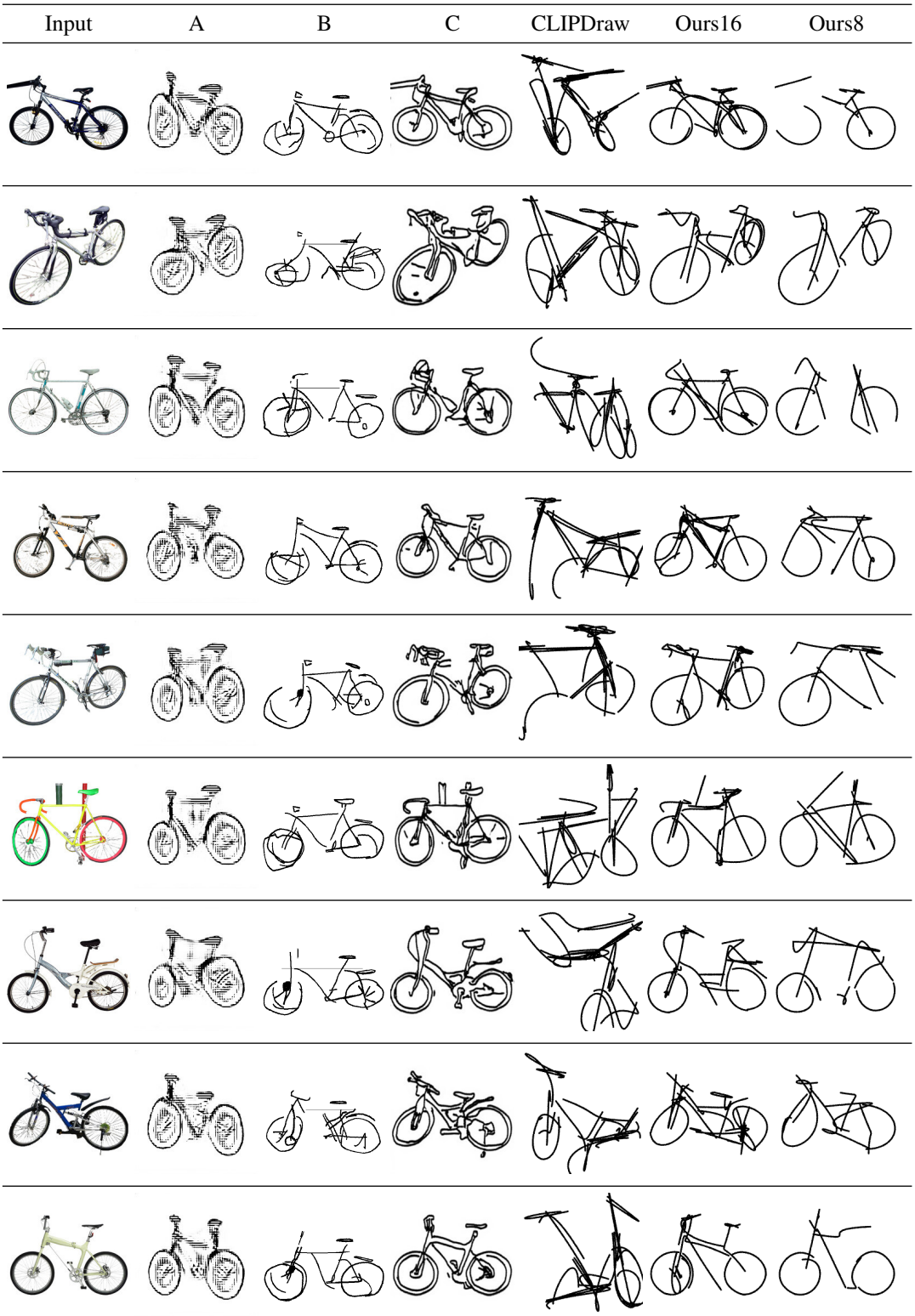


Figure 44. Comparison to competitor methods on images from the ”bicycle” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

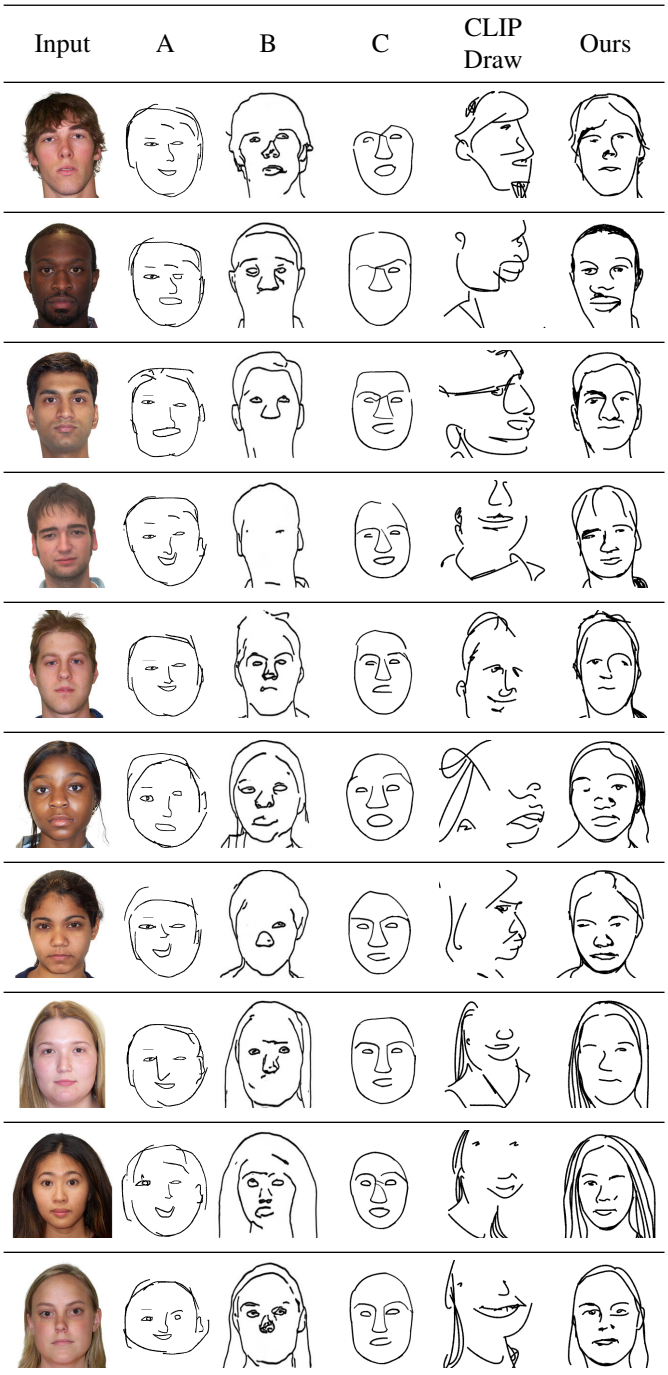


Figure 45. Comparison to competitor methods on human faces. Left to right: (A) Li et al. [25], (B) Li et al. [23], (C) Mo et al. [29], and CLIPDraw [11].

H. Additional Qualitative Results

As stated in the paper, we mainly use four control points to define each Bezier curve, however, it is also possible to change the degree of the curves to achieve different styles and levels of abstraction. In Figures 46, 47, and 48 we provide additional results generated by our method, demonstrating three levels of abstraction for each image, and three different styles. The numbers on top indicate the total number of control points used to generate the sketches (64, 24, and 16). In each block, each row shows a different style: first row – 4 control points per stroke, second row – 3 control points per stroke, third row – 2 control points per stroke (i.e. straight lines).

In Figure 49 we show additional results on portrait-style human faces from the NPR benchmark [36]. The sketches produced using 64, 32, 16, and 8 strokes, with 4 control points each.

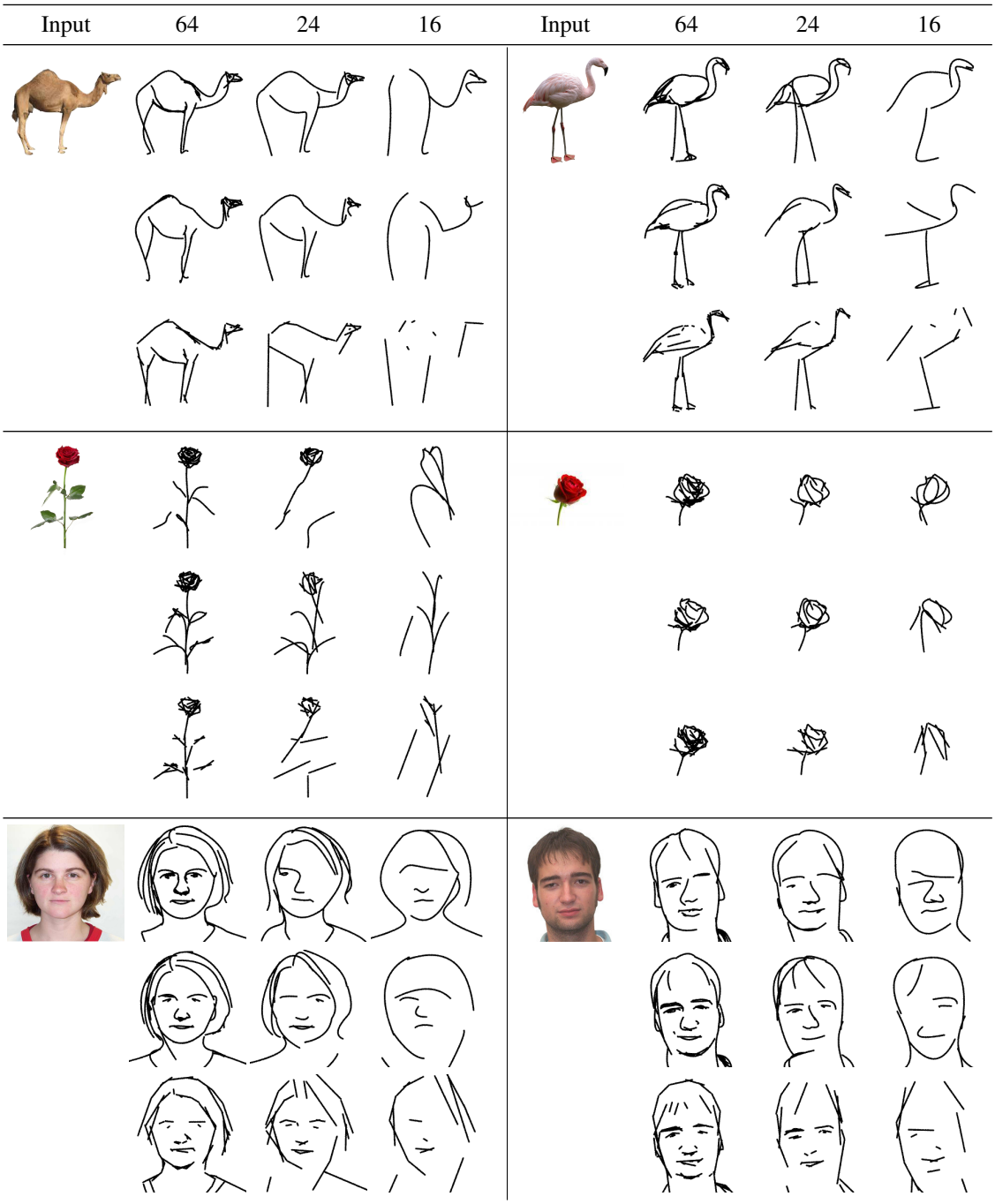


Figure 46. Sketches produced by our method with different styles and levels of abstraction. The columns indicate the total number of control points used to generate the sketch - 64, 24, and 16. The rows indicate the number of control points used for each stroke, which defines the style - 4, 3, and 2 control points respectively.

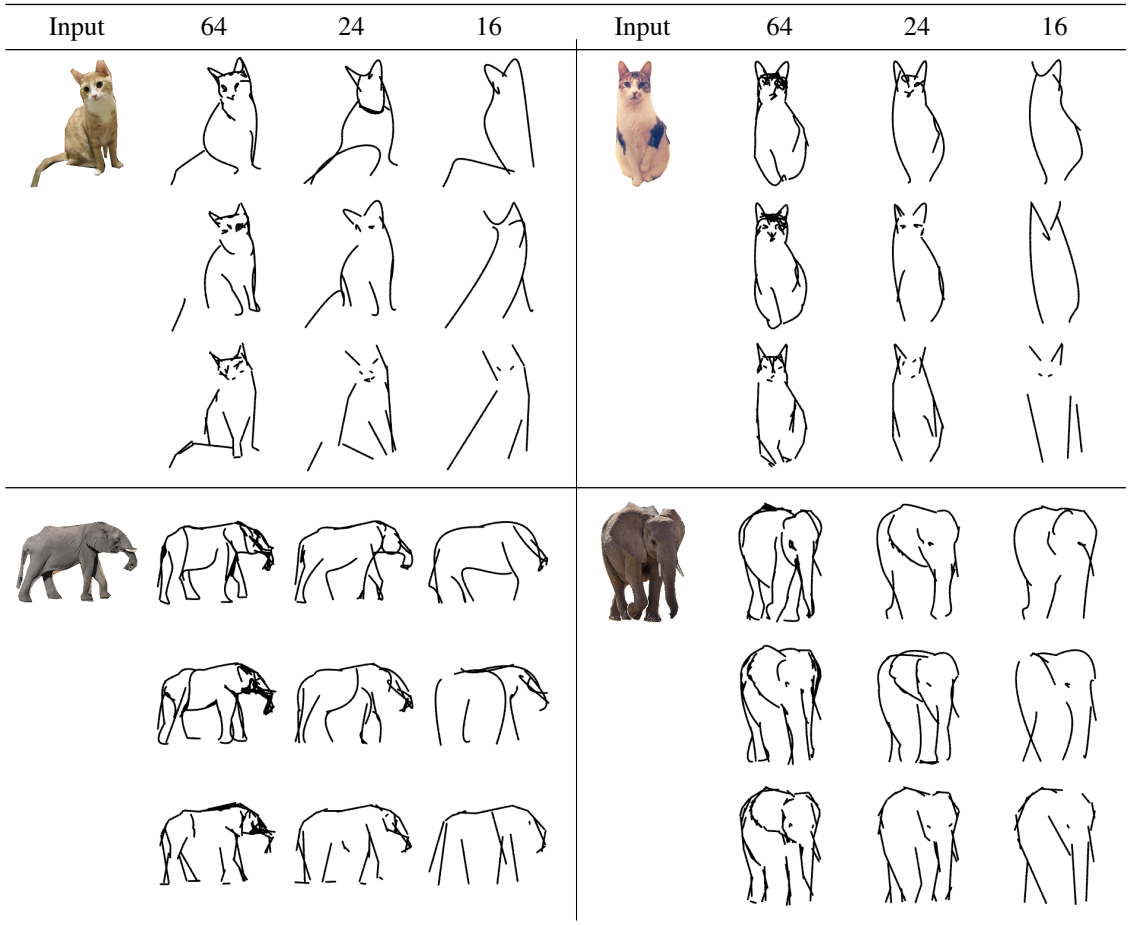


Figure 47. Sketches produced by our method with different styles and levels of abstraction. The columns indicate the total number of control points used to generate the sketch - 64, 24, and 16. The rows indicate the number of control points used for each stroke, which defines the style - 4, 3, and 2 control points respectively.

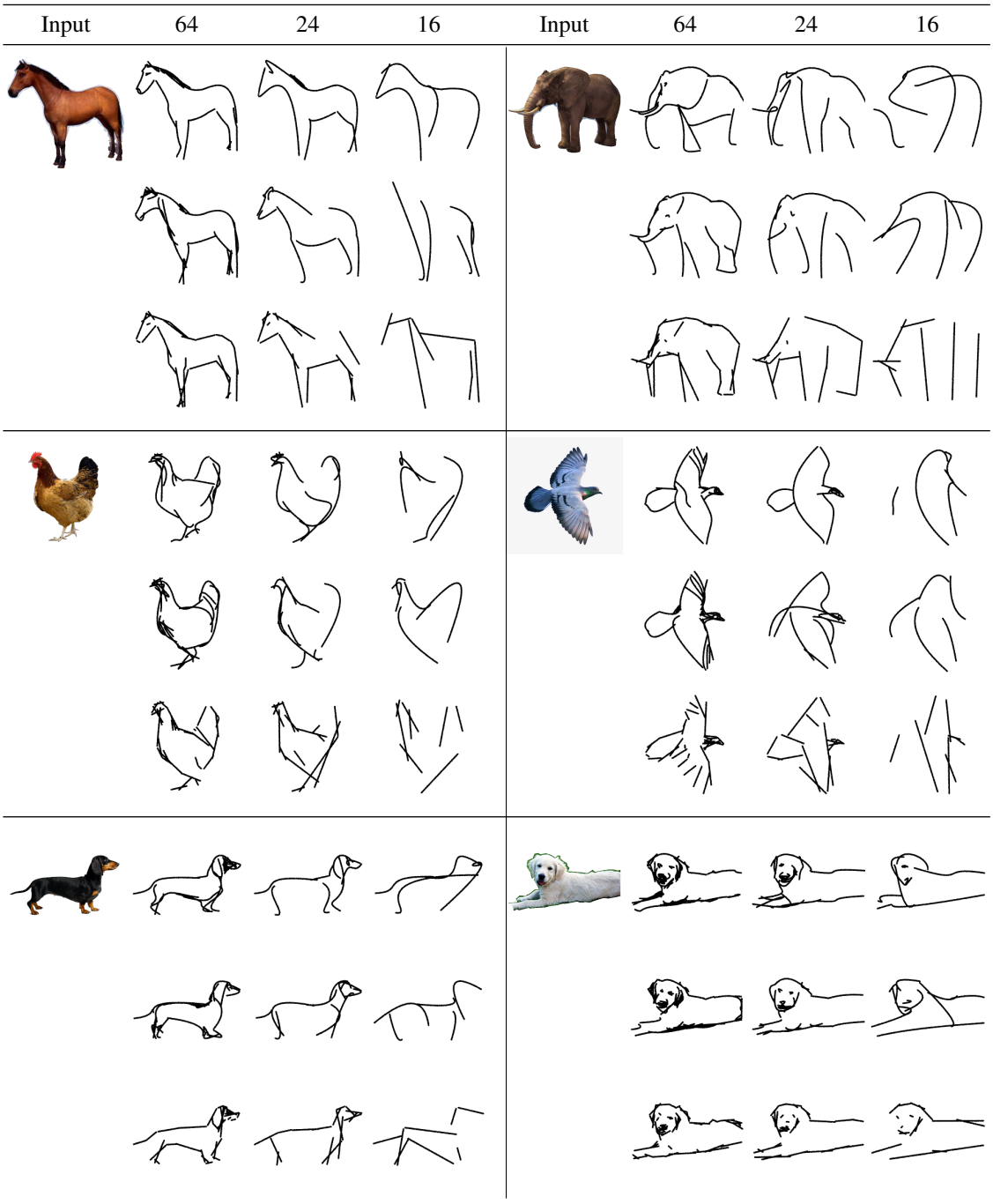


Figure 48. Sketches produced by our method with different styles and levels of abstraction. The columns indicate the total number of control points used to generate the sketch - 64, 24, and 16. The rows indicate the number of control points used for each stroke, which defines the style - 4, 3, and 2 control points respectively.

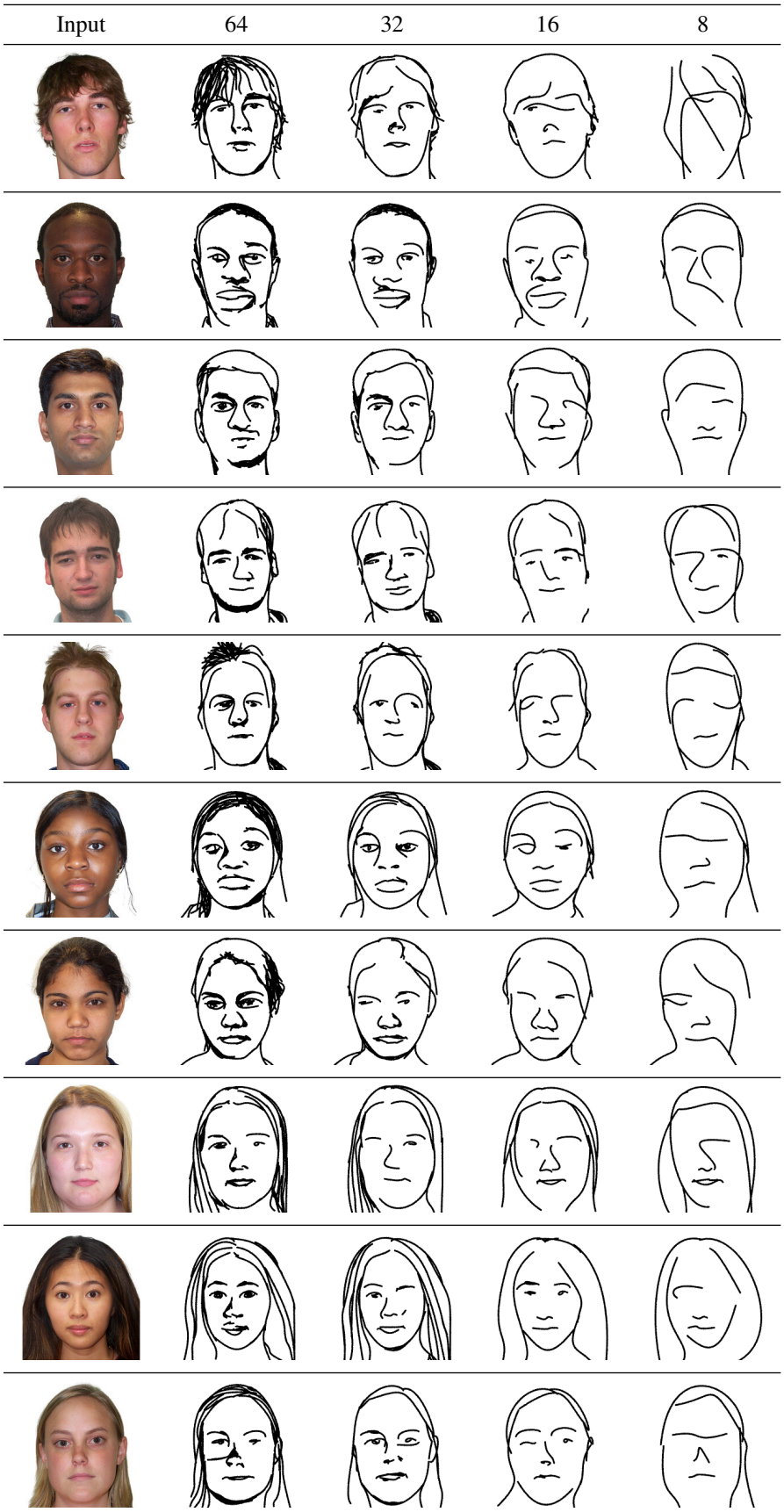


Figure 49. Sketches of human faces produced by our method with different levels of abstraction.

Figure 50. Sketching ”in the wild”: results of 100 random images of cats from SketchyCOCO [13].

Figure 51. Sketching ”in the wild”: results of 100 random images of cats from SketchyCOCO [13].