

**CONTROLLABLE CONTENT BASED IMAGE SYNTHESIS AND IMAGE
RETRIEVAL**

A Dissertation
Presented to
The Academic Faculty

By

Patsorn Sangkloy

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Interactive Computing
College of Computing

Georgia Institute of Technology

May 2022

© Patsorn Sangkloy 2022

CONTROLLABLE CONTENT BASED IMAGE SYNTHESIS AND IMAGE RETRIEVAL

Thesis committee:

Dr. James Hays (Advisor)
School of Interactive Computing
Georgia Institute of Technology

Dr. Mark Riedl
School of Interactive Computing
Georgia Institute of Technology

Dr. Devi Parikh
School of Interactive Computing
Georgia Institute of Technology

Dr. Subhansu Maji
College of Information and Computer Sci-
ences
University of Massachusetts, Amherst

Dr. Diyi Yang
School of Interactive Computing
Georgia Institute of Technology

Date approved: February 10, 2022

Drawing is not what one sees but what one can make others see

Edgar Degas

ACKNOWLEDGMENTS

This thesis is only possible because of many people who were there for me when I needed support. I am grateful to everyone who played a part in my PhD journey. First and foremost, I would like to thank my Phd supervisor James Hays, who provided not only expert guidance on technical matters, but also encouragements throughout my journey. James has always been kind, patient, and understanding.

I thank Cynthia Jingwan Lu for being a great host for my internship at Adobe Research, and for being my inspiration. I thank all my collaborators for all the cool research we did together. I thank my labmates – Amit, Cusuh, John, Sean, Samarth, Nam, Ben, and Wenqi – for all the fun research discussion and exciting game night. I thank my family members and friends for providing me strong emotional support. Special thanks to Witta for all the invaluable support. I am grateful for Royal Thai Government Scholarship for partially funding my study.

TABLE OF CONTENTS

Acknowledgments	iv
List of Tables	viii
List of Figures	ix
Chapter 1: Introduction	2
Chapter 2: Content Based Image Retrieval	4
2.1 The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies	4
2.1.1 Related Work	7
2.1.2 Creating the Sketchy Database	9
2.1.3 Learning a Cross-Domain Mapping	15
2.1.4 Quantitative Evaluation	22
2.1.5 Visualization: Average Objects	26
2.1.6 Limitations and Future Opportunities	27
2.2 A Sketch Is Worth a Thousand Words: Image Retrieval with Text and Sketch	31
2.3 Related work	34
2.4 Proposal: TASK-former	37
2.4.1 Training pipeline	38

2.4.2	Objective function	39
2.4.3	Sketch generation and data augmentation	40
2.4.4	Data collection: sketching from memory	41
2.5	Results and discussion	42
2.5.1	Ablation study	43
2.5.2	The robustness of sketch and text	45
2.5.3	On sketch complexity	47
2.6	Limitations and future work	48
Chapter 3: Content Based Image Synthesis		50
3.1	Scribbler: Controlling Deep Image Synthesis with Sketch and Color	50
3.1.1	Related Work	52
3.1.2	Overview of Scribbler	56
3.1.3	Sketch-Based Image Synthesis	60
3.1.4	User-Guided Colorization	66
3.1.5	Applications	69
3.1.6	Network Training Details	70
3.1.7	Conclusion and Future Work	72
3.2	TextureGAN: Controlling Deep Image Synthesis with Texture Patches	73
3.2.1	Related Work	75
3.2.2	TextureGAN	78
3.2.3	Training Setup	86
3.2.4	Results and Discussions	88

3.2.5 Conclusion	92
Appendices	93
Appendix A: The Sketchy Database	94
Appendix B: TASK-former	114
Appendix C: Scribbler	123
Appendix D: TextureGAN	126
References	134

LIST OF TABLES

2.1	Sketch retrieval benchmark of Li et al. [22]	22
2.2	Recall on COCO dataset	43
2.3	results from our ablation study	45
A.1	Parameter details for training Triplet/Siamese network	95
A.2	Comparison between different output dimension	95
B.1	Recall based on text completeness	115
D.1	Architecture of our generator network	126
D.2	Architecture of our global discriminator network	127
D.3	Architecture of our texture discriminator network	127
D.4	Architecture of each residual block	127

LIST OF FIGURES

2.1	Samples of photo–sketch pairs from the Sketchy database.	4
2.2	A spectrum of “sketchability”	10
2.3	Sketch collection interface	11
2.4	Stroke length vs time	14
2.5	Overview of our pipeline for cross-domain embedding.	16
2.6	t-SNE visualization of the Sketchy database test set	20
2.7	Average recall at $K = 1$	23
2.8	Evaluation on our test set	25
2.9	Align and Average retrieval results	28
2.10	Qualitative retrieval results	29
2.11	Overview of our model and the training pipeline	35
2.12	Example retrieved images from TASK-former	46
2.13	Retrieval performance when using with partial query	47
2.14	sketch completeness	48
3.1	A user can sketch and scribble colors to control deep image synthesis.	50
3.2	Network Architecture	57
3.3	Results comparison	60

3.4	Sketch generation examples	61
3.5	Sketch-based photo synthesis on hand-drawn sketches	62
3.6	Interesting network behavior	63
3.7	Guided sketch colorization results	65
3.8	Guided image colorization	68
3.9	Interactive image generation and editing	69
3.10	Visual search application	71
3.11	TextureGAN teaser	73
3.12	TextureGAN pipeline	80
3.13	The effect of losses	88
3.14	Effect of Proposed <i>local</i> losses	89
3.15	Results for shoes and handbags	90
3.16	Applying multiple texture patches on the sketch	91
3.17	Results on human-drawn sketches.	91
A.1	Benchmark plot with additional models and baselines	95
A.2	Per class Recall at $K = 1$	96
A.3	Comparison between network trained on COCO vs Sketchy	96
A.4	Contribution of different pre-training schemes.	97
A.5	Beyond SBIR	98
A.6	Additional SBIR results	99
A.7	Additional SBIR results	100
A.8	Additional SBIR results	101

A.9	Additional SBIR results	102
A.10	Additional SBIR results	103
A.11	Additional SBIR results	104
A.12	Additional IBIR results	105
A.13	Additional IBIR results	106
A.14	Additional IBIR results	107
A.15	Additional IBIR results	108
A.16	Additional SBSR results	109
A.17	Additional SBSR results	110
A.18	Additional SBSR results	111
A.19	Additional IBSR results	112
A.20	Additional IBSR results	113
B.1	Incomplete sketch	116
B.2	Retrieved images from our model	117
B.3	Examples of sketch captions generated from our model	118
B.4	Examples of synthetically generated sketches of [96].	119
B.5	GUI of our data collection process	121
B.6	Examples of the hand-drawn sketches in our dataset	122
C.1	Additional results on held out bedroom sketches	123
C.2	Additional results on held out face sketches	124
C.3	Additional results on held out car sketches	125
C.4	Additional face colorization results	125

D.1	Additional results on held out handbag sketches	128
D.2	Additional results on held out shoes sketches	129
D.3	Additional results on held out clothes sketches	130
D.4	Comparison of results before and after fine-tuning on an external texture database.	131
D.5	Failure cases	132
D.6	The effect of varying input patch sizes during testing.	132
D.7	Comparison RGB versus Lab inputs	133

SUMMARY

In this thesis, we address the problem of returning target images that match user queries in image retrieval and image synthesis. We investigate line drawing sketch as the main query, and explore several additional signals from the users that can help clarify the type of images they are looking for. These additional queries may be expressed in one of the following two convenient forms:

1. visual content (sketch, scribble, texture patch);
2. language content.

For image retrieval, we first look at the problem of sketch based image retrieval. We construct cross-domain networks that embed a user query and a target image into a shared feature space. We collected Sketchy Database; a large-scale dataset of matching sketch and image pairs that can be used as training data. The dataset has been made publicly available, and has become one of the few standard benchmarks for sketch-based image retrieval.

To incorporate both sketch and language content as queries, we propose a late-fusion dual-encoder approach, similar to CLIP; a recent successful work on vision and language representation learning. We also collected the dataset of 5,000 hand drawn sketch, which can be combined with existing COCO caption annotation to evaluate the task of image retrieval with sketch and language.

For image synthesis, we present a general framework that allows users to interactively control the generated images based on specification of visual features (e.g., shape, color, texture).

CHAPTER 1

INTRODUCTION

Drawing has been one of the oldest forms of communication in human history. It allows us to communicate and express our interpretation of the world to other people, all without the barrier of language. In this thesis, we explore the role of sketch, along with additional signals from the user (scribbler, texture patch, text) as user queries. Our goal is to return the target image(s) that match user queries. Two related but subtly different tasks are studied:

1. **Content Based Image Retrieval** (where target images are retrieved from a database)
2. **Content Based Image Synthesis** (where target images are generated).

In this thesis, we are interested in the following two convenient forms of user queries:

1. **Visual content** (where a query can be expressed as a simple line drawing sketch, an image patch, or a color scribble)
2. **Language content** (where a query can be expressed as a textual description of desired target images)

While these forms of querying are user-oriented, designing a model which can accurately capture users' intents is challenging for a number of reasons. For instance, querying with a hand drawn sketch often involves simplification, exaggeration, and iconic representation that can make it non-trivial to measure similarity across domains of sketch and image. The problem is more pronounced for the language content due to the inherent difference in natural language and image domains.

In this thesis, we explore both forms of querying (visual content and language content) for image retrieval and various type of visual content for image synthesis. A framework for cross-domain measurement of query-target (e.g., text-image, sketch-image) similarity

is proposed for each problem setting. For image retrieval, we construct cross-domains networks that embed a user query and a target image into a common feature space. We also collected a large scale dataset of matching sketch and image pairs that can be used as training data. The dataset has been made publicly available, and has become one of the few standard benchmarks for sketch-based image retrieval. For image synthesis, we present a general framework that allows users to interactively control the generated images based on specification of visual features (e.g., shape, color, texture).

This thesis contains content from published papers [1], [2], and [3].

CHAPTER 2

CONTENT BASED IMAGE RETRIEVAL

Given a set of image I_1, \dots, I_n , our goal is to correctly retrieve the target image I_t based on user queries. In this chapter, we first explore sketch based image retrieval in Section 2.1. Then we incorporate sketch and text for image retrieval in Section 2.2.

2.1 The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies

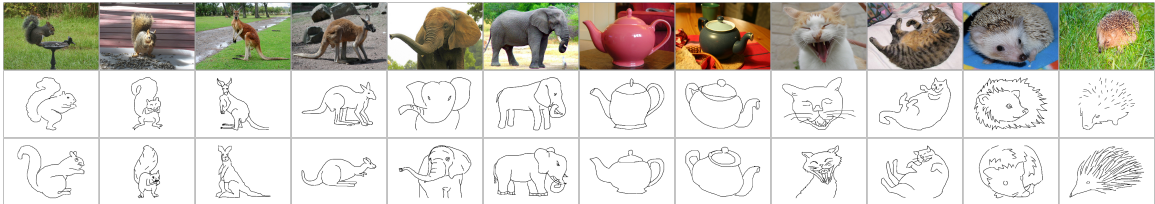


Figure 2.1: Samples of photo-sketch pairs from the Sketchy database.

The goal of Sketch-based image retrieval is to allow non-artist users to draw visual content (usually objects) and then find matching examples in an image collection. Sketch-based image retrieval is an alternative or a complement to widely used language-based image querying (e.g. Google image search). In the computer graphics community, sketch-based image retrieval has been used to drive image synthesis approaches such as Sketch2Photo [4] and PhotoSketcher [5]. Sketch-based image retrieval has been studied for nearly 25 years [6], but is especially relevant as touch and pen-based devices have proliferated over the last few years.¹

Sketch-based image retrieval is challenging because it requires comparison across two domains (sketches and photos) that are individually difficult to understand and have distinct appearance. Typical approaches propose a hand-designed feature, usually focused

¹Touch-enabled smartphones and tablets outnumber desktop and portable PCs by 4 to 1 as of 2014, while the market was roughly equal in 2010 [7].

on edges or gradients, which is somewhat invariant across domains. But as we will show (Section 2.1.4) there is a lot of room for improvement over such features.

The primary reason sketch-to-image comparison is hard is that humans are not faithful artists. We tend to draw only salient object structures and we tend to draw them poorly. Shapes and scales are distorted. Object parts are caricatured (big ears on an elephant), anthropomorphized (smiling mouth on a spider), or simplified (stick-figure limbs). Nonetheless these sketches are usually understandable to other humans.

It is appealing to try and *learn* the correspondence between human sketches and photographic objects since the task seems to require high-level understanding. Recent progress in deep convolutional networks has enabled better understanding of sketches and object images, individually. The 250 categories of sketches collected by Eitz et al. [8] and the 1000 categories in the ImageNet challenge [9] can be recognized with roughly human-level accuracy. This suggests a “retrieval by categorization” approach in which relevant images are returned if they appear to be the same category as a query sketch. In fact, this strategy is consistent with common sketch-based image retrieval benchmarks [10] where retrieval results are correct as long as they are the same category.

However, we argue that sketch-based image retrieval needs to go beyond category recognition. If a user wants objects of a particular category, then language already gives them an efficient way to find huge amounts of relevant imagery. The appeal of sketching is that it allows us to specify *fine-grained* aspects of an object – pose, parts, sub-type (e.g. office chair versus dining chair) – and many of these fine-grained attributes are clumsy to specify with language (e.g. “office chair with no arms viewed from slightly above and to the side” or “castle with two pointy towers and a crenelated battlement joining them”). Even if such text descriptions did encapsulate user intent, fine-grained text-based image retrieval is an open problem. This need to retrieve semantically and structurally appropriate objects leads Sketch2Photo [4] to synthesize scenes using a combination of language and sketch – language typically specifies the category and sketch specifies the shape. We

believe sketches alone can be sufficient.

Our goal in this paper is to learn a cross-domain representation for sketches and photos which is reliable not just at retrieving objects of the correct category, but also objects with *fine-grained* similarity to the sketch. We utilize deep learning techniques which have led to dramatic improvements in various recognition tasks, but which require large amounts of supervised training data. In particular, for our cross-domain deep learning we will need thousands of pairs of matching sketches and photos. Because no such database exists we ask crowd workers to draw thousands of photographic objects spanning 125 categories. We do not allow participants to trace photos, but instead reveal and then hide photos so that they must sketch from memory similar to the way in which a user of a sketch-based image retrieval system would be drawing based on some mental image of a desired object. The cross-domain representation we learn proves effective for sketch-based image retrieval and we believe the Sketchy database supports many interesting investigations into sketch and image understanding.

We make the following contributions:

- We develop a crowd data collection technique to collect representative (often bad!) object sketches prompted by but not traced from particular photos. The Sketchy database contains 75,471 sketches of 12,500 objects spanning 125 categories.²
- We demonstrate the first deep learning approach to sketch-based image retrieval. We learn a common feature space for sketches and photos which enables not only sketch-based image retrieval but also image-based sketch retrieval or sketch-to-sketch and photo-to-photo comparisons.
- We show that our learned representation leads to significantly better fine-grained sketch retrieval than existing methods and even outperforms a hypothetical ideal “re-

²the database can be downloaded from <http://sketchy.eye.gatech.edu/>

trieval by categorization” method.

In the next section we discuss related work. Section 2.1.2 describes the creation of the Sketchy database. Section 2.1.3 describes our deep learning experiments. Section 2.1.4 evaluates our learned representation for the task of sketch-based image retrieval. Section 2.1.5 shows sketch-constrained average images inspired by AverageExplorer [11]. Section 2.1.6 discusses limitations and possible future applications of the Sketchy database.

2.1.1 Related Work

Sketch-based image retrieval.

Numerous representations have been proposed to retrieve images from sketch queries [6, 12, 13] or color drawings [14]. See Smeulders et al. [15] for a survey of classical approaches. More recent methods propose increasingly sophisticated feature representations, often inspired by object recognition approaches in the computer vision community [16, 17, 18, 19].

Benchmarks for sketch-based image retrieval are fairly small, e.g. 43 sketch-photo pairs [20] or 31 sketches each with 40 photos ranked by similarity [21]. The Flickr15k dataset [10] contains 330 sketches and 14,660 photos spanning 33 categories, but there are no fine-grained associations across domain – the benchmark is equivalent to categorization. These benchmarks have been useful to the field but are not large enough to *learn* from.

The tendency to evaluate sketch-based image retrieval as category retrieval was noted by Li et al. [22] and they propose a “fine-grained” retrieval method based on deformable part models [23] trained on the 14 overlapping categories of the Pascal VOC [24] and Eitz 2012 datasets. At training time there is still no instance level sketch-photo association, but their test set scores sketch-photo pairs in terms of viewpoint, zoom, pose, and shape similarity. We share the motivation for “fine-grained” retrieval and we evaluate our representation learned from the Sketchy database on their benchmark (Table 1).

The most similar related work is the concurrent “Sketch Me that Shoe” [25] which also

collects a database of sketch-photo pairs and uses deep learning to learn a shared embedding. Their dataset is smaller – 1,432 sketch-photo pairs of shoes and chairs – but more densely annotated with 32,000 triplet rankings. Because the dataset is smaller, edge detection is used to render photos like sketches instead of *learning* the entire cross-domain transformation as we do. The paper also proposes more aggressive, domain-specific “jittering” to amplify the value of each training sketch.

Sketch classification.

“How do Humans Sketch Objects?” [8] introduced a dataset of 20,000 sketches spanning 250 categories and demonstrated that bag-of-features representations built on gradient features could achieve reasonable classification accuracy (56%). Schneider and Tuytelaars [26] showed that Fisher vector encoding of local features significantly improved recognition accuracy (69%). Sketch-a-Net [27] showed that deep features can surpass human recognition accuracy – 75% compared to the 73% accuracy from crowd workers in Eitz 2012. Su et al. [28] focus on 3D model classification with deep features but show that their network can be adapted to match the state of the art in sketch recognition or sketch-based 3D model retrieval. We use the Eitz 2012 dataset to pre-train the sketch half of our cross-domain embedding approach.

Deep learning for cross-domain embedding.

Our technical approach is similar to recent cross-domain embedding methods that train deep networks to learn a common feature space for Sketches and 3D models [29], ground and aerial photographs [30], Iconic and in-the-wild product photos [31], and Images and 3D models [32]. We experiment with the tools used in these works such as Siamese networks trained with contrastive loss [33, 34], but find that triplet or ranking loss [35] performs better. Our best performing method combines the Triplet loss with a classification loss similar to Bell and Bala’s [31] combination of Siamese and classification losses.

Sketch-based image retrieval for image synthesis.

We are motivated by Sketch2Photo [4] and PhotoSketcher [5], which synthesize scenes by compositing objects and backgrounds retrieved based on user sketches. PoseShop [36], follow on work from Sketch2Photo, addresses the fact that bad user sketches cannot be matched reliably by letting users pose a 2D skeleton and then generating a contour query from a mesh attached to the 2D skeleton. Improved sketch-based object retrieval would make these approaches simpler or more reliable.

Sketch-photo pairs for sketch synthesis.

Berger et al. [37] and Limpaecher et al. [38] collect expert and amateur portrait drawings, respectively, to aid in the creation of new portraits. The 14,270 face portraits collected through a Facebook game by Limpaecher et al. [38] is one of the largest collections of sketches to date.

2.1.2 Creating the Sketchy Database

In this section we describe the creation of the Sketchy database, which spans 125 categories and consists of 12,500 unique photographs of objects and 75,471 human sketches of objects inspired by those photographs.

Category Selection

In order to choose the 125 categories in our data set, we use the same criteria defined in “How Do Humans Sketch Objects?” [8]: exhaustive, recognizable, and specific. That is, the categories should cover a large number of common objects and each category should have recognizable sketch representations. In addition, we add a “sketchability” criterion (Figure 2.2). Some object categories may be fairly easy to sketch from memory, but photographs of those objects tend to be too challenging to sketch or the resulting sketches may be too



Figure 2.2: A spectrum of “sketchability” for three ImageNet categories: *horse*, *apple*, and *rabbit*. This subjective ranking is determined by answering the question, “How easily could a novice artist capture the subject category and pose?” The most difficult photographs, such as those shown on the far right of the figure, are excluded from our data set.

uninformative due to the nature of photographs common to those objects. This concept is illustrated in Figure 2.2 and further discussed in the following sections.

To start, we consider all ImageNet [9] categories for which there are bounding box annotations, with preference given to categories contained within the Eitz 2012 sketch data set. We hope our data set will both complement and extend these data sets. Ultimately, 125 categories are included in our data set. Of these, 100 categories exist within the Eitz 2012 data set. Where appropriate, multiple, related ImageNet categories (e.g. specific dog breeds) are combined into a single category in order to add visual diversity and increase the number of “sketchable” photographs.

Photograph Selection

We cannot expect novice artists to draw photographs of horse eyes and crocodile teeth in meaningful ways, yet these extreme photographs are prevalent in Internet-scale image data sets. In order to select appropriate photographs for our data set, we first eliminate all photographs that do not have exactly one bounding box annotation. Next, we manually review the remaining photographs and eliminate those with 1) disturbing or inappropriate content, 2) poor or degraded image quality, 3) significant manipulation or watermarks, 4) incorrect category label, and/or 5) ambiguous content due to occlusion or object pose. Overall, we review a total of 69,495 photographs and deem 24,819 as “sketchable”. This

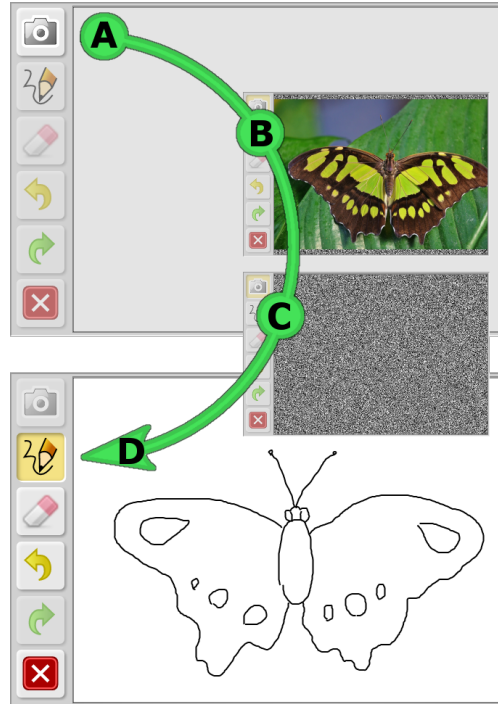


Figure 2.3: Sketch collection interface. A participant initially sees a blank canvas. (a) Pressing a button reveals (b) a photo for 2 seconds followed by (c) a noise mask for one second. (d) The participant then uses pencil, eraser, and undo tools to create their sketch.

process results in a median of 147 sketchable photographs per category. Categories with fewer than 100 sketchable photographs are not included in our data set. As part of this process, a volunteer annotates each remaining photograph with a subjective “sketchability” score, ranging from 1 (very easy to sketch) to 5 (very difficult to sketch). Each category’s 100 photographs are chosen at random from this pool with a targeted distribution of 40 very easy, 30 easy, 20 average, 10 hard, and 0 very hard photographs.

Sketch Collection

The creation of sketch-photo pairs is the most critical challenge of creating the Sketchy database. There are two broad strategies – prompt the creation of sketches from particular photos [20] or have people associate existing sketches to photos, e.g. by ranking a list of potentially matching photos [21]. We choose the first strategy because it is better able to create fine-grained, instance-level associations. With the second strategy, there may not

exist a photo that is particularly similar to a sketch.

However, the first strategy, photo-prompted sketch creation, is somewhat the inverse of motivating task – sketch-based image retrieval. Does a user drawing of a particular photo resemble a sketch retrieval query? For instance, a naive approach to sketch creation would be to have users trace over a particular photo. Such drawings would effectively be boundary annotations as in the Berkeley Segmentation Dataset [39]. If we thought faithful object boundaries were satisfactory “sketch” training data we could use existing segment annotations from datasets such as LabelMe/SUN [40] or MS COCO [41]. As Figure 1 shows, the sketches we collect are very different from such annotations. We also attempt to train on MS COCO boundaries with no success in Section 2.1.4.

The key to collecting “realistic” sketches is to prompt workers with a particular photo but then hide it so they must draw from memory. This strategy was used in Eitz et al. [20] to collect 43 sketches and also in Antol et al. [42] to collect “clipart” annotations which correspond to the most salient scene structures. The choice of which object structures people preserve (or hallucinate) when sketching a particular photo is interesting in itself, independent from the goal of sketch-based image retrieval, and could support research on human visual memory of objects [43, 44]. We use the fact that the Sketchy database implicitly encodes image salience to produce visualizations in Section 2.1.5.

Figure 2.3 shows our sketch collection interface. Each participant is provided with a randomly selected category name and a blank canvas on which to sketch. Upon pressing a specified button, the participant is shown a photograph containing an example of the selected category. The photograph is only visible for two seconds, but the participant can view it as many times as needed. However, each viewing clears the sketch canvas. In order to discourage rote boundary reproduction, a noise mask is displayed after the photograph is revealed and before the participant may begin sketching. Noise masking, often used in psychology experiments, aims to destroy low-level visual representations in visual working memory [45, 46]. In our case, this prevents participants from “tracing” an afterimage in

visual working memory and hopefully produces more diverse and realistic sketches.

A participant is instructed to 1) sketch the named subject object with a pose similar to that of the object in the photograph, 2) sketch only the subject object, and 3) avoid shading in regions. Since we have bounding box details for the object in the photograph, we are not concerned with sketch size nor location in the canvas as they can be aligned after the fact. The sketch canvas supports touch-enabled devices and provides a means to sketch, undo, redo, clear, and erase. Each sketch is stored as an SVG file. We extend the SVG format to include high resolution time details; not only do we record the stroke start and end times, but also fine-grained timing along each stroke. Figure 2.4 examines the drawing tendencies of our participants. Prior research has shown that stroke order and length are important in determining the relative importance of a given stroke [27]. If the speed at which a stroke is drawn indicates care and/or concentration, then speed, too, may be useful in determining the relative importance of a stroke. The vector representation of strokes allows us to re-render the drawings in various ways (different stroke widths, stroke width related to velocity, etc.) but we maintain the same fixed width stroke representation as used in the data gathering through all experiments. Varying the rendering style did not significantly influence the learning experiments in Section 2.1.3, but we believe there is potential to artificially increase the training set size through numerous stroke-based augmentations as in concurrent work [25].

Participants.

When it comes to sketching a photograph, there is no single correct answer. The artist's skill, motivation, subject familiarity, and input device can all influence the resultant sketch. In order to capture this diversity we collect five sketches per photograph each from a different participant. We use Amazon Mechanical Turk (AMT) to increase the number of potential participants and resultant sketch diversity. We use a qualification test to ensure each potential participant understands the sketching process. The qualification test asks

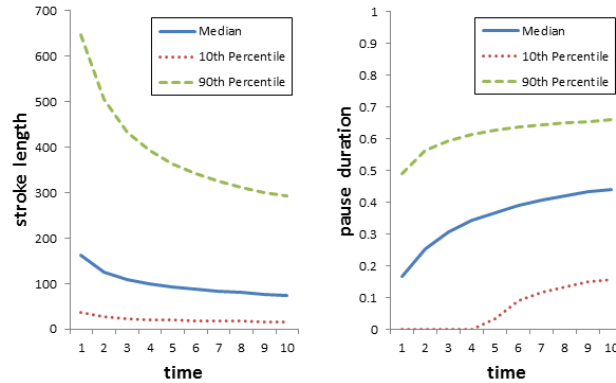


Figure 2.4: Left: as in Eitz 2012 we find that the participants follow a coarse-to-fine sketching strategy and draw shorter strokes over time. Right: Over their drawing time, participants spend increasingly more time deliberating between strokes as measured by the fraction of time spent idle. For both plots, the duration of each sketch session is normalized to the same time duration.

the candidate to sketch three different photographs, each carefully chosen to test the candidate’s understanding of one or more rules. We manually review these qualification results for compliance and allow those who pass to work on our sketch collection tasks. We receive 1,204 qualification test submissions, from which we identify 644 qualified individuals. Over the course of six months, these participants collectively spent 3,921 hours sketching for our data set.

Data Validation.

Even though we use a qualification test to screen the crowd workers, mistakes, misunderstandings, and abuses are inevitable. Instead of using the crowd to validate its own work, we opt to manually review all sketches. We use a custom software tool to display each photograph along with all its sketches and then tag each sketch in one of five ways: 1) correct, 2) contains environment details or shading, 3) incorrect pose or perspective, 4) ambiguous, or 5) erroneous. All of these sketches remain in the data set, even if deemed incorrect in some way. Inclusion is up to the judgment of the database user and is highly task dependent. The only truly incorrect sketches are those marked as erroneous. In order to increase the overall quality of the data set and bring the total number of sketches per

photograph closer to five, we collect additional sketches to replace those that are not tagged as correct. This results in a total of 75,471 sketches, with 64,560 correct, 6,249 ambiguous, 2,683 with an incorrect pose, 1,061 including environment details, and 918 erroneous.

Comparison to Eitz 2012.

Our participants spent a similar amount of time drawing sketches as is reported in Eitz 2012. The median sketch time is 85 seconds, with the 10th and 90th percentile at 41 and 281 seconds (compared to 86, 31, and 280 seconds, respectively). The median number of strokes per sketch is 14, compared with 13 in Eitz 2012.

While the stroke-level statistics are similar to the 20,000 sketches of Eitz 2012, we believe the distribution of sketches is quite different. Eitz 2012 prompted workers with an object category and nothing more. The resulting sketches are very iconic. For example 85% of buses are drawn directly from the side and the remainder from 45°. Not a single duck is drawn in flight. 80% of calculators are upright and viewed from the front. People chose easy-to-draw poses and viewpoints, which is partly why the dataset is fairly easy as a recognition benchmark. But the Sketchy database contains objects in a variety of poses, states, and viewpoints because workers were prompted with particular photographs. While the ImageNet photos are themselves somewhat iconic, they still lead to greater sketch diversity than free recall. Working from a particular photo also constrained the sketches to be somewhat less caricatured (fewer big teeth and ears on rabbits).

2.1.3 Learning a Cross-Domain Mapping

In this section we use the Sketchy database to learn a shared embedding for sketches and photos such that distances in the learned feature space are related to structural and semantic similarity between sketches and photos. We follow the trend of recent works which use deep convolutional networks (CNNs) to learn cross-domain embeddings [29, 30, 31, 32], but the details of our network architectures and training strategies vary considerably.

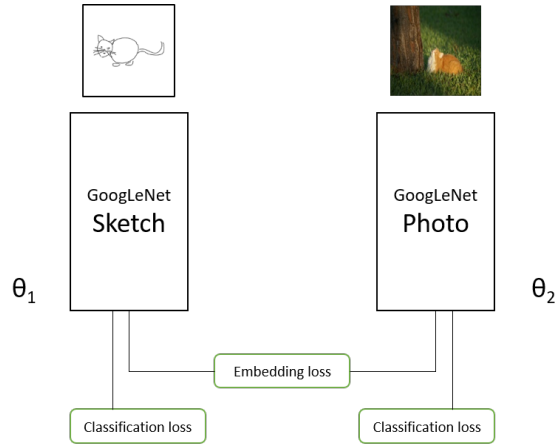


Figure 2.5: Our adaptation of the GoogLeNet architecture to cross-domain embedding. The embedding loss can be either contrastive loss (as used in a Siamese network) or triplet loss. For triplet loss, the photo branch would take two inputs, one for the matching photo and one for a dissimilar photo. The sketch and photo networks learn independent weights θ_1 and θ_2 .

Our deep networks which achieve the highest performance on our benchmarks required rather complex training regimes. There are two reasons for this: (1) While the Sketchy database is large relative to existing sketch databases, deep networks have tens of millions of free parameters to learn and thus demand large amounts of training data. We see benefits but also complications from pre-training on additional datasets. (2) We have two separate notions of similarity – instance-level similarity between a sketch and the exact photo that elicited it, and category-level similarity between a sketch and all photos in that category. Making use of both forms of supervision during training increases performance at the cost of added complexity.

We examine combinations of three different loss functions for training deep networks – Siamese loss, Triplet loss, and Classification loss. We first give an overview of these loss functions.

Siamese Network.

In a Siamese network [33, 34] a pair of convolutional networks each accept an input. Supervision is binary – either the input pair should be similar or dissimilar. Siamese net-

works use a “contrastive” loss function of the form $L = l * d(S, I+) + (1 - l) * \max(0, m - d(S, I-))$ where S is an embedded sketch, $I+$ is an embedded image of the same object instance, $I-$ is an embedded image of a different object instance, $d()$ is Euclidean distance, m is a margin, and $l \in \{0, 1\}$ is label with 1 for positive and 0 for negative pair. Dissimilar sketch–image pairs will be pushed apart unless their distance is already greater than the margin. Similar sketch–image pairs will be pulled together in the feature space.

Triplet Network.

Triplet networks [35] are similar to Siamese networks except that the supervision is of the form “input a should be closer to input b than to input c ”. Triplet networks use a “ranking” loss function of the form $L = \max(0, m + d(S, I+) - d(S, I-))$. This ranking loss function can express more fine-grained relationships than the Siamese loss which can only say pairs of points should be near or far.

Traditionally, while a Siamese network can be conceptualized as two networks and a Triplet network can be conceptualized as three networks there is really only a single network used repeatedly for all inputs. This makes sense when the embedding being learned is not cross-domain. For example, Hadsell et al. [34] addresses the within-domain task of face verification where input faces are classified as same or different identity. But our inputs – sketches and photos – are dramatically different and it makes sense to train two networks with independent weights to embed the sketches and photos. This is a departure from most previous deep embedding works, even those that are cross-domain. Lin et al. [30] find that independent weights barely improved ground-to-aerial image matching and Bell and Bala [31] use shared weights for iconic-to-internet product photo matching.

Using shared weights also makes sense if you can convert between domains before passing inputs into the CNNs. For example, Wang et al. [29] addresses sketch-based 3D model retrieval and renders the 3D models like sketches. An analogous approach for us would be to run edge detection on photos before using a Siamese or Triplet network with

shared weights. Yu et al. [25] take this approach. But we suspect that detected edges are not a good proxy for the image structures that humans draw, and it makes more sense to let the deep networks learn the cross-domain transformation from data.

Classification loss.

A successful sketch-based image retrieval system needs to respect both the semantics (e.g. object category) and fine-grained details (e.g. pose, shape, viewpoint, and other attributes) of a user query sketch. If a user sketches a horse, then it is not satisfactory to retrieve cows, zebras, and dogs in the same pose. While the Siamese or Triplet losses encourage CNNs to be sensitive to fine-grained sketch-to-photo similarities, we can improve performance by including a *classification loss* when training our deep networks. We use the traditional “softmax” classification loss with the 125 categories of the Sketchy database and this helps ensure that retrieval results match the category of a query. The same approach was used by Bell and Bala [31] to improve image retrieval.

Network architectures.

We experiment with two deep network architectures – AlexNet [47] as implemented in Caffe [48] and the deeper GoogLeNet [49]. We omit the auxiliary classification loss layers of the GoogLeNet architecture since their effect was negligible in our experiments. Figure 2.5 visualizes our cross-domain network.

Data preparation and Pre-training

We first train each subnetwork for sketch and image classification. The networks independently learn weights appropriate for each domain without any initial constraints for common embedding. We start with AlexNet or GoogLeNet trained on ImageNet. The sketch network is fine-tuned to recognize the 250 categories from Eitz 2012 [8]. We divide the dataset into a train/test split of 18k/2k and after fine-tuning achieve 77.29% accuracy

with AlexNet and 80.85% accuracy with GoogLeNet (this represents the state of the art on Eitz 2012 to the best of our knowledge).

Cross-domain classification.

Up to this point, each network is trained separately for their specific domain and the ‘features’ computed by these networks are not comparable (or even the same dimensionality at the highest layer). As before, we train the network with classification loss only, but switch to training using the 125 sketch categories of the Sketchy database for sketch network branch. We also collect 1000 Flickr photos for each of our 125 categories and use them to further train image branch. This means that the activations at the top of each network are comparable 125-dimensional features, where each dimension of the feature is a measure of confidence of the presence of a particular category. These 125 dimensional features are used as a “retrieval by categorization” baseline which we evaluate in Section 2.1.4.

Fine-grained training data.

As discussed in Section 2.1.2, the Sketchy database provides fine-grained correspondence between sketches and photos that can be used as positive pairs for training our cross-domain CNNs. We hold out 10% of the data for testing. The only ‘jittering’ we use is mirroring. Positive sketch-photo pairs are mirrored jointly because we do not want to be mirror invariant. After mirroring, the 90% of data used for training provides more than one hundred thousand positive pairs (22,500 images with *at least* 5 sketches each) and more than a billion negative pairs.

Sketch Normalization.

We uniformly scale and center sketches from the Sketchy database so that the learned representation is not sensitive to the absolute location and scale of a sketch. This normalization makes our benchmark harder because it breaks the spatial correspondence between

sketches and photos. But in a realistic sketch-based image retrieval scenario we assume the user wants to be invariant to location and scale and instead wants to match pose, shape, and other attributes. We release the aligned sketch-photo data, though, as it would be more appropriate training data for some scenarios (e.g. learning to render photos like sketches).

Training Cross-Domain Embeddings

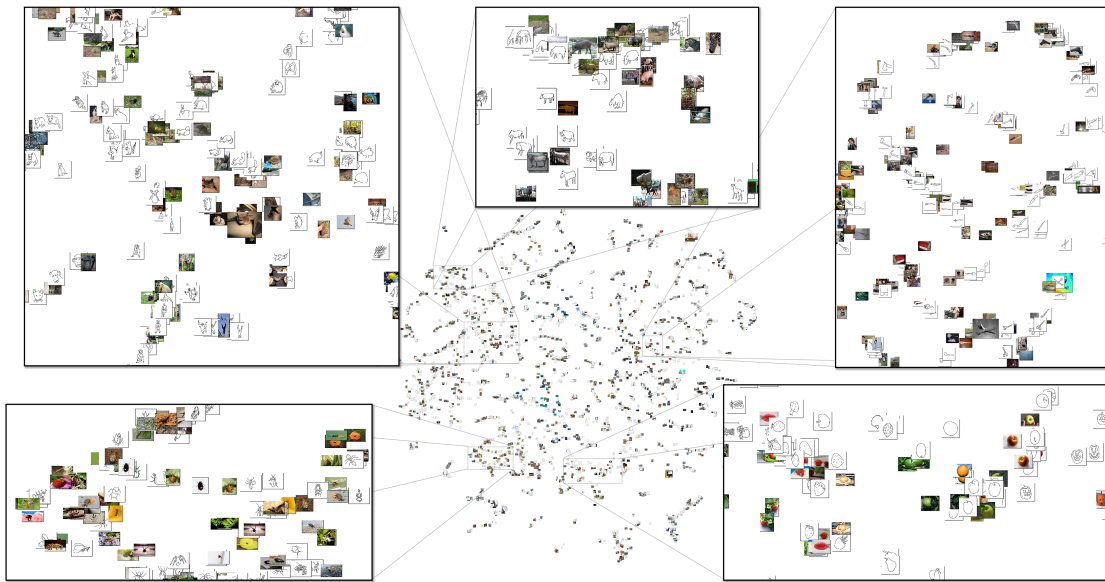


Figure 2.6: t-SNE visualization of the Sketchy database test set of 1250 images and sketches. The images and sketches are embedded into a common feature space with our best performing Triplet network. Note that nearby sketches and photos share not just the same category but also similar shape and pose. Categories also seem to have arranged themselves by semantic similarity even though no such supervision was provided – insects such as bees, scorpions, ants are grouped (lower left), as are small animals (top left), grazing animals (top center), hand-held devices (top right), and fruits (bottom right).

We now train deep networks to embed sketches and photos into a 1024 dimensional space with Siamese and Triplet loss functions. These Siamese and Triplet losses are used simultaneously with a softmax classification loss.

Training with Siamese contrastive loss.

The inputs for Siamese training are a sketch-photo pair, each passed to the correspond-

ing subnetwork, and supervision as to whether the pair is matching or not. The contrastive loss has a free parameter, the margin, which can significantly affect the learned embedding. Since our dataset can be interpreted as having two discrete levels of similarity – categorical similarity and instance similarity – we train the Siamese network in two phases. First, we use a large margin and label all sketches and photos from the same category as matching (this is effectively categorical supervision but *not* the typical categorical softmax loss function). Next, we use a small margin and treat only instance level sketch-photo pairs as matching. E.g. a sketch of a rabbit and a photo of a rabbit would be pushed apart if the sketch was not generated from that photo. In each epoch, we train with 10 times more negative than positive pairs while maintaining a 50% ratio by repeating positive pairs accordingly. Since negative pairs are plentiful we resample new negative pairs between training epochs.

Training with Triplet ranking loss.

For the triplet loss we need input tuples of the form $(S, I+, I-)$ corresponding to a sketch, a matching image, and a non-matching image. For the ranking loss, our experiments suggest it is better to sample $I+$ and $I-$ only from the sketch category (e.g. a zebra sketch with a matching zebra photo and non-matching zebra photo), because we will simultaneously use a classification loss which differentiates sketches from photos of different categories. Note that for our Triplet loss network the two image branches still share weights. As a result, both Siamese and Triplet networks will have one set of weights for the sketch domain and one set of weights for the image domain.

We train networks using Caffe [48]. Training parameters such as learning rate decay and batch size can be found in the supplemental material. Figure 2.6 visualizes the embedding learned by this final Triplet network. The non-linear projection from 1024D to 2D is performed using t-SNE [50].

Matching Sketches and Images

Our network consists of two subnetworks responsible for mapping the input sketch and photo into a shared 1024 dimensional feature space.³ We can precompute features for large sketch or image databases and rapidly search for matches when given a query.

2.1.4 Quantitative Evaluation

We evaluate our learned representation on two sketch-based image retrieval benchmarks – the fine-grained benchmark of Li et al. [22] and the held out test set of the Sketchy database.

The Li et al. [22] benchmark evaluates *intra-category* sketch retrieval for a known category, e.g. given a sheep sketch rank a set of sheep photos according to similarity. Retrieval results are scored based on how often the $K = 5$ top retrieved images share discrete, hand-coded attributes of viewpoint, zoom, part configuration, and body shape. Higher scores are better. We evaluate on the 10 common categories between our datasets. This benchmark assumes category-specific models but our Triplet network is trained to simultaneously embed and recognize 125 object categories. Nonetheless it slightly outperforms Li et al.’s category-specific deformable part models on average, although their method is better on three categories. Both methods outperform a spatial pyramid (SP) baseline.

Table 2.1: Sketch retrieval benchmark of Li et al. [22]

	ours	Li et al.	SP
airplane	27.2	22	20.33
bicycle	21.5	11.67	13.83
car	15.8	18.83	14.5
cat	13.8	12.17	7.67
chair	21.7	20	20.33
cow	19.8	19.67	14
dog	21	9.5	6.83
horse	23.2	31.67	7.33
motorbike	13	22.5	9
sheep	21	17.67	5
average	19.8	18.57	11.88

³The networks also output 125 dimensions corresponding to classification confidences, but we discard these for retrieval applications.

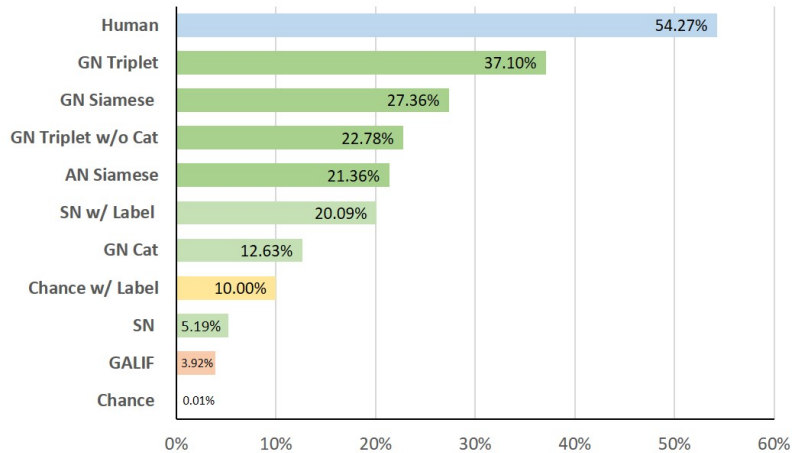


Figure 2.7: Average recall at $K = 1$ on the Sketchy database test set. This is the percentage of time that the highest ranking (smallest distance) retrieved image is the single matching photo among 1250 test images. See main text for explanation of algorithm variants. Green bars correspond to methods and features proposed in this paper.

We also evaluate several deep networks and baselines for sketch-based retrieval on our held out test set of 6312 query sketches and 1250 photos spanning 125 categories. For each sketch query there is a single matching photo (the photo that prompted the creation of that sketch), and we measure performance by recall @ K . For a particular sketch query, recall @ K is 1 if the corresponding photo is within the top K retrieved results and 0 otherwise. We average over all queries to produce Figures 2.7 and 2.8. Figure 2.7 focuses on the most challenging $K = 1$ case. It shows how often the top retrieval result is the single matching photo in the 1250 photo test set. Figure 2.8 plots recall @ K for $K = 1$ to 10.

We estimate human performance at $K = 1$ by having participants manually find the matching photo for a query sketch from our test set. To ease the task we sort the photos into categories so that participants need not browse 1,250 photos, but participants are not told the ground truth category. After 772 trials, the human participants select the correct photograph 54% of the time. This is not an “upper bound”, though – there was large variance in accuracy between participants with some achieving greater than 70% $K = 1$ recall. We include this human baseline in Figure 2.7.

We compare the following retrieval methods:

GN Triplet. This is our top-performing model, GoogLeNet trained with Triplet and classification loss.

GN Siamese. GoogLeNet trained with Siamese and classification loss.

GN Triplet w/o Cat. GoogLeNet trained exclusively with the Triplet loss.

AN Siamese. AlexNet trained with Siamese and classification loss.

GN Cat. GoogLeNet trained exclusively with the classification loss on the Sketchy database. The sketch and photo networks are trained independently, but as they predict the same 125 categories their final layer outputs are comparable. GN Cat is meant to represent a “retrieval by categorization” method.

Chance. Photos are retrieved in random order.

Chance w/ Label. This represents a hypothetical algorithm which can achieve perfect category recognition but ranks the results within that category randomly. This is a strong baseline for our test set – with 125 categories and 10 images per category, this baseline will always find the matching photo within the top 10 out of 1250 retrieved photos.

GALIF. Gabor local line based feature [Eitz2012sbsr]. This feature describes sketches using a bank of Gabor filters. It was used for sketch-based shape retrieval by rendering 3D objects as sketches. We apply it to sketch-based image retrieval by performing Canny edge detection on the photos before computing the feature.

SN. For another “retrieval by categorization” baseline, we fine-tune GoogLeNet with the 250 category Eitz 2012 dataset and then use the network as a feature extractor. Similar to GALIF approach, we first use edge detection on photos then extract features from both sketch and edge images. We use the 1024 dimensional penultimate network layer activations as the feature.

SN w/ Label. Same as SN, except that we assume category recognition is perfect as in the Chance w/ Label baseline. The SN representation outperforms that baseline because it can still sort results within class. Even though this method has an oracle telling it ground truth sketch and photo category, it still performs worse than networks trained on

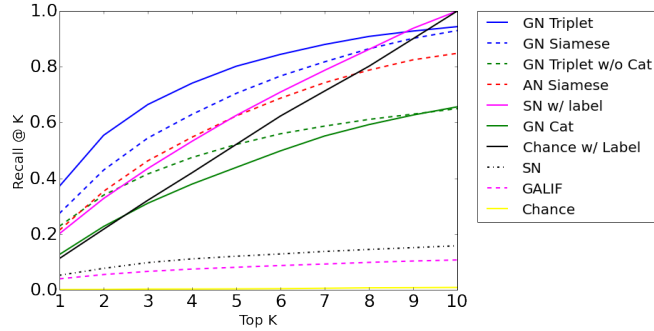


Figure 2.8: Evaluation on our test set. We measure recall at K – whether or not the network can retrieve target image within Top K nearest neighbors. See main text for explanation of algorithm variants.

fine-grained sketch-photo pairs for small values of K .

Our benchmark leads us to conclude that the deeper GoogLeNet significantly outperforms AlexNet for our task. Likewise, Triplet loss significantly outperforms Siamese loss. Perhaps most surprising is the effect of combining classification loss with Triplet or Siamese losses – incorporating classification loss significantly improves fine-grained sketch retrieval even though it is the least successful loss function on its own. Classification loss alone leads to $K = 1$ recall of 12.6%, and Triplet loss alone leads to $K = 1$ recall of 22.8% but together the recall improves to 37.1%. This means that one third of the time the single correct match out of 1250 test photos is the first retrieval result. The correct match is within the top 8 results 90% of the time.

Figure 2.10 shows examples of sketch-based image retrieval. For each sketch query we show the top 10 retrieval results (smallest distance in the Triplet GoogLeNet feature space). We search a collection of 255,828 Flickr images. 125,000 of these photos come from downloading 1,000 Flickr photos with tag searches based on the 125 Sketchy database categories. 130,828 more images are downloaded randomly. The query sketches come from the Eitz 2012 database. The retrieved photos generally depict objects of the desired type in similar poses.

Evaluation of pre-training.

We quantify the effect of pre-training by incrementally adding pre-training with ImageNet, Eitz 2012, and Flickr. All experiments conclude with 160 thousand iterations of training on the Sketchy database with triplet and classification loss and the GoogLeNet architecture. No pre-training (random initial weights) leads to recall at $K = 1$ of 2%. Pre-training with ImageNet leads to 28%. ImageNet and Eitz 2012 leads to 30%. ImageNet and Flickr leads to 33%. Using all three for pre-training leads to 36% recall at $K = 1$. This is slightly less than our best reported model because of simplified training during these experiments.

Training on MS COCO object boundaries.

We claim that human object sketches are not particularly similar to object boundaries or silhouettes. To test this, we train a network using object boundaries from MS COCO [41] rendered like our sketches. We omit MS COCO instances that are partially occluded by other objects or truncated by the image boundary. We train and evaluate on the 13 object categories that overlap between the Sketchy and COCO databases and have at least 500 object instances. The MS COCO “sketches” are centered and scaled in the same manner as the Sketchy database and used to train networks with only ImageNet pre-training. For these 13 categories, the network trained on the Sketchy database performs significantly better – 35% vs 11% $K = 1$ recall. This suggests that faithful object boundaries are not a good proxy for human-drawn object sketches.

2.1.5 Visualization: Average Objects

If our learned representation allows us to retrieve semantically similar objects with similar poses it could enable an object synthesis or data exploration tool in the spirit of AverageExplorer [11]. AverageExplorer operates on constrained image sets, e.g. image search results for “kids with Santa” or “a brown tabby cat” but we aim to create average images from

a diverse collection of 255,828 Flickr images with no user intervention beyond the initial query sketch.

The primary challenge is that our retrieval results (Figure 2.10) contain objects at different scales and locations, even when the pose and object attributes are fairly similar. This means that a naive average of the top retrieved photos is very blurry. In Section 2.1.2 we speculated that our sketch-photo training pairs implicitly encode information about which image structures are salient. We use the approach of Zeiler and Fergus [51] to localize “salient” regions as learned by our deep network. The approach is conceptually simple – we grey out image regions and if that leads to a large change in the final network layer activations then we must have removed a salient structure. Figure 2.9a shows that this does indeed tend to localize objects in photos. Figure 2.9b shows an average image computed by centering and rescaling top retrieval results such that the salient regions overlap. Such averages are sharper than unaligned averages but still blurry.

To improve the averages further we use FlowWeb [52] to find correspondences among the retrieved photos *and* the query sketch. Aligning the sketch to the detected edges of the photos works slightly better than using the photos directly. Figure 2.9b shows averages computed after the retrieved photos have been transformed according to FlowWeb correspondences. The FlowWeb algorithm was generally unable to find correspondences if we skipped the salience-based alignment and centering. We think that these average images demonstrate the ability of our learned representation to find cross-domain matches and hints at the possibility of new sketch-based image synthesis techniques.

2.1.6 Limitations and Future Opportunities

Limitations.

Our data collection and training assumes that there are three discrete levels of similarity between sketches and photos – the same instance, the same category, or completely dissimilar. But sketch-photo similarity as perceived by a human would of course be more

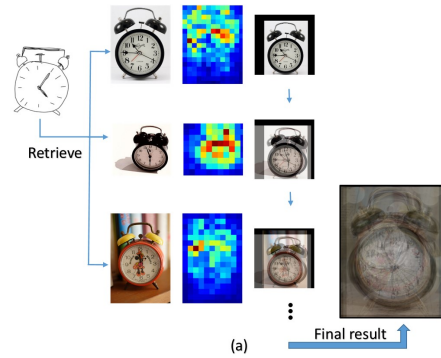


Figure 2.9: In (a) we align top retrieval results such that the salient regions overlap and this produces a cleaner average image. (b) shows examples of average images from our top 20 results. Fine-grained alignment with the overlaid sketch is based on corresponding points from Flowweb [52].

continuous. It is not economical to collect all pairs of perceptual distances between 75,471 sketches and 12,500 photos and it doesn't appear necessary to learn a useful representation. Still, it would probably help to have more annotations, especially within categories where it seems unsatisfactory to assume that there is only one valid match for any sketch query – there could be several photos that match a sketch reasonably well. The Triplet ranking loss naturally accommodates such finer-grained supervision. This limitation is addressed by “Sketch Me that Shoe” [25], but their database is considerably smaller.

Possible applications of the Sketchy Database

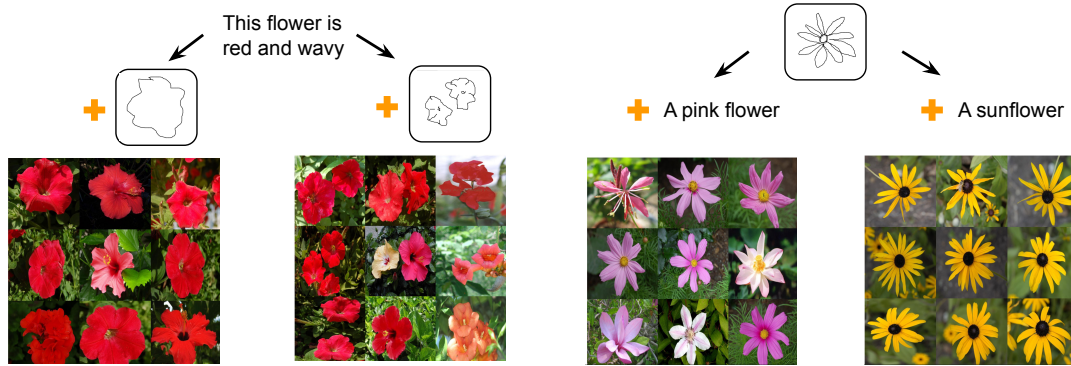
Previous portrait synthesis methods [37, 38] used sketch-photo pairs to aid the generative photo to sketch process. Our database could be used as training data for generating



Figure 2.10: Qualitative retrieval results. Queries are sampled from the Sketchy database test set. Matching photos are retrieved from a held out collection of 450 thousand Flickr photos, of which 50k are retrieved by querying categories in the Sketchy database and 400k are random. There is no ground truth, but clearly incorrect retrieval results are highlighted in red.

sketches of photographic objects. In the other direction, previous methods synthesized images from sketches [4, 5] and the Sketchy database could be used to improve the sketch-based retrieval at the core of these methods or to directly learn deep generative image models [53]. The database could also enable the study of the humans artists analogous to “Where do people draw lines” [54] which studied drawings of 3D shapes. Finally, while we were motivated to collect the Sketchy database to provide training data for deep networks, it can also serve as an image-retrieval benchmark for a field that has a shortage of large scale benchmarks.

2.2 A Sketch Is Worth a Thousand Words: Image Retrieval with Text and Sketch



Cross-modal retrieval [55] is a retrieval problem where the query and the set of retrieved objects take different forms. A representative problem of this class is Text-Based Image Retrieval (TBIR), where the goal is to retrieve relevant images from an input text query. TBIR has been studied extensively, with recent interest focused on transformer-based models [56]. At the core of TBIR is a scoring function that assesses the similarity between a text description and an image. With a scoring function, given a text description, retrieving relevant images amounts to finding the top k images that achieve the highest scores.

Recent approaches for defining the scoring function can be broadly categorized into two types based on when information from the two modalities is fused: 1) early fusion, and 2) late fusion. In early fusion, the scoring function takes as input the text query and a candidate image and outputs a scalar score. Modern choices for the scoring function involve defining a deep neural network that jointly takes the two inputs. Cross modality attention over both modalities is a common approach to this task [57, 58, 59]. In the late fusion approach, a separate feature extraction network is employed for each modality to co-embed the two input objects into the same embedding space (i.e., dual encoders) [60, 61]. The similarity score is then given by the inner product of the two embeddings. Unlike late fusion where the scoring function is factorized, it is generally observed that early fusion can

achieve higher capacity for modeling a complex scoring function because of its unrestricted form. The factorization in the late fusion approach, however, has a clear advantage of better computational efficiency, since it allows pre-indexing embeddings of retrievable images in the database for fast maximum inner product search at retrieval time. The dual encoders approach is what we focus on in this paper.

While a text description is suitable for describing qualitative attributes of an object (e.g., object color, object shape), it can be cumbersome when there is a need to describe multiple objects or a complex shape. Take the query “This flower is red and wavy” in Figure 3.11 as an example. This query alone can match red flowers of a wide range of shapes. Further, with multiple objects, it becomes necessary to specify their relative positions, which makes it too cumbersome to be practical. These problems naturally led to a related thread of research on Sketch-Based Image Retrieval (SBIR), where the goal is to retrieve images from an input sketch [1, 62, 63, 64, 65]. Compared to text, specifying object positions with a hand-drawn sketch is relatively easy.

SBIR has gathered attention of late to address the aforementioned limitations of TBIR, among others. Owing to the difficulty of SBIR, recent works on SBIR tend to focus on a specific set of retrievable images. For instance, [1] considers object based retrieval; that is, each image has only one object centered in the image. Compared to [1], [63] allows more free-form sketch input, but from a single category e.g., shoes. While suitable for an e-commerce product search, searching from an arbitrarily drawn sketch, also known as *in-the-wild SBIR*, was not commonly studied in [63]. This is the problem we tackle in this work.

In-the wild SBIR presents two challenges. Firstly, there is semantic ambiguity in a sketch drawn by a non-artist user. To a non-artist, it takes effort to draw a sketch which sufficiently accurately represents the desired target image to retrieve. While adding details to the sketch would naturally narrow down the candidate image set, to a non-artist user, the extra effort required to do so may outweigh the intended convenience of SBIR. Ideally the

retrieval system should be able to extract relevant information from a poorly drawn sketch. Secondly, the few publicly available SBIR datasets contain only either images of single objects [1, 63], or images describing single concepts [66]. These images are different from target images in in-the-wild image search, which may contain multiple objects with each belonging to a distinct category.

In this work, we address these two challenges in in-the-wild image retrieval by proposing to *use both a sketch and a text query as input*. The (optional) input sketch is treated as a supplement to the text query to provide more information that may be difficult to express with text (e.g., positions of multiple objects, object shapes). In particular, we do not require the input sketch to be drawn well. As will be seen in our results, when coupled with a text query, an extra input sketch (even a poorly drawn one) can help considerably narrow down the set of candidate images, leading to an increase in retrieval recall. We show that both input modalities can complement each other in a manner that cannot be easily achieved by either one alone. To illustrate this point concretely, two example queries can be found in Figure 3.11 where dropping one input modality would make it difficult or impossible to retrieve the same set of images. This idea directly addresses the first challenge of in-the-wild SBIR – sketch ambiguity.

Combining two input modalities is made possible with our proposed similarity scoring model, *TASK-former*, which follows the late-fusion dual-encoder approach, and allows efficient retrieval (see Figure 2.11 for the model and our training pipeline). Our proposed training objective comprises 1) an embedding loss (to learn a shared embedding space for text, sketch, and image); 2) a multi-label classification loss (to allow the model to recognize objects); and 3) a caption generation loss (to encourage a strong correspondence between the learned joint embedding and text description). Crucially, training our model only requires synthetically generated sketches, not human-drawn ones. These sketches are generated from the target images, and are further transformed by appropriate augmentation procedures (e.g., random affine transformation, dropout) to provide robustness to ambiguity

in the input sketch. The ability of our model to make use of synthetically generated sketches during training addresses the second challenge of in-the-wild SBIR – lack of training data. We show that our model is robust to sketches with missing strokes (Figure 2.14), and able to operate on human-drawn input sketches (Figure 2.13).

Our contributions are as follows.

1. We present TASK-former (Text And SKetch transformer), a scalable, end-to-end trainable model for image retrieval using a text description and a sketch as input. *To our knowledge, our work is one of the first that proposes a model that jointly takes a sketch and a text description as input for in-the-wild image retrieval.*
2. We collect 5000 hand-drawn sketches for images in the test set of COCO [67], a commonly used dataset to benchmark image retrieval methods. The collected sketches will be made publicly available.
3. We empirically demonstrate (in Section 2.5) that using an input sketch (even with a poorly drawn one) in addition to text helps increase retrieval recall, compared to the traditional TBIR where only a text description is used as input.

2.3 Related work

Sketch Based Image retrieval (SBIR) There are several works that tackle SBIR [1, 62, 63, 64, 65]. The works of [62, 68, 69, 70, 64] consider zero-shot SBIR where retrieving from unseen categories is the focus. Zhang et al. [71] tackles SBIR using only weakly labeled data. Fine-grained SBIR of a restricted set of images (e.g., from a single category) is studied in [65, 72, 73]. An interactive variant of SBIR was considered in [74] where images are retrieved on-the-fly as an input sketch is being drawn. They propose a form of sketch completion system for SBIR based on user’s choice of image cluster to refine the retrieval results. Bhunia et al. [75] present a reinforcement learning based pipeline for SBIR and allows a retrieval even before the sketch is completed. Being able to retrieve

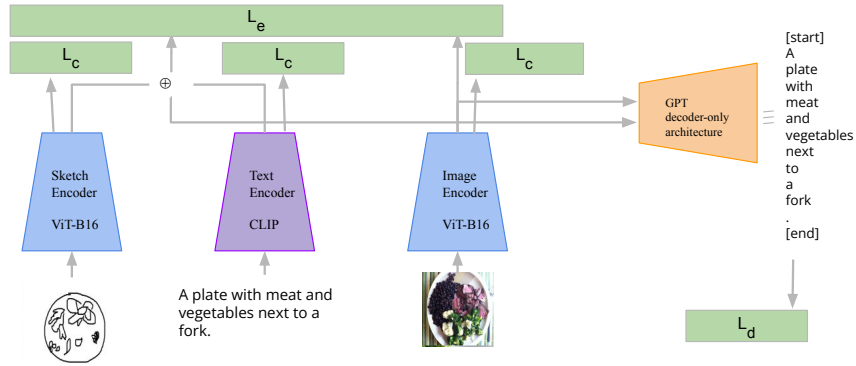


Figure 2.11: Overview of our model and the training pipeline. ViT is the vision transformer [76], and \oplus represents an element-wise addition. We extend CLIP [61] to incorporate additional sketch input from user. We add two auxiliary tasks: multi-label classification (L_c) and caption generation (L_d). Our objective losses are explained in Section 2.4.2. We encourage the network to learn a discriminative representation to simultaneously distinguish at the instance level and at the category level, both of which are crucial for a successful retrieval.

relevant images even with incomplete sketch information is an important aspect of SBIR. In this work, we achieved this by supplementing the (incomplete) sketch with an input text description.

Text-based image retrieval (TBIR) The problem of retrieving images based on a text description is closely related to the ability to learn a good representation between these two different domains. Previous successes in TBIR relied on the cross attention mechanism (a specific form of early fusion) to learn the similarity between text and images [77, 78, 57]. It is generally believed that, compared to a late fusion model, an early fusion model can offer more capacity for modeling a complex scoring function owing to its non-factorized form [79, 80]. Interestingly however the recently proposed CLIP [61], a late fusion model, is able to achieve comparable TBIR results on COCO to existing early fusion models. This feat is impressive and is presumably due to the use of a large proprietary dataset of text-image pairs crawled from the Internet. The representation learned by CLIP is incredibly rich, and correlates well with human perception on image-text similarity as observed in [81].

Similar to CLIP, ALIGN [60] also proposes a similar contrastive training scheme on a large-scale data. Fine-tuning from a pre-trained network in ALIGN is able to achieve state-of-the-art (SOTA) accuracy of 59.9% in retrieving a single correct image based on text description in the COCO image retrieval benchmark (karphathy split).⁴ In our work, we opt to use the pre-trained CLIP network to initialize our network for training.

Image retrieval with multi-modal queries Using text and image as queries for image retrieval has been explored extensively in the literature, for example as a textual instruction for a desired modification of the query image [83, 84, 85, 86, 87]. Jia et al [60] demonstrate that it is possible to directly combine representations of the two input modalities (text and image in this case) with a simple element-wise addition, for the purpose of image retrieval with maximum inner product search. In our work, we take this insight and combine the representation of sketch and text inputs in the same way. Using a sketch as input which is further supplemented by additional information (such as a classification label) has also been studied in Sketch2Tag [88] and Sketch2Photo [89]. Instead of sketch, [90] explores using fine grain association between mouse trace and text description for image retrieval. Surprisingly, combining sketch and full text description as queries for image retrieval has been relatively under explore despite its potential usefulness. [91, 92, 93] explore the benefit of training with both sketch and text inputs for the purpose of improving image retrieval performance for each modality separately. Only [93] demonstrate examples of retrieving with both sketch and text inputs by linearly combining their score during test time. Note that unlike ours, the proposed loss function of [93] does not explicitly consider interactions between sketch and text inputs, since the goal is to simply improve retrieving with each modality separately. Further to find the similarity between a sketch and a given image, the proposed method of [93] requires that the image be first converted to a sketch via edge detection. This conversion would incur a loss of information such as the color.

⁴More recently, Microsoft team also presented T-Bletchley [82] with a similar pipeline, achieving even higher score than ALIGN at 62.3%.

The proposed method also requires a conversion of image into sketch via edge detection, and would lose color information which is not ideal. While [93] is one of the most related works to our work, we are unfortunately not able to obtain the data used, or the trained model for a quantitative comparison.

2.4 Proposal: TASK-former

TASK-former can be considered an extension of the training pipeline proposed in CLIP [61]. We start with the same network architectures and contrastive learning as in CLIP, but modify them and add additional loss terms to allow the use of both a sketch and an input text query. Specifically, we include two additional auxiliary tasks: multi-label classification and caption generation. Our motivation is to improve the learned embedding space so as to achieve the following goals.

1. Discriminate between the positive and negative pairs;
2. Distinguish objects of different categories; and
3. Contain sufficient information to reconstruct the original text caption from embeddings of an image and its sketch.

We achieve these goals via CLIP’s symmetric cross entropy, multi-label classification objective, and caption generation objective, respectively. While these three objectives appear to seek conflicting goals (e.g., the classification loss might encourage discarding class-invariant information, whereas the image captioning loss would suffer less if all describable details are kept), we observe that combining them with appropriate weights can lead to gains in performance for image retrieval as shown in Table 2.3.

We start by describing our model and training pipeline in Section 2.4.1, and describe the three aforementioned loss terms in Section 2.4.2.

2.4.1 Training pipeline

Our pipeline is summarized in Figure 2.11. Each input query consists of 1) a hand-drawn sketch, and 2) a text description of desired target images. We use the same image and text encoder architecture as described in CLIP [61] in order to leverage networks that have been pre-trained with large-scale training data. This choice is in accord with the common practice of using, for instance, an ImageNet pre-trained network for various vision tasks. We use CLIP’s publicly available pre-trained model ViT-B/16, in which the image encoder is based on the Vision Transformer [76] which we found to be the best performer for our task. For the details of the network architecture of each encoder that we use, please refer to CLIP [61] and ViT [76].

There are three encoders in our pipeline for: 1) input sketch, 2) input text description, and 3) a candidate retrieved image. The output embeddings from the sketch and the text encoders are combined together and used for contrastive learning with the image embedding as the target. We explore several options for combining features in Section 2.5.1 Since images and sketches are in the same domain (visual), we use the same architecture for them (ViT-B/16 pre-trained on CLIP). We also found that the performance is much better when sharing weight parameters across the sketch and the image encoders. Yu et al. [63] also observed similar results where the Siamese network performs significantly better than heterogeneous networks for the SBIR task. It was speculated that weight sharing is advantageous because of relatively small training datasets. This explanation may also hold in our case as well given the complexity of our task. For classification, we feed each embedding to two additional fully connected layers with ReLU activation.

Additionally, we also train a captioning generator from the embeddings of sketches and images using a transformer based text decoder. This decoder is an autoregressive language model similar to GPT decoder-only architecture [56]. We use absolute positional embedding, with six stacks of decoder blocks, each with eight attention heads. More details can be found in the supplementary materials.

2.4.2 Objective function

Our objective function consists of three main components: symmetric cross entropy, asymmetric loss for multi-label classification, and auxiliary caption generation.

Embedding Loss (L_e) To learn a shared embedding space for text, image and sketch, we follow the contrastive learning objective from CLIP [61], which use a form of *InfoNCE Loss* as originally proposed in [94] to learn to match the right image-text pairs as observed in the batch. This proxy task is accomplished via a symmetric cross entropy loss over all possible pairs in each batch, effectively maximizing cosine similarity of each matching pair and minimizing it for non-matching ones. We add the sketch as an additional query, and replace the text embedding in CLIP with our combined embedding constructed by summing the text and sketch embeddings.

Classification Loss (L_c) For classification, we consider this as a multi-label classification problem as each image can belong to multiple categories. We follow a common practice in multi-label classification, which frames the problem as a series of many binary classification problems. We use object annotation available in the datasets as ground truth. Specifically, we use Asymmetric Loss For Multi-Label Classification (ASL Loss), proposed in [95]. The loss is designed to help alleviate the effect of the imbalanced label distribution in multi-label classification.

Auxiliary caption generation (Decoder Loss, L_d) For caption generation, the decoder attempts to predict the most likely token given the accumulated embedding and the previous tokens. The decoded output tokens are then compared with the ground truth sentences via the cross entropy loss $\sum_t^T \log(p_t|p_1, \dots, p_{t-1})$, where T is the maximum sequence length.

Our final objective is given by a weighted combination of all the loss terms. We refer to supplementary materials for our hyperparameter choices.

2.4.3 Sketch generation and data augmentation

During training, we synthetically generate sketches using the method proposed in [96]. In general the method produces drawings similar to human sketches. The synthesized sketches are however in exact alignment with their source images, which would not be the case in sketches drawn by humans. To achieve invariance to small misalignment, we further apply a random affine transformation on the synthetic sketch as an augmentation. We also apply a similar transformation to the image (with different random seeds). Introducing this misalignment is crucial for the network to generalize to hand-drawn input sketches at test time.

To help deal with partial sketches at test time, we also randomly occlude parts of each sketch. We randomly replace black strokes with white pixel. The completion level of each synthesized sketch in the training set is between 60% to 100%

Evaluation with sketch-text-image tuples Since our approach retrieves images with sketches and text, evaluating our approach naturally requires an annotated dataset where each record contains a hand-drawn sketch, a human-annotated text description, and a source image. SketchyCOCO [97] fits this description and is a candidate dataset. However, the dataset was constructed by having the sketches drawn first based on categories. The sketch for each category was then pasted into their supposed areas based on incomplete annotation (not all objects were annotated). As a result, the sketch in each record may poorly represent the image because 1) each category contains the same sketch, 2) the choice of which object to draw is based on categories, rather than human judgement of what is salient in each particular image. More related is the dataset mentioned in [93] which provides 1,112 matching sketch/image/text of shoes. However, the dataset is not yet available at the time of writing. Also worth mentioning is [98], which also proposes a dataset containing synchronized annotation between text description and the associated location (in a form of mouse trace) in the image. However, we argue that a line drawing sketch represents more than just the location of the objects; even a badly drawn sketch can provide information such as shape,

details, or even relative scale. Owing to lack of an appropriate evaluation dataset, we construct a new benchmark by collecting hand-drawn sketches for images in the COCO image retrieval benchmark. These images are in the COCO 5k split from [99], which is widely used to evaluate text based image retrieval methods. As part of our contributions, the collected data will be made publicly available. We describe the sketch collection process in the next section.

2.4.4 Data collection: sketching from memory

We collect sketches for COCO 5k via Amazon Mechanical Turk crowd sourcing platform (AMT). We follow the split from [99], which has been widely used as benchmark for text based image retrieval. The test set contains 5000 images which are disjoint from the training split.

To solicit a sketch, we first show the participant (Turker) the target image for 15 seconds, before replacing it with a noise image. This is to mimic the typical scenario at deployment time where there is no concrete reference image, but instead only a mental image of a retrieval target. The participant is then asked to draw a sketch from memory. In the instructions, we ask the participants to draw as if they are explaining the image to a friend but using a sketch. We do not put any restrictions on how the sketch has to be drawn, other than no shading. The participants are free to draw any parts of the image they think are important and distinctive enough to be included in the sketch. The sketches are collected in SVG (a vector format), and contain information of all individual strokes. In experiments, we use this information to randomly drop individual strokes to test the robustness of our approach to incomplete input sketches (see Figure 2.14). As a sanity check, we also ask the participants to put one or two words describing the image in an open-ended fashion.

From the results, we manually filter out sketches that are not at all representative of the target image (e.g., empty sketch, random lines, wrong object). Our only criteria is that

each sketch has to be recognizable as describing the target image (even if only remotely recognizable). Our goal is not to collect complete or perfect sketches, but to collect in-the-wild sketches, which may be poorly drawn, that can be used along side the text description to explain image.

2.5 Results and discussion

For quantitative evaluation, we calculate Recall@K, which is the fraction of times that the target image is included in the top-K retrieved images. We evaluate on COCO [67]; a commonly used datasets for language based image retrieval. We use the same data split as proposed in [99]. Specifically, COCO’s evaluation set contains 5,000 images. Each image is annotated with multiple captions, and we additionally add a sketch to each image. For COCO, we collect hand drawn sketch as described in 2.4.4.

Implementation details We use Adam [100] as the optimization method to train the model. Training hyperparameters are set in accordance with Open Clip [101] with the initial learning rate set to 10^{-5} . To provide robustness to incomplete input, for each (text query, sketch, image) training tuple, we set a 20% probability to drop either the sketch (replaced with a white image) or the text query (replaced with an empty text). We demonstrate the robustness of our model to incomplete input in Section 2.5.2. Code to reproduce our results will be made publicly available.

For evaluation, our model TASK-former is initialized with the publicly released CLIP model (ViT-16) [61], a text-based image retrieval model. Table 2.2 compares Recall@{1, 5, 10} of our method and that of current state-of-the-art approaches on text based image retrieval: Uniter, OSCAR, ALIGN, and T-Bletchley as reported in [59, 60, 57, 82], respectively. To provide a fairer comparison, we further finetune the publicly released CLIP model (ViT-16) on COCO.⁵ We observe that our method is able to achieve a considerably

⁵Note that this baseline makes use of our best-effort implementation and training. We do not have access to the official training code and there is no reported result on using a fine-tuned CLIP on an image retrieval

Table 2.2: Recall@{1, 5, 10} of state-of-the-art methods on COCO dataset. We use our collected hand-drawn sketches for evaluation. Some examples of these sketches are shown in Figure 2.12. Among ALIGN, T-Bletchley, and CLIP, only CLIP has released pre-trained model. Our TASK-former is initialized for training with this CLIP model.

Type	Method	Input	R@1	R@5	R@10
Early fusion	Uniter [59]	Text	0.529	0.799	0.880
Early fusion	OSCAR [57]	Text	0.575	0.828	0.898
Dual encoders	ALIGN [60] (fine-tuned)	Text	0.599	0.833	0.898
Dual encoders	T-Bletchley [82] (fine-tuned)	Text	0.623	-	-
Dual encoders	CLIP [61] (fine-tuned)	Text	0.518	0.811	0.891
Dual encoders	TASK-former (ours)	(Text, Sketch)	0.635	0.871	0.931

higher recall than other methods, and CLIP in particular, owing in part to the use of a sketch as a supplemental input to text. Our method can retrieve the correct image with recall@1 of 63.5%, compared to using text alone, which achieves 51.8% on the same CLIP architecture (ViT-16).

At the time of writing, models for ALIGN and T-Bletchley have not been released. Both methods have been shown to perform better than CLIP on text based image retrieval, and without any early fusion of text and image. We note that both ALIGN and T-Bletchley can be used as part of our framework by simply replacing the pre-trained encoders for text and image.

2.5.1 Ablation study

In this section, we seek to understand the effect of each of our proposed loss functions (L_e, L_c, L_d), as described in Section 2.4. Baselines used as part of this ablation study are:

- **CLIP zero shot.** Recall@K that has been reported in [61] for zero-shot image retrieval. We note that their best model for the reported results (ViT-L/14@336px) is

dataset.

not available publicly.

- **Ours:** L_e (ablated TASK-former). We start adding a sketch as an additional query, as shown in Figure 2.11. The only objective in this baseline is to correctly classify the correct matching pair between the query (sketch+text) and the image via *symmetric cross entropy loss*.
- **Ours:** $L_e + L_c$ (ablated TASK-former). In this baseline, we add the multi-label classification loss term in the objective.
- **Ours:** $L_e + L_c + L_d$ (ablated TASK-former). In this baseline, we add both classification loss and decoder loss

For the above three baselines, text and sketch embeddings are combined by adding them, as described in Figure 2.11. We further compare two additional ways to combine sketch and text embeddings: coordinate-wise maximum, and concatenation:

- **Ours: Feature max** (ablated TASK-former). Embeddings from sketch and text are combined using element-wise max.
- **Ours: Feature concatenate** (ablated TASK-former). Embeddings from sketch and text are concatenated, and projected into the same dimension as embedding from image.

We use the full objective (i.e., $L_e + L_c + L_d$) for the above two variants.

- **Ours (final)** This is our complete model trained with the full objective ($L_e + L_c + L_d$). We augment training sketches and images with random affine transformation, randomly remove parts of each sketch (as describe in section 2.4) and train for 50 epochs.

Table 2.3: Results from our ablation study as described in Section 2.5.1. We construct variants of our proposed method by selectively including only some training losses. These are denoted by L_e , $L_e + L_c$, and $L_e + L_c + L_d$ (see Section 2.4.2). Sketch and text inputs are combined by adding their embeddings. For “Feature max” and “Feature concat”, the embeddings are combined by coordinate-wise maximum, and concatenation, respectively, and the full objective is used.

Method	R@1	R@5	R@10
CLIP (zero shot)	0.378	0.624	0.722
Ours: L_e	0.493	0.748	0.836
Ours: $L_e + L_c$	0.509	0.764	0.850
Ours: $L_e + L_c + L_d$	0.527	0.778	0.862
Ours: Feature max	0.443	0.704	0.804
Ours: Feature concat	0.357	0.650	0.768
Ours (final)	0.635	0.871	0.931

Except for the final model, we train each baseline for 10 epoches. For ablation study, we simplify the training by only performing simple augmentation (random cropping and flipping).

Table 2.3 reports recalls of each baselines. Both L_c and L_d further improve the retrieval performance compare to our baseline with only embedding loss (L_e). Surprisingly, our experiment also show that a simple element wise addition leads to the best performance compare to element wise max, and concatenation. We hypothesize that direct combination of the sketch and text embedding implicitly helps with feature alignment, compare to concatenate them.

2.5.2 The robustness of sketch and text

One key benefit of our pipeline is the ability to retrieve image even when missing one of the input modalities (sketch or text). We accomplish this by adding a query drop-out augmentation which replaces either the sketch or text with an empty sketch or an empty string. This ensures the network can operate even when no sketch or text description is given. Figure 2.13 shows the results breaking down into querying with sketch only, querying with text only, and querying with both modalities. We emphasize that sketch based image retrieval

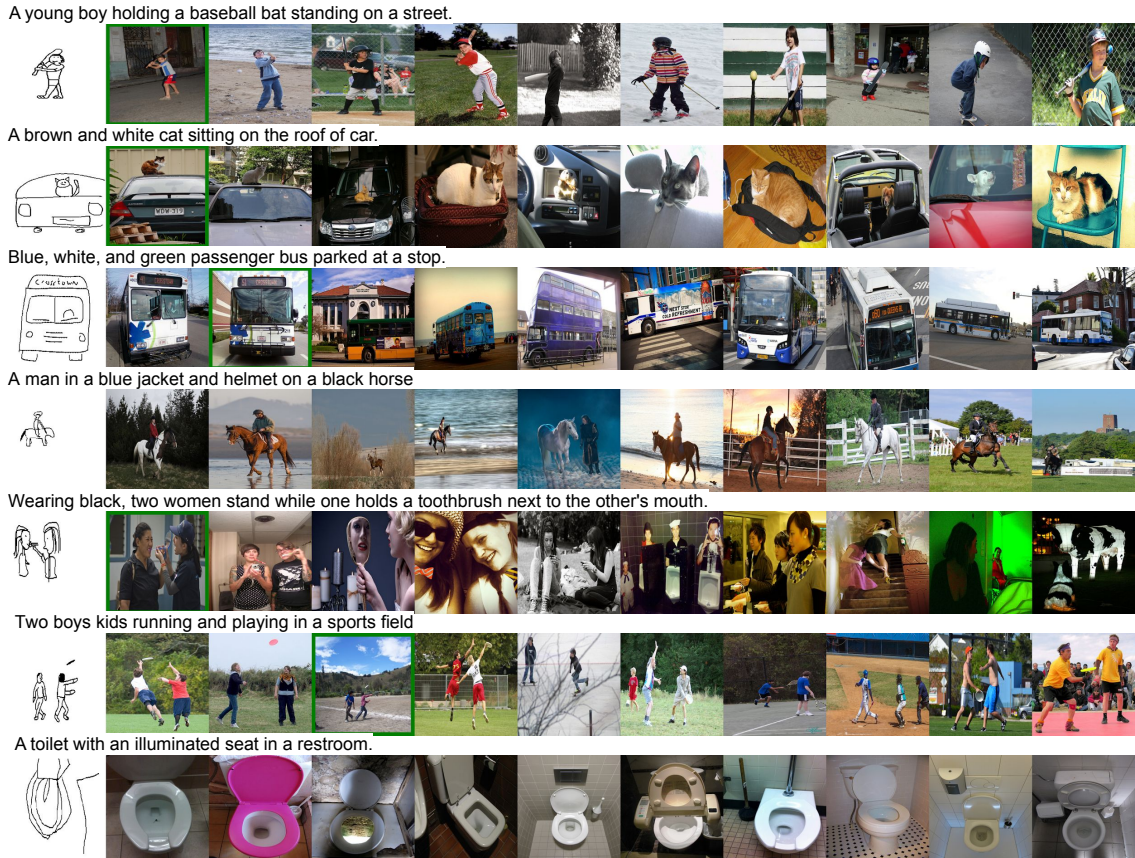


Figure 2.12: Example retrieved images from TASK-former, randomly selected from our benchmark. Each query consists of a text description (shown at the top of each block, and an input sketch (at the top left of each block). In each case, the image that forms a matching pair to the sketch is highlighted with a green border. See Section 2.4.4 for details on how we collect human drawn sketches for images in the evaluation set.

is particularly challenging for unconstrained, in-the-wild images with multiple objects and no fixed categories. In spite of that, our network can retrieve reasonably relevant results for sketch query alone, all without requiring any hand drawn sketch during training.

Nevertheless, we find that using the sketch or text as a sole input is often sub-optimal for image retrieval in this setup. Sketch or Text as a single input can only provide at best a partial description of the target image. A combination of sketch and text can provide a more complete picture of the target image, leading to an improve performance. For instance, in the first example of Figure 2.13, it can be difficult to draw a *brown* desk with sketch alone, but this can be included easily with text description.



Figure 2.13: Retrieval performance of our model when using with only sketch query, only text query, and with both queries at the same time. Our model is robust to the missing query and while not optimal, can still retrieve relevant image for each modality independently. We observe that the presence of an input sketch clearly helps rank the most relevant image higher.

2.5.3 On sketch complexity

In this section, we investigate the effect of varying the level of sketch details on the retrieval performance. We sample 300 sketches from our collected COCO hand-drawn sketches, and randomly subsample strokes to keep only 20% to 100%. Table 2.14 shows the recall performance on each level of sketch complexity. We observe a large gain near the lower end of the completeness level, with a diminish gain as the completeness level increases. In particular, this suggests that even a poorly drawn sketch helps in retrieving more relevant

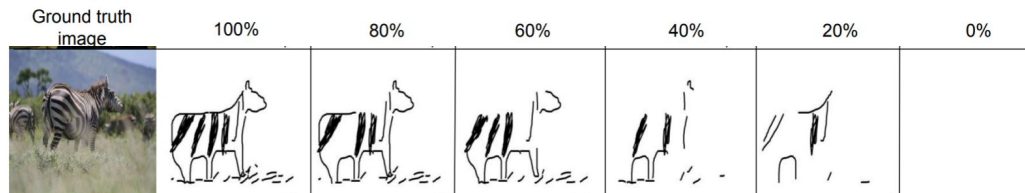


Figure 2.14: Recall@{1, 5, 10} of TASK-former, averaged over 300 randomly sampled (text query, sketch, image) tuples in the COCO dataset. For each test instance, we vary the completeness of the sketch, where completeness level $x\%$ is defined as the original sketch with only $x\%$ of all strokes randomly chosen. Candidate retrievable images are the full COCO test set of 5k. The completeness 0% corresponds to using only text as input. We observe that even with a small fraction of strokes retained (e.g., 20%), there is gain in recall compared to using only text as input (i.e., 0% sketch completeness)

Sketch completeness	100%	80%	60%	40%	20%	0%
R@1	.615	.609	.602	.582	.515	.458
R@5	.883	.876	.870	.843	.833	.776
R@10	.947	.926	.940	.923	.916	.870

images.

2.6 Limitations and future work

Figure 2.12 also shows examples of when our method fails to retrieve the target image. For example, in the forth row, the network is unable to retrieve the correct image (with blue jacket and black horse). This is likely due to the confusion with the color (both black and blue color are in the text description). While the network does place images with both black and blue color closer to the query embedding, it is unable to find the target image with black horse, and blue jacket within the top 10 results. This shows the current limitation of the model in understanding complicated text description.

Beyond these examples, we generally observe that the network suffers when the sketch is not representative of the target image. For example, the difference in scale and location can contribute to incorrect retrieval results (sixth row of Figure 2.12). Designing a model that is more tolerant to scale mismatch will be an interesting topic for future research. On the text encoder, automatically augmenting the text query to be more concrete by leveraging

text augmentation techniques also deserves further attention.

CHAPTER 3

CONTENT BASED IMAGE SYNTHESIS

In the problem of image retrieval, we assume that the target image exists in a pre-defined set of images. This, however, that is not always the case. In this chapter, we remove this assumption and attempt to generate an entirely new image that match with user queries.

3.1 Scribbler: Controlling Deep Image Synthesis with Sketch and Color

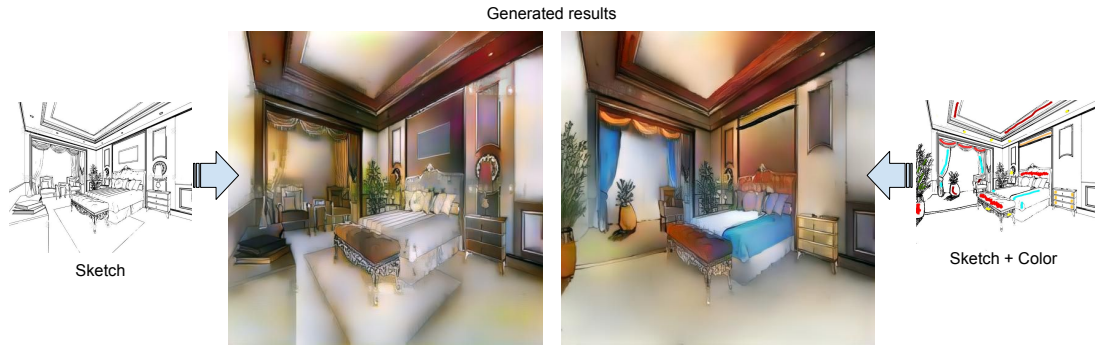


Figure 3.1: A user can sketch and scribble colors to control deep image synthesis.

Recently, numerous image synthesis methods built on neural networks have emerged [102, 103, 104, 105, 106, 107]. These methods can generate detailed and diverse (if not quite photorealistic) images in many domains. However, it is still unclear how to *control* these powerful new tools. How can we enable everyday users (non-artists) to harness the power of deep image synthesis methods and produce realistic imagery? Several recent methods have explored controllable deep synthesis [53, 108, 109, 110, 111, 112, 113] and we focus on two complementary forms of control – sketches and color strokes. Sketches are a compelling form of control because anyone can draw (potentially very badly) and because it is easy to edit sketches, e.g. to remove or add objects, whereas the equivalent operations in the image domain require artistic expertise. Color is a compelling form of control

because many sketches or grayscale scenes are fundamentally ambiguous with respect to color [114], but it is easy for a user to intervene, e.g. to scribble that drapes should be blue and the valance should be red. Both forms of control are relatively sparse and require a deep network to synthesize image detail beyond what is contained in the input. The deep network must also implicitly learn a significant amount of high-level image understanding, e.g. what colors are allowable for particular objects, the boundaries of objects such that color does not bleed beyond a single semantic region, and the appropriate high frequency textures for different scene elements.

We propose a deep adversarial (GAN) image synthesis architecture trained to generate realistic images from sparse and simple sketched boundaries and color strokes. We train our network on a diverse set of synthetic sketches optionally augmented with randomly sampled color strokes. The network learns to recover the color and detail lost to the sketching process and to extrapolate the sparse color indications to semantic scene elements. We show qualitative results of image synthesis in three domains – faces, cars, and bedrooms. We test on synthetic sketches as well as imperfect hand-drawn sketches.

Our approach is similar to Sketch Inversion [110], which also generates images from sketches, although we show the benefit of adversarial training, introduce color control signals, demonstrate results on image domains beyond faces, and demonstrate that users can perform simple edits to sketches to control the synthesis. Our control signals are most similar to Zhu et al. [111] – they also demonstrate that GANs can be constrained by sketch and color strokes. However, our architecture is a feed-forward mapping from sketch and color to images while Zhu et al. perform an optimization to map user sketches into the latent GAN space in order to find the most similar image on the natural image manifold (as understood by the GAN). Their approach does not see user inputs at training time and thus cannot learn the complex mapping between user inputs and desired image outputs. Their method is also significantly slower because it is not a strictly feed-forward process and this hinders interactive image editing. The concurrent work of Isola et al. [112] significantly

overlaps with our own. Both approaches use conditional GANs for the sketch to photo as well as grayscale to color synthesis tasks, although they do not focus on user control of the synthesis.

The contributions of this paper include:

- First and foremost, we are the first to demonstrate an adversarial deep architecture that can learn to generate realistic images from imperfect sketches with sparse color 'scribbles'. Our feed-forward architecture is fast and interactive.
- We improve the quality of sketch-to-image synthesis compared to existing work [110]. We produce higher resolution, more diverse images spanning more image domains (bedrooms and cars in addition to faces).
- We show that our method can generate realistic images from diverse sketch styles, including imperfect human sketches or edits of synthetic sketches. We achieve this generality by augmenting our training data with multiple sketch styles.
- Finally, we demonstrate that our adversarial architecture is also promising for image colorization. We show encouraging results for grayscale to RGB conversion and introduce controllable colorization using the same sparse color strokes used with sketches.

3.1.1 Related Work

Synthesizing images by learning from image collections is a long standing interest of the computer graphics and computer vision communities. Previously, the most successful methods tended to be non-parametric approaches which found clever ways to reuse existing image fragments [115, 116, 117, 118, 119].

In the last few years, *parametric* models built on deep convolutional networks have shown promising results [104, 53, 105, 106, 107]. While deep image synthesis methods cannot yet create realistic, high-resolution images they have an implicit ability to generalize that is difficult for data-driven non-parametric methods (e.g. the ability to hallucinate unseen viewpoints of particular chairs based on the appearance changes of other chairs [53]). Because our visual world is both enormously complex (with appearance depending on viewpoints, materials, attributes, object identity, lighting, etc.) *and* heavy-tailed, non-parametric methods are limited even in the “big data” era. But deep image synthesis methods might implicitly factorize our visual world and thus generalize to situations beyond the training examples.

A common approach to deep image synthesis is to learn a low dimensional latent representation that can later be used to reconstruct an image, e.g. with Variational Autoencoders (VAEs) [106] or Generative Adversarial Networks (GANs) [104]. In general, deep image synthesis can be conditioned on any input vector [113], such as attributes [108], 3d viewpoint parameters and object identity [53], image and desired viewpoint [109], or grayscale image [114, 120, 121].

Generative Adversarial Networks (GANs)

Among the most promising deep image synthesis techniques are Generative Adversarial Networks (GANs) [104, 105] in which a *generative* network attempts to fool a simultaneously trained *discriminator* network that classifies images as real or synthetic. The discriminator discourages the network from producing obviously fake images. In particular, straightforward regression loss for image synthesis often leads to ‘conservative’ networks which produce blurry and desaturated outputs which are close to the mean of the data yet perceptually unrealistic. After training, the generator network is able to produce diverse images from a low dimensional latent input space. Although optimizing in this latent space can be used to ‘walk’ the natural image manifold (e.g. for image editing [122, 111] or

network visualization [123, 124]), the space itself is not semantically well organized – the particular dimensions of the latent vector do not correspond to semantic attributes although mapping them to an intermediate structure image [125] can give us more insight.

Conditional GANs

Instead of synthesizing images from latent vectors, several works explore *conditional* GANs where the generator is conditioned on more meaningful inputs such as text [126, 127], low resolution images (super-resolution) [128, 129], or incomplete images (inpainting) [130, 131, 132]. Conditional GANs have also been used to transform images into different domains such as a product images [133] or different artistic styles [134]. Conditional GANs can also condition the *discriminator* on particular inputs, e.g. Reed et al. [126] condition both the generator and discriminator on an embedding of input text. This effectively makes the discriminator more powerful. In this paper, only our generator is conditioned on input sketches and color strokes leaving the discriminator to discern real vs fake and not to evaluate the appropriateness of an output given the particular input.

Controlling deep image synthesis

Several recent works share our motivation of adding user editable *control* to deep image generation. Examples of control signals include 3d pose of objects [53], natural language [126], semantic attributes [108], semantic segmentation [135], and object keypoints and bounding box [127].

The artistic style transfer approach of Gatys et al. [136] could also be considered a mechanism to control deep image synthesis. Their method does not ‘learn’ transformations end-to-end but instead uses a pre-trained network and optimizes for output images which have deep network feature *activations* (content) similar to one input image and deep network feature *correlations* (style) similar to another input image. The approach does not perform well for transformations which requires the synthesis of realistic detail (e.g. trying

to preserve the ‘content’ of a sketch and the ‘style’ of a photograph).

The most similar previous deep image synthesis approach in terms of *control* is Zhu et al. [111] which optimizes for an image that is similar to an input sketch (potentially with color strokes) that lies on a learned natural image manifold. However, identifying a matching image within this manifold that is similar content-wise to the sketch can be challenging when the sketch and the image are significantly different. For the ‘sketch brush’ in [111], they get around this by optimizing for image with the same edges as user sketch that also lies within a natural image manifold as approximated by a pre-trained GAN. However, image edges are not necessarily a good proxy for human sketched strokes [1] and their method has no capacity to *learn* the mapping between user inputs and desired outputs. In contrast, our method enables control via sketch and color strokes in a unified framework learned end-to-end. Sketch Inversion [110] is also closely related to our work although they do not address color control. We compare our sketch-to-photo results with Sketch Inversion.

Controllable Colorization

Our color control strokes are inspired by Colorization Using Optimization [137] which interpolates sparse color strokes such that color changes tend to happen at intensity boundaries. The algorithm does not learn the association between objects and colors and thus can only interpolate user provided colors (e.g. a tree in the background of a scene will not be green if the user only marked foreground objects). The algorithm also does not learn the spatial extent of objects, and thus colors might ‘snap’ to spurious boundaries or bleed over weak intensity edges that are none-the-less salient boundaries. Our deep network learns object color tendencies and object extent and thus can cleanly color objects either with no color strokes or with color strokes on a subset of scene elements. Similar control strokes have been applied to sketch and manga imagery [138, 139], but the results remain non-photorealistic and lack lighting and shading.

We are unaware of sparse scribbles being used as input constraints to deep generative

networks, although ScribbleSup [140] uses sparse scribbles to *supervise* the output of semantic segmentation networks. The scribbles are training data and there is no user control at test time.

Concurrent work

Concurrent to our work at the time of writing, the ‘pix2pix’ method of Isola et al. [112] also uses conditional GANs for sketch to photo and grayscale to color synthesis. Additionally, they explore several other interesting image-to-image ‘translation’ tasks. Unlike our approach, they use a “U-Net” architecture [141] which allows later layers of the network to be conditioned on early layers where more spatial information is preserved. They condition both their generator and discriminator on the input whereas we condition only the generator. Their results are high quality and they are able to synthesize shoes and handbags from coarse sketches [8] even though their training data was simple image edges. In contrast, we take care to train on a diversity of synthetic sketch styles. The most significant difference between our works is that we introduce sparse color control strokes and demonstrate how to train a network so that it learns to intelligently interpolate such control signals, whereas Isola et al. [112] does not emphasize controllable synthesis.

3.1.2 Overview of Scribbler

In this paper, we explore adding direct and fine-grained user controls to generative neural networks. We propose a generic feed-forward network that can be trained end-to-end to directly transform users’ control signals, for example a hand-drawn sketch and color strokes, to a high-res photo with realistic textural details.

Our proposed network is essentially a deep generative model that is conditioned on control signals. The network learns a transformation from control signal to the pixel domain. It learns to fill in missing details and colors in a realistic way. Section 3.1.2 discusses the network structure that is shared by all applications presented in the paper. Section 3.1.2

introduces the objective functions, in particular the combination of content loss and adversarial loss, which encourages the result to be photo-realistic while satisfying user’s fine-grained control. Section 3.1.3 and 3.1.4 show how to enforce two different user controls in the proposed framework – using hand-drawn sketches to determine the gist or shape of the contents and using sparse color strokes to propagate colors to semantic regions. Section 3.1.5 applies the proposed framework in several interactive applications.

Network Architecture

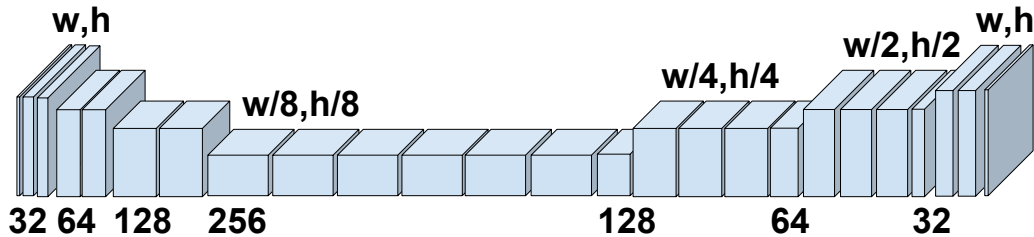


Figure 3.2: Network Architecture. Following the encoder-decoder design, we use three downsampling steps, seven residual blocks at the bottleneck resolution and three upsampling steps. Residual blocks use stride 1. Downsampling uses convolutions with stride 2. Upsampling uses bilinear upsampling followed by residual blocks.

We design a feed-forward neural network that takes an image as input and generates a photo of the same resolution as output. When generating an image conditioned on a high dimensional input in the same domain (i.e. from image to image), typically an encoder-decoder type of network architecture is adopted, for example in sketch inversion [110], image colorization [114, 120], and sketch simplification [142]. In a typical network structure, the input gets downsampled several times to a lower dimension, then goes through a sequence of non-linear transformations, and finally gets upsampled to the desired output size. Recently, He et al. [143] proposed the residual connection that uses skip layers allowing network blocks to learn only the *residual* component. The use of residual block eases the training of deeper networks which improves the capability of neural network for more

complex tasks.

We employ an encoder-decoder architecture with residual connections. Starting from the network design in Sketch Inversion [110], we introduce several important modifications to improve the visual quality of output and accommodate higher resolution input and more challenging image categories, such as car and bedroom. In particular, we add one more up/downsampling layer and double the number of filters in all convolutional layers between the last downsampling layer and the first upsampling step. In addition, we replace the deconvolutional layers with the bilinear upsampling step followed by two residual blocks, due to the recent finding that deconvolutional layers have the tendency to produce checkerboard artifacts commonly seen in deep generative models [144]. Overall, our architecture has around 7.8 millions learnable parameters, while the Sketch Inversion network we implemented has around 1.7 millions. See Figure 3.2 for a diagram of our architecture.

Objective Function

Given pairs of training images (input, ground-truth), where the input image is derived from the ground-truth photo (synthetically generated sketches and color strokes in our case), the simplest and most common loss is the average per-pixel L2 difference between the generated image and the ground-truth, which we denote as L_p .

Previous work [110] showed that adding a *feature* loss to the objective function is beneficial for image generation tasks. Feature loss L_f is defined as the L2 difference in a feature space, where a feature is extracted from a certain layer of a pre-trained neural network representing high-level information of images.

While pixel and feature losses are widely used to explicitly correlate synthesized output with input, using them alone is often not sufficient to generate diverse, realistic images. More importantly, in our problem setup, conditioning on coarse user controls leaves us with a highly ill-posed problem where the potential solution space is multimodal. Therefore, with only pixel and feature losses, the network tends to average over all plausible solutions,

due to the lack of a loss which pushes for realism and diversity.

For image categories like face, the generated results tend to have similar skin tones [110]. For more complicated categories like cars and bedrooms, where the foreground and background contents can have large variety of shapes and colors, the generated results might not be visually plausible, since neutral colors are chosen by the network to minimize MSE. The second and third rows in Figure 3.3 demonstrate the problems.

To encourage more variations and vividness in generated results, we experiment with adding an adversarial loss to the objective function. Generative adversarial networks (GAN), proposed by Goodfellow et al [104], have attracted considerable attention recently. A generative network G_θ is jointly trained with a discriminative adversarial network D_ϕ , so that the discriminator tries to distinguish between the generated images and ground-truth images, while the generator tries to fool the discriminator into thinking the generated result is real. Dosovitskiy et al [145] showed that complimenting the feature loss with an adversarial loss leads to more realistic results. The adversarial loss L_{adv} is defined as:

$$L_{adv} = - \sum \log D_\phi(G_\theta(x_i)) \quad (3.1)$$

We find that adversarial loss is also beneficial for our sketch-based image synthesis problem (Figure 3.3). With adversarial training, the network puts less emphasis on exactly reproducing ground-truth, but instead focuses on generating more realistic results with plausible color and shape deviation from ground-truth.

Adversarial training tends to be unstable, especially at the start of training when the generator does not produce anything meaningful and the discriminator can easily distinguish between real and fake. We find that using a weak discriminator D_ϕ helps stabilize the training. We also avoided conditioning the discriminator on the input image, as this tends to increase the instability [130]. In particular, we use a fully convolutional structure without fully connected layers and batch normalization. Section 3.1.6 introduces additional



Figure 3.3: Results comparison. From top to bottom: input sketch, Sketch Inversion with content loss, our network with content loss, our network with content loss and adversarial loss

tricks for successful adversarial training.

Finally, we also add a total variation loss L_{tv} to encourage smoothness in the output [146].

Our final objective function becomes:

$$L = w_p L_p + w_f L_f + w_{adv} L_{adv} + w_{tv} L_{tv} \quad (3.2)$$

3.1.3 Sketch-Based Image Synthesis

In this section, we explore how to apply the proposed feed-forward network to hallucinate content, color and texture to reconstruct a photo based on an input sketch of arbitrary style. To train such a deep neural network, we need lots of training sketch-photo pairs. Though high quality hand-drawn sketches are readily available online, the corresponding photos based on which sketches are drawn are not. Therefore, we apply high-quality line drawing synthesis algorithms to generate synthetic sketches from photos. In order to handle real hand-drawn sketches at test time, we apply various data augmentations to the training data to improve the generality of the network. In this paper, we experiment with three image classes – faces [147], cars, and bedrooms [148]. We believe the proposed framework can generalize well to other categories given similar amounts of training data and training time.

Generation of Training Sketches

For each image category – face, car, or bedroom – we apply the boundary detection filter XDoG [149] on 200k photos to generate the corresponding synthetic sketches. The input (and output) resolution to our network during the training phase is 128x128.

To make the network invariant to the exact locations of the objects, we randomly crop both the input and the ground-truth images. For the face and bedroom categories, we first resize the images to 256x256 before randomly cropping them to 128x128. For the car category, we scale the images to 170x170 before cropping, since most cars already occupy large image areas, enlarging them too much means losing the global spatial arrangement and the contexts around the cars.

In addition to randomly cropping an image and its corresponding sketch, we also randomly adjust the brightness level of the sketch to get different levels of details from the same sketch (i.e. some sketch lines will disappear with higher brightness level). Finally, we also randomly cut off some lines in the sketch, by overlaying a random number of white strokes (the background color of sketch input) on top of the sketch. We randomize the length, width and locations of the white strokes.

Network Generalization

Real hand-drawn sketches exhibit a large variety of styles, from abstract pen-and-ink

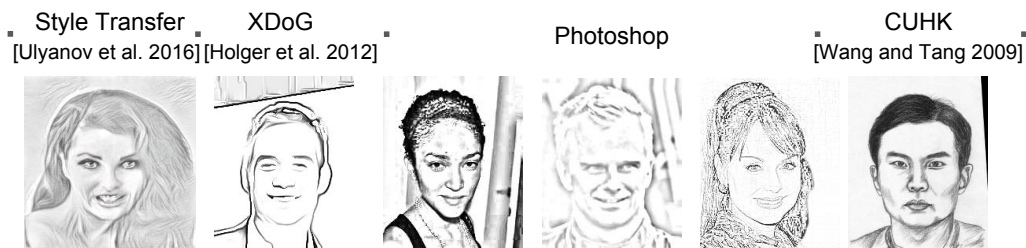


Figure 3.4: We generate synthetic sketches from photos using five different algorithms. We also include and augment a small set of hand-drawn sketch-photo pairs to help generalize the network to handle real hand-drawn sketch inputs.

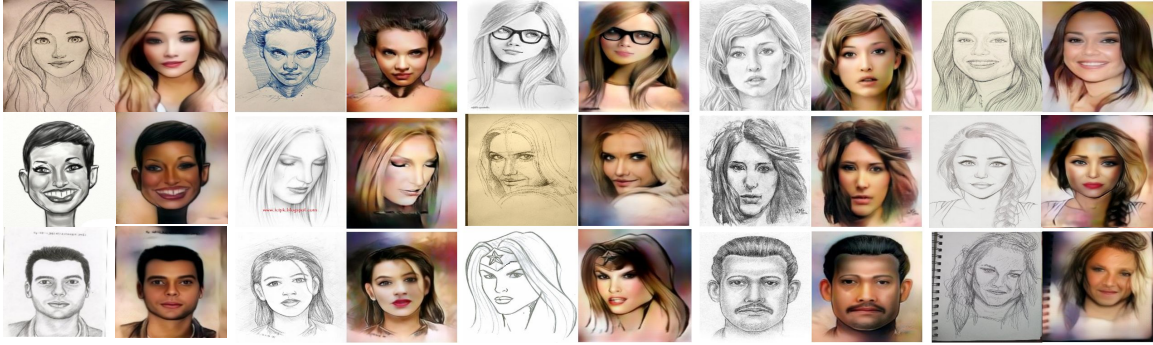


Figure 3.5: Sketch-based photo synthesis of hand-drawn test sketches. Despite the diverse sketch styles, our network usually produces high quality, diverse results. Notice the skin tone and hair color variations in the output. Some outputs are non-photorealistic because they are being somewhat faithful to caricatured input sketches. Unfortunately, some results have unrealistically high low-frequency contrast and appear unnaturally lit.

illustrations to elaborate pencil-like drawings with shading. The characteristics of the hand-drawn sketches might be very different from the synthetic sketches we generated algorithmically. Even with the various augmentations, random cropping, random brightness adjustment and random cut-off, the trained network might still overfit to that particular style of sketches. To improve the network generality, we further augment the training data by adding multiple styles of sketches.

For the face category, we obtain 20k additional images and for each image we randomly choose one of the following four algorithms to synthesize a corresponding sketch. See example sketches in Figure 3.4.

- **StyleNet** [136] We apply neural network-based style transfer algorithm to transfer the texture style of a pencil drawing to the ground-truth photo.
- **Photoshop filters** [150] Applying Photoshop’s ‘photocopy’ effect to ground-truth images, we can generate two different versions of sketches with different levels of details and stroke darkness.
- **Gaussian blur on inverse image** [151] Using Photoshop, we can also synthesize another sketch style by performing Gaussian blur on an inverse (grayscale) image

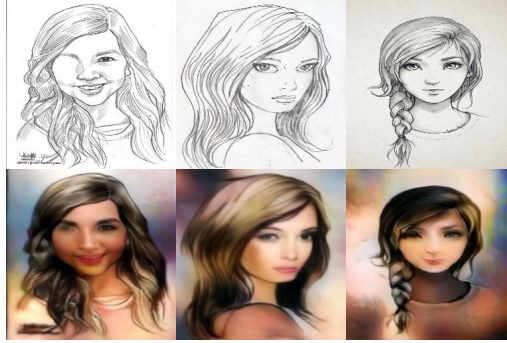


Figure 3.6: Interesting network behavior. Left: the network hallucinates the missing eye. Middle: the adversarial training reduces the size of the eyes to more realistic proportions. Right: The network curves the lips upward changing the overall expression to a subtle smile.

in Photoshop color dodge mode. This creates detailed line drawings with very little shading.

- **CUHK** Finally, we add the CUHK dataset, which contains 188 hand-drawn portrait sketches and their corresponding photos [152]. To give higher weights to the high quality hand-drawn sketches, we apply mirroring and varying degrees of rotation to the sketches and end up with 1869 images in total.

At this point, we have 21848 images of 6 different sketch styles. Pre-trained on the 200k sketches of the XDoG style, the network is fine-tuned using the 20k multi-style sketches. We use the same parameter settings as before and train the network on these additional data for 5 epochs.

Results and Discussions

For comparison purposes, we implemented the Sketch Inversion architecture as described in [110]. We trained both the Sketch Inversion network and our deeper network using the same training data and parameter settings. Figure 3.3 shows side-by-side comparisons of the results generated by Sketch Inversion (second row), our deeper network

trained without (third row) and with adversarial loss (fourth row) on three different image categories. Compared to Sketch Inversion, our deeper network even without adversarial loss produces sharper results on complex bedroom scenes and performs better at hallucinating missing details (shapes of eyes and eyebrows) given simplified sketches with few lines. With adversarial loss, our network is encouraged to generate images with sharper edges, higher contrast and more realistic color and lighting. As discussed in Section 3.1.2, adversarial loss helps the network generate more diversified results, avoiding always producing similar skin tones and hair colors for portraits and dull and unrealistic colors for the bedrooms and cars. Figure 3.5 shows diverse hair colors and skin tones in the result.

Among the three image categories, bedroom is arguably most challenging, since each bedroom scene can contain multiple object categories. The fact that our current network handles it successfully with only 200K training data leads us to believe the possibility of training a general sketch-to-photo network across several image categories using an even deeper network.

After training with multiple sketch styles and various data augmentations (Section 3.1.3), our network generates much more realistic results given arbitrary hand-drawn sketches as input. Figure 3.5 shows reconstruction results based on sketches found by Google search. Note that the sketches are drawn with diverse styles, some detailed and realistic, some abstract and simplified. The results show that our network generalizes well to arbitrary hand-drawn sketches and is robust to the variations in head pose, background colors, and textures. Data augmentation such as random cropping and cutoff also helps our network hallucinate missing details. Figure 3.6 (left) shows that the network can fill in the missing eye to some extent. However, generating missing object parts is a challenge itself, therefore we consider it beyond the scope of this paper.

The network trained with adversarial loss has an interesting behavior. When applying it to cartoonish or unprofessional sketches with intentional or unintentional exaggeration of facial features, the network tends to ‘realistify’ or beautify the input sketch to generate

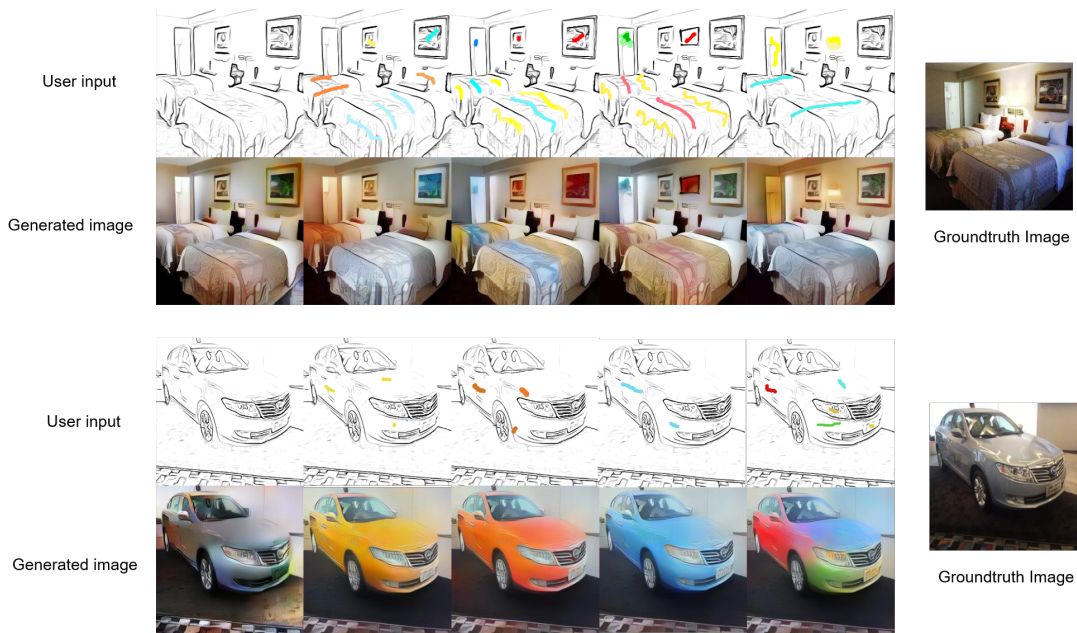


Figure 3.7: Guided sketch colorization results on held out test sketches. Without color strokes, our network produces synthesis results that closely follow the input sketch (left-most column). With color strokes, our network can adapt the synthesis results to satisfy different color constraints. Note that the fourth and fifth bedroom sketches are edited to add a framed picture on the wall and a lamp using only simple user edits.

result more photo-like at the cost of not strictly following the sketch constraints. For example, eyes that are inhumanly large will get reduced to a realistic size or faces with weird shapes will be smoothed and ‘beautified’ (see Figure 3.6). To produce realistic results, the network learns not to blindly trust the sketch input, but resorts to its understanding about the natural image manifold acquired during the adversarial training.

3.1.4 User-Guided Colorization

The previous section focuses on using a gray-scale sketch to guide the generation of a color photos. The lack of color information in the input causes the problem to be under-determined, since one sketch can correspond to photos colored in many different ways.

Although the use of adversarial loss constrains the output to lie on an approximated manifold of natural images and therefore limits the color choices, it is still up to the generator (and the discriminator) to choose a specific color.

In this section, we explore how to allow users to directly control the colors in the output. To do that, we need to modify the input to the network to include rough color information during training (Section 3.1.4). We investigated adding color controls in two applications, guided sketch colorization (Section 3.1.4) and guided image colorization (Section 3.1.4).

Generation of Training Color Strokes

One of the most intuitive ways to control the outcome of colorization is to ‘scribble’ some color strokes to indicate the preferred color in a region

To train a network to recognize these control signals at test time, we need to synthesize color strokes for the training data. We generate synthetic strokes based on the colors in the ground-truth image.

To emulate arbitrary user behaviors, we blur the ground-truth image and sample a random number of color strokes of random length and thickness at random locations. We pick the ground-truth pixel color at the stroke starting point as the stroke color and continue to

grow the stroke until the maximum length is reached.

When growing a stroke, if the difference between the current pixel color and the stroke color exceeds a certain threshold, we restart the stroke with a new color sampled at the current pixel. By randomizing various stroke parameters, we are able to synthesize color strokes similar to what human would draw during test time.

Guided Sketch Colorization

The goal here is to add color control to our sketch-based image synthesis pipeline. Our previous objective function still holds: we want the output to have the same content as the input (pixel and feature loss), and appear realistic (adversarial loss). Pixel loss is essential here as it forces the network to be more precise with color by paying more attention to the color strokes. We modify the training data by placing color strokes on top of the input sketches. We then train the network as before using a parameter setting that emphasizes content loss and de-emphasizes adversarial loss, so that the results better satisfy color constraints (Section 3.1.6).

Figure 3.7 shows the results of reconstructing bedroom and car scenes based on an input sketch and color strokes. Note that the colors of the strokes deviate a lot from the colors in the ground-truth image, nevertheless, the network is able to propagate the input color to the relevant regions respecting object boundaries. In the bedroom scene (two rightmost columns), based on the crude outline of a picture frame and a yellow lamp, our network successfully generates plausible details in the results. See more results in the supplementary material.

Guided Image Colorization

Recent work [120, 114] explores training deep neural network models for the image colorization tasks. However, the selection of colors in the output is entirely up to the network. In this section, we investigate using color strokes (Section 3.1.4) to guide the colorization



Figure 3.8: Guided Image Colorization: a) grayscale input, b) original color image, c) deep colorization result [114], d) First and third rows: color strokes overlaid on top of the grayscale input (zoom in to see the color strokes). Second and fourth rows: colorization results.

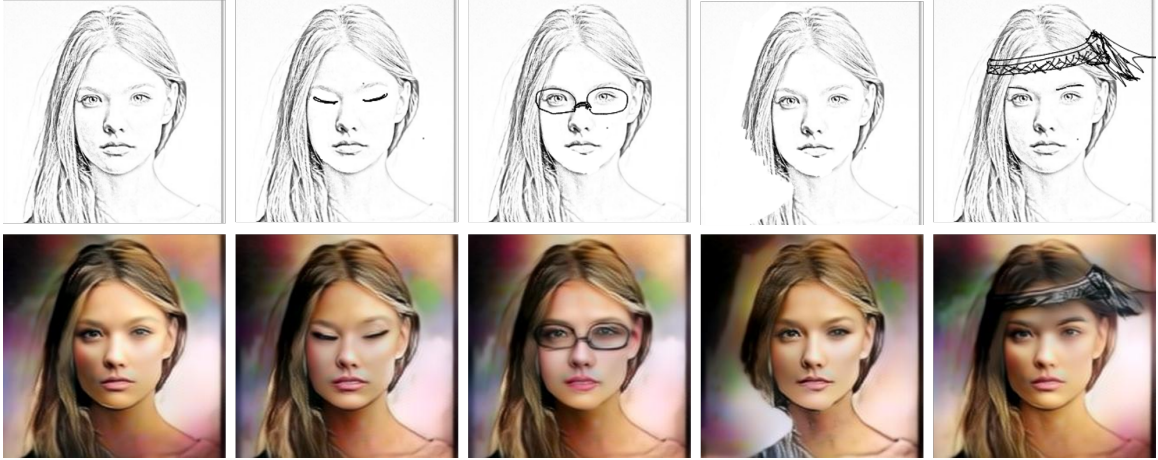


Figure 3.9: Interactive image generation and editing. The user can incrementally modify the sketch to change the eyes, hair, and head decorations.

process. We generate training data by extracting a one-channel grayscale image from the ground-truth photo and combining it with the three-channel image containing color strokes.

Figure 3.8 shows various colorization results on a car image. Given a gray-scale image, our system synthesizes realistic looking cars based on strokes drawn with different colors at random locations. Note that most strokes are placed on the body of the car and therefore do not influence the colorization of the other regions. Due to adversarial training, the sky is colored blue and the trees are colored green, regardless of the colors of the foreground object. Internally, the network learns to recognize semantic contents and therefore can put right colors in the relevant regions while at the same time satisfying user constraints wherever possible.

3.1.5 Applications

We believe being able to control the generated output using sketch and color allows for many useful applications, especially in the artistic domain.

Interactive Image Generation Tools

Given an input image of resolution 256x256, our network takes about 20ms to transform

it to a photo-like result. The real-time performance enables instant visual feedback after incremental edits in image generation and editing applications.

Using sketch and color strokes to enforce fine-grained control is useful for several design applications. For example, an interior designer can quickly sketch out rough shapes of the objects, specify colors in various regions and let our system fill in missing details and textures to generate a plausible bedroom scene. After seeing the result, the designer can interactively modify the shapes and colors of the objects and receive instant visual feedback. Figure 3.7 illustrates the potential design workflow. Similarly, a car designer can follow similar workflow to design new cars and test out the looks in different background settings.

Our portrait synthesis system provides a tool for artists to design virtual characters (see Figure 3.9). Based on the initial design, one can change the shape of eyes and/or hairstyle, add glasses and/or head decorations, etc. In addition to design, portrait reconstruction technology is useful for forensic purposes, for example the law enforcement department can use it to help identify suspects [110].

Sketch and Color Guided Visual Search

Our image generation tools provide flexible ways to perform visual search. With a target scene in mind, one can sketch out the object boundaries and color constraints, based on which our network can reconstruct a plausible arrangement. The reconstructed image can then be used in a typical visual search tool to identify high-res images with similar contents (see Figure 3.10).

3.1.6 Network Training Details

With the unpredictable nature of adversarial training, we find it helpful to separate the training into two stages.

Optimizing for Content Loss



Figure 3.10: Given the reconstructed images (leftmost column), Google’s visual search engine retrieves visually similar photos.

In the first stage, we set the adversarial weight w_{adv} from equation 3.2 to 0 and let the network focus on minimizing the content loss which is a combination of pixel and feature loss. To enforce a fine-grained control using the input sketch, we choose the ReLU2-2 layer of the VGG-19 net [153] to compute the feature loss, since higher level feature representations tend to encourage the network to ignore important details such as the exact locations of the pupils.

We set the weights of pixel loss and feature loss w_p , w_f to 1, and the weight of TV loss w_{tv} to $1e-5$. We train the network for around 3 epochs using a batch size of 32 before moving on to the second stage of the training.

Adding Adversarial Loss

Given the network pretrained for content loss, we fine tune it with different loss settings for different applications. For photo reconstruction from gray-scale sketches (Section 3.1.3), we turn off the pixel loss, keep the feature loss and add the adversarial loss with

the following weight setting, $w_f = 1$, $w_p = 0$, $w_{tv} = 0$, $w_{adv} \approx 1e8$. For colorization applications (Section 3.1.4), we emphasize the feature and pixel loss and de-emphasize the adversarial loss, so that the output better follows the color controls, $w_f = 10$, $w_p = 1$, $w_{tv} = 0$, $w_{adv} \approx 1e5$.

We train the adversarial discriminator alongside our generative network for three epochs using a learning rate between $1e-5$ and $1e-6$.

3.1.7 Conclusion and Future Work

In this paper, we propose a deep generative framework that enables two types of user controls to guide the result generation – using sketch to guide high-level visual structure and using sparse color strokes to control object color pattern.

Despite the promising results, our current system suffers from several limitations. First, we sometimes observe blurry boundaries between object parts or regions of different colors which diminish the overall realism of the results.

Figure 3.7 shows the color leaking problem on the car results, where the color of the car’s hood leaks into the background. Second, our system struggles between strictly following color/sketch controls and minimizing adversarial loss. In other words, adversarial loss prohibits the generated images from taking uncommon colors and shapes. If the user specifies a rare color, for example, purple for car, red for trees, our network will map it to a different color deemed more realistic by the adversarial loss. Third, the network sees objects of similar scale during training, and would expect to see the same scale at testing. As future work, we can add multi-scale support to the network by randomizing the ratio between the cropping size and the image size during training.

3.2 TextureGAN: Controlling Deep Image Synthesis with Texture Patches



Figure 3.11: Top row: Sketch with texture patch overlaid. Bottom row: Results from TextureGAN.

One of the “Grand Challenges” of computer graphics is to allow *anyone* to author realistic visual content. The traditional 3d rendering pipeline can produce astonishing and realistic imagery, but only in the hands of talented and trained artists. The idea of short-circuiting the traditional 3d modeling and rendering pipeline dates back at least 20 years to image-based rendering techniques [154]. These techniques and later “image-based” graphics approaches focus on re-using image content from a database of training images [155]. For a limited range of image synthesis and editing scenarios, these non-parametric techniques allow non-experts to author photorealistic imagery.

In the last two years, the idea of direct image synthesis without using the traditional rendering pipeline has gotten significant interest because of promising results from deep network architectures such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). However, there has been little investigation of fine-grained *texture* control in deep image synthesis (as opposed to coarse texture control through “style transfer” methods).

In this paper we introduce TextureGAN, the first deep image synthesis method which allows users to control object texture. Users “drag” one or more example textures onto sketched objects in a scene and the network realistically applies these textures to the indicated objects.

This “texture fill” operation is difficult for a deep network to learn for several reasons:

- (1) Existing deep networks aren’t particularly good at synthesizing high-resolution texture details even without user constraints. Typical results from recent deep image synthesis methods are at low resolution (e.g. 64x64) where texture is not prominent or they are higher resolution but relatively flat (e.g. birds with sharp boundaries but few fine-scale details).
- (2) For TextureGAN, the network must learn to propagate textures to the relevant object boundaries – it is undesirable to leave an object partially textured or to have the texture spill into the background. To accomplish this, our network must implicitly segment the sketched objects and perform texture synthesis, tasks which are individually difficult.
- (3) The network should additionally learn to foreshorten textures as they wrap around 3d object shapes, to shade textures according to ambient occlusion and lighting direction, and to understand that some object parts (handbag clasps) are not to be textured but should occlude the texture. These texture manipulation steps go beyond traditional texture synthesis in which a texture is assumed to be stationary. To accomplish these steps the network needs a rich implicit model of the visual world that involves some partial 3d understanding.

Fortunately, the difficulty of this task is somewhat balanced by the availability of training data. Like recent unsupervised learning methods based on colorization [114, 121], training pairs can be generated from unannotated images. In our case, input training sketches and texture suggestions are automatically extracted from real photographs which in turn serve as the ground truth for initial training. We introduce *local* texture loss to further fine-tune our networks to handle diverse textures unseen on ground truth objects.

We make the following contributions:

- We are the first to demonstrate the plausibility of fine-grained texture control in deep image synthesis. In concert with sketched object boundaries, this allows non-experts to author realistic visual content. Our network is feed-forward and thus can run interactively as users modify sketch or texture suggestions.

- We propose a “drag and drop” texture interface where users place particular textures onto sparse, sketched object boundaries. The deep generative network directly operates on these localized texture patches and sketched object boundaries.
- We explore novel losses for training deep image synthesis. In particular we formulate a local texture loss which encourages the generative network to handle new textures never seen on existing objects.

3.2.1 Related Work

Image Synthesis.

Synthesizing natural images has been one of the most intriguing and challenging tasks in graphics, vision and machine learning research. Most of the existing approaches fall into non-parametric and parametric types. On one hand, non-parametric approaches have a long-standing history. They are typically data-driven or example-based, i.e., directly exploit and borrow existing image pixels for the target tasks [119, 4, 156, 157, 154]. Therefore, non-parametric approaches often excel at generating realistic results while having limited generalization ability, i.e., being restricted by the limitation of data and examples, e.g., data bias and long-tail distribution. On the other hand, parametric approaches, especially deep learning based approaches, have achieved promising results in recent years. Different from non-parametric methods, these approaches utilize image datasets as training data to fit deep parametric models, and have shown superior modeling power and generalization ability in image synthesis [104, 106], e.g., hallucinating diverse and relatively realistic images that are different from training data.

Generative Adversarial Networks (GANs) [104] are a type of parametric method that has been widely applied and studied for image synthesis. The main idea is to train paired generator and discriminator networks jointly, where the goal of the discriminator is to classify between ‘real’ images and generated ‘fake’ images, and the generator aims to fool the discriminator so that the generated images are indistinguishable from real images. Once

trained, the generator can be used to synthesize images driven by a compact vector of noise. Compared to the blurry and low-resolution outcome from other deep learning methods [106, 53], GAN-based methods [105, 158, 159, 160] generate more realistic results with richer local details and of higher resolution.

Controllable Image Synthesis and Conditional GANs.

Practical image synthesis tools require perceptually controllable interfaces, ranging from high-level attributes, such as object classes [161], object poses [53], natural language descriptions [162], to fine-grained details, such as segmentation masks [159], sketches [2, 110], color scribbles [2, 163], and cross-domain images [164, 133]. While the ‘vanilla’ GAN is able to generate realistic looking images from noise, it is controllable. *Conditional* GANs are models that synthesize images based on input modalities other than simple noise, thus offering more control over the generated results. Compared to vanilla GANs, conditional GANs introduce additional discriminators or losses to guide generators to output images with desired properties, e.g., an object category discriminator [161], a discriminator to judge visual-text association [162], or a simple pixel-wise loss between generated images and target images [159].

It is worth highlighting several recent and concurrent works on sketch or color-constrained deep image synthesis. Scribbler [2] demonstrates an image synthesis framework that takes as input user sketches and short color strokes, and generates realistic looking output that follows the input sketch and has colorization schemes consistent to color strokes. A similar system is employed for automatically painting cartoon images [165]. Recently, a user-guided interactive image colorization system was proposed in [163], offering users the control of color when coloring or recoloring an input image. Distinct from these works, our system simultaneously supports richer user guidance signals including structural sketches, color patches and texture swatches. Moreover, we offer studies on the effect of several variants of improved loss functions and show synthesized results of various object categories.

Texture Synthesis and Style Transfer.

Texture synthesis and style transfer are two closely related topics in image synthesis. Given an input texture image, texture synthesis aims at generating new images with visually similar textures. Style transfer has two inputs – *content* and *style* images – and aims to synthesize images with the layout and structure of the content image and the texture of the style image. Non-parametric texture synthesis and style transfer methods typically re-sample given example images to form the output [156, 166, 167, 168]. TextureShop [169] is similar to our method in that it aims to texture an object with a user-provided texture, although TextureShop used non-parametric texture synthesis and shape-from-shading to foreshorten the texture so that it appears to follow the object surface.

Recently, a new deep style transfer method [170, 164] demonstrated that the correlations (i.e., Gram matrix) between features extracted from a pre-trained deep neural network capture the characteristics of textures well and showed promising results in synthesizing textures and transferring styles. In [170, 164], texture synthesis and style transfer are formalized as an optimization problem, where an output image is generated by minimizing a loss function of two terms, one of which measures content similarity between the input content image and the output, and the other measures style similarity between the input style and the output using the Gram matrix. Shortly after, there have been many work on improving [170, 164] from the aspects for generalization [171, 172, 173], efficiency [174, 175] and controllability [176].

Recently, several texture synthesis methods have used GANs to improve the quality of the generated results. [134] uses adversarial training to discriminate between real and fake textures based on a feature patch from the VGG network. Instead of operating on feature space, [177, 178] apply adversarial training at the pixel level to encourage the generated results to be indistinguishable from real texture. Our proposed texture discriminator in Section 3.2.2 differs from prior work by comparing a *pair* of patches from generated and

ground truth textures instead of using a single texture patch. Intuitively, our discriminator is tasked with the fine-grained question of “is this the same texture?” rather than the more general “is this a valid texture?”. Fooling such a discriminator is more difficult and requires our generator to synthesize not just realistic texture but also texture that is faithful to various input texture styles.

Similar to texture synthesis, image completion or inpainting methods have also showed promising results using GANs. Our task has similarities to the image completion problem, which attempts to fill in missing regions of an image, although our missing area is significantly larger and partially constrained by sketch, color, or texture. Similar to our approach, [179] computes texture loss between patches to encourage the inpainted region to be faithful to the original image regions. However, their texture loss only accounts for similarity in feature space. Our approach is similar in spirit to [180] which proposes using both global and local discriminators to ensure that results are both realistic and consistent with the image context, whereas our texture discriminator is instead checking consistency between input texture patch and output image.

3.2.2 TextureGAN

We seek an image synthesis pipeline that can generate natural images based on an input *sketch* and some number of user-provided *texture* patches. Users can provide rough sketches that outline the desired objects to control the generation of semantic content, e.g. object type and shape, but sketches do not contain enough information to guide the generation of texture details, materials, and patterns. To guide the generation of fine-scale details, we want users to somehow control texture properties of objects and scene elements.

Towards this goal, we introduce **TextureGAN**, a conditional generative network that learns to generate realistic images from input sketches with overlaid textures. We argue that instead of providing just an unanchored texture sample, users can more precisely control the generated appearance by directly placing small texture patches over the sketch,

since locations and sizes of patches provide important information that influence the visual appearance. In this setup, the user can ‘drag’ rectangular texture patches of arbitrary sizes into different sketch regions as additional input to the network. For example, the user can specify a striped texture patch for a shirt and a dotted texture patch for a skirt. The input patches guide the network to propagate the texture information to the relevant regions respecting semantic boundaries (e.g. dots should appear on the skirt but not on the legs).

A major challenge for a network learning this task is the uncertain pixel correspondence between the input texture and the unconstrained sketch regions. To encourage the network to produce realistic textures, we propose a patch-based texture loss 3.2.2 based on a texture discriminator and a Gram matrix style loss. This not only helps the generated texture follow the input faithfully, but also helps the network learn to propagate the texture patch and synthesize new texture.

TextureGAN also allows users to more precisely control the colors in the generated result. One limitation of previous color control with GANs [2] is that the input color constraints in the form of **RGB** need to fight with the network’s understanding about the semantics, e.g., bags are mostly black and shoes are seldom green. To address this problem, we train the network to generate images in the **Lab** color space. We convert the groundtruth images to **Lab**, enforce the content, texture and adversarial losses only on the **L** channel, and enforce a separate color loss on the **ab** channels. We show that combining the controls in this way allows the network to generate realistic photos closely following the user’s color and texture intent without introducing obvious visual artifacts.

Figure 3.12 shows our training pipeline. We use the network architecture proposed in Scribbler [2] with additional skip connections. Details of our network architecture are included in the supplementary material. We use a 5-channel image as input to the network. The channels support three different types of controls – one channel for sketch, two channels for texture (one intensity and one binary location mask), and two channels for color (including but not limited to texture color). Section 3.2.3 describes the method we used to

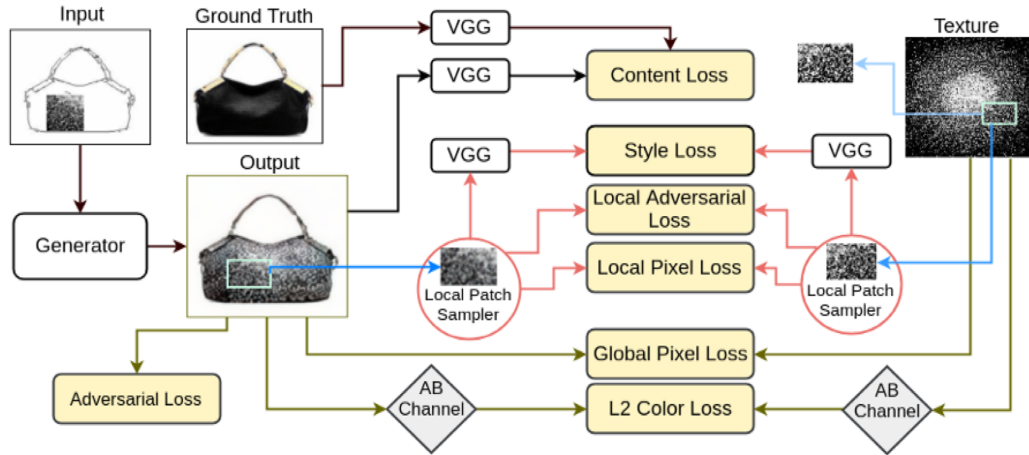


Figure 3.12: TextureGAN pipeline. A feed-forward generative network is trained end-to-end to directly transform a 5-channel input to a high-res photo with realistic texture details. The red arrows indicate losses that are active for “texture fine tuning”. See text for detailed description of various losses.

generate each input channel of the network.

We first train TextureGAN to reproduce ground-truth shoe, handbag, and clothes photos given synthetically generated input control channels. We then generalize TextureGAN to support a broader range of textures and to propagate unseen textures better by fine-tuning the network with a separate texture-only database.

Ground-truth Pre-training

We aim to propagate the texture information contained in small patches to fill in an entire object. As in Scribbler [2], we use feature and adversarial losses to encourage the generation of realistic object structures. However, we find that these losses alone cannot reproduce fine-grained texture details. Also, Scribbler uses pixel loss to enforce color constraints, but fails when the input color is rare for that particular object category. Therefore, we redefine the feature and adversarial losses and introduce new losses to improve the replication of texture details and encourage precise propagation of colors. For initial training, we derive the network’s input channels from ground-truth photos of objects. When computing the losses, we compare the generated images with the ground-truth. Our objective

function consists of multiple terms, each of which encourages the network to focus on different image aspects.

Feature Loss \mathcal{L}_F .

It has been shown previously that the features extracted from middle layers of a pre-trained neural network, VGG-19 [153], represent high-level semantic information of an image [110, 2]. Given a rough outline sketch, we would like the generated image to loosely follow the object structures specified by the sketch. Therefore, we decide to use a deeper layer of VGG-19 for feature loss (relu 4_2). To focus the feature loss on generating structures, we convert both the ground-truth image and the generated image from RGB color space to Lab and generate grayscale images by repeating the L channel values. We then feed the grayscale image to VGG-19 to extract features. The feature loss is defined as the L2 difference in the feature space. During back propagation, the gradients passing through the L channel of the output image are averaged from the three channels of the VGG-19 output.

Adversarial Loss \mathcal{L}_{ADV} .

Generative adversarial networks (GANs) have been shown to generate realistic images from random noise seeds [104]. In GANs, a generator network and a discriminator network are trained simultaneously in a minimax game. The discriminator tries to distinguish generated images from real photos while the generator tries to generate realistic images tricking the discriminator into thinking they are real. By alternating the optimization of the generator and the discriminator until convergence, the generator would ideally generate images indistinguishable from real photos.

In recent work, the concept of adversarial training has also been adopted in the context of image to image translation. In particular, one can attach a trainable discriminator network at the end of the image translation network and use it to constrain the generated result

to lie on the training image manifold. Previous work proposed to minimize the adversarial loss (loss from the discriminator network) together with other standard losses (pixel, feature losses, etc). The exact choice of losses depends on the different applications [2, 159, 110]. Our work follows the same line. We use adversarial loss on top of feature, texture and color losses. The adversarial loss pushes the network towards synthesizing sharp and realistic images, but at the same time constrains the generated images to choose among typical colors in the training images. The network’s understanding about color sometimes conflicts with user’s color constraints, e.g. a user provides a rainbow color constraint for a handbag, but the adversarial network thinks it looks fake and discourages the generator from producing such output. Therefore, we propose applying the adversarial loss \mathcal{L}_{adv} only on grayscale image (the L channel in Lab space). The discriminator is trained to disregard the color but focus on generating sharp and realistic details. The gradients of the loss only flow through the L channel of the generator output. This effectively reduces the search space and makes GAN training easier and more stable. We perform the adversarial training using the techniques proposed in DCGAN [105] with the modification proposed in LSGAN [158]. LSGAN proposed replacing the cross entropy loss in the original GAN with linear least square loss for higher quality results and stable training.

Style Loss \mathcal{L}_S .

In addition to generating the right content following the input sketch, we would also like to propagate the texture details given in the input texture patch. The previous feature and adversarial losses sometimes struggle to capture fine-scale details, since they focus on getting the overall structure correct. Similar to deep learning based texture synthesis and style transfer work [170, 164], we use style loss to specifically encourage the reproduction of texture details, but we apply style loss on the L channel only. We adopt the idea of matching the Gram matrices (feature correlations) of the features extracted from certain layers of the pretrained classification network (VGG-19). The Gram matrix $\mathcal{G}_{ij}^l \in \mathcal{R}^{N_i \times N_i}$

is defined as:

$$\mathcal{G}_{ij}^l = \sum_k \mathcal{F}_{ik}^l \mathcal{F}_{jk}^l \quad (3.3)$$

where, N_l is the number of feature maps at network layer l , \mathcal{F}_{ik}^l is the activation of the i th filter at position k in layer l . We use two layers of the VGG-19 network (relu3_2, relu4_2) to define our style loss.

Pixel Loss \mathcal{L}_P .

We find that adding relatively weak L2 pixel loss on the L channel stabilizes the training and leads to the generation of texture details that are more faithful to the user’s input texture patch.

Color Loss \mathcal{L}_C .

All losses above are applied only on the L channel of the output to focus on generating sketch-conforming structures, realistic shading, and sharp high-frequency texture details. To enforce the user’s color constraints, we add a separate color loss that penalizes the L2 difference between the ab channels of the generated result and that of the ground-truth.

Our combined objective function is defined as:

$$\mathcal{L} = \mathcal{L}_F + \mathbf{w}_{ADV} \mathcal{L}_{ADV} + \mathbf{w}_S \mathcal{L}_S + \mathbf{w}_P \mathcal{L}_P + \mathbf{w}_C \mathcal{L}_C \quad (3.4)$$

The exact training details can be found in section 3.2.3.

External Texture Fine-tuning

One problem of training with “ground-truth” images is that it is hard for the network to focus on reproducing low-level texture details due to the difficulty of disentangling the texture from the content within the same image. For example, we do not necessarily have training examples of the same object with different textures applied which might help the

network learn the factorization between structure and texture. Also, the Gram matrix-based style loss can be dominated by the feature loss since both are optimized for the same image. There is not much room for the network to be creative in hallucinating low-level texture details, since it tends to focus on generating high-level structure, color, and patterns. Finally, many of the ground-truth texture patches contain smooth color gradients without rich details. Trained solely on those, the network is likely to ignore “hints” from an unseen input texture patch at test time, especially if the texture hint conflicts with information from the sketch. As a result, the network often struggles to propagate high-frequency texture details in the results especially for textures that are rarely seen during training.

To train the network to propagate a broader range of textures, we fine-tune our network to reproduce and propagate textures *for which we have no ground truth output*. To do this, we introduce a new patch-based, local texture loss and adapt our existing losses to encourage faithfulness to a *texture* rather than faithfulness to a ground truth output *object photo*. We use all the losses introduced in the previous sections except the global style loss \mathcal{L}_S . We keep the *feature and adversarial losses*, $\mathcal{L}_F, \mathcal{L}_{ADV}$, unchanged, but modify the *pixel and color losses*, $\mathcal{L}'_P, \mathcal{L}'_C$, to compare the generated result with the entire input texture from which input texture patches are extracted. To prevent color and texture bleeding, the losses are applied only on the foreground object, as approximated by a segmentation mask (Section 3.2.3).

Patch-based Texture Loss

To encourage better propagation of texture, we propose a **patch-based texture loss** \mathcal{L}_t , that is only applied to small local regions of the output image. For handbags and shoes, we utilize foreground background separation (section 3.2.3) to locate the foreground object and apply the texture loss within the foreground. For clothes, we use the detailed segmentation of clothes items [181, 182] to apply texture loss within the semantic region where the input

patch is placed. The texture loss \mathcal{L}_t is composed of three terms:

$$\mathcal{L}_t = \mathcal{L}_s + \mathbf{w}_p \mathcal{L}_p + \mathbf{w}_{adv} \mathcal{L}_{adv} \quad (3.5)$$

Local Discriminator Loss \mathcal{L}_{adv} . We introduce a patch-based adversarial loss that decides whether a pair of texture patches have the same texture. We train the discriminator to recognize a pair of cropped patches from the same texture as a positive example, and a pair of patches from different textures (one from the input texture and one from the generated result) as a negative example. We define L_{adv} as follows: $\mathbf{L}_{adv} = -\sum (\mathbf{D}_{\text{txt}}(\mathbf{G}(x_i), I_t) - 1)^2$. We use 0 to indicate fake and 1 to indicate real.

Local Style Loss \mathcal{L}_s and Pixel Loss \mathcal{L}_p .

To strengthen the texture propagation, we also use Gram matrix-based style loss and L2 pixel loss on the cropped patches.

We randomly sample two patches of size 50x50 from the generated result and the input texture (from the separate texture database), compute texture loss on the L channel of the patches and average them. We propagate the gradients of the texture loss only through the corresponding patch region in the output.

While performing the texture fine-tuning, the network is trying to adapt itself to understand and propagate new types of textures, and might ‘forget’ what it learnt from the ground-truth pretraining stage. Therefore, when training on external textures, we mix in iterations of ground-truth training fifty percent of the time.

Our final objective function becomes:

$$\mathcal{L} = \mathcal{L}_F + \mathbf{w}_{ADV} \mathcal{L}_{ADV} + \mathbf{w}_P \mathcal{L}'_P + \mathbf{w}_C \mathcal{L}'_C + \mathcal{L}_t \quad (3.6)$$

3.2.3 Training Setup

We train TextureGAN on three object-centric datasets – **handbags** [160], **shoes** [183] and **clothes** [184, 182, 185, 186]. Each photo collection contains large variations of colors, materials, and patterns. These domains are also chosen so that we can demonstrate plausible product design applications. For supervised training, we need to generate (input, output) image pairs. For the output of the network, we convert the ground-truth photos to Lab color space. For the input to the network, we process the ground-truth photos to extract 5-channel images. The five channels include one channel for the binary sketch, two channels for the texture (intensities and binary location masks), and two channels for the color controls.

In this section, we describe how we obtain segmentation masks used during training, how we generate each of the input channels for the ground-truth pre-training, and how we utilize the separate texture database for the network fine-tuning. We also provide detailed training procedures and parameters.

Segmentation Mask

For our local texture loss, we hope to encourage samples of output texture to match samples of input texture. But the output texture is localized to particular image regions (e.g. the interior of objects) so we wouldn't want to compare a background patch to an input texture. Therefore we only sample patches which fall inside an estimated foreground segmentation mask. Our handbag and shoe datasets are product images with consistent, white backgrounds so we simply set the white pixels as background pixels. For clothes, the segmentation mask is already given in the dataset. With the clothes segmentation mask, we process the ground-truth photos to white out the background. Note that segmentation masks are *not used* at test time.

Data Generation for Pre-training

Sketch Generation. For handbags and shoes, we generate sketches using the deep edge detection method used in pix2pix [187, 159]. For clothes, we leverage the clothes parsing information provided in the dataset [184, 182]. We apply Canny edge detection on the clothing segmentation mask to extract the segment boundaries and treat them as a sketch. We also apply xDoG [149] on the clothes image to obtain more variation in the training sketches. Finally, we mix in additional synthetic sketches generated using the methods proposed in Scribbler [2].

Texture Patches. To generate input texture constraints, we randomly crop small regions within the foreground objects of the ground-truth images. We randomly choose the patch location from within the segmentation and randomize the patch size. We convert each texture patch to the Lab color space and normalize the pixels to fall into 0-1 range. For each image, we randomly generate one or two texture patches. For clothes, we extract texture patches from one of the following regions – top, skirt, pant, dress, or bag. We pass a binary mask to the network to indicate the spatial support of the texture.

Data Generation for Fine-tuning

To encourage diverse and faithful texture reproduction, we fine-tune TextureGAN by applying external texture patches from a leather-like texture dataset. We queried “leather” in Google and manually filtered the results to 130 high resolution leather textures. From this clean dataset, we sampled roughly 50 crops of size 256x256 from each image to generate a dataset of 6,300 leather-like textures. We train our models on leather-like textures since they are commonly seen materials for handbags, shoes and clothes and contain large appearance variations that are challenging for the network to propagate.

Training Details

For **pre-training**, we use the following parameters on all datasets. $w_{ADV} = 1$, $w_S =$



Figure 3.13: The effect of texture loss and adversarial loss. a) The network trained using all proposed losses can effectively propagate textures to most of the foreground region; b) Removing adversarial loss leads to blurry results; c) Removing texture loss harms the propagation of textures.

0.1, $w_P = 10$ and $w_C = 100$. We use the Adam optimizer [188] with learning rate $1e-2$.

For **fine-tuning**, we optimize all the losses at the same time but use different weight settings. $w_{ADV} = 1e4$, $w_S = 0$, $w_P = 1e2$, $w_C = 1e3$, $w_s = 10$, $w_p = 0.01$, and $w_{adv} = 7e3$. We also decrease the learning rate to $1e-3$. We train most of the models at input resolution of 128×128 except one clothes model at the resolution of 256×256 (Figure 3.16).

3.2.4 Results and Discussions

Ablation Study.

Keeping other settings the same, we train networks using different combinations of losses to analyze how they influence the result quality. In Figure 3.13, given the input sketch, texture patch and color patch (first column), the network trained with the complete objective function (second column) correctly propagates the color and texture to the entire



Figure 3.14: Effect of Proposed *local* losses. a) Results from the ground-truth model without any local losses, b) with local pixel loss, c) with local style loss, d) with local texture discriminator loss. With local discriminator loss, the network tends to produce more consistent texture throughout the object.

handbag. If we turn off the texture loss (fourth column), the texture details within the area of the input patch are preserved, but difficult textures cannot be fully propagated to the rest of the bag. If we turn off the adversarial loss (third column), texture is synthesized, but that texture is not consistent with the input texture. Our ablation experiment confirms that style loss alone is not sufficient to encourage texture propagation motivating our local patch-based texture loss (Section 3.2.2).

External Texture Fine-tuning Results.

We train TextureGAN on three datasets – shoes, handbags, and clothes – with increasing levels of structure complexity. We notice that for object categories like shoes that contain limited structure variations, the network is able to quickly generate realistic shading and structures and focus its remaining capacity for propagating textures. The texture propagation on the shoes dataset works well even without external texture fine-tuning. For more sophisticated datasets like handbags and clothes, external texture fine-tuning is critical for



Figure 3.15: Results for shoes and handbags on different textures. Odd rows: input sketch and texture patch. Even rows: generated results.

the propagation of difficult textures that contain sharp regular structures, such as stripes.

Figure 3.14 demonstrates how external texture fine-tuning with our proposed texture loss can improve the texture consistency and propagation. The “ground truth” pre-trained model is faithful to the input texture patch in the output only directly under the patch and does not propagate it throughout the foreground region. By fine-tuning the network with texture examples and enforcing local style loss, local pixel loss, and local texture loss we nudge the network to apply texture consistently across the object. With local style loss (column c) and local texture discriminator loss (column d), the networks are able to propagate texture better than without fine tuning (column a) or just local pixel loss (column b). Using local texture discriminator loss tends to produce more visually similar results to the input texture than style loss.

Figure 3.15 shows the results of applying various texture patches to sketches of handbags and shoes. These results are typical of test-time result quality.

Figure 3.16 shows results on the clothes dataset trained at a resolution of 256x256. The clothes dataset contains large variations of structures and textures, and each image in the dataset contains multiple semantic regions. Our network can also handle multiple texture patches. As shown in figure 3.16, we put different texture patches on different parts of the clothes (middle left and bottom left). The network can propagate the textures within semantic regions of the sketch while respecting the sketch boundaries.



Figure 3.16: Applying multiple texture patches on the sketch. Our system can also handle multiple texture inputs and our network can follow sketch contours and expand the texture to cover the sketched object.



Figure 3.17: Results on human-drawn sketches.

Figure 3.17 shows results on human-drawn handbags. These drawings differ from our synthetically generated training sketches but the results are still high quality.

3.2.5 Conclusion

We have presented an approach for controlling deep image synthesis with input sketch and texture patches. With this system, the user can draw the object structure through sketching and precisely control the generated details with texture patches. TextureGAN is feed-forward which allows users to see the effect of their edits in real time. By training TextureGAN with local texture constraints, we demonstrate its effectiveness on sketch and texture-based image synthesis. TextureGAN also operates in **Lab** color space, which enables separate controls on color and content. Furthermore, our results on fashion datasets show that our pipeline is able to handle a wide variety of texture inputs and generates texture compositions that follow the sketched contours. In the future, we hope to apply our network on more complex scenes.

Appendices

APPENDIX A

THE SKETCHY DATABASE

This supplemental material contains (1) a more exhaustive benchmark of sketch-based image retrieval variants, (2) more details about the training parameters for the deep networks, (3) per-category retrieval evaluation for all 125 categories, (4) qualitative examples of image-based sketch retrieval, sketch-based sketch retrieval, and image-based image retrieval, and (5) numerous additional within-domain and across-domain retrieval results.

GN - GoogLeNet[49]

AN - AlexNet[47] refers to the caffe variation—CaffeNet

SN GN - GoogLeNet fine tuned for sketch classification

SN AN - AlexNet fine tuned for sketch classification

GN Cat - GoogLeNet cross domain classification network

AN Cat - AlexNet cross domain classification network

GFHOG - Gradient field HOG based on code from the authors [10]

GALIF - Gabor local line based feature, based on OpenSSE implementation by Zhang Dongdong available on github at <https://github.com/zddhub/opensse>

¹The learning rate drops by factor of 0.1 for each step size.

²Shows final value. The value has been adjusted during the training phase to avoid getting stuck in local optima.

³ For positive pairs and negative pairs, respectively.

Momentum was set to 0.9 for all experiments.

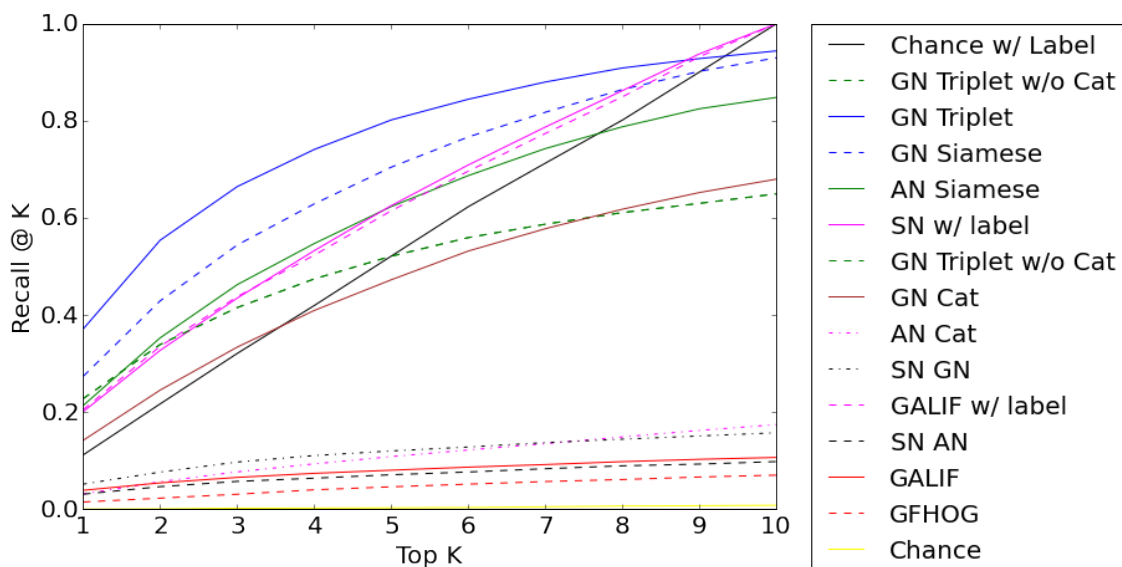


Figure A.1: Benchmark plot with additional models and baselines

Table A.1: Parameter details for training Triplet/Siamese network

Model	Learning rate	Iterations	Step size ¹	Batch size	Weight decay
GN Triplet	10e-5	160k	100k	16	2e-3
GN Siamese	10e-5	120k	100k	32	2e-3
AN Siamese	10e-5	80k	50k	128	5e-4
Model	Classification loss weight	Embedding weight	Margin(s)		
GN Triplet	² 10	1	15		
GN Siamese	1	0.1	³ 2500, 2.5		
AN Siamese	1	10e-3	³ 2500, 2.5		

Table A.2: Comparison between different output dimension

Output dimension	Recall @ K=1
2048	35.98%
1024	36.09%
512	35.7%
256	35.49%
128	36.33%
64	34.71%
32	31.62%
2	4.07%

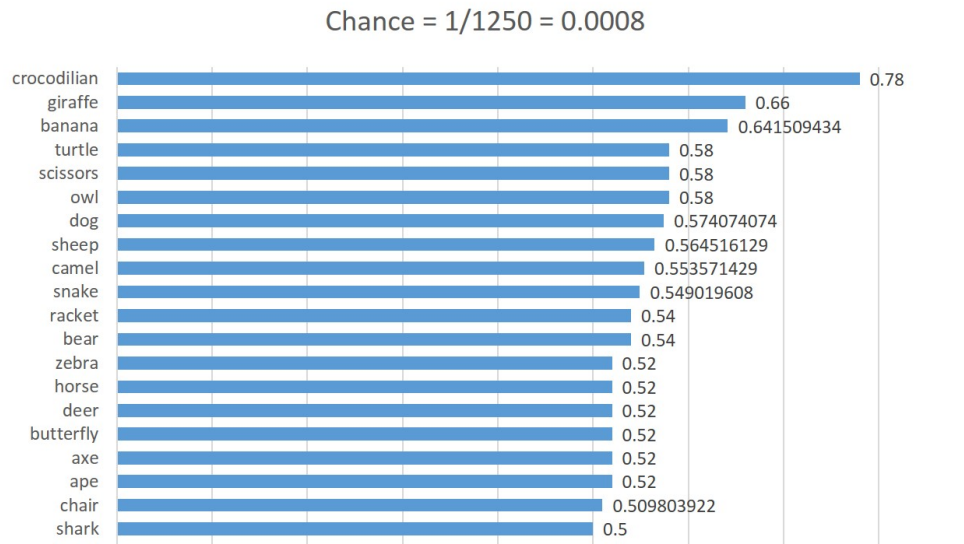


Figure A.2: Categories in the Sketchy database that give top 20 per-class recall@1. Note that chance level is $1/1250$ because there is only one correct photo for each sketch query in the test set of size 1250.

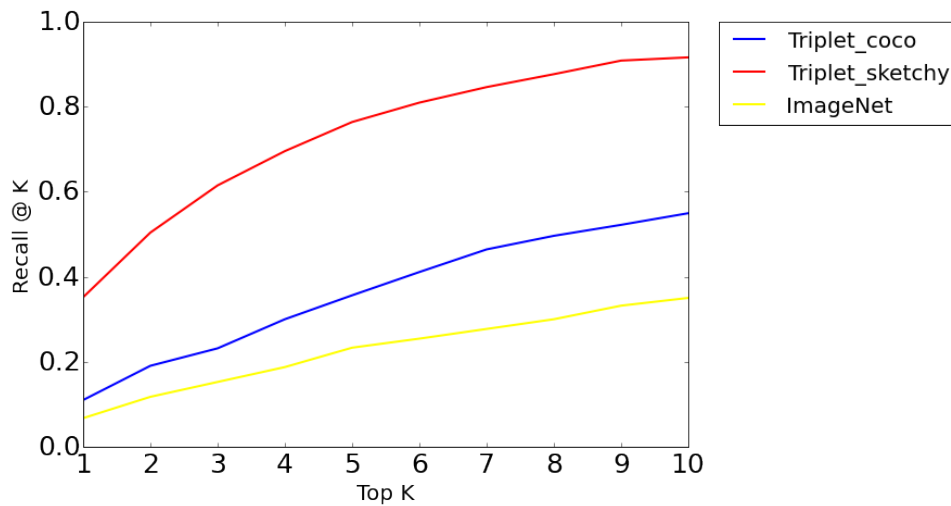


Figure A.3: Comparison between networks trained for one epoch with MS COCO boundaries vs the Sketchy database. Both networks were fine tuned from an ImageNet trained model with no additional pre-training. Note, this experiment covers only the 13 MS COCO categories that have enough training data and overlap with Sketchy database categories, thus the accuracies are not comparable to other experiments.

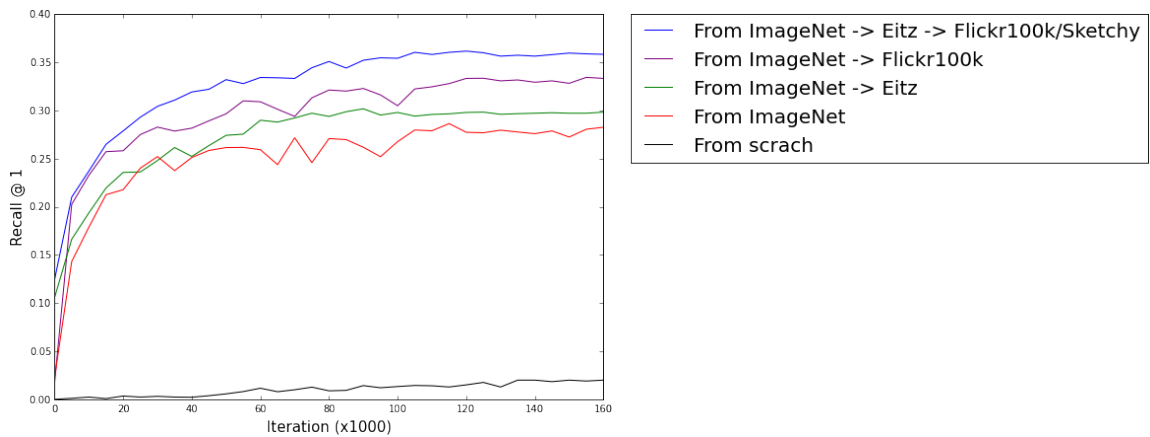


Figure A.4: Contribution of different pre-training schemes. We train each network with our Sketchy database for 160k iterations and then measure the recall at $K = 1$.

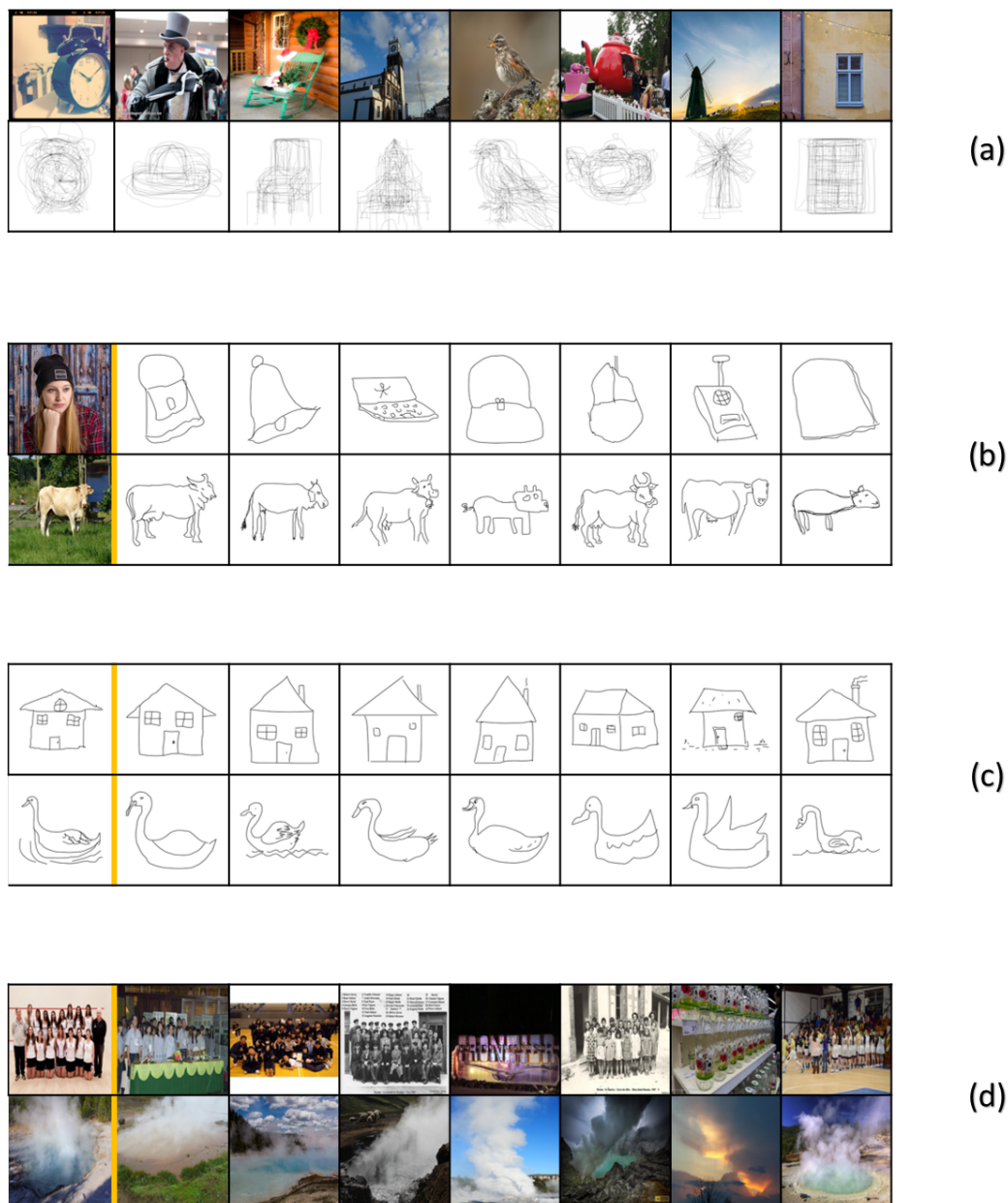


Figure A.5: In our paper, we evaluate our learned cross-domain embedding with sketch-based image retrieval. But the embedding just as naturally supports (b) image-based sketch retrieval, (c) sketch-based sketch retrieval, and (d) image-based image retrieval. (a) Shows the average of aligned sketches retrieved for various query photos. The photo query in (b) top did not work particularly well because “person” is not a category in the Sketchy database. The remainder of this supplementary material contains additional cross-domain and within-domain retrieval results. All results use the “GN Triplet” network pair (or half of the network pair in the case of image-to-image and sketch-to-sketch retrieval). It may be that focusing on within-domain retrieval at learning time would improve the results. Or the opposite could be true – that image-to-image retrieval is improved by the presence of cross-domain sketches because because it trains the image networks to encode the salient object details. Our aim with those qualitative results is to show that our learned embedding space is reasonable, not that it is “state-of-the-art” for these retrieval tasks.

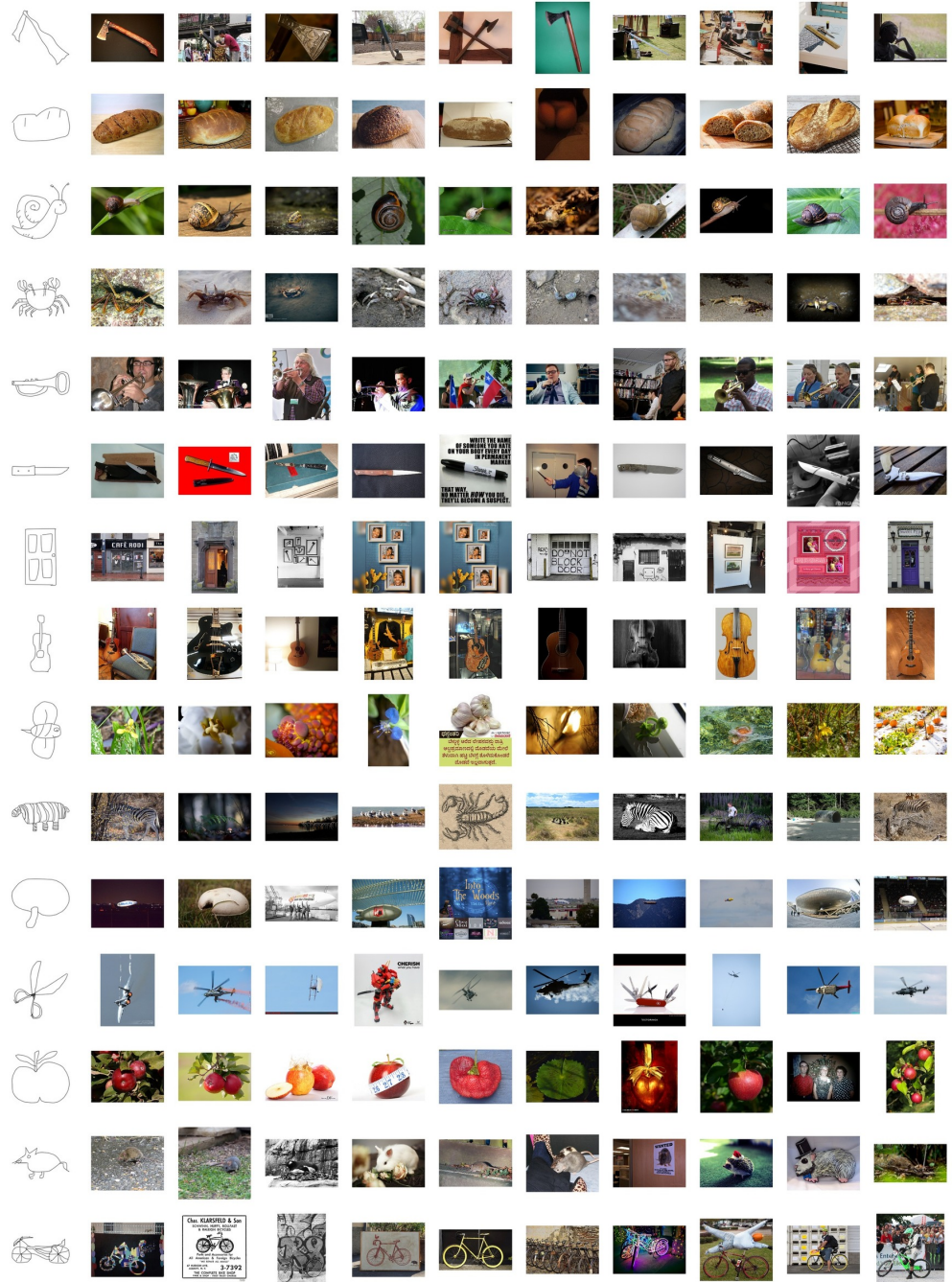


Figure A.6: Sketch-based image retrieval results

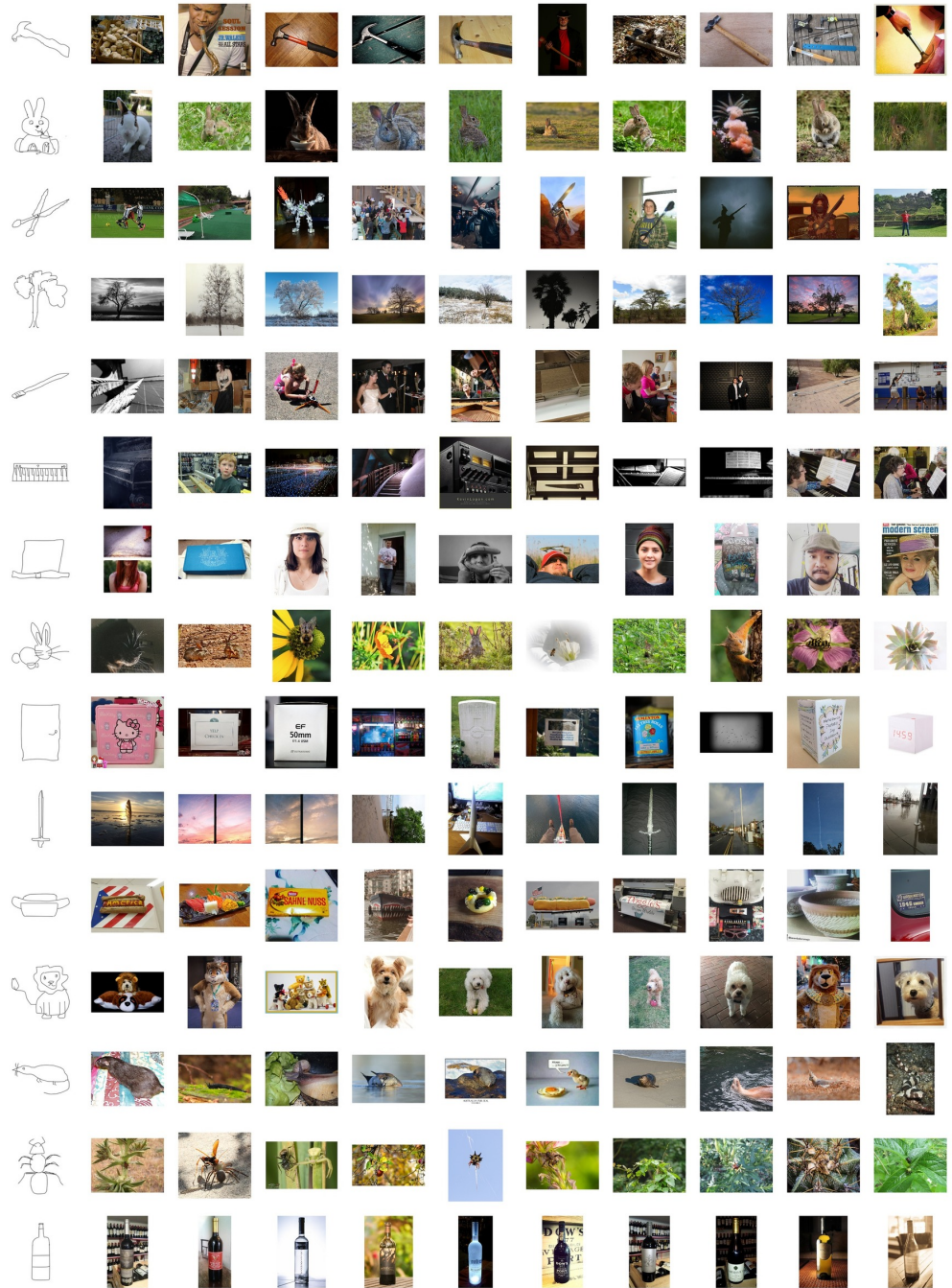


Figure A.7: Sketch-based image retrieval results
100

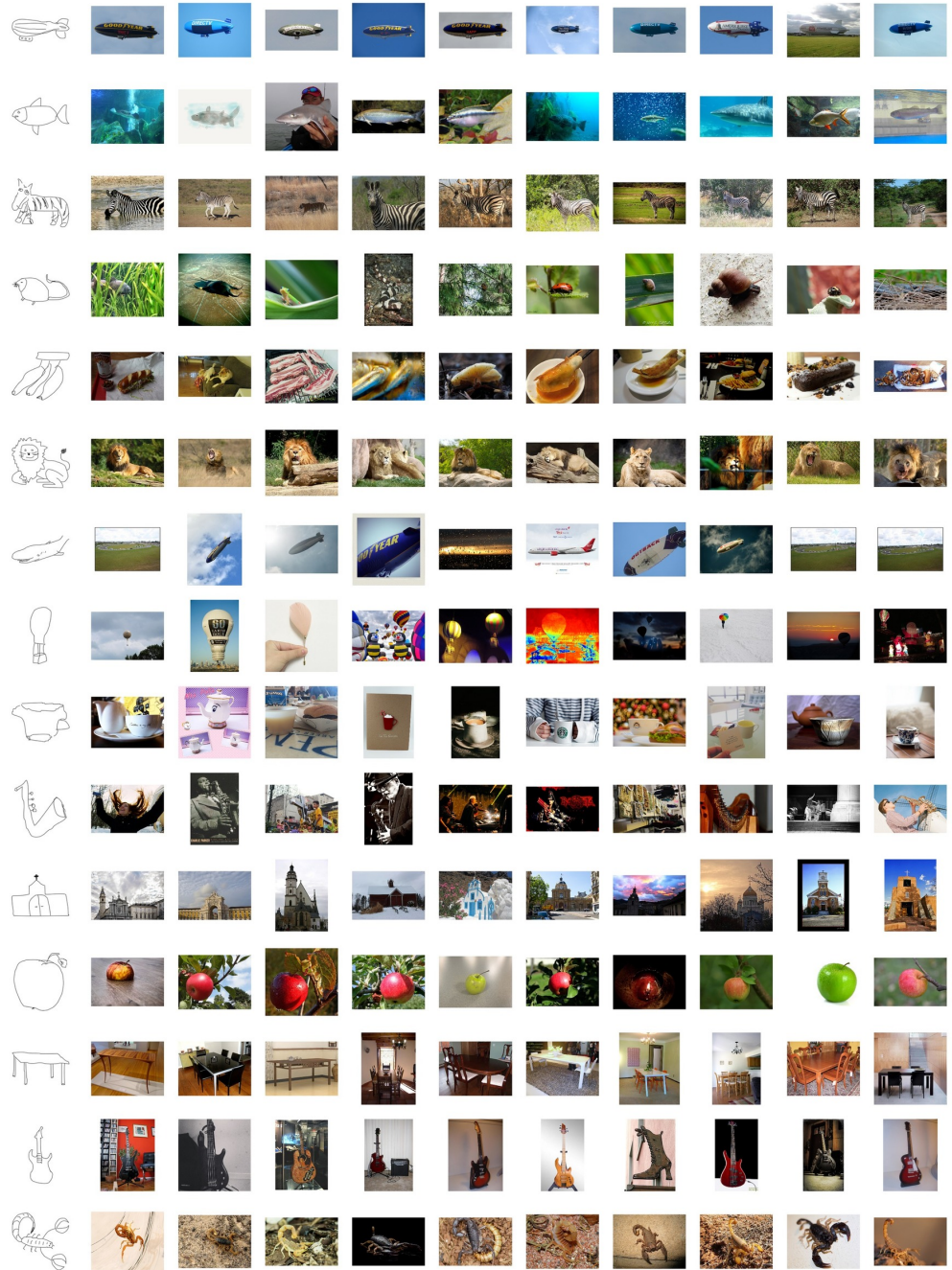


Figure A.8: Sketch-based image retrieval results

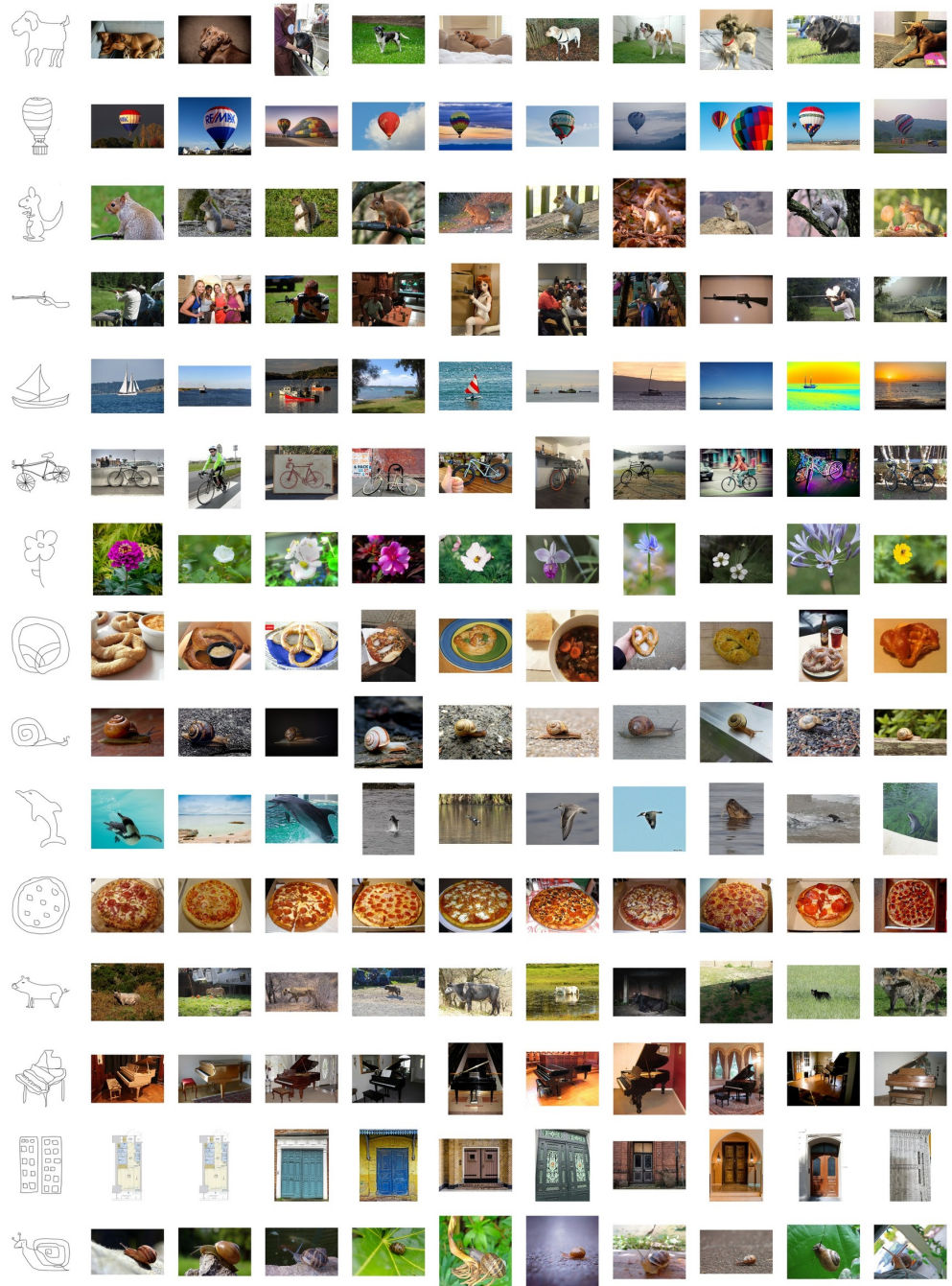


Figure A.9: Sketch-based image retrieval results

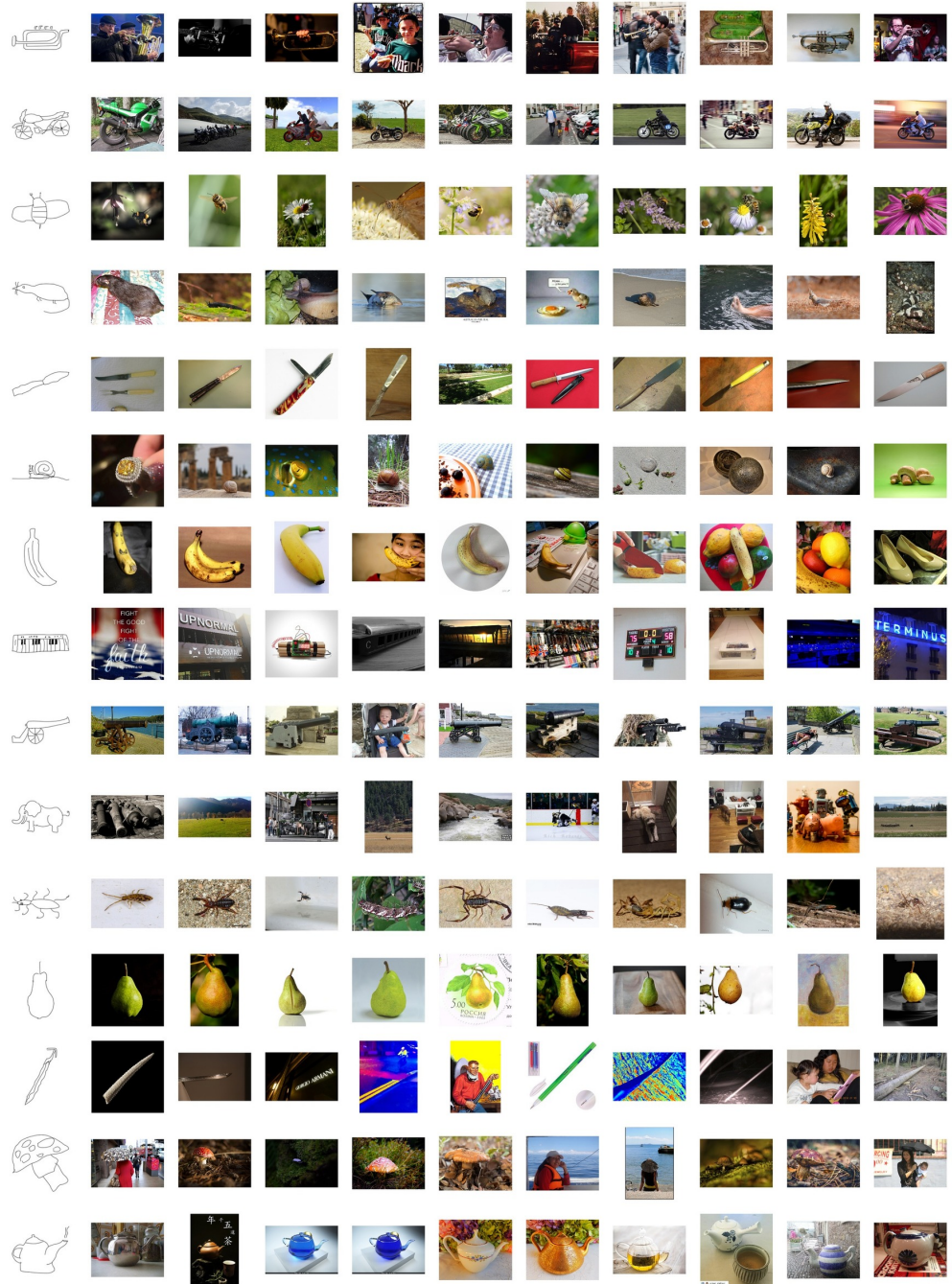


Figure A.10: Sketch-based image retrieval results
103



Figure A.11: Sketch-based image retrieval results
104

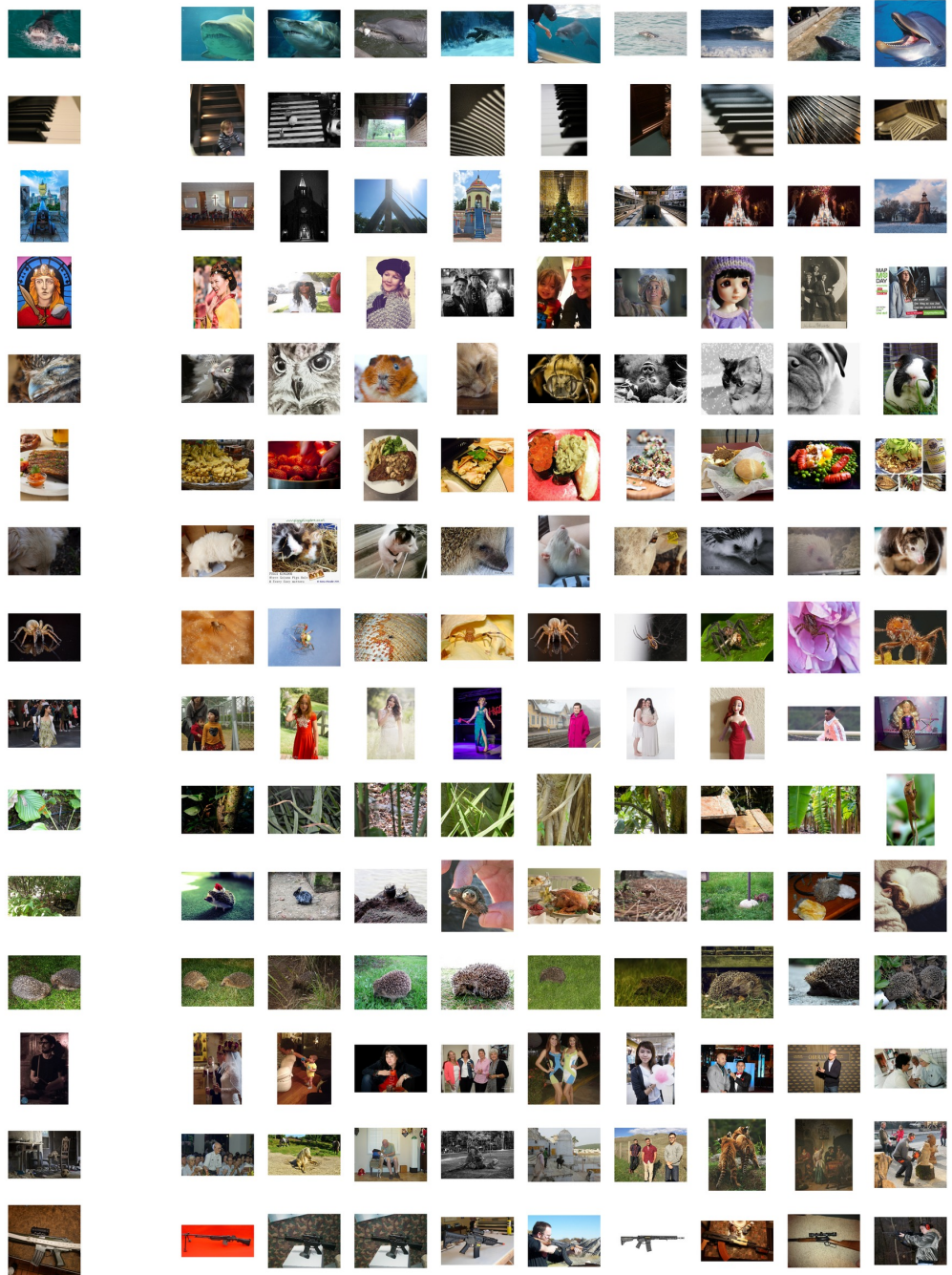


Figure A.12: Image-based image retrieval results
105

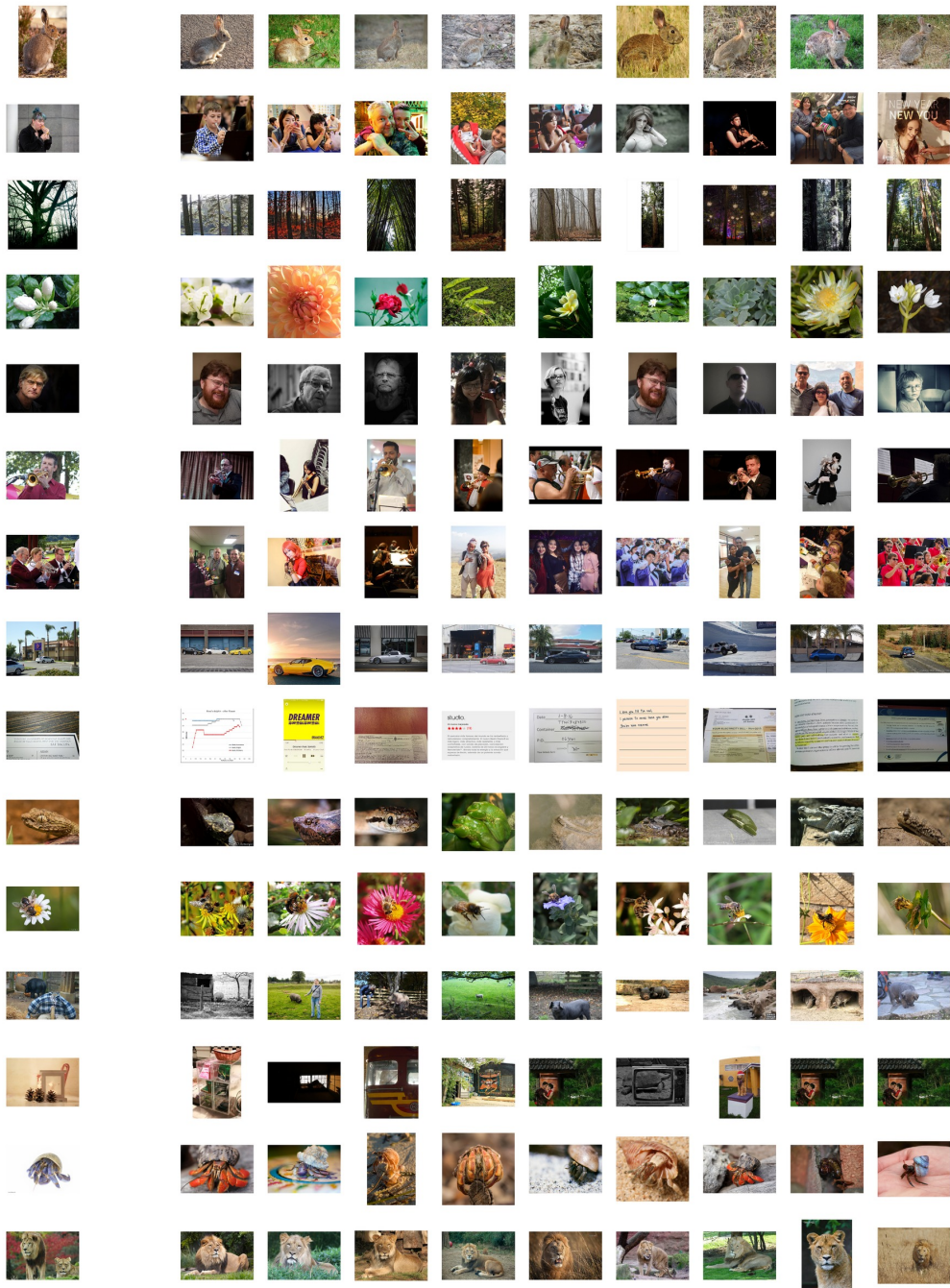


Figure A.13: Image-based image retrieval results
106

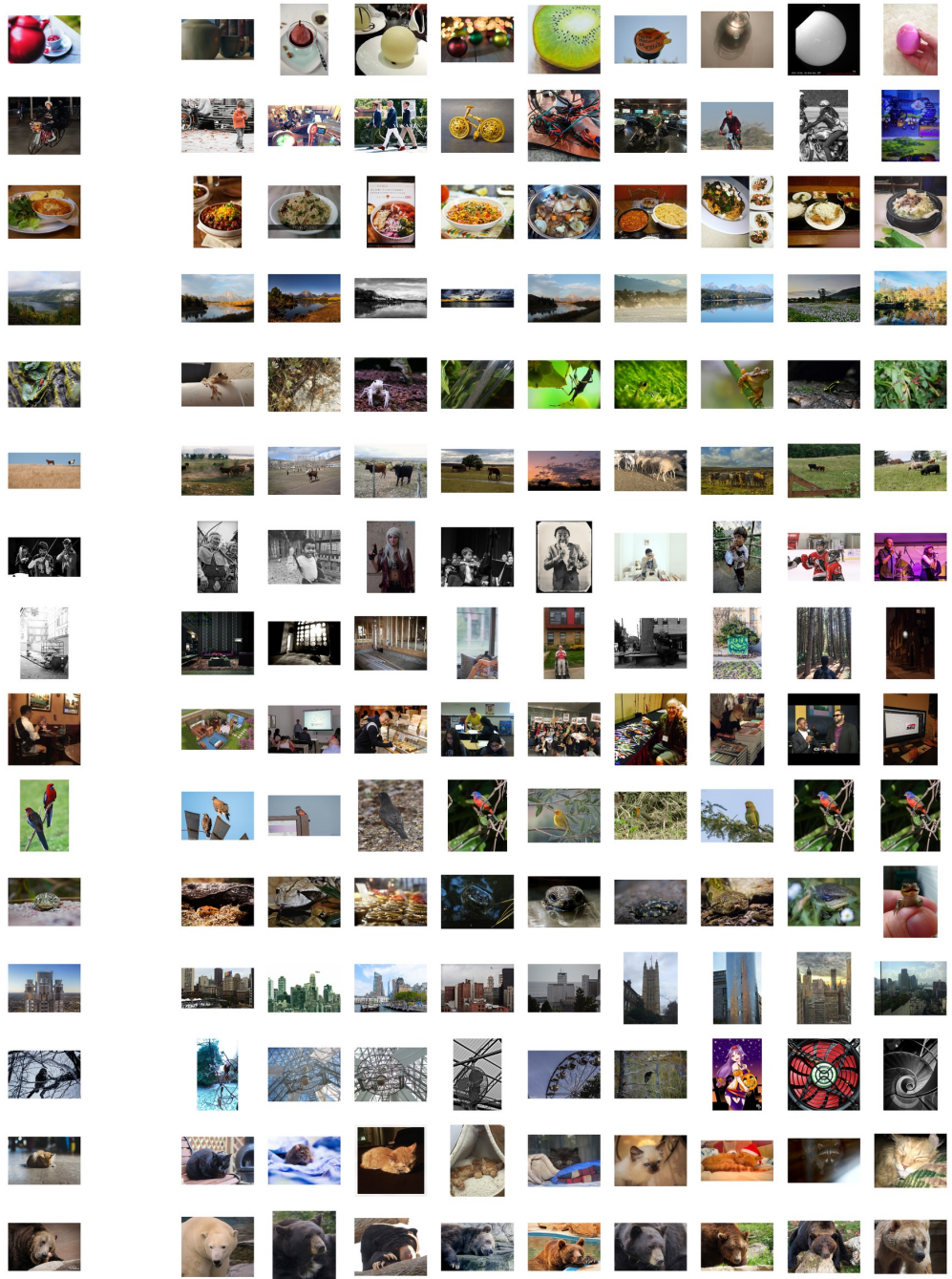


Figure A.14: Image-based image retrieval results
107

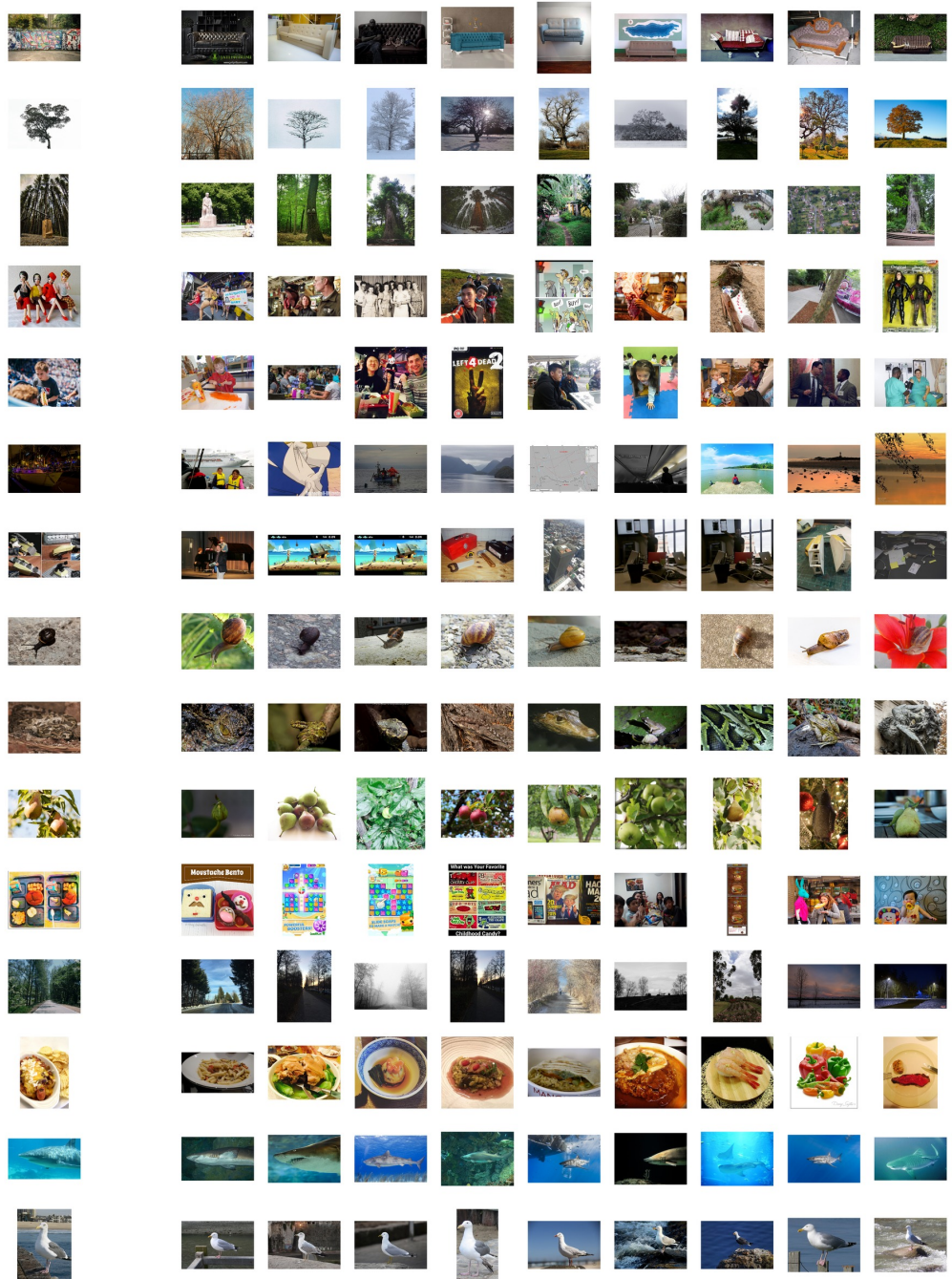


Figure A.15: Image-based image retrieval results
108

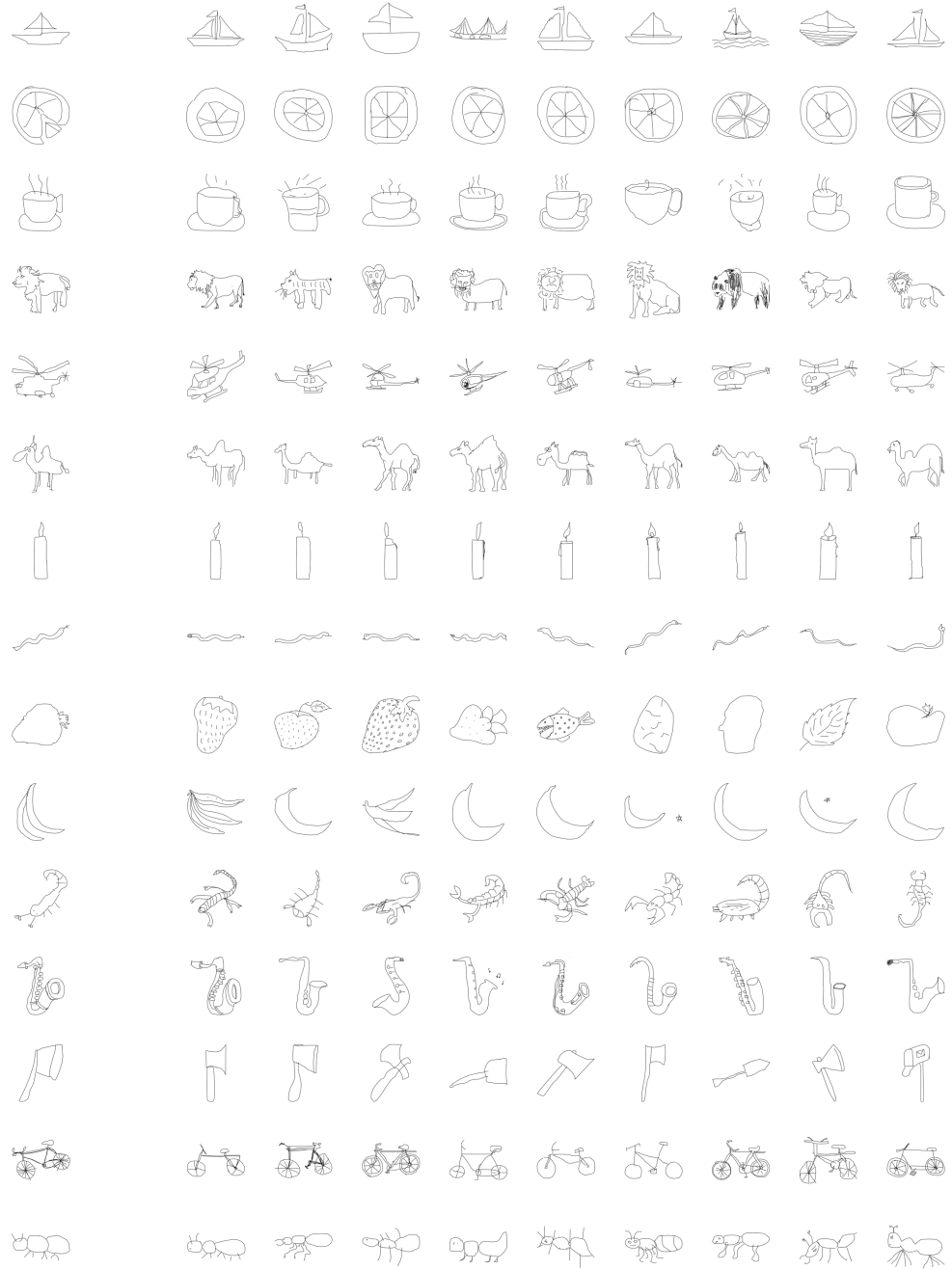


Figure A.16: Sketch-based sketch retrieval results
109

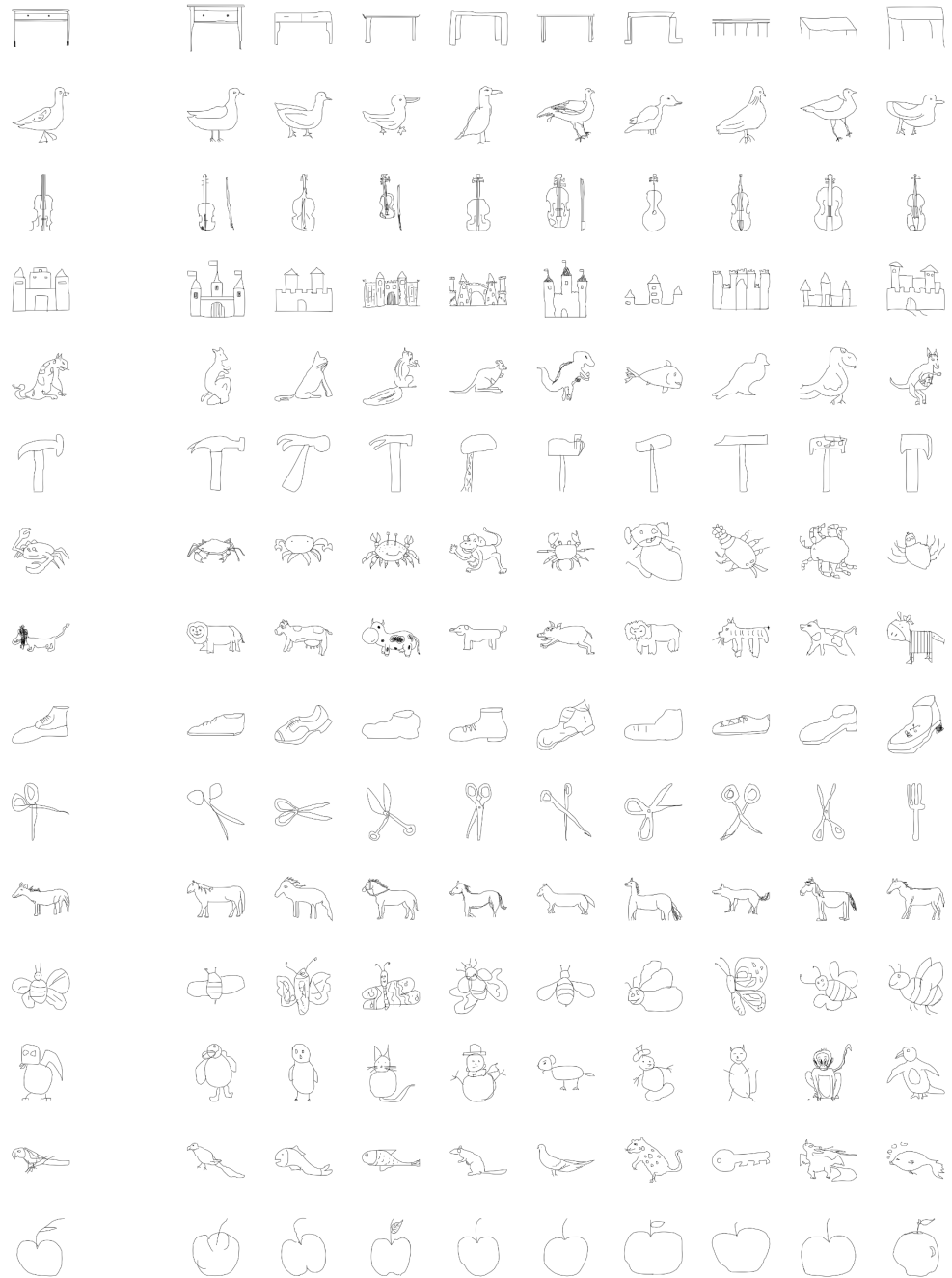


Figure A.17: Sketch-based sketch retrieval results

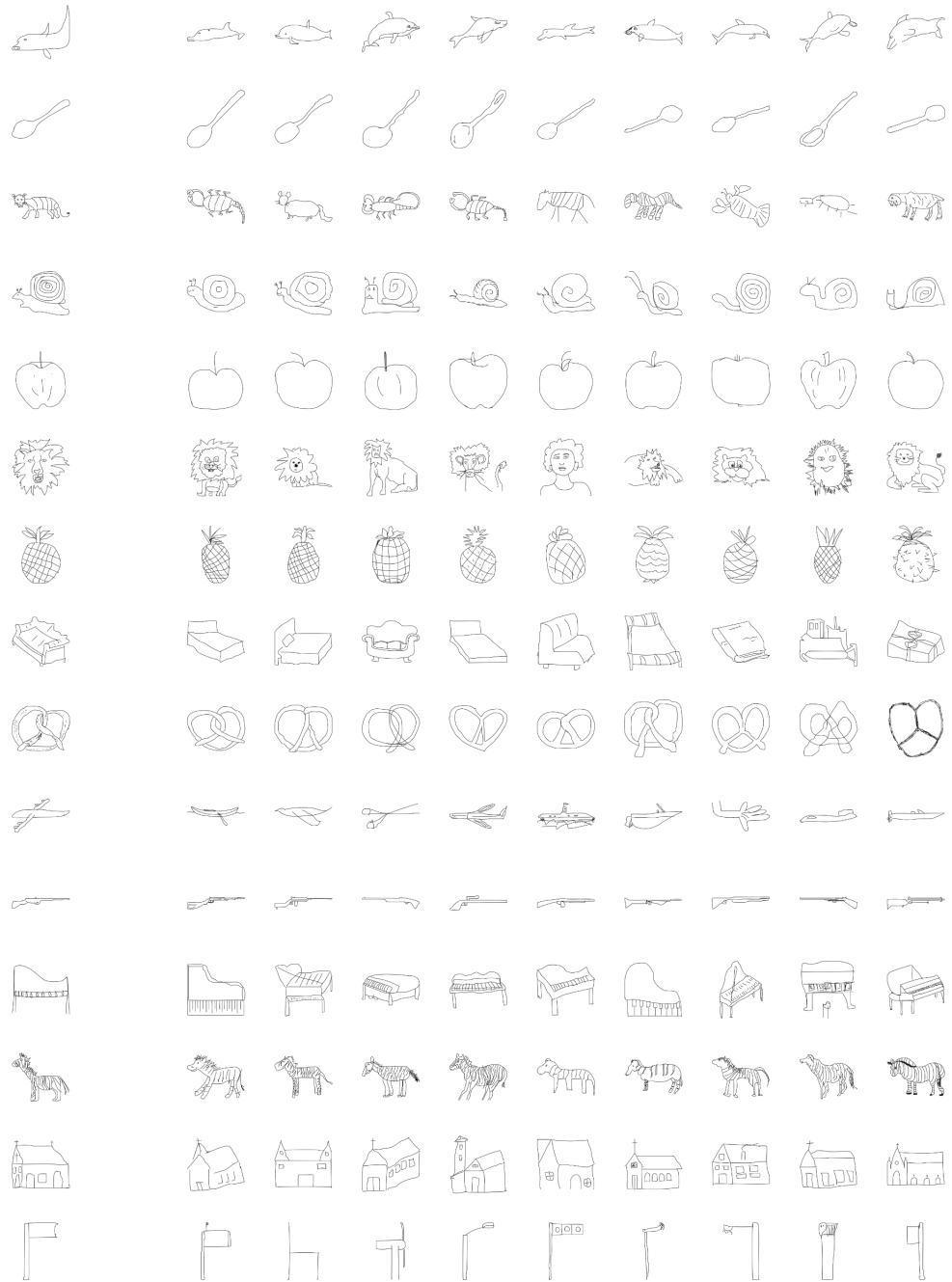


Figure A.18: Sketch-based sketch retrieval results



Figure A.19: Image-based sketch retrieval results
112

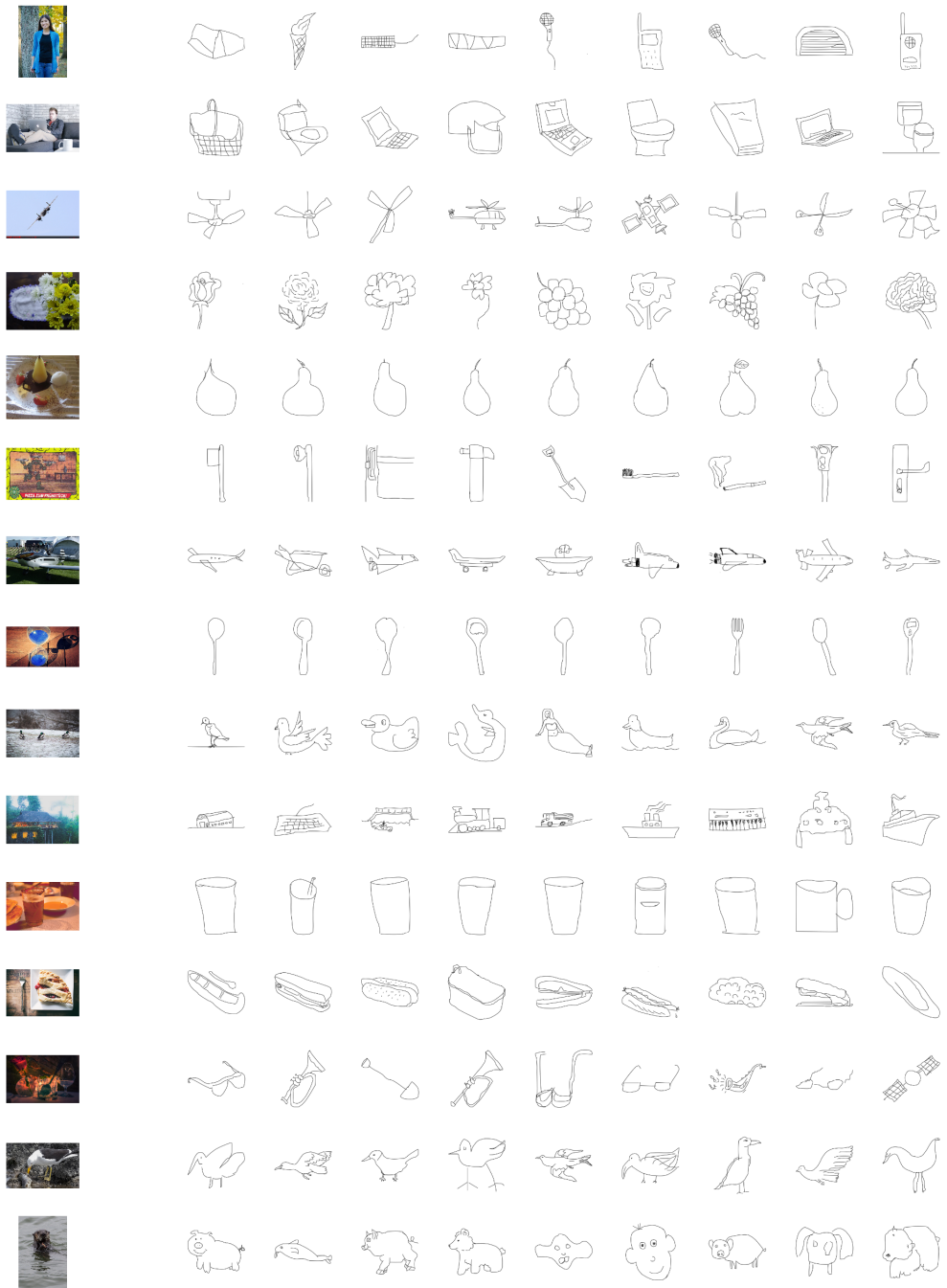


Figure A.20: Image-based sketch retrieval results
113

APPENDIX B

TASK-FORMER

B.1 Training Details

Each competing method including ours is trained for 20 epochs. For multi-label classification, we first train a multi-label classifier on top of a pre-trained CLIP embedding (freezing all CLIP pre-trained parameters), and use them to initialize the classifier for fine-tuning. This helps accelerate training as the network is well closer to an optimum than training from scratch with a random initialization. We do train the decoder for the caption generation task from scratch, however. We observe that the presence of caption generation as a helper task helps increase retrieval performance. We start training of the multi-label classifier with a learning rate of 10^{-4} which is then quickly decreased to 10^{-5} . In the final network, we use the weight ratio of 10, 1, and 100 for multi label classification, caption generation, and the contrastive learning term respectively.

B.2 On the Effect of Text Completeness

In Section 2.5.3, we investigate the effect of an incomplete input sketch on the retrieval recall. Recall that we vary the complexity of each input sketch, while keeping the text description intact. The goal of this section is to provide an analogous analysis on the effect of an incomplete text input. We sample 300 records of from our collected COCO dataset. Each record consists of three objects: a sketch, a text description, and an image. For each text description, we vary its degree of completeness by randomly subsampling $x\%$ (for a number of values of x) of the tokens (words), while keeping its corresponding sketch intact. Table B.1 shows the resulted retrieval recall for each degree of text completeness. We observe that compared to when no text query is given as an input (0%), recall@1 increases

from 0.099 to 0.316 when only 20% of tokens are included. While recall unsurprisingly increases as more tokens are included, the trend appears to exhibit a diminishing return property.

Table B.1: Retrieval performance of our model as measured by $\text{recall}@\{1, 5, 10\}$ when both a complete sketches, and text queries are inputted to the model. Words in each text query are sub-sampled to investigate the effect of incomplete input text on retrieval.

Text completeness	R@1	R@5	R@10
0%	0.099	0.195	0.257
20%	0.316	0.545	0.640
40%	0.357	0.613	0.708
60%	0.446	0.710	0.813
80%	0.530	0.807	0.884
100%	0.607	0.866	0.930

B.3 Example Sketches at Different Degrees of Complexity

In this section we show examples of sketches at different levels of complexity, as discussed in Section 2.5.3. For each hand-drawn sketch we collected (100%), we randomly sub-sample and keep only a fraction of strokes in the sketch. These incomplete sketches are only used for investigating the robustness of our model to inaccurate input sketches (see Section 2.5.3). Examples can be found in Figure B.1.

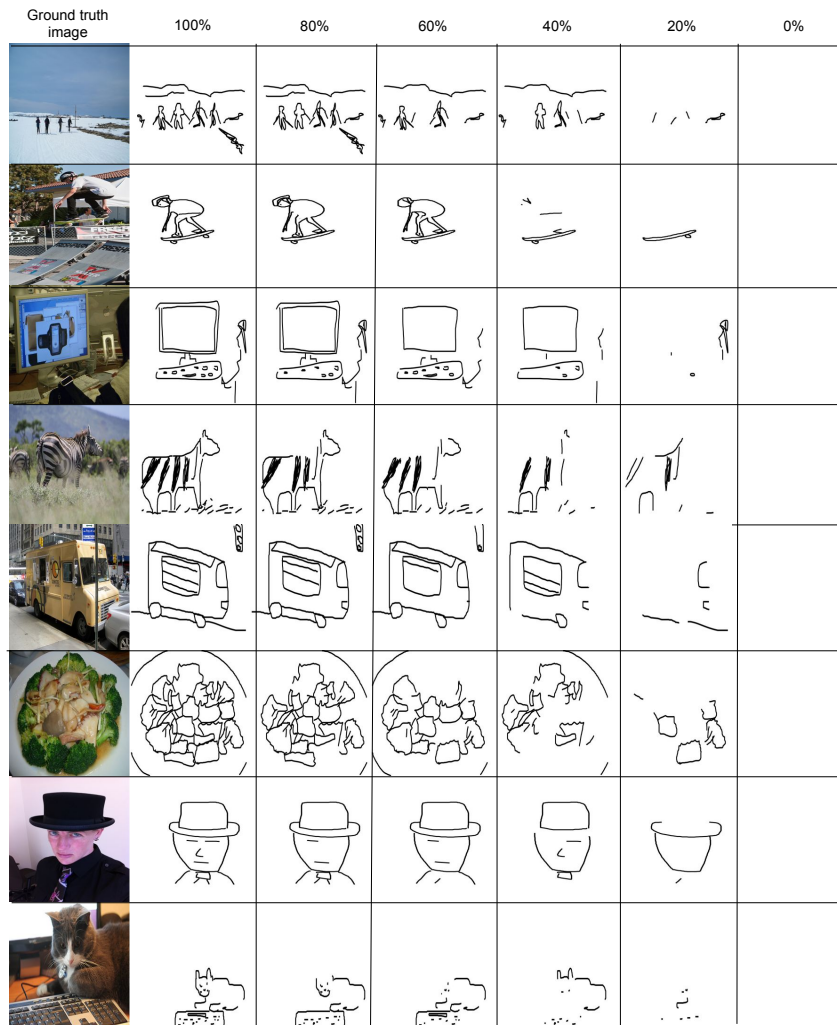


Figure B.1: For each collected sketch (100%), we randomly sub-sample a fraction of strokes to investigate the robustness of our model to incomplete sketches (see Section 2.5.3).

B.4 Retrieval Results

We present retrieved images from our model for a number of randomly selected input sketch-text pairs in Figure B.2.



Figure B.2: Retrieved images from our model for a number of randomly selected sketch-text pairs.

B.5 Sketch Captioning

As discussed in Section ??, owing to the use of a caption generation loss term, our model can also be used to generate captions from sketches as a convenient side benefit. The goal of this section is to provide more examples of generated captions to supplement what are shown in Figure ?? in the main text. We emphasize that caption generation is not part of the main goals of our proposal. We inspect captions generated from our model as a way to diagnose and gain insights into what the network sees. These examples can be found in Figure B.3.









	a person is riding a horse in the grass .
	a man is sitting on a bench with a dog .
	a man in a suit and tie standing in front of a large building .
	a group of people sitting at a table with food .
	a woman in a black jacket is holding a tennis racquet .
	a man is riding a wave on a surfboard .
	a person is holding a sandwich and a plate of food .
	a person riding a bike with a dog on the back .

Figure B.3: Examples of sketch captions generated from our model. The tendency to start each caption with “a woman” or “a man“ stems from the skewness in the label distribution of the training dataset (COCO) where more than half of images contain people. Besides this anticipated issue, our network appears to be able to generate coherent and satisfactorily accurate captions.

B.6 Synthetic Sketches

Examples of synthetically generated sketches of [96] from images in the COCO dataset [67] can be found in Figure B.4. Recall from the main text that we use these synthetically sketches to train our model. For evaluation on COCO, we use hand-drawn sketches. Details of how we collect hand-drawn sketches can be found in Section B.7.



Figure B.4: Examples of synthetically generated sketches of [96].

B.7 Collecting Hand-drawn Sketches

While our model is trained with synthetically generated sketches [96], to investigate the retrieval performance on real data, we collect hand-drawn sketches for images in the COCO test set. The collected sketches are drawn by Amazon Mechanical Turk (AMT) workers (Turkers).

The workers draw sketches by going through the following process. An image from the COCO test set is randomly selected as the target image to draw. The image is displayed to the worker for 15 seconds and the worker is asked to memorize crucial details in the image. After 15 seconds, the image disappears and the worker is asked to draw a sketch that represents the image from memory. Drawing takes place on a graphical user interface (GUI) that we provide (see Figure B.5). On the provided GUI, the worker can draw, erase, undo a stroke, redo a stroke, or clear the entire canvas. The size of the sketch pad always matches the size of the target image. Some example sketches collected can be found in Figure B.6.

When you are ready, press the start button to start the timer.

Start

00:00:11

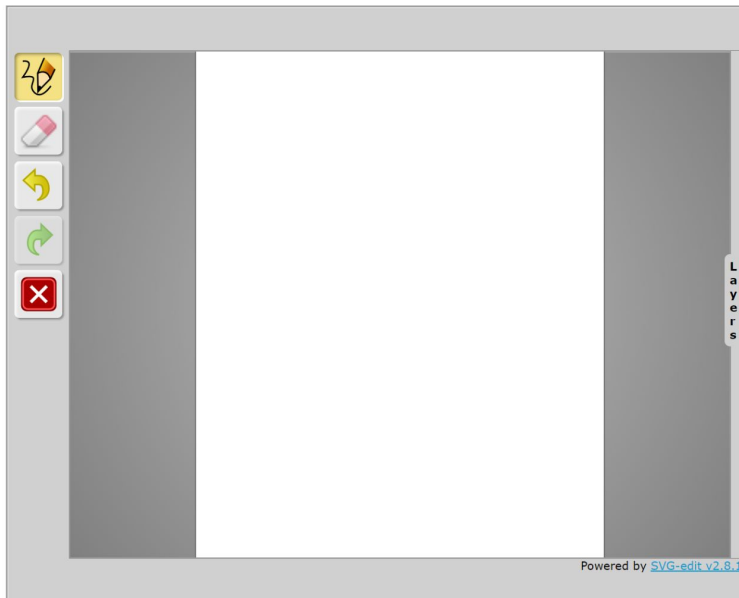


Figure B.5: The graphical user interface that workers on Amazon Mechanical Turk use to draw sketches for images in the COCO test set. Workers can draw, erase, undo/redo, or clear the entire canvas.



Figure B.6: Examples of the hand-drawn sketches collected via Amazon Mechanical Turk.

APPENDIX C

SCRIBBLER

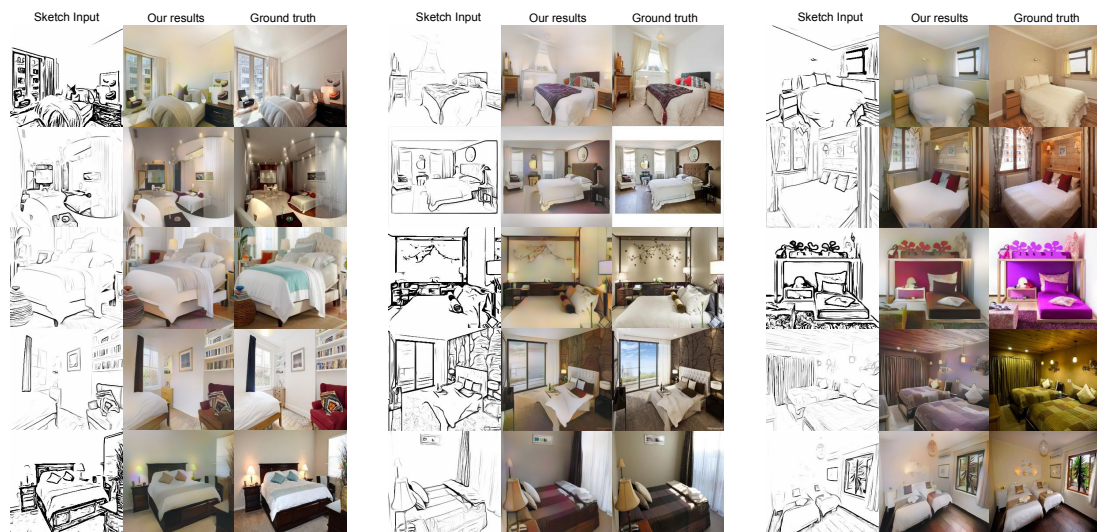


Figure C.1: Additional results on held out bedroom sketches

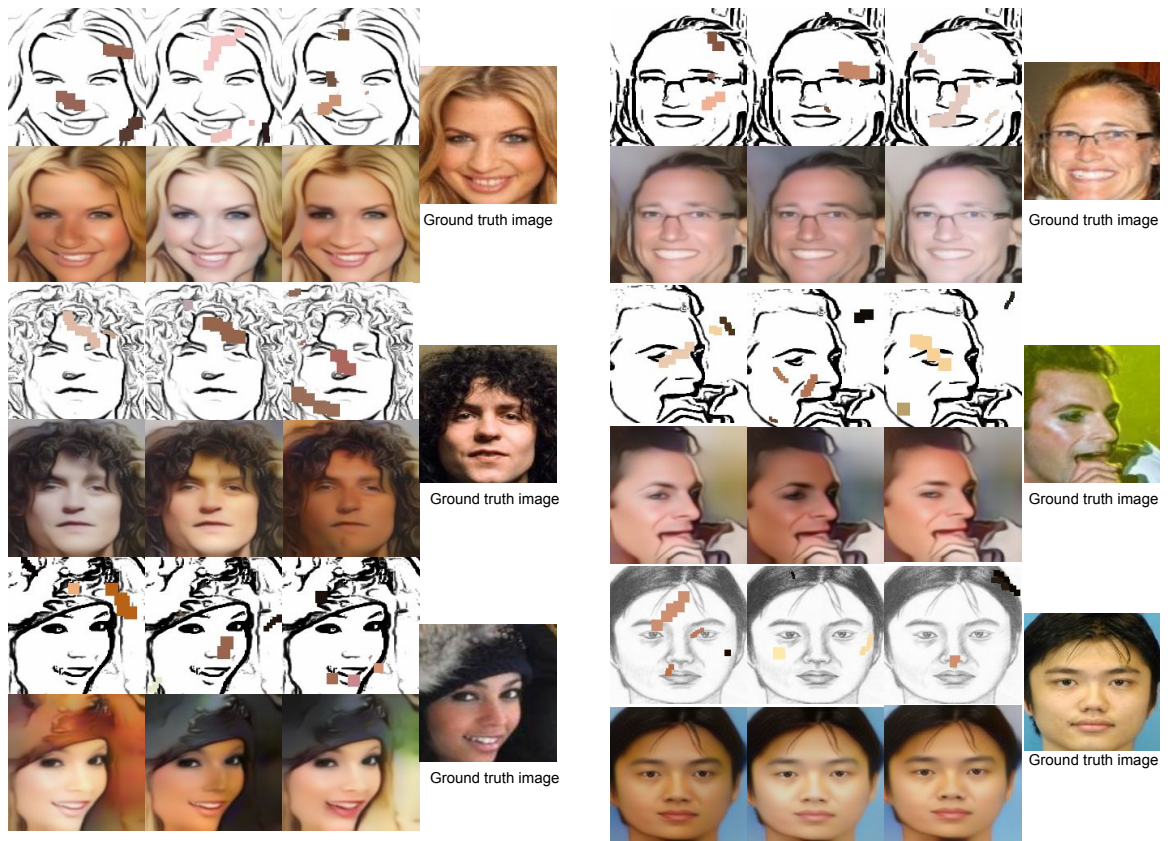


Figure C.2: Additional results on held out face sketches of multiple styles with random color strokes. The color strokes are generated by sampling curves from random face images.

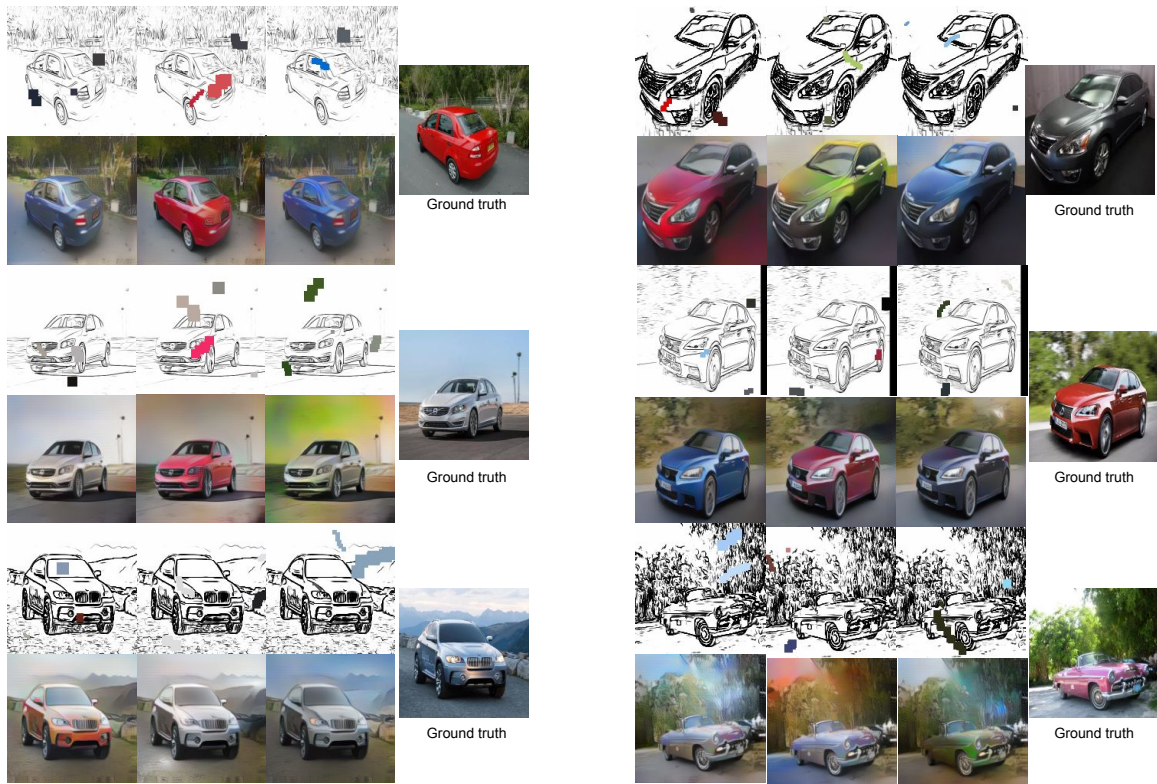


Figure C.3: Additional results on held out car sketches with random color strokes. The color strokes are generated by sampling curves from random car images.

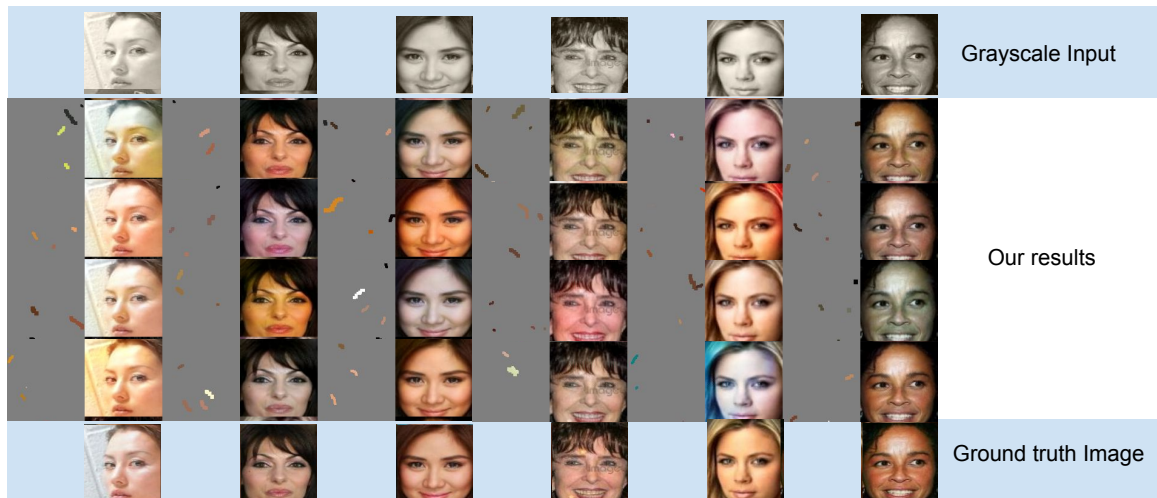


Figure C.4: Additional face colorization results with random color suggestions. The color strokes are sampled from random curves in other face images.

APPENDIX D
TEXTUREGAN

Table D.1: Generator network. Every conv layer is followed by batch normalization and relu activation. Residual blocks consist of 2 conv layers with stride of 1 (table 4), in which the input to the block is added to the output as proposed in [189]. Biup refers to the upsampling with scale of 2x, using bilinear interpolation. We also add a single skip connection from input to layer 21.

Layer	Type	In	Out	Kernel Size	Stride	Pad
1	conv [skip connection]	5	32	3	1	1
2	residual block	32	32	3	1	1
3	conv	32	64	3	2	1
4	residual block	64	64	3	1	1
5	conv	64	128	3	2	1
6	residual block	128	128	3	1	1
7	conv	128	256	3	2	1
8	residual block	256	256	3	1	1
9	residual block	256	256	3	1	1
10	residual block	256	256	3	1	1
11	residual block	256	256	3	1	1
12	residual block	256	256	3	1	1
13	conv + biup	256	128	3	1	1
14	residual block	128	128	3	1	1
15	residual block	128	128	3	1	1
16	conv + biup	128	64	3	1	1
17	residual block	64	64	3	1	1
18	residual block	64	64	3	1	1
19	conv + biup	64	32	3	1	1
20	residual block	32	32	3	1	1
21	conv [skip connection]	32 + 5	64	3	1	1
22	residual block	64	64	3	1	1
23	residual block	64	64	3	1	1
24	conv	64	3	3	1	1

Table D.2: Global Discriminator network. The input is a single L channel normalized to 0/1 range (equivalent to greyscale image).

Layer	Type	Input Channels	Output Channels	Kernel Size	Stride	Pad
1	conv	1	32	9	2	1
2	conv	32	64	5	2	1
3	conv	64	256	5	2	1
4	residual block	256	256	3	1	1
5	residual block	256	256	3	1	1
6	conv	256	128	4	2	1
7	conv	128	1	4	2	1

Table D.3: Texture Discriminator network. The input to the network is a pair of texture patches, of size 50×50 .

Layer	Type	Input Channels	Output Channels	Kernel Size	Stride	Pad
1	conv	2	32	3	2	1
2	conv	32	128	3	2	1
3	residual block	128	128	3	1	1
4	residual block	128	128	3	1	1
5	conv	128	64	3	2	1
6	conv	64	1	3	2	1

Table D.4: Residual Block. Each conv is followed by batch normalization, then relu activation.

Layer	Type	Kernel Size	Stride	Pad
1	conv	3	1	1
2	conv	3	1	1
3	+ input	-	-	-



Figure D.1: Additional results on held out handbag sketches [152x152]. On the right is the “ground truth” photo from which the sketch was synthesized. On the far left, a texture patch is also sampled from the original handbag. We show three additional results with diverse textures.



Figure D.2: Additional results on held out shoes sketches [152x152]. On the right is the “ground truth” photo from which the sketch was synthesized. On the far left, a texture patch is also sampled from the original shoe. We show three additional results with diverse textures.



Figure D.3: Additional results on held out clothes sketches [256x256].



Figure D.4: Comparison of results before and after fine-tuning on an external texture database. The base model is trained on handbag and shoe sketches with ground truth images. The fine-tuned model with local texture loss improves texture propagation on difficult texture with high contrast and strong regularity, e.g. striped, dotted, checkerboard patterns, which are rarely seen in the ground-truth.



Figure D.5: In some cases, especially with complicated or rare texture patterns, the network fails to propagate texture throughout the object. In these cases, the input texture is blended into the object instead. A user can alleviate such effect by placing more texture swatches.



Figure D.6: The effect of varying input patch sizes during testing.



Figure D.7: Comparison RGB versus Lab inputs. Enforcing feature, texture and adversarial losses on the L channel and adding a separate color loss on the ab channels significantly improve the color propagation.

REFERENCES

- [1] P. Sangkloy, N. Burnell, C. Ham, and J. Hays, “The sketchy database: Learning to retrieve badly drawn bunnies,” *ACM Transactions on Graphics (proceedings of SIGGRAPH)*, 2016.
- [2] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, “Scribbler: Controlling deep image synthesis with sketch and color,” *Computer Vision and Pattern Recognition, CVPR*, 2017.
- [3] W. Xian *et al.*, “Texturegan: Controlling deep image synthesis with texture patches,” in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 8456–8465.
- [4] T. Chen, M.-m. Cheng, P. Tan, A. Shamir, and S.-m. Hu, “Sketch2photo: Internet image montage,” *ACM SIGGRAPH Asia*, 2009.
- [5] M. Eitz, R. Richter, K. Hildebrand, T. Boubekeur, and M. Alexa, “Photosketcher: Interactive sketch-based image synthesis,” *IEEE Computer Graphics and Applications*, 2011.
- [6] T. Kato, T. Kurita, N. Otsu, and K. Hirata, “A sketch retrieval method for full color image database-query by visual example,” in *Pattern Recognition, 1992. Vol. I. Conference A: Computer Vision and Applications, Proceedings., 11th IAPR International Conference on*, Aug. 1992, pp. 530–533.
- [7] T. Mainelli, M. Chau, R. Reith, and M. Shirer, *Idc worldwide quarterly smart connected device tracker*, <http://www.idc.com/getdoc.jsp?containerId=prUS25500515>, Mar. 2015.
- [8] M. Eitz, J. Hays, and M. Alexa, “How do humans sketch objects?” *ACM Trans. Graph.*, vol. 31, no. 4, Jul. 2012.
- [9] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.
- [10] R. Hu and J. Collomosse, “A performance evaluation of gradient field hog descriptor for sketch based image retrieval,” *Computer Vision and Image Understanding*, vol. 117, no. 7, pp. 790–806, 2013.
- [11] J.-Y. Zhu, Y. J. Lee, and A. A. Efros, “Averageexplorer: Interactive exploration and alignment of visual data collections,” *ACM Transactions on Graphics (SIGGRAPH 2014)*, vol. 33, no. 4, 2014.

- [12] A. Del Bimbo and P. Pala, “Visual image retrieval by elastic matching of user sketches,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 2, pp. 121–132, Feb. 1997.
- [13] S. Sclaroff, “Deformable prototypes for encoding shape categories in image databases,” *Pattern Recognition*, vol. 30, no. 4, pp. 627–641, 1997.
- [14] C. E. Jacobs, A. Finkelstein, and D. H. Salesin, “Fast multiresolution image querying,” in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’95, ACM, 1995, pp. 277–286.
- [15] A. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain, “Content-based image retrieval at the end of the early years,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 12, pp. 1349–1380, Dec. 2000.
- [16] Y. Cao, C. Wang, L. Zhang, and L. Zhang, “Edgel index for large-scale sketch-based image search,” in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 2011, pp. 761–768.
- [17] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros, “Data-driven visual similarity for cross-domain image matching,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 30, 2011, p. 154.
- [18] X. Cao, H. Zhang, S. Liu, X. Guo, and L. Lin, “Sym-fish: A symmetry-aware flip invariant sketch histogram shape descriptor,” in *Computer Vision (ICCV), 2013 IEEE International Conference on*, 2013, pp. 313–320.
- [19] J. M. Saavedra and J. M. Barrios, “Sketch based image retrieval using learned keyshapes (lks),” in *Proceedings of the British Machine Vision Conference (BMVC)*, Sep. 2015, pp. 164.1–164.11.
- [20] M. Eitz, K. Hildebrand, T. Boubekeur, and M. Alexa, “An evaluation of descriptors for large-scale image retrieval from sketched feature lines,” *Computers & Graphics*, vol. 34, no. 5, pp. 482–498, 2010.
- [21] ———, “Sketch-based image retrieval: Benchmark and bag-of-features descriptors,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 17, no. 11, pp. 1624–1636, 2011.
- [22] Y. Li, T. M. Hospedales, Y.-Z. Song, and S. Gong, “Fine-grained sketch-based image retrieval by matching deformable part models,” in *British Machine Vision Conference (BMVC)*, 2014.

- [23] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1627–1645, Sep. 2010.
- [24] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, “The pascal visual object classes (voc) challenge,” *Int. J. Comput. Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.
- [25] Q. Yu, F. Liu, Y. Song, T. Xiang, T. Hospedales, and C. C. Loy, “Sketch me that shoe,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.
- [26] R. G. Schneider and T. Tuytelaars, “Sketch classification and classification-driven analysis using fisher vectors,” *ACM Trans. Graph.*, vol. 33, no. 6, 174:1–174:9, Nov. 2014.
- [27] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales, “Sketch-a-net that beats humans,” in *British Machine Vision Conference (BMVC)*, 2015.
- [28] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, “Multi-view convolutional neural networks for 3d shape recognition,” in *Proc. ICCV*, 2015.
- [29] F. Wang, L. Kang, and Y. Li, “Sketch-based 3d shape retrieval using convolutional neural networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.
- [30] T.-Y. Lin, Y. Cui, S. Belongie, and J. Hays, “Learning deep representations for ground-to-aerial geolocalization,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.
- [31] S. Bell and K. Bala, “Learning visual similarity for product design with convolutional neural networks,” *ACM Trans. Graph.*, vol. 34, no. 4, Jul. 2015.
- [32] Y. Li, H. Su, C. R. Qi, N. Fish, D. Cohen-Or, and L. J. Guibas, “Joint embeddings of shapes and images via cnn image purification,” *ACM Trans. Graph.*, vol. 34, no. 6, 234:1–234:12, Oct. 2015.
- [33] S. Chopra, R. Hadsell, and Y. LeCun, “Learning a similarity metric discriminatively, with application to face verification,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, Jun. 2005, pp. 539–546.
- [34] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping,” in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 1735–1742.

- [35] J. Wang *et al.*, “Learning fine-grained image similarity with deep ranking,” *CoRR*, vol. abs/1404.4661, 2014.
- [36] T. Chen, P. Tan, L.-Q. Ma, M.-M. Cheng, A. Shamir, and S.-M. Hu, “Poseshop: Human image database construction and personalized content synthesis,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 5, pp. 824–837, May 2013.
- [37] I. Berger, A. Shamir, M. Mahler, E. Carter, and J. Hodgins, “Style and abstraction in portrait sketching,” *ACM Trans. Graph.*, vol. 32, no. 4, 55:1–55:12, Jul. 2013.
- [38] A. Limpaecher, N. Feltman, A. Treuille, and M. Cohen, “Real-time drawing assistance through crowdsourcing,” *ACM Trans. Graph.*, vol. 32, no. 4, 54:1–54:8, Jul. 2013.
- [39] D. Martin, C. Fowlkes, D. Tal, and J. Malik, “A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics,” in *Proc. 8th Int’l Conf. Computer Vision*, vol. 2, Jul. 2001, pp. 416–423.
- [40] J. Xiao, K. A. Ehinger, J. Hays, A. Torralba, and A. Oliva, “Sun database: Exploring a large collection of scene categories,” *International Journal of Computer Vision*, pp. 1–20, 2014.
- [41] T. Lin *et al.*, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014.
- [42] S. Antol, C. L. Zitnick, and D. Parikh, “Zero-Shot Learning via Visual Abstraction,” in *ECCV*, 2014.
- [43] T. F. Brady, T. Konkle, G. A. Alvarez, and A. Oliva, “Visual long-term memory has a massive storage capacity for object details,” *Proceedings of the National Academy of Sciences*, vol. 105, no. 38, pp. 14 325–14 329, 2008.
- [44] T. F. Brady, T. Konkle, J. Gill, A. Oliva, and G. A. Alvarez, “Visual long-term memory has the same limit on fidelity as visual working memory,” *Psychological Science*, vol. 24, no. 6, 2013.
- [45] K. Grill-Spector and N. Kanwisher, “Visual recognition: As soon as you see it, you know what it is,” *Psychological Science*, vol. 16, no. 2, pp. 152–160, 2005.
- [46] M. Nieuwenstein and B. Wyble, “Beyond a mask and against the bottleneck: Retroactive dual-task interference during working memory consolidation of a masked visual target,” *Journal of Experimental Psychology: General*, vol. 143, pp. 1409–1427, 2014.

- [47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *26th Annual Conference on Neural Information Processing Systems (NIPS)*, 2012, pp. 1106–1114.
- [48] Y. Jia *et al.*, “Caffe: Convolutional architecture for fast feature embedding,” *arXiv preprint arXiv:1408.5093*, 2014.
- [49] C. Szegedy *et al.*, “Going deeper with convolutions,” *arXiv preprint arXiv:1409.4842*, 2014.
- [50] L. van der Maaten and G. Hinton, “Visualizing high-dimensional data using t-sne,” *Journal of Machine Learning Research*, vol. 9, no. 3, pp. 2579–2605, Nov. 2008.
- [51] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision—ECCV 2014*, Springer, 2014, pp. 818–833.
- [52] T. Zhou, Y. Jae Lee, S. X. Yu, and A. A. Efros, “Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.
- [53] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox, “Learning to generate chairs with convolutional neural networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.
- [54] F. Cole *et al.*, “Where do people draw lines?” *ACM Transactions on Graphics (Proc. SIGGRAPH)*, vol. 27, no. 3, Aug. 2008.
- [55] B. Wang, Y. Yang, X. Xu, A. Hanjalic, and H. T. Shen, “Adversarial cross-modal retrieval,” in *Proceedings of the 25th ACM international conference on Multimedia*, 2017, pp. 154–162.
- [56] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” 2019.
- [57] X. Li *et al.*, “Oscar: Object-semantics aligned pre-training for vision-language tasks,” in *European Conference on Computer Vision*, Springer, 2020, pp. 121–137.
- [58] D. Qi, L. Su, J. Song, E. Cui, T. Bharti, and A. Sacheti, “Imagebert: Cross-modal pre-training with large-scale weak-supervised image-text data,” *arXiv preprint arXiv:2001.07966*, 2020.
- [59] Y.-C. Chen *et al.*, “Uniter: Universal image-text representation learning,” in *ECCV*, 2020.

- [60] C. Jia *et al.*, “Scaling up visual and vision-language representation learning with noisy text supervision,” *arXiv preprint arXiv:2102.05918*, 2021.
- [61] A. Radford *et al.*, “Learning transferable visual models from natural language supervision,” in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 8748–8763.
- [62] A. Pandey, A. Mishra, V. K. Verma, A. Mittal, and H. Murthy, “Stacked adversarial network for zero-shot sketch based image retrieval,” in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2020, pp. 2540–2549.
- [63] Q. Yu, F. Liu, Y.-Z. Song, T. Xiang, T. M. Hospedales, and C.-C. Loy, “Sketch me that shoe,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 799–807.
- [64] A. Dutta and Z. Akata, “Semantically tied paired cycle consistency for zero-shot sketch-based image retrieval,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5089–5098.
- [65] K. Pang *et al.*, “Generalising fine-grained sketch-based image retrieval,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.
- [66] R. Hu and J. Collomosse, “A performance evaluation of gradient field hog descriptor for sketch based image retrieval,” *Comput. Vis. Image Underst.*, vol. 117, no. 7, p. 790 806, Jul. 2013.
- [67] T. Lin *et al.*, “Microsoft COCO: common objects in context,” *CoRR*, vol. abs/1405.0312, 2014. arXiv: 1405.0312.
- [68] S. Dey, P. Riba, A. Dutta, J. Lladós, and Y.-Z. Song, “Doodle to search: Practical zero-shot sketch-based image retrieval,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2019.
- [69] O. Tursun, S. Denman, S. Sridharan, E. Goan, and C. Fookes, “An efficient framework for zero-shot sketch-based image retrieval,” *arXiv preprint arXiv:2102.04016*, 2021.
- [70] Q. Liu, L. Xie, H. Wang, and A. L. Yuille, “Semantic-aware knowledge preservation for zero-shot sketch-based image retrieval,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 3662–3671.

- [71] H. Zhang, S. Liu, C. Zhang, W. Ren, R. Wang, and X. Cao, “Sketchnet: Sketch classification with web images,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 1105–1113.
- [72] J. Song, Q. Yu, Y.-Z. Song, T. Xiang, and T. M. Hospedales, “Deep spatial-semantic attention for fine-grained sketch-based image retrieval,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 5551–5560.
- [73] K. Pang, Y.-Z. Song, T. Xiang, and T. M. Hospedales, “Cross-domain generative learning for fine-grained sketch-based image retrieval.,” in *BMVC*, 2017, pp. 1–12.
- [74] J. Collomosse, T. Bui, and H. Jin, “Livesketch: Query perturbations for guided sketch-based visual search,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 2879–2887.
- [75] A. K. Bhunia, Y. Yang, T. M. Hospedales, T. Xiang, and Y.-Z. Song, “Sketch less for more: On-the-fly fine-grained sketch-based image retrieval,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 9779–9788.
- [76] A. Dosovitskiy *et al.*, “An image is worth 16x16 words: Transformers for image recognition at scale,” *ICLR*, 2021.
- [77] Q. Zhang, Z. Lei, Z. Zhang, and S. Z. Li, “Context-aware attention network for image-text retrieval,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3536–3545.
- [78] K.-H. Lee, X. Chen, G. Hua, H. Hu, and X. He, “Stacked cross attention for image-text matching,” *arXiv preprint arXiv:1803.08024*, 2018.
- [79] Y. Yang, N. Jin, K. Lin, M. Guo, and D. Cer, “Neural retrieval for question answering with cross-attention supervised data augmentation,” *arXiv preprint arXiv:2009.13815*, 2020.
- [80] C. Alberti, J. Ling, M. Collins, and D. Reitter, “Fusion of detected objects in text for visual question answering,” *arXiv preprint arXiv:1908.05054*, 2019.
- [81] J. Hessel, A. Holtzman, M. Forbes, R. L. Bras, and Y. Choi, *Clipscore: A reference-free evaluation metric for image captioning*, 2021. arXiv: 2104.08718 [cs.CV].
- [82] S. Tiwary, *Turing Bletchley: A Universal Image Language Representation model by Microsoft*, <https://www.microsoft.com/en-us/research/blog/turing-bletchley-a-universal-image-language-representation-model-by-microsoft/>, [Online; accessed 7-March-2021], 2021.

- [83] I. Tautkute, T. Trzcinski, A. Skorupa, L. Brocki, and K. Marasek, *Deepstyle: Multimodal search engine for fashion and interior design*, 2019. arXiv: 1801.03002 [cs.CV].
- [84] N. Vo *et al.*, “Composing text and image for image retrieval - an empirical odyssey,” in *CVPR*, 2019.
- [85] X. Han *et al.*, “Automatic spatially-aware fashion concept discovery,” in *ICCV*, 2017.
- [86] Y. Chen and L. Bazzani, “Learning joint visual semantic matching embeddings for language-guided retrieval,” in *Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, Springer, 2020, pp. 136–152.
- [87] H. Dong, Z. Wang, Q. Qiu, and G. Sapiro, *Using text to teach image retrieval*, 2020. arXiv: 2011.09928 [cs.LG].
- [88] C. Wang, Z. Sun, L. Zhang, and L. Zhang, “Sketch2tag: Automatic hand-drawn sketch recognition,” *ACM Conference on Multimedia*, Jan. 2012.
- [89] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, “Sketch2photo: Internet image montage,” *ACM Trans. Graph.*, vol. 28, no. 5, p. 110, Dec. 2009.
- [90] S. Changpinyo, J. Pont-Tuset, V. Ferrari, and R. Soricut, “Telling the what while pointing to the where: Multimodal queries for image retrieval,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 136–12 146.
- [91] S. Dey, A. Dutta, S. K. Ghosh, E. Valveny, J. Lladós, and U. Pal, “Learning cross-modal deep embeddings for multi-object image retrieval using text and sketch,” in *2018 24th international conference on pattern recognition (ICPR)*, IEEE, 2018, pp. 916–921.
- [92] T. Han and D. Schlangen, “Draw and tell: Multimodal descriptions outperform verbal- or sketch-only descriptions in an image retrieval task,” in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Taipei, Taiwan: Asian Federation of Natural Language Processing, Nov. 2017, pp. 361–365.
- [93] T. X. Jifei Song Yi-zhe Song and T. Hospedales, “Fine-grained image retrieval: The text/sketch input dilemma,” in *Proceedings of the British Machine Vision Conference (BMVC)*, G. B. Tae-Kyun Kim Stefanos Zafeiriou and K. Mikolajczyk, Eds., BMVA Press, Sep. 2017, pp. 45.1–45.12.

- [94] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.
- [95] E. Ben-Baruch *et al.*, *Asymmetric loss for multi-label classification*, 2020. arXiv: 2009.14119 [cs.CV].
- [96] M. Li, Z. Lin, R. Měch, E. Yumer, and D. Ramanan, “Photo-sketching: Inferring contour drawings from images,” *WACV*, 2019.
- [97] C. Gao, Q. Liu, Q. Xu, L. Wang, J. Liu, and C. Zou, “Sketchycoco: Image generation from freehand scene sketches,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020.
- [98] J. Pont-Tuset, J. Uijlings, S. Changpinyo, R. Soricut, and V. Ferrari, “Connecting vision and language with localized narratives,” in *ECCV*, 2020.
- [99] A. Karpathy and L. Fei-Fei, *Deep visual-semantic alignments for generating image descriptions*, 2015. arXiv: 1412.2306 [cs.CV].
- [100] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [101] G. Ilharco *et al.*, *Openclip*, version 0.1, If you use this software, please cite it as below., Jul. 2021.
- [102] R. Salakhutdinov and G. Hinton, “Deep Boltzmann machines,” in *Proceedings of the International Conference on Artificial Intelligence and Statistics*, vol. 5, 2009, pp. 448–455.
- [103] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, “Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 609–616.
- [104] I. Goodfellow *et al.*, “Generative adversarial nets,” in *Advances in neural information processing systems*, 2014, pp. 2672–2680.
- [105] A. Radford, L. Metz, and S. Chintala, “Unsupervised representation learning with deep convolutional generative adversarial networks,” *arXiv preprint arXiv:1511.06434*, 2015.
- [106] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.

- [107] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, “Draw: A recurrent neural network for image generation,” in *ICML*, 2015.
- [108] X. Yan, J. Yang, K. Sohn, and H. Lee, “Attribute2image: Conditional image generation from visual attributes,” in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [109] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros, “View synthesis by appearance flow,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [110] Y. Güçlütürk, U. Güçlü, R. van Lier, and M. A. van Gerven, “Convolutional sketch inversion,” in *Proceeding of the ECCV workshop on VISART Where Computer Vision Meets Art*, 2016.
- [111] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, “Generative visual manipulation on the natural image manifold,” in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [112] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arxiv*, 2016.
- [113] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, *et al.*, “Conditional image generation with pixelcnn decoders,” in *Advances In Neural Information Processing Systems*, 2016, pp. 4790–4798.
- [114] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” *ECCV*, 2016.
- [115] L. McMillan and G. Bishop, “Plenoptic modeling: An image-based rendering system,” in *Proceedings of the 22Nd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’95, 1995, pp. 39–46.
- [116] A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, 1033–1038 vol.2.
- [117] J. Hays and A. A. Efros, “Scene completion using millions of photographs,” in *ACM SIGGRAPH 2007 Papers*, ser. SIGGRAPH ’07, 2007.
- [118] T. Chen, M.-M. Cheng, P. Tan, A. Shamir, and S.-M. Hu, “Sketch2photo: Internet image montage,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 5, p. 124, 2009.
- [119] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, “Patchmatch: A randomized correspondence algorithm for structural image editing,” *ACM Transactions on Graphics-TOG*, vol. 28, no. 3, p. 24, 2009.

- [120] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Let there be Color!: Joint End-to-end Learning of Global and Local Image Priors for Automatic Image Colorization with Simultaneous Classification,” *ACM Transactions on Graphics (Proc. of SIGGRAPH 2016)*, vol. 35, no. 4, 2016.
- [121] G. Larsson, M. Maire, and G. Shakhnarovich, “Learning representations for automatic colorization,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [122] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Neural Photo Editing with Intrinsic Adversarial Networks,” *ArXiv e-prints*, Sep. 2016. arXiv: 1609.07093 [cs.LG].
- [123] A. Nguyen, A. Dosovitskiy, J. Yosinski, T. Brox, and J. Clune, “Synthesizing the preferred inputs for neurons in neural networks via deep generator networks,” in *Advances in Neural Information Processing Systems 29*, 2016.
- [124] A. Nguyen, J. Yosinski, Y. Bengio, A. Dosovitskiy, and J. Clune, “Plug & play generative networks: Conditional iterative generation of images in latent space,” in *arXiv pre-print*, 2016.
- [125] X. Wang and A. Gupta, “Generative image modeling using style and structure adversarial networks,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [126] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text-to-image synthesis,” in *Proceedings of The 33rd International Conference on Machine Learning*, 2016.
- [127] S. Reed, Z. Akata, S. Mohan, S. Tenka, B. Schiele, and H. Lee, “Learning what and where to draw,” in *NIPS*, 2016.
- [128] C. Ledig *et al.*, “Photo-realistic single image super-resolution using a generative adversarial network,” *arXiv preprint arXiv:1609.04802*, 2016.
- [129] C. Kaae Sønderby, J. Caballero, L. Theis, W. Shi, and F. Huszár, “Amortised MAP Inference for Image Super-resolution,” *ArXiv e-prints*, Oct. 2016. arXiv: 1610.04490 [cs.CV].
- [130] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros, “Context encoders: Feature learning by inpainting,” in *IEEE Conf. Comput. Vis. Pattern Recog.*, 2016.
- [131] A. v. d. Oord, N. Kalchbrenner, and K. Kavukcuoglu, “Pixel recurrent neural networks,” in *Proceedings of the 33th International Conference on Machine Learning, ICML 2016*, 2016.

- [132] R. Yeh, C. Chen, T. Y. Lim, M. Hasegawa-Johnson, and M. N. Do, “Semantic image inpainting with perceptual and contextual losses,” *arXiv preprint arXiv:1607.07539*, 2016.
- [133] D. Yoo, N. Kim, S. Park, A. S. Paek, and I. S. Kweon, “Pixel-level domain transfer,” in *European Conference on Computer Vision*, Springer, 2016, pp. 517–532.
- [134] C. Li and M. Wand, “Precomputed real-time texture synthesis with markovian generative adversarial networks,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [135] A. J. Champandard, “Semantic style transfer and turning two-bit doodles into fine artworks,” *arXiv preprint arXiv:1603.01768*, 2016.
- [136] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2414–2423.
- [137] A. Levin, D. Lischinski, and Y. Weiss, “Colorization using optimization,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 23, 2004, pp. 689–694.
- [138] Y. Qu, T.-T. Wong, and P.-A. Heng, “Manga colorization,” *ACM Transactions on Graphics (SIGGRAPH 2006 issue)*, vol. 25, no. 3, pp. 1214–1220, Jul. 2006.
- [139] D. Sykora, J. Dingliana, and S. Collins, “Lazybrush: Flexible painting tool for hand-drawn cartoons,” in *Computer Graphics Forum*, Wiley Online Library, vol. 28, 2009, pp. 599–608.
- [140] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, “Scribblesup: Scribble-supervised convolutional networks for semantic segmentation,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [141] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. 2015, pp. 234–241.
- [142] E. Simo-Serra, S. Iizuka, K. Sasaki, and H. Ishikawa, “Learning to Simplify: Fully Convolutional Networks for Rough Sketch Cleanup,” *ACM Transactions on Graphics (SIGGRAPH)*, vol. 35, no. 4, 2016.
- [143] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [144] A. Odena, V. Dumoulin, and C. Olah, *Deconvolution and checkerboard artifacts*, <http://distill.pub/2016/deconv-checkerboard/>, 2016.
- [145] A. Dosovitskiy and T. Brox, “Generating images with perceptual similarity metrics based on deep networks,” in *NIPS*, 2016.
- [146] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 694–711, ISBN: 978-3-319-46475-6.
- [147] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.
- [148] F. Yu, Y. Zhang, S. Song, A. Seff, and J. Xiao, “LSUN: construction of a large-scale image dataset using deep learning with humans in the loop,” *CoRR*, vol. abs/1506.03365, 2015.
- [149] H. Winnemöller, J. E. Kyprianidis, and S. C. Olsen, “Xdog: An extended difference-of-gaussians compendium including advanced image stylization,” *Computers & Graphics*, vol. 36, no. 6, pp. 740–753, 2012.
- [150] *Create filter gallery photocopy effect with single step in photoshop*, https://www.youtube.com/watch?v=QNmniB_5Nz0/, 2016.
- [151] *Convert photo to line drawing*, <https://www.youtube.com/watch?v=Gyu2yPwiQvA>, 2012.
- [152] X. Wang and X. Tang, “Face photo-sketch synthesis and recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 11, pp. 1955–1967, 2009.
- [153] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.
- [154] L. McMillan and G. Bishop, “Plenoptic modeling: An image-based rendering system,” in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, ACM, 1995, pp. 39–46.
- [155] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi, “Photo clip art,” *ACM Transactions on Graphics (SIGGRAPH 2007)*, vol. 26, no. 3, p. 3, Aug. 2007.

- [156] A. A. Efros and T. K. Leung, “Texture synthesis by non-parametric sampling,” in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, IEEE, vol. 2, 1999, pp. 1033–1038.
- [157] J. Hays and A. A. Efros, “Scene completion using millions of photographs,” in *ACM Transactions on Graphics (TOG)*, ACM, vol. 26, 2007, p. 4.
- [158] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. P. Smolley, “Least squares generative adversarial networks,” *arXiv preprint ArXiv:1611.04076*, 2016.
- [159] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, “Image-to-image translation with conditional adversarial networks,” *arXiv preprint arXiv:1611.07004*, 2016.
- [160] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, “Generative visual manipulation on the natural image manifold,” in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016.
- [161] A. Odena, C. Olah, and J. Shlens, “Conditional image synthesis with auxiliary classifier gans,” *arXiv preprint arXiv:1610.09585*, 2016.
- [162] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee, “Generative adversarial text to image synthesis,” in *Proceedings of The 33rd International Conference on Machine Learning*, vol. 3, 2016.
- [163] R. Zhang *et al.*, “Real-time user-guided image colorization with learned deep priors,” *ACM Transactions on Graphics (TOG)*, vol. 9, no. 4, 2017.
- [164] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2414–2423.
- [165] Y. Liu, Z. Qin, Z. Luo, and H. Wang, “Auto-painter: Cartoon image generation from sketch by using conditional generative adversarial networks,” *arXiv preprint arXiv:1705.01908*, 2017.
- [166] A. A. Efros and W. T. Freeman, “Image quilting for texture synthesis and transfer,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 2001, pp. 341–346.
- [167] L.-Y. Wei and M. Levoy, “Fast texture synthesis using tree-structured vector quantization,” in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 479–488.

- [168] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, “Image analogies,” in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 2001, pp. 327–340.
- [169] H. Fang and J. C. Hart, “Textureshop: Texture synthesis as a photograph editing tool,” *ACM Trans. Graph.*, vol. 23, no. 3, pp. 354–359, Aug. 2004.
- [170] L. Gatys, A. S. Ecker, and M. Bethge, “Texture synthesis using convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2015, pp. 262–270.
- [171] H. Zhang and K. Dana, “Multi-style generative network for real-time transfer,” *arXiv preprint arXiv:1703.06953*, 2017.
- [172] X. Huang and S. Belongie, “Arbitrary style transfer in real-time with adaptive instance normalization,” *arXiv preprint arXiv:1703.06868*, 2017.
- [173] Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, and M.-H. Yang, “Diversified texture synthesis with feed-forward networks,” *arXiv preprint arXiv:1703.01664*, 2017.
- [174] D. Ulyanov, V. Lebedev, A. Vedaldi, and V. Lempitsky, “Texture networks: Feed-forward synthesis of textures and stylized images,” in *International Conference on Machine Learning (ICML)*, 2016.
- [175] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution,” in *European Conference on Computer Vision*, Springer, 2016, pp. 694–711.
- [176] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, “Controlling perceptual factors in neural style transfer,” *arXiv preprint arXiv:1611.07865*, 2016.
- [177] N. Jetchev, U. Bergmann, and R. Vollgraf, “Texture synthesis with spatial generative adversarial networks,” *arXiv preprint arXiv:1611.08207*, 2016.
- [178] U. Bergmann, N. Jetchev, and R. Vollgraf, “Learning texture manifolds with the periodic spatial gan,” *arXiv preprint arXiv:1705.06566*, 2017.
- [179] C. Yang, X. Lu, Z. Lin, E. Shechtman, O. Wang, and H. Li, “High-resolution image inpainting using multi-scale neural patch synthesis,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.
- [180] S. Iizuka, E. Simo-Serra, and H. Ishikawa, “Globally and Locally Consistent Image Completion,” *ACM Transactions on Graphics (Proc. of SIGGRAPH 2017)*, vol. 36, no. 4, 107:1–107:14, 2017.

- [181] C. Lassner, G. Pons-Moll, and P. V. Gehler, “A generative model for people in clothing,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2017.
- [182] X. Liang *et al.*, “Human parsing with contextualized convolutional neural network,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1386–1394.
- [183] A. Yu and K. Grauman, “Fine-grained visual comparisons with local learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 192–199.
- [184] X. Liang *et al.*, “Deep human parsing with active template regression,” *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 12, pp. 2402–2414, Dec. 2015.
- [185] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang, “Deepfashion: Powering robust clothes recognition and retrieval with rich annotations,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [186] Z. Liu, S. Yan, P. Luo, X. Wang, and X. Tang, “Fashion landmark detection in the wild,” in *European Conference on Computer Vision (ECCV)*, 2016.
- [187] S. Xie and Z. Tu, “Holistically-nested edge detection,” in *Proceedings of IEEE International Conference on Computer Vision*, 2015.
- [188] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [189] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.