CLIPasso: Semantically-Aware Abstract 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.

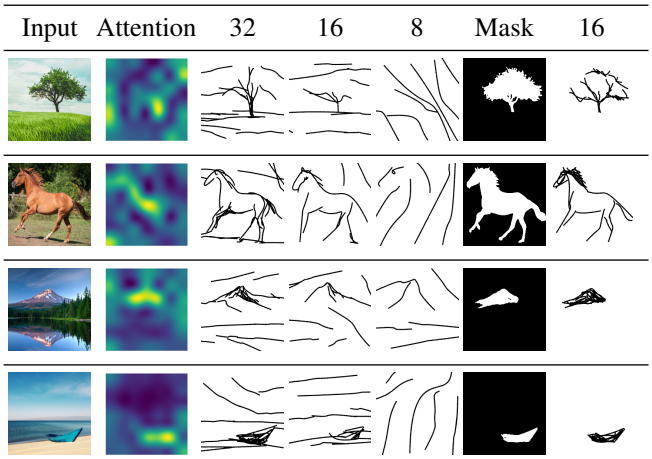

Figure 14. Background Analysis - the first column shows the input images with background that were used for our optimization, the second column shows the extracted attention map. Columns 3, 4, and 5 show the output sketches produced by our method, using 32, 16, and 8 strokes respectively. Column 6 shows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

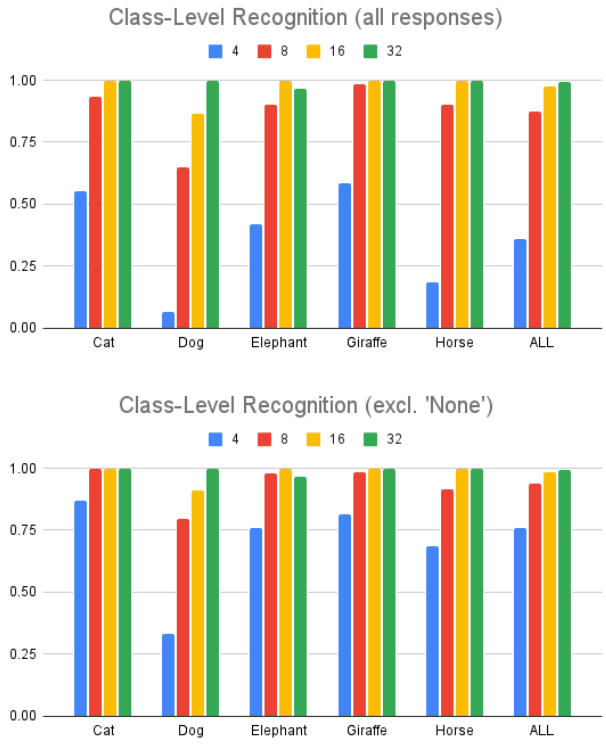

Figure 15. Class level recognizability user study results.

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.

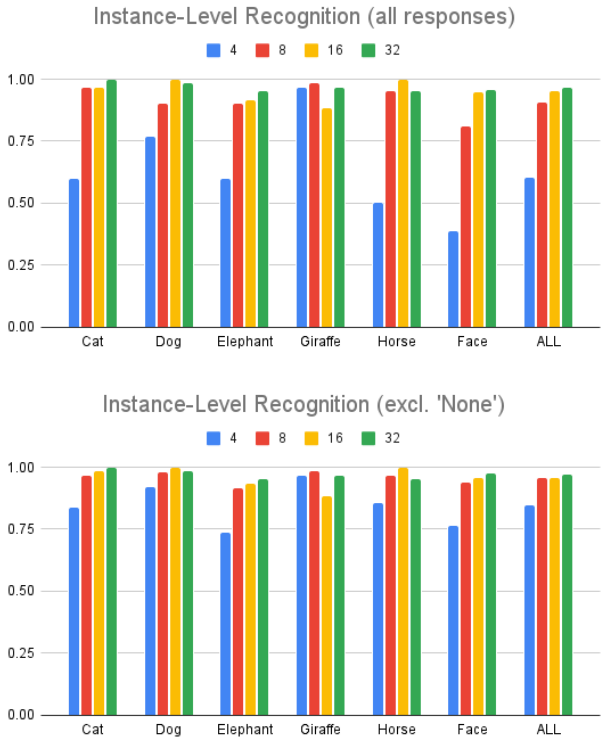

Figure 16. Instance level recognizability user study results.

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.

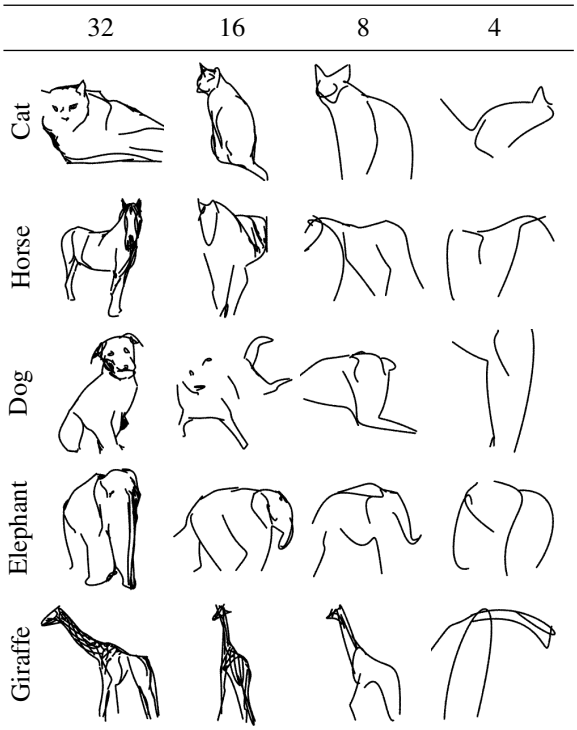

Figure 19. Sample animal sketches produced by our method, that were used in the user study.

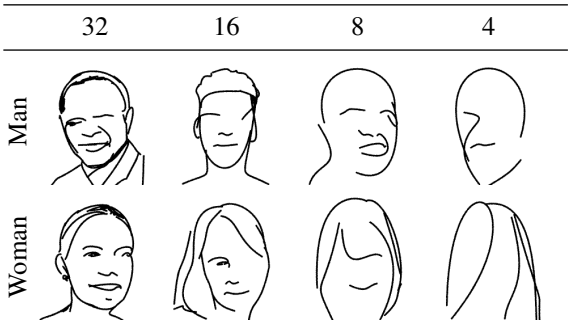

Figure 20. Sample face sketches produced by our method that were used in the user study.

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.

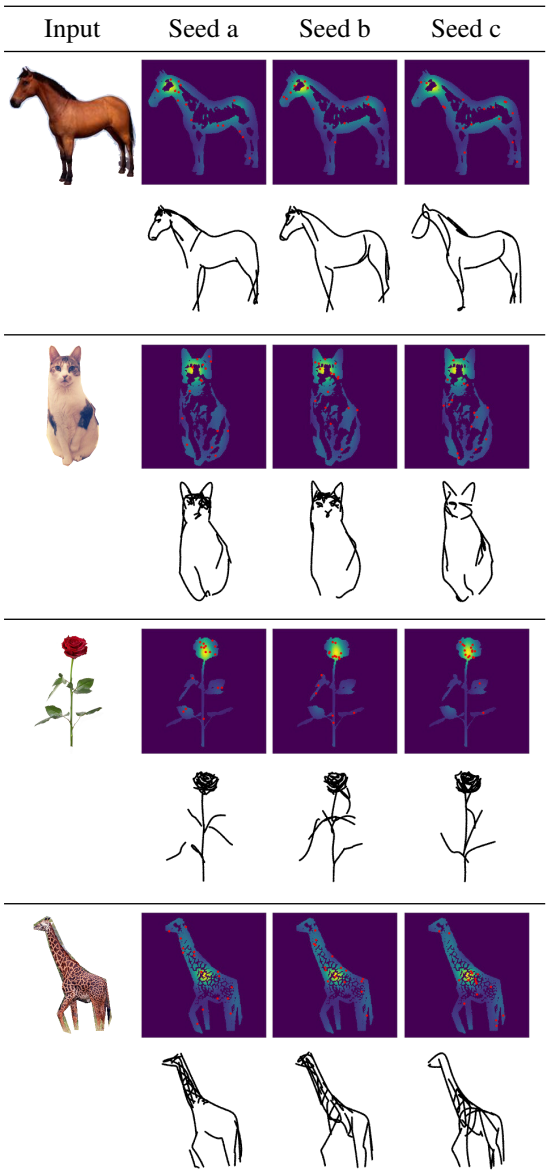

Figure 21. Different Seeds — sorted by final loss score from left to right (best to worst).

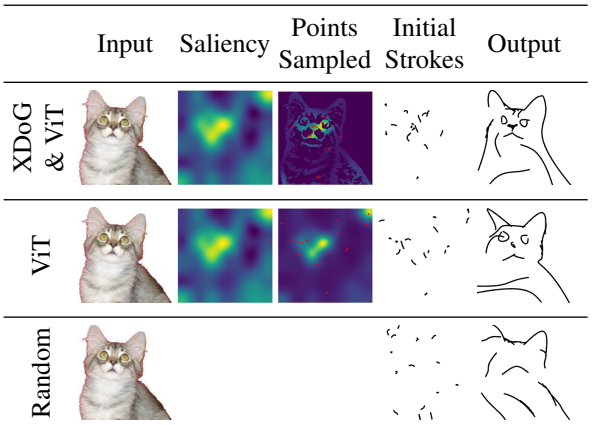

Figure 22. Comparison of different initialization approaches.

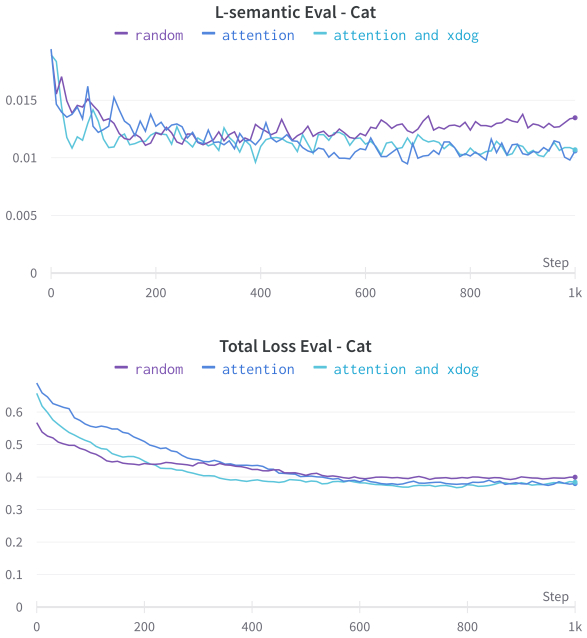

Figure 23. Convergence comparison with different initialization approaches.

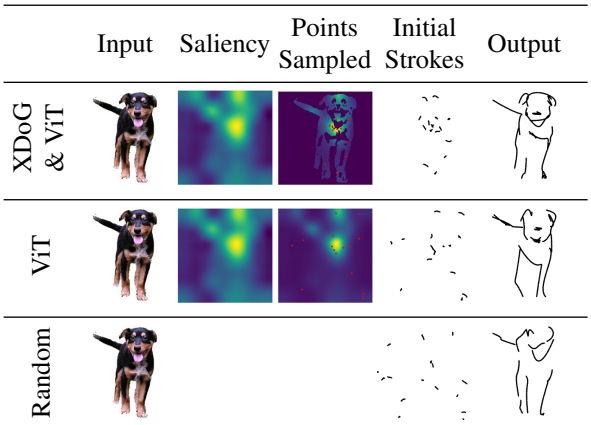

Figure 24. Comparison of different initialization approaches.

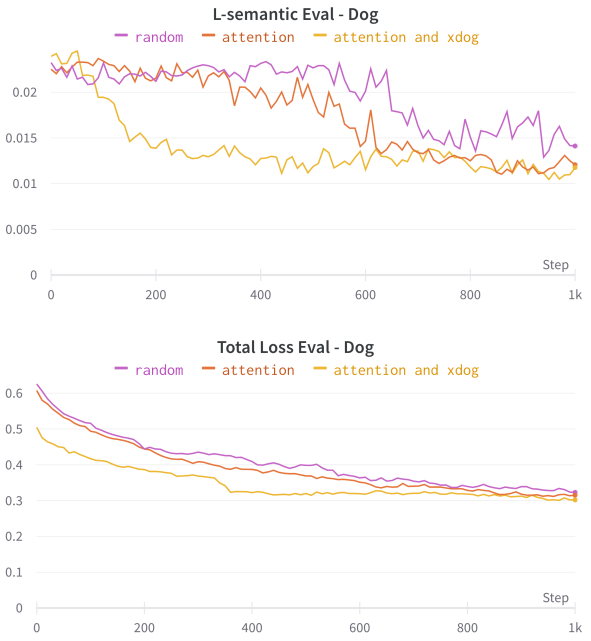

Figure 25. Convergence comparison with different initialization approaches.

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.

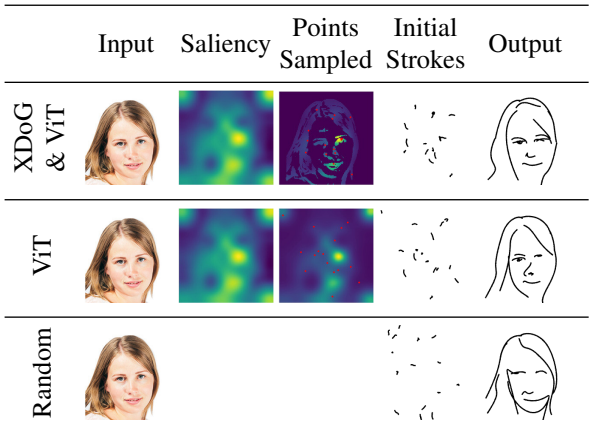

Figure 26. Comparison of different initialization approaches.

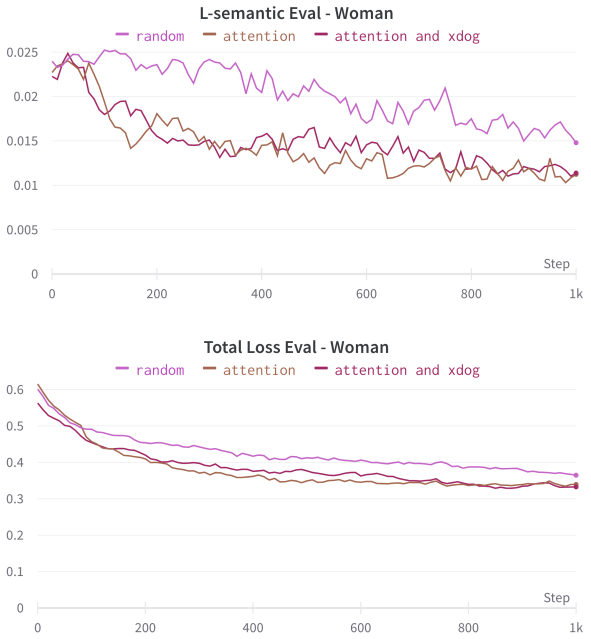

Figure 27. Convergence comparison with different initialization approaches.

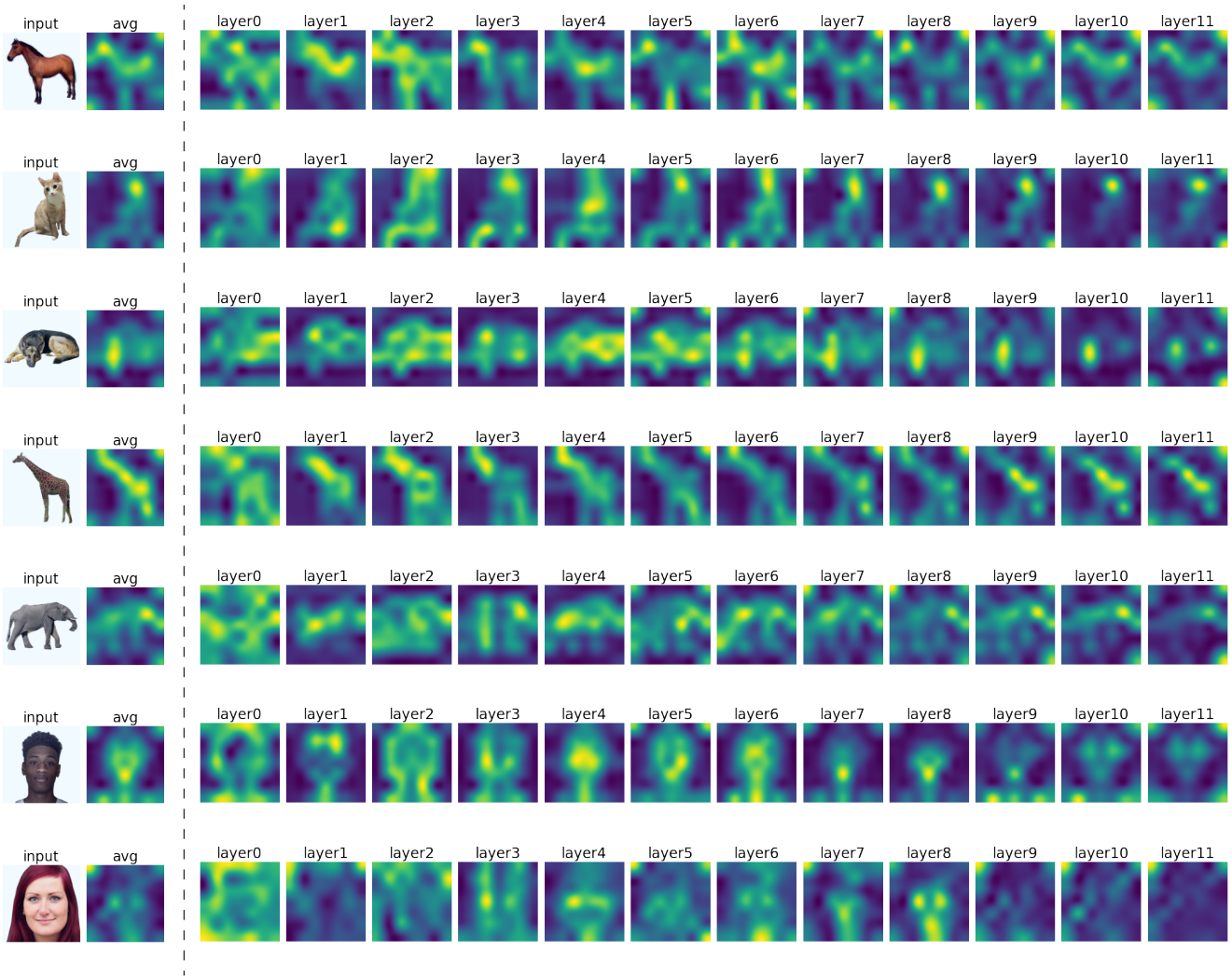

Figure 28. Attention heads of CLIP among all layers.

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-

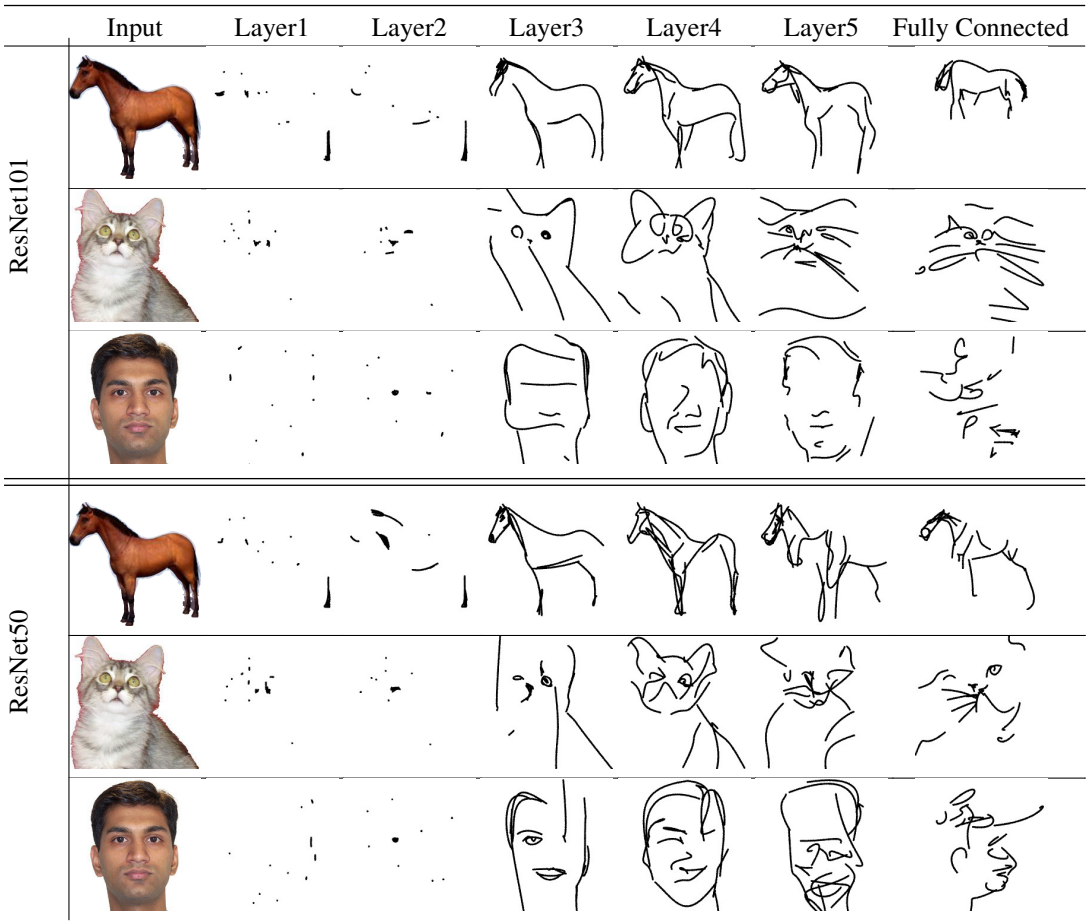

Figure 29. Synthesized sketches by optimizing w.r.t the different layers of RN101 and RN50 architectures

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.

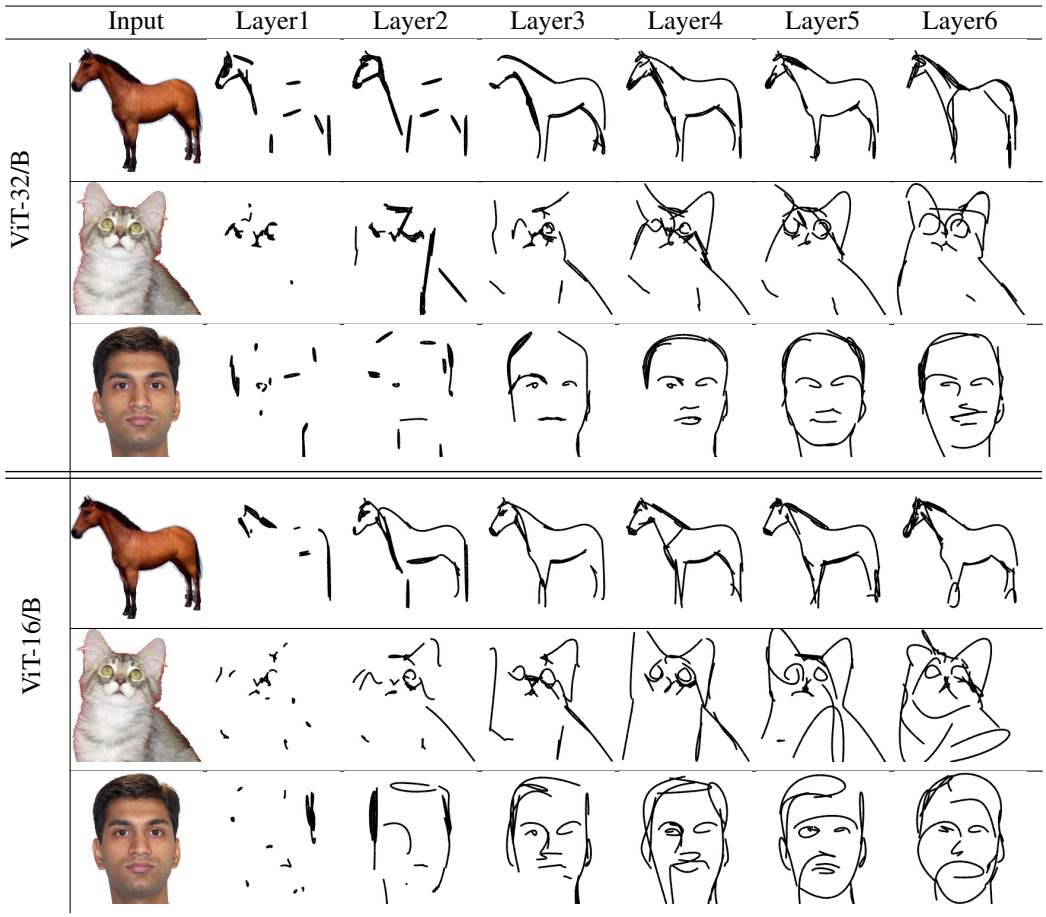

Figure 30. Synthesized sketches by optimizing w.r.t the different layers of ViT32 and ViT16 architectures

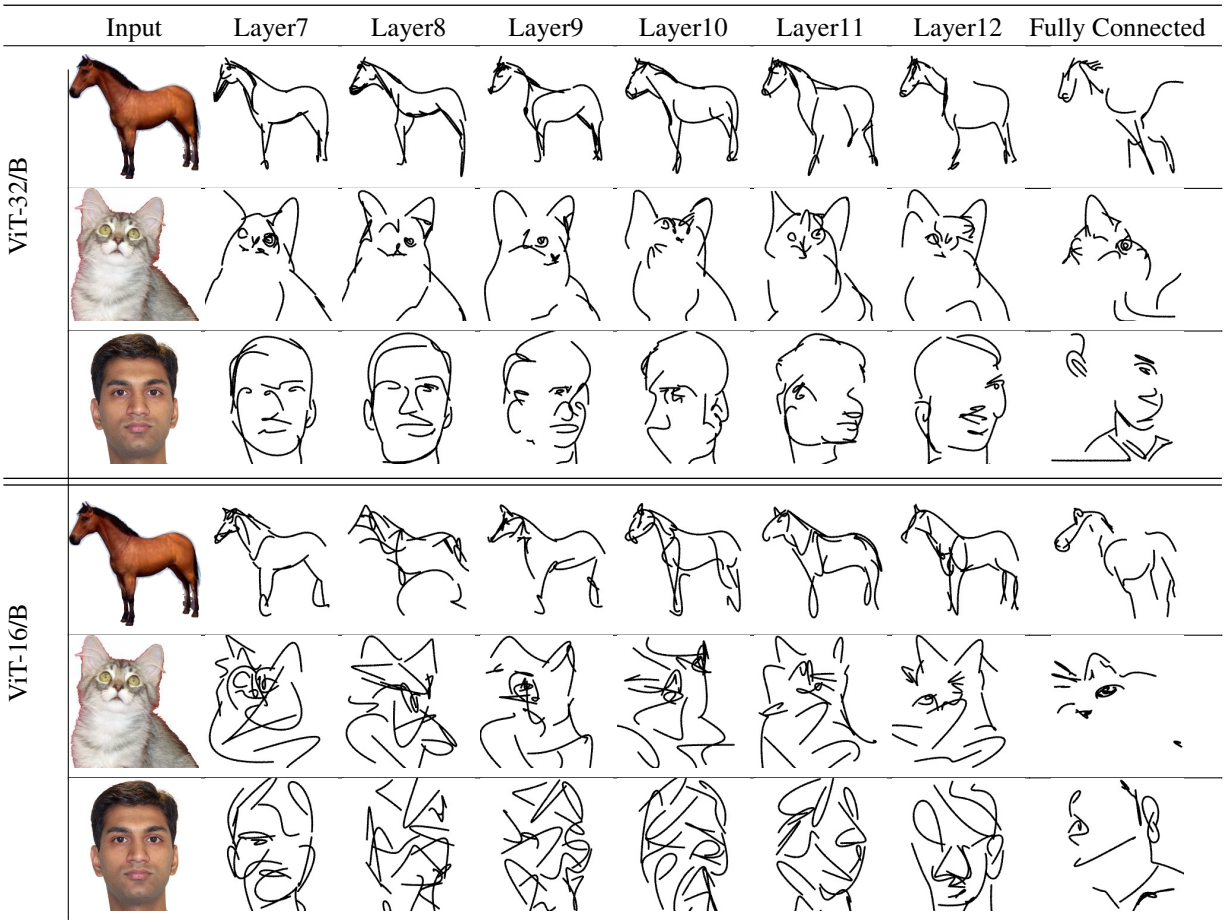

Figure 31. Synthesized sketches by optimizing w.r.t the different layers of ViT32 and ViT16 architectures

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

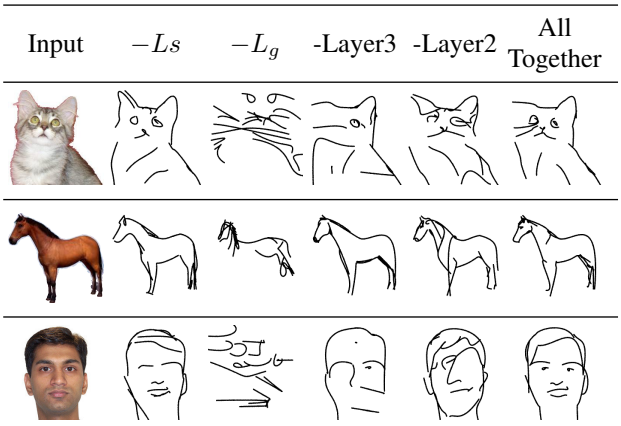

Figure 32. Excluding terms from the loss function - in the left column are the input images, and from left to right are the sketches produced when omitting - $L_{semantic}$, $L_{geometric}$, layer3 from $L_{geometric}$, and layer2 from $L_{geometric}$. In the rightmost column are the results of the proposed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

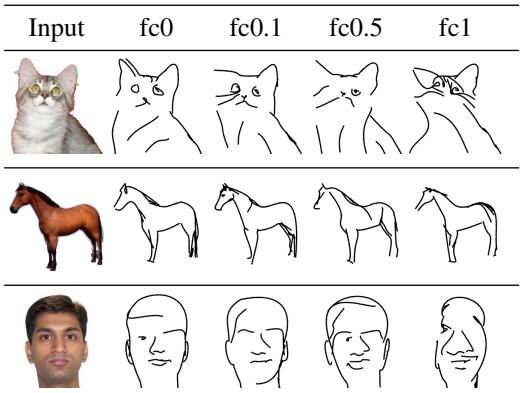

Figure 33. Weights intervals of semantic loss, columns 2 - 5 shows the optimization result when $L_{semantic}$ is weighted by 0, 0.1, 0.5, and 1 correspondingly.

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.

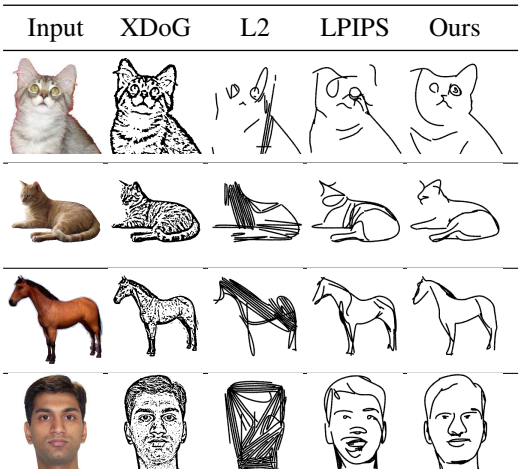

Figure 34. Replacing our CLIP-based geometric loss with LPIPS, and L2. The animal sketches are drawn using 16 strokes and the face sketch is drawn using 32 strokes.

F. Quantitative Comparison

Figure 37 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 35, 36 show sample 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.

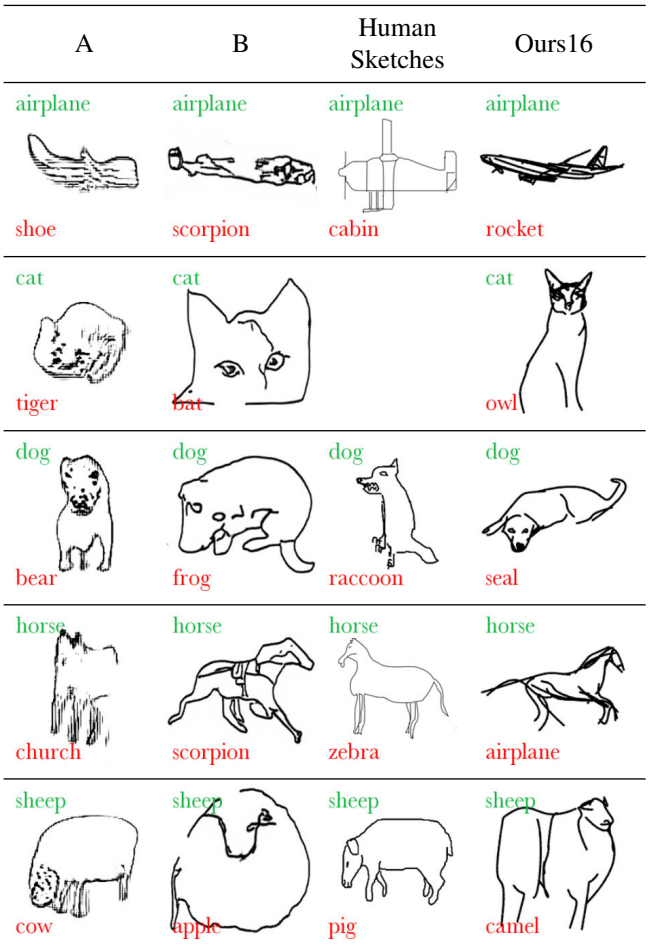

Figure 35. ResNet34 miss-classifications on the sketches generated by different methods (A) Kampelmühler and Pinz [21], (B) Li et al. [23]. The green text on each sketch shows to correct class and the red text shows the predicted class.

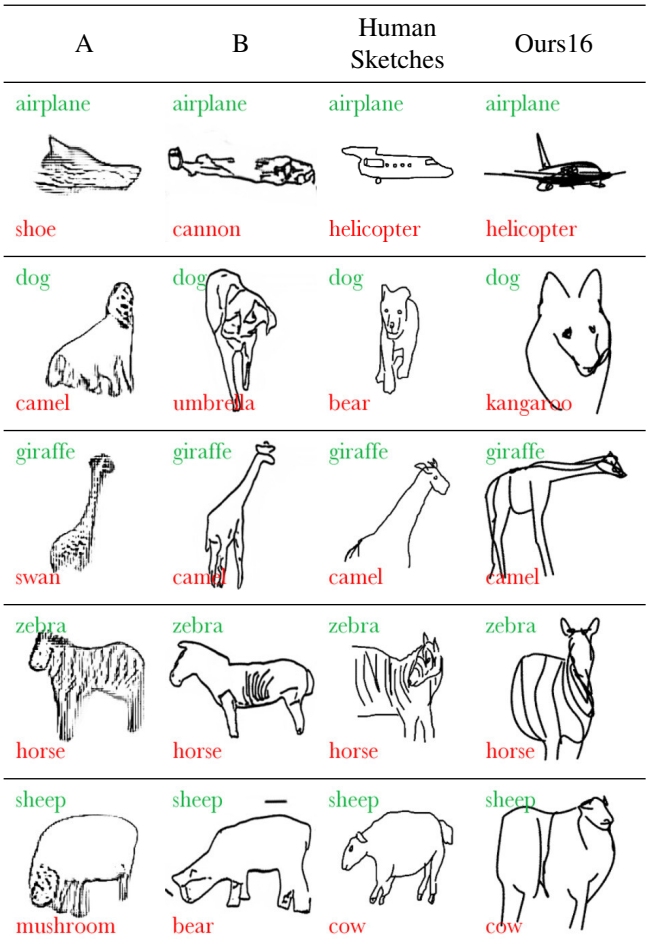

Figure 36. CLIP ViT-B/32 miss-classifications on the sketches generated by different methods (A) Kampelmühler and Pinz [21], (B) Li et al. [23]. The green text on each sketch shows to correct class and the red text show the predicted class.

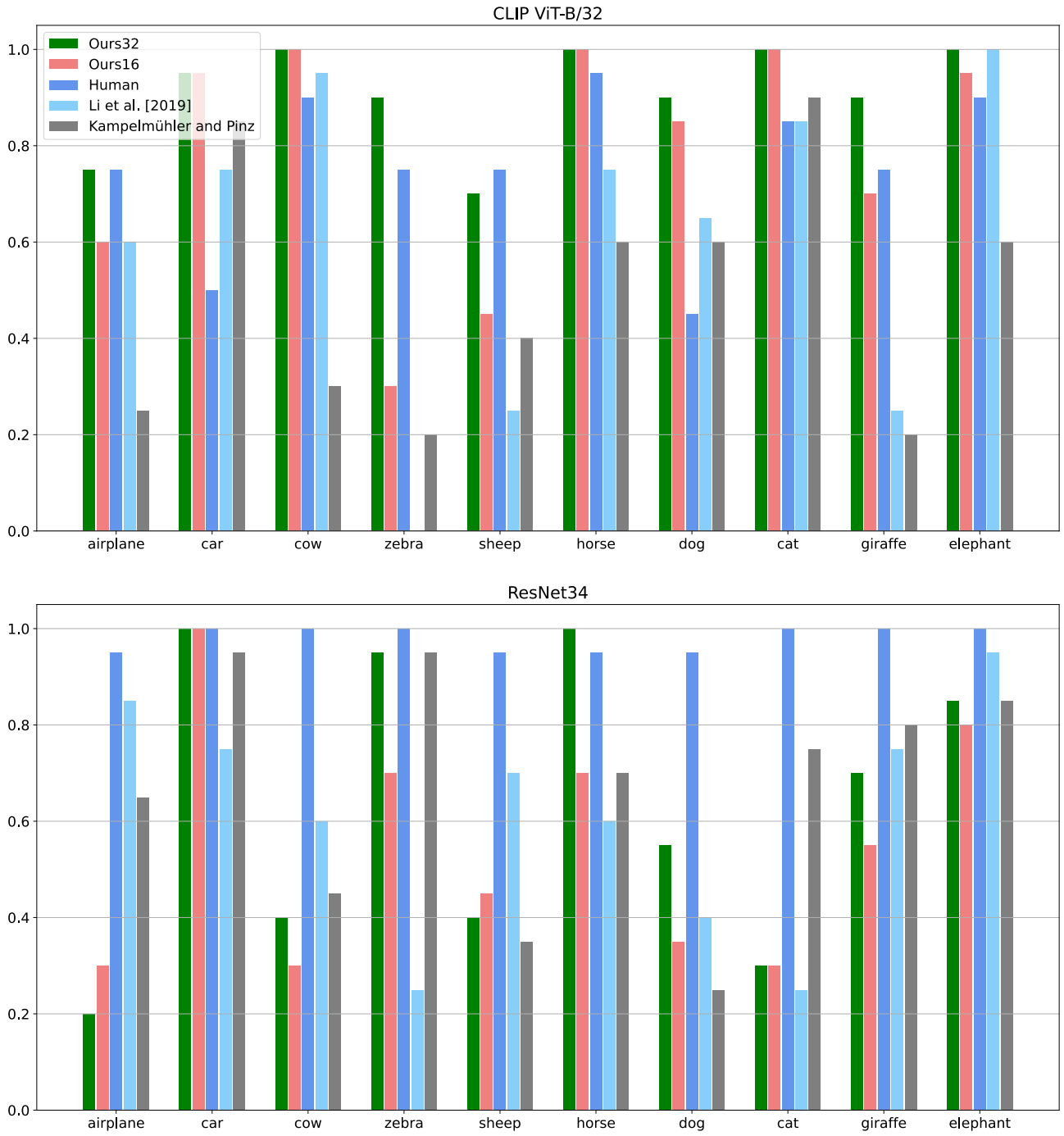

Figure 37. Per-class classification accuracy of the two classifiers we used in our quantitative analysis. The figure on top shows the results for CLIP ViT-B/32 and the figure below shows the results for ResNet34 classifier. Each color represent sketches by the corresponding sketch synthesis model.

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.

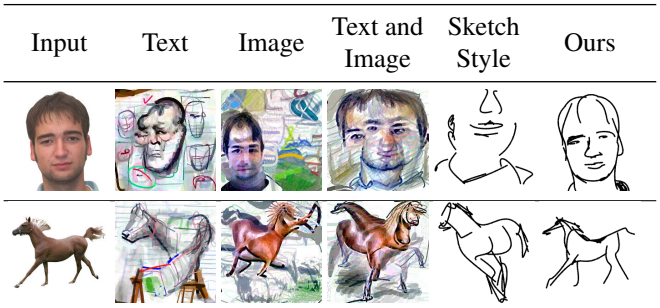

Figure 38. Comparison to CLIPDraw [11].

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 Figures 45, 46 we provide additional comparison on portrait-style human faces from the NPR benchmark [36].

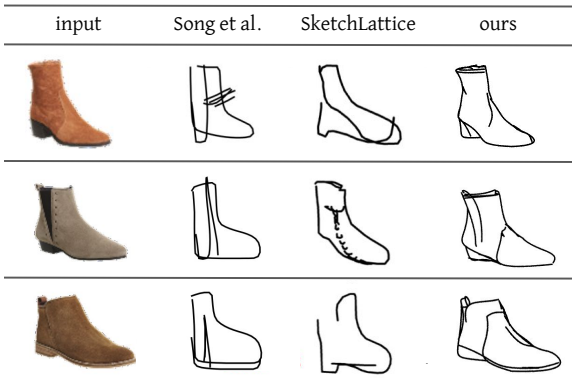

Figure 39. Comparison to Song et al. [38] and SketchLattice [32]

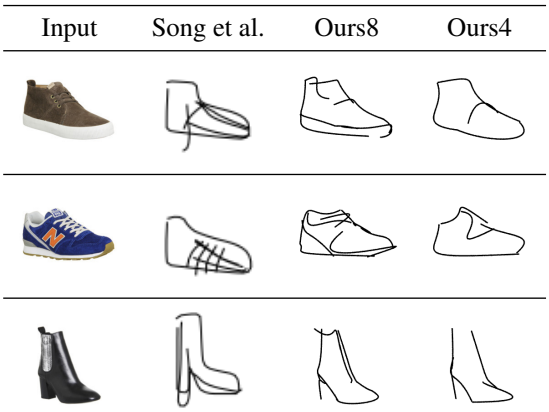

Figure 40. Comparison to Song et al.

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].

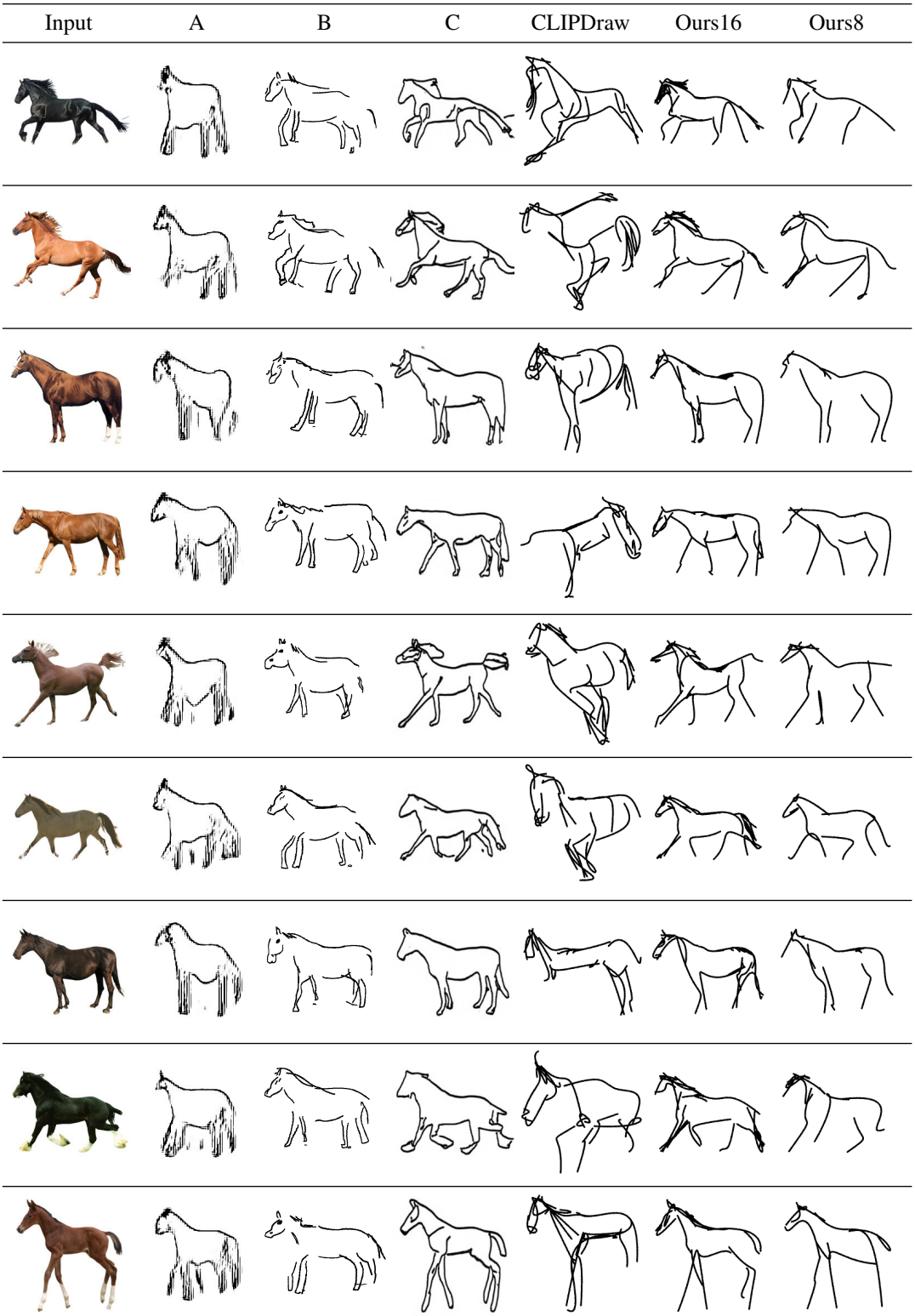

Figure 42. Comparison to competitor methods on images from the ”horse” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

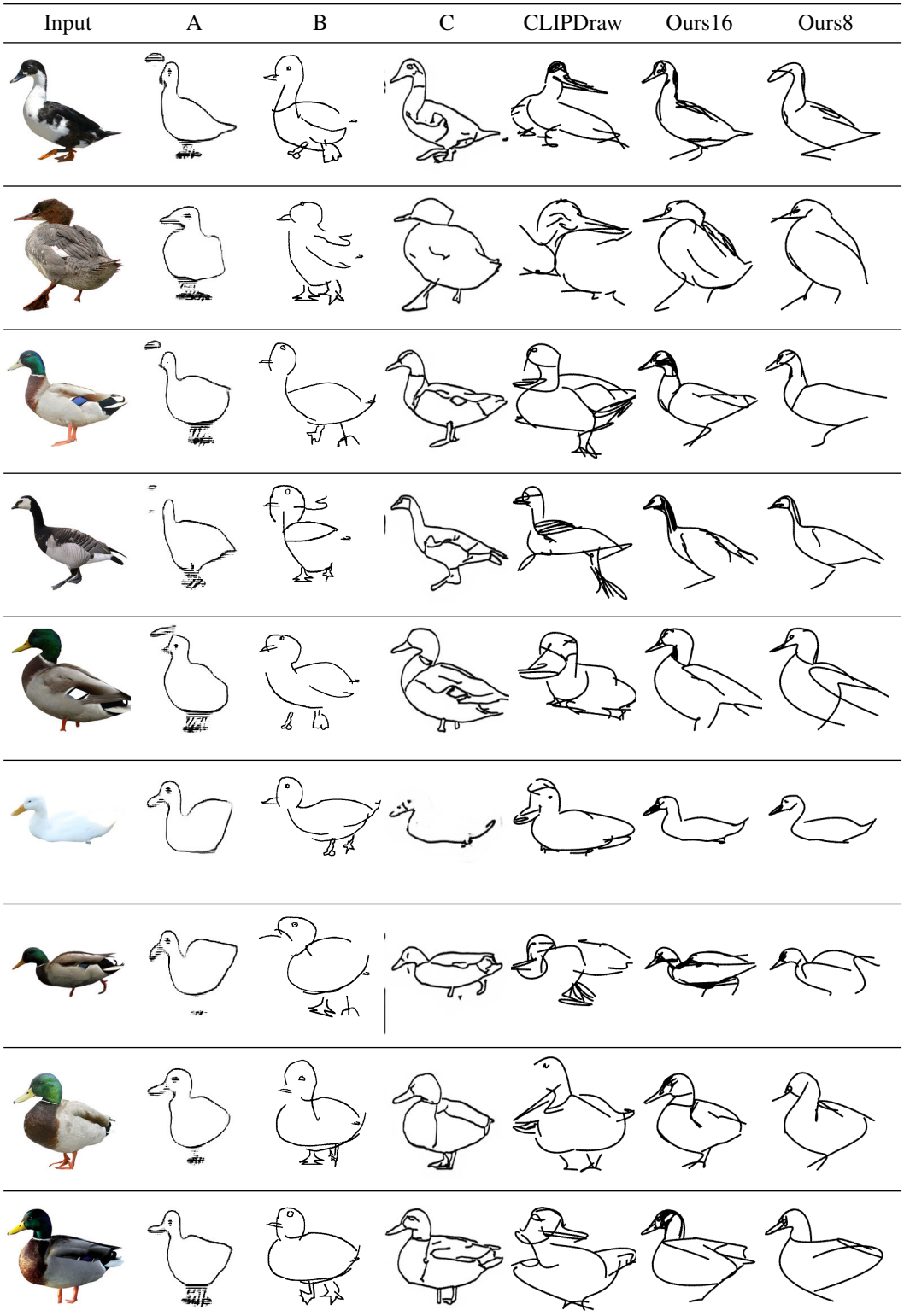

Figure 43. Comparison to competitor methods on images from the ”duck” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

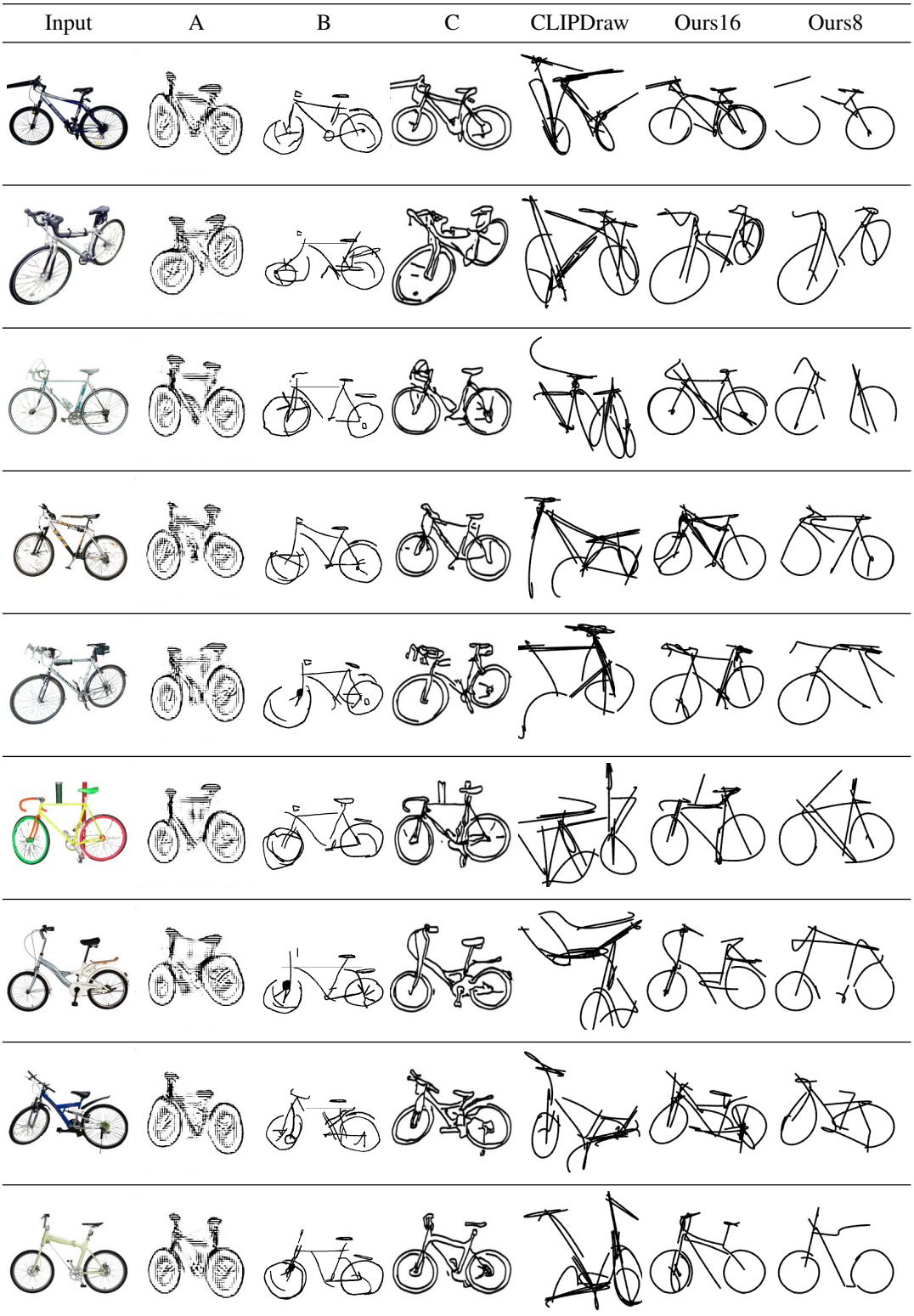

Figure 44. Comparison to competitor methods on images from the ”bicycle” class. Left to right : (A) Kampelmühler and Pinz [21], (B) Li et al. [25], (C) Li et al. [23], and CLIPDraw [11].

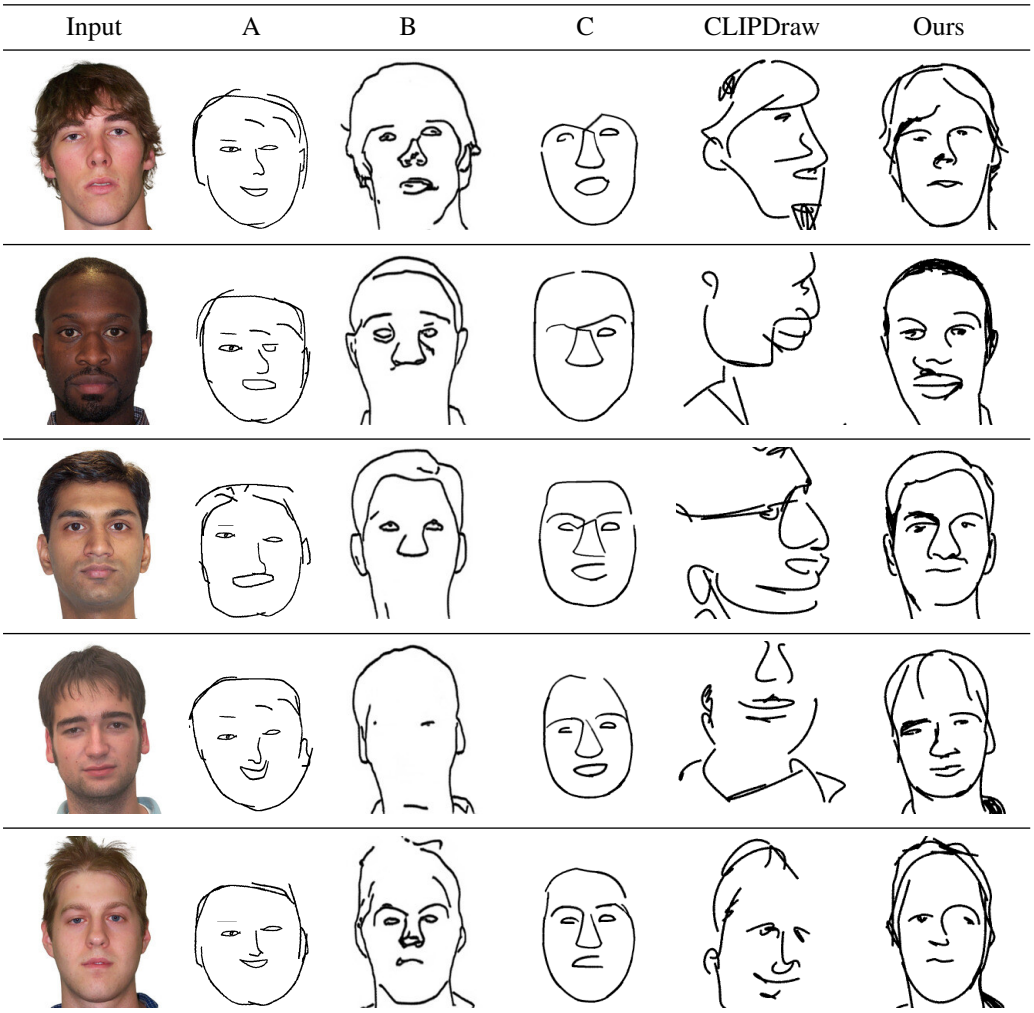

Figure 45. Comparison to competitor methods on human faces. Left to right: (A) Li et al. [25], (B) Li et al. [23], (C) Mo et al. [29], and CLIPDraw [11].

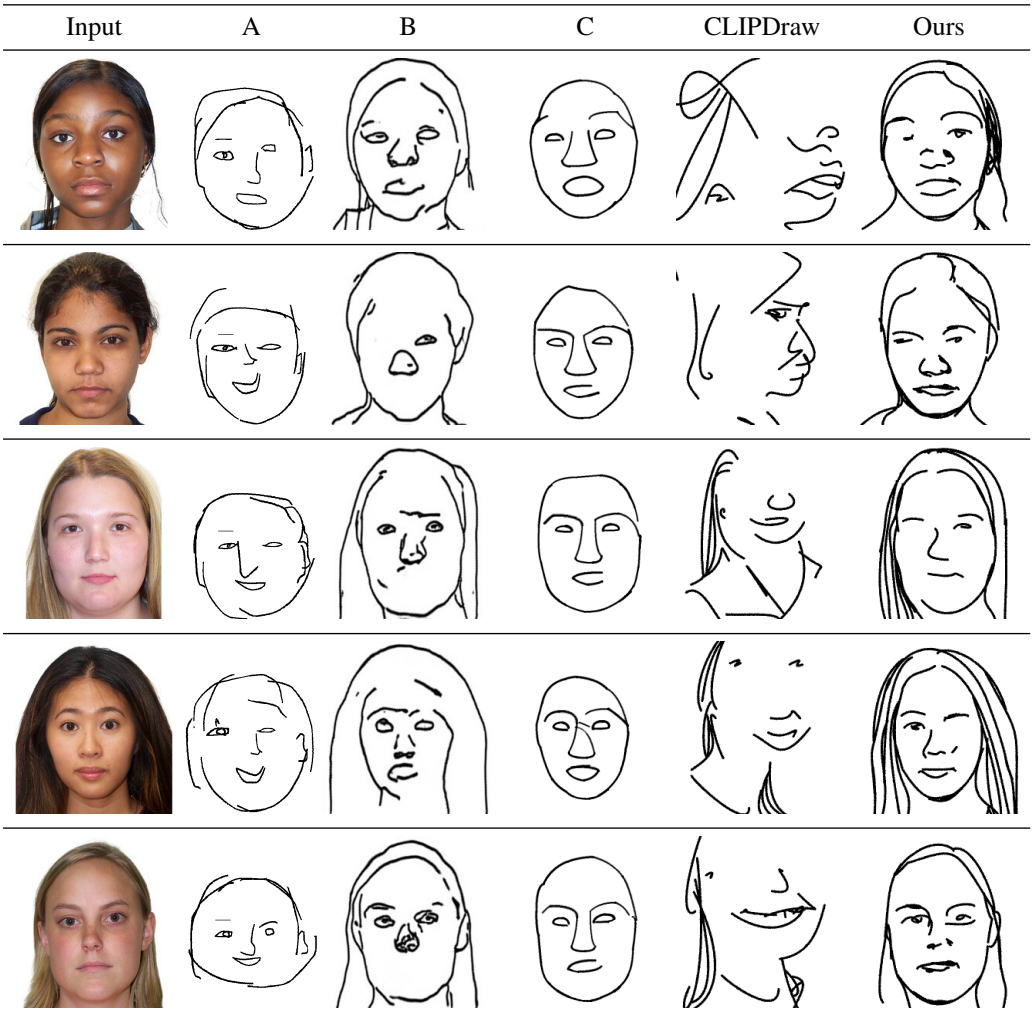

Figure 46. Comparison to competitor methods on human faces. Left to right: (A) Li et al. [25], (B) Li et al. [23], (C) Mo et al. [29], and CLIPDraw [11].

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 47, 48, and 49 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 Figures 50, 51 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.

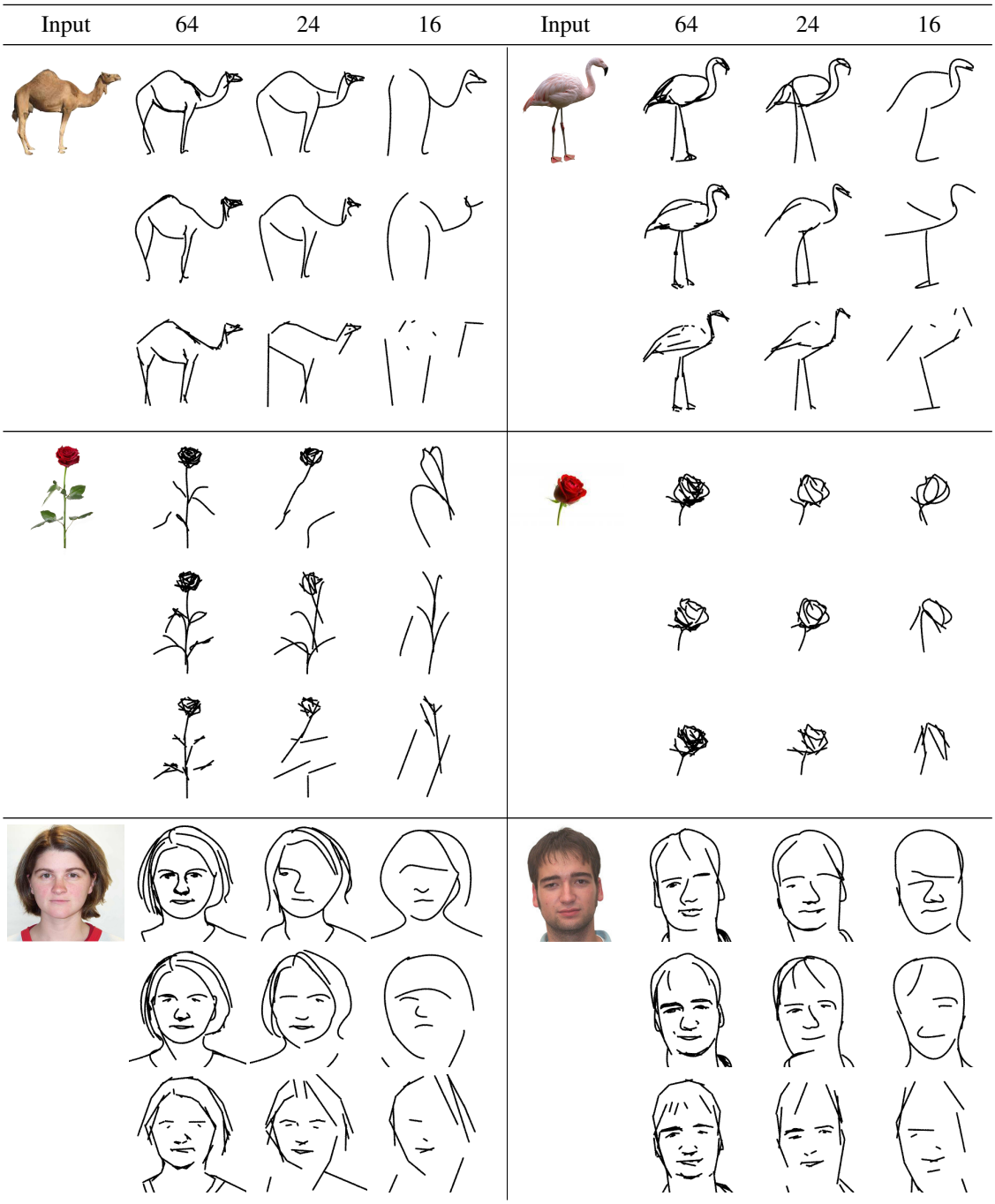

Figure 47. Sketches produced by our method with different styles and levels of abstraction. The columns indicate the total number of control points used to generate the sketch - 64, 24, and 16. The rows indicate the number of control points used for each stroke, which defines the style - 4, 3, and 2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

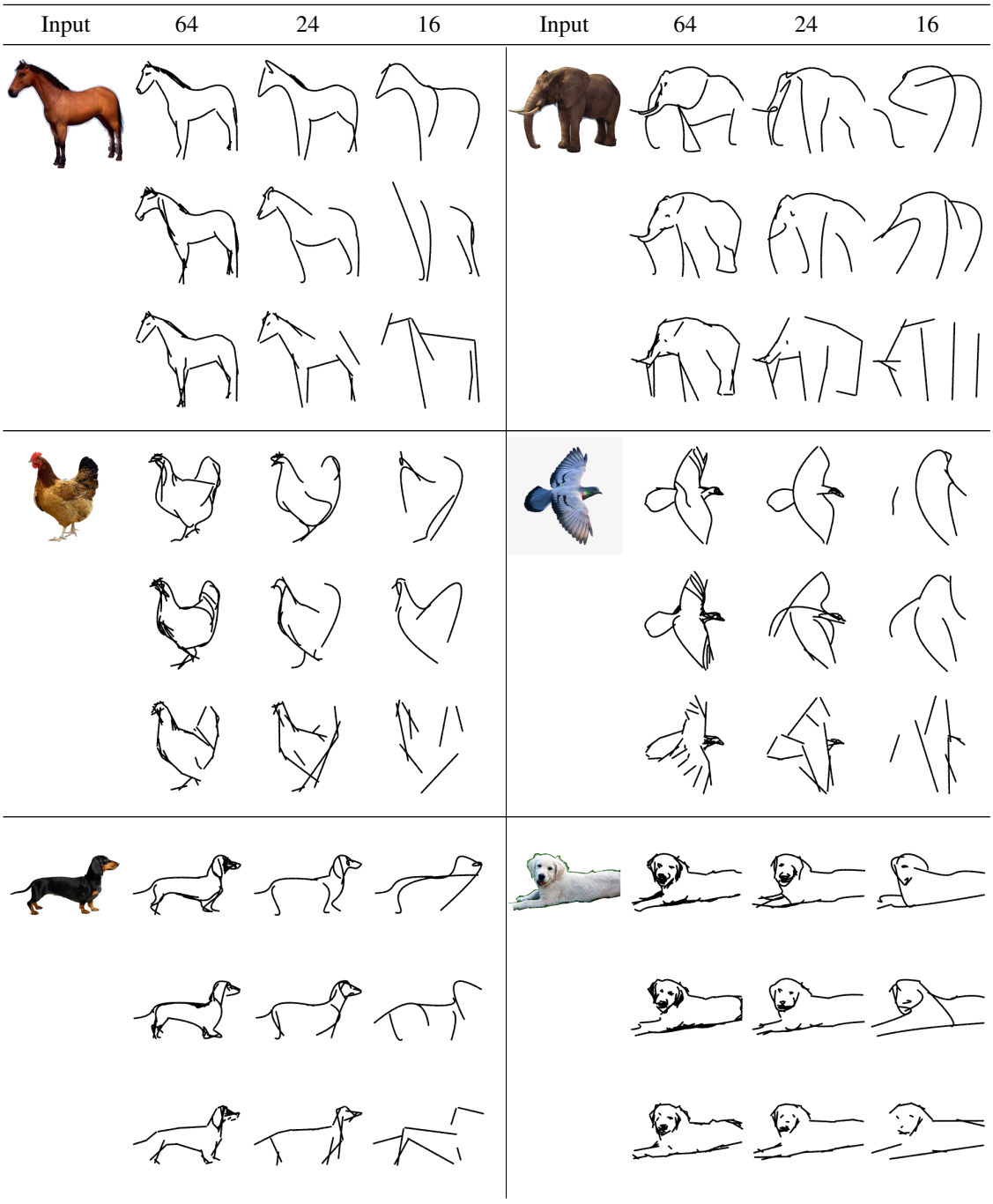

Figure 49. Sketches produced by our method with different styles and levels of abstraction. The columns indicate the total number of control points used to generate the sketch - 64, 24, and 16. The rows indicate the number of control points used for each stroke, which defines the style - 4, 3, and 2 control points respectively.

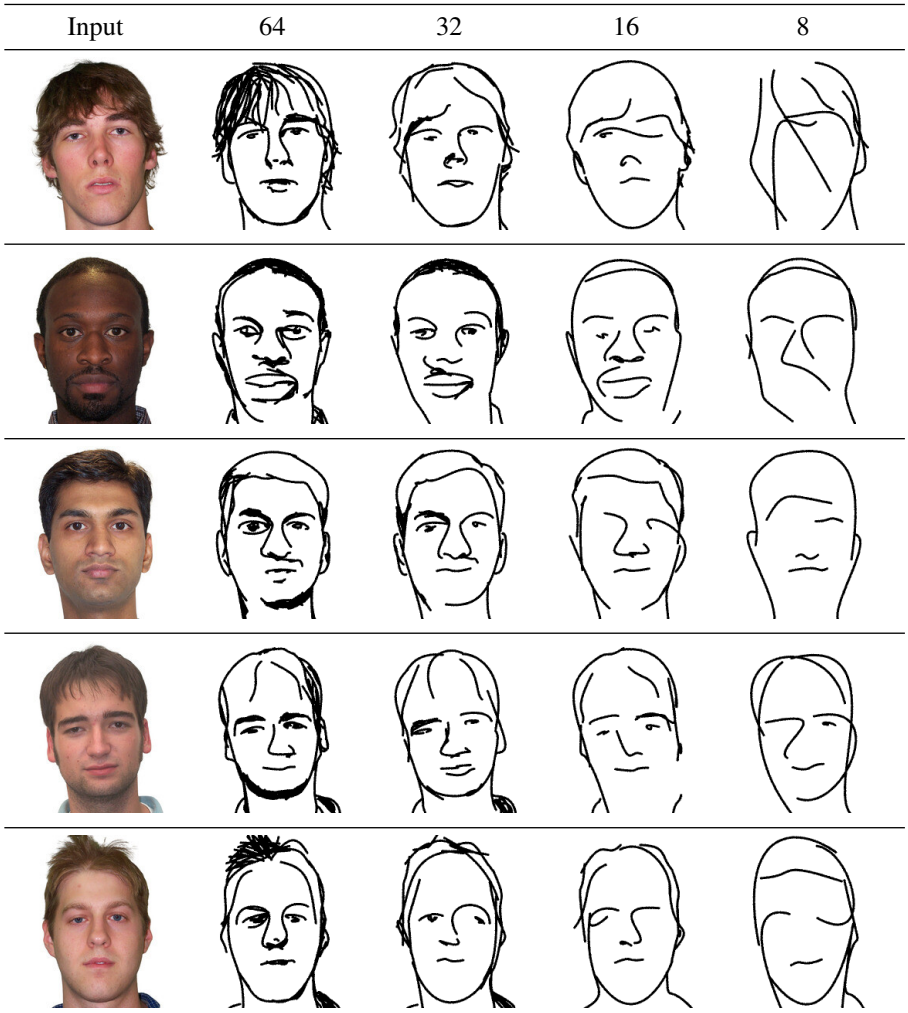

Figure 50. Sketches of human faces produced by our method with different levels of abstraction.

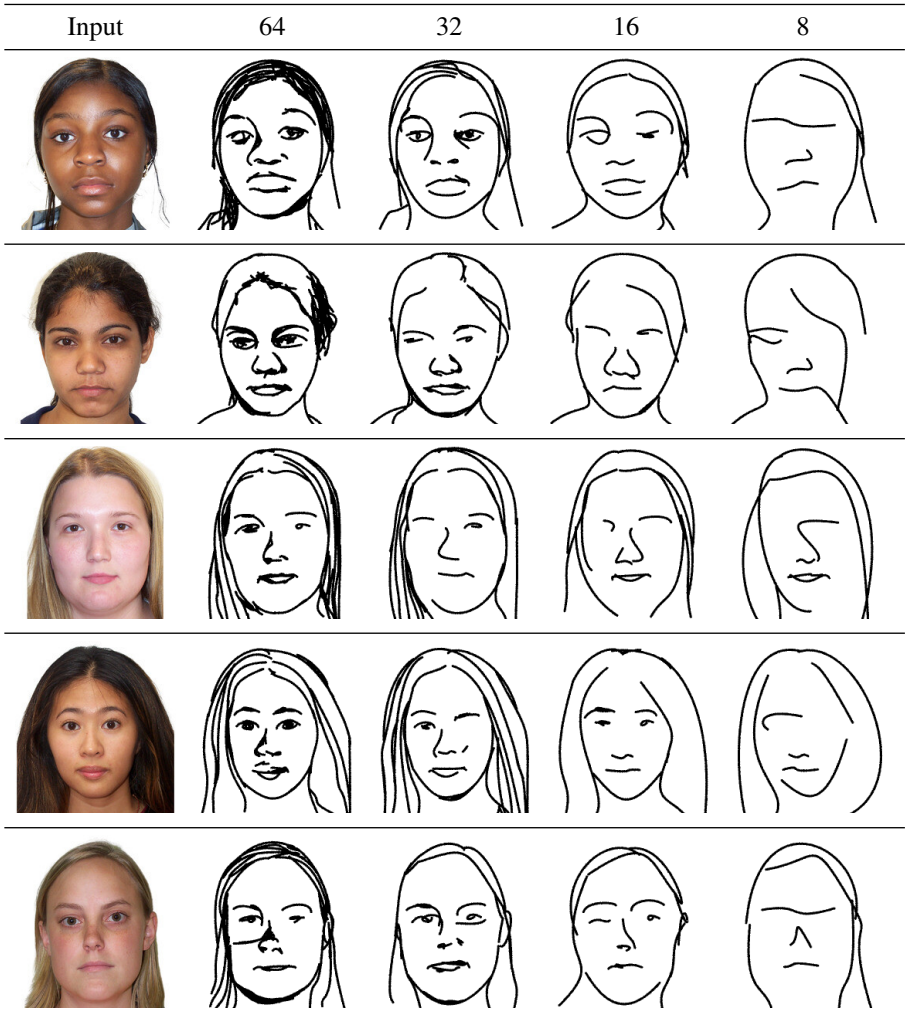

Figure 51. Sketches of human faces produced by our method with different levels of abstraction.

Figure 52. Sketching ”in the wild”: results of 100 random images of cats from SketchCOCO [13].

Figure 53. Sketching ”in the wild”: results of 100 random images of cats from SketchCOCO [13].