Emergent Capabilities of Generative Models: "Software 3.0" and Beyond

by

Alexander Andonian

B.S., Bates College (2017)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2021

Author . . . .
Department of Electrical Engineering and Computer Science
August 27, 2021

Certified by . . . .
Aude Oliva
Senior Research Scientist, Computer Science Artificial Intelligence Laboratory
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Emergent Capabilities of Generative Models: "Software 3.0" and Beyond

by

Alexander Andonian

Submitted to the Department of Electrical Engineering and Computer Science
on August 27, 2021, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Modern AI algorithms are rapidly becoming ubiquitous in everyday life and have even been touted as the new "Software 2.0" stack by prominent researchers in the field. Indeed, these algorithms are fundamentally changing the way we interact with, potentially even how we will *program* computers to achieve desired outcomes. In this thesis, we advocate that wielding control over these increasingly powerful models is important for progression in the field, and more importantly, for ensuring that models deployed in the real world behave in the ways we would like and preventing cases where they may do unintended harm. First, we present an empirical study in which we train a large-scale Generative Adversarial Network (GAN) on the MIT Places365 dataset, achieving state-of-the-art Inception scores and Fréchet Inception distance, metrics that are used to evaluate image synthesis quality. We then introduce a GAN framework, *GANalyze*, that allows one to make targeted manipulations to various cognitive attributes of GAN generated imagery, such as memorability and emotional valence, and use this framework to surface "visual definitions" of these properties. Through behavioral experiments, we verify that our method discovers image manipulations that causally affect human memory performance. Finally, we build on this framework by incorporating a powerful new pretrained text-image semantic similarity model to create a novel image editing application that allows users to "paint by word." All together, this progression of work underscores the advantages of the emerging "Software 3.0" stack, whereby programmers are tasked with orchestrating and finetuning the interactions between large-scale *foundation* models to carry out higher-order tasks.

Thesis Supervisor: Aude Oliva
Title: Senior Research Scientist, Computer Science Artificial Intelligence Laboratory

Acknowledgments

Thank you to my advisor Aude Oliva, for all of her support, guidance, energy and trust over the course of a this work, especially during the unprecedented year of 2020.

Thank you to every member of the entire Oliva Lab, for you have all influenced and enhanced my life in ways too numerous to list here.

Finally, thank you to my family - my Mom, Dad and Sammy - who have always been my greatest source of support and inspiration.

Contents

1 Introduction **17**

1.1 A new type of software 18

1.2 Software 3.0 19

1.3 A case study: Controllable Neural Image Synthesis 22

2 Generative Modeling **25**

2.1 Introduction 25

2.2 Related Work 27

2.2.1 GAN Framework 27

2.2.2 Evaluation Metrics 29

2.3 Methods 30

2.3.1 Network Architectures 30

2.3.2 Optimization and Inference 30

2.3.3 Implementation 31

2.4 Experiments 33

2.5 Analysis 35

2.6 Conclusion 38

3 GANalyze **41**

3.1 Introduction 42

3.1.1 Related work 44

3.2 Model 45

3.2.1 Formulation 45

3.2.2 Implementation 45
3.3 Experiments 46
3.3.1 Model validation 46
3.3.2 Emerging factors 46
3.3.3 Realness 49
3.3.4 Do our changes causally affect memory? 51
3.4 Other image properties 54
3.5 Conclusion 56

4 Paint By Word **61**
4.1 Introduction 61
4.2 Related Work 63
4.3 Method 65
4.3.1 Semantic consistency in a region 67
4.3.2 Generative modeling in a region 70
4.3.3 Avoiding adversarial attack using CMA 72
4.4 Results 74
4.4.1 Testing full image generation 74
4.4.2 A large scale user study of bedroom image edits 77
4.4.3 Demonstrating paint-by-word on a broader diversity of image domains 78
4.5 Discussion 79

5 Conclusion **81**
5.1 Contributions 81
5.2 Future directions 81

List of Figures

1-1 Comparing the I/O structure of software 1.0 vs software 2.0. In software 1.0, programmers provide inputs and the explicit instructions to be performed on those inputs to produce outputs. In contrast, software 2.0 consumes examples of inputs and outputs and must produce models that have learned the rules of the desired program. 19

1-2 **Software 3.0** is concerned with harnessing the emergent capabilities of existing pretrained AI foundation models and orchestrating their interactions to carry out higher-order tasks. 21

2-1 Preliminary samples from a 256x256 pixel resolution BigGAN-deep model trained on the Places365-Challenge dataset containing over 8 million annotated scene-centric images. A truncation threshold of 0.5 was used for sampling the random noise vector z. This model was only trained for 100k iterations. 26

2-2 Typical plots of various metrics monitored during the training phase . 34

2-3 A collage of samples from the fully trained 256x256 pixel resolution model highlights the diversity and quality of generated samples. . . . 35

2-4 The effects of varying the amount of truncation during sampling from the 128 pixel BigGAN model for the "Skyscraper" scene category. From left to right, the threshold is set to 2, 1, 0.5, 0.04. 36

2-5 FID vs. IS at 128 pixel resolutions. 36

2-6 Linear interpolations between two randomly sampled coordinates in Z for a given class y shows that the latent space has developed smooth transitions, with each sample appearing as a visually plausible intermediate version of its right and left neighbors. 37

2-7 Interpolations between classes c with z held constant. Scene semantics and spatial layouts are frequently maintained between endpoints. . . 38

3-1 **Schematic of the model**. The model learns how to transform a $\mathbf{z}$ vector such that when fed to a Generator, the resulting image's property of interest changes. The transformation is achieved by the *Transformer*, who moves the $\mathbf{z}$ vector along a learned direction, θ, in the Generator's latent space. An *Assessor module* (e.g., MemNet) predicts the property of interest (e.g., memorability). Finally, α acts as a knob to set the desired degree of change in the Assessor value (e.g., MemNet score), telling the *Transformer* how far to move along θ. 41

3-2 **Model results**. A) Graph shows the mean MemNet score across the images in every α condition. Our model successfully learned how to modify a GAN image to decrease (negative α) or increase (positive α) its MemNet score. B) List of emerging factors potentially underlying the effect observed in (A), and graph of how they change in function of α. The factors emerged from visualizations generated by the GANalyze framework (examples shown in Figures 3-3 and in the Supplementary Materials). 47

3-3 **Examples of generated images** along the memorability dimension. The middle column represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed to create a series of clone images more or less memorable. 48

3-4 **Realness measures as a function of** α. A) Two popular automatic measures for evaluating the realness of a set of GAN images. Note that lower FID values indicate higher realness. B) Human fakeness discriminability, measured as the mean proportion correct in a 2AFC-task in which AMT workers had to discriminate GAN-images (fake) from real photographs. 50

3-5 **Schematic of the visual memory game**. Each image is shown for 600 ms, with a blank interstimulus interval of 800 ms. Workers are asked to respond whenever they recognize a repeat of a previously shown image. For a correct response, the frame around the image briefly turns green. A red frame, on the other hand, indicates a mistake. 50

3-6 **Human memory performance** for images modified according to different Assessors: A) MemNet, B) Object size and C) AestheticsNet. Performance is measured as the hit rate across all images and workers in the memory game for each property. 53

3-7 **Model results for additional Assessors**. Graphs show the mean Assessor (A: Object size, B: AestheticsNet, C: EmoNet) score across images in every α condition. 54

3-8 **Examples of generated images** along the aesthetics (top) and emotional valence (bottom) dimension. The middle column represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed. An images' Assessor score is shown in its top left corner. More examples are included in the Supplementary Materials. 57

3-9 **Comparison of examples generated according to different Assessors.** The top row represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed to create a series of images with a higher or lower Assessor value. The respective Assessor values are indicated in the top left corner. Note that for object size, we used a different α range: {-0.8,0.8}. 58

4-1 An overview of our method. (a) we begin with an image x_0 that is decoded from a latent z_0 by a pretranied generative model of realistic images $x_0 = G(z_0)$. (b) To edit the image, the user selects a region m and gives an arbitrary run of descriptive text t. The image generator $G(z) = h(f(z))$ is decomposed to expose a image representation w that can be split spatially between the painted and unpainted region. Then a text-image semantic similarity model $C(x, t)$ is used to define a semantic consistency loss that is used together with an image loss (minimizing changes outside the mask) to optimize w inside the region to make the synthesized region match the given text. 62

4-2 Comparing the effect of optimizing with and without a spatially split generator. The masked loss $C_{t,m}(x)$ is used to generate modifications of a generated image (a) that matches the description "A photo of a purple bed." where the loss is masked to the user-painted region m as shown in (b). The modification is done in two ways. In (c), the original unmodified StyleGAN2 [57] model (trained on LSUN [117] bedrooms) is used and an optimizer finds a $G(z)$ that minimizes the masked loss. In (d), the optimization is done over a spatially-split generator $G_{z,m}(w)$. Although in both cases, the semantic loss examines only the region of the bed, only the split generator allows changes to be made in the user-specified region without altering the rest of the scene. 66

4-3 Avoiding adversarial examples when optimizing an image to match the description "A photo of a yellow bird flying through the air." The gradient descent method (Adam) obtains good scores while yielding an unrealistic image that looks neither like it is flying, nor like a real bird. In practice, we obtain better results by using the non-gradient sampling method (CMA-ES). Although samples do not achieve scores that are as good, the generated samples avoid becoming adversarial, and user studies (Section 4.4.1) show that results are both more realistic and accurate in practice. 68

4-4 Qualitative comparison of a handful of birds from the user study conducted in Section 4.4.1. Our method (third row) is applied to synthesizing full images based on description of birds. It is compared to an ablation where the Adam optimizer is used alone instead of CMA-ES; and we also compare our method to DALL-E [88]. Note that images from our method tend to be more realistic more often when compared to the other implementations. 69

4-5 User study comparing the semantic accuracy and realism of our basic method (Equation 4.1). We ask users to compare images generated by our method to two baselines; three workers examined each pair of images to make each assessment. We compared our method as implemented using CMA [34] versus to our method using the Adam optimizer [61]. Our CMA method outperforms our Adam method in both accuracy and realism. To compare to a state-of-the-art baseline, we also compare our method to DALL-E [88], which is text-to-image system that was trained on a much broader and more difficult domain. Our method outperforms DALL-E in this setting, possibly because here our generator is trained to specialize on the narrow domain of birds. In this test, our method pairs a 256-pixel StyleGAN2 [55, 57] trained on CUB birds [111] with the CLIP [84] `ViT/32B` text-image semantic similarity network. 73

4-6 Representative successes and failures in a large-scale user study of image edits. 3000 edits were made on 300 randomly sampled StyleGAN2 bedroom outputs, and the edited image was compared to the original by crowdsourced workers. In cases a-e, 2/2 workers rated the modified image as a better match to the word than the original, and in case (f), both workers rated the modified image as a worse match to the word. In case (a) 2/2 workers rated the edited image as more realistic than the original, in cases (b) and (f) the two workers split on realism, and in cases c-e, both workers rated the edited image as less realistic than the original. 74

4-7 A detailed breakdown of user edits that were considered accurate by 2/2 workers, according to individual word. The most accurate edits among those we tested are for the color orange; and the least accurate are for the word "Scandinavian". Edits that were considered less realistic than the original are shown in orange. 75

4-8 Exploring edits using our method with BigGAN. Because BigGAN models have been pretrained on very broad distributions, they provide an opportunity to experiment with a broad range of types of edits. . . 76

List of Tables

2.1 Computational configurations employed in this study. The 128x128 models trained on Google Cloud and Core Scientific used a single multi-GPU machine, which allows for low-latency GPU-to-GPU synchronization. The 256x256 model was trained for 100k iterations using 42 2xGPU nodes on IBM Cloud. Despite having an order of magnitude more GPUs and fewer gradient accumulations, the rate of training in this configuration is significantly lower than its single node counterparts, suggesting that the distributed nature of this setup introduces a heavy synchronization time penalty. In contrast, the infiniband network connections in the MIT-Satori cluster mitigated these synchronization penalties. 32

2.2 Evaluation of models for different datasets, resolutions and training iterations via Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better). 33

3.1 **Relation between emerging factors and human memory performance.** We show the output of logistic mixed-effects regressions. From left to right: the regression weight, the confidence interval (CI) for that weight, the p-value for statistical significance, and Tjur's coefficient of discrimination (D), being the regression model's goodness of fit [108]. The emerging factor values were normalized before running the regression models. . . . 55

4.1 User study of edits, by semantic category. Percentages shown include only cases where both evaluations agree. “Accurate” counts cases where the edited image is a better match for the text. “Realistic” counts cases where the edited image is more realistic than the original. In both cases the opposite counts are also shown. 77

Chapter 1

Introduction

Since the resurgence of deep neural networks at the beginning of the last decade, modern deep learning continues to triumph in domains such as computer vision and natural language processing. As a result, these algorithms are rapidly infiltrating many aspects of modern day life in ways that we can and cannot see [95]. For example, these algorithms may influence what content and media we consume, which purchases we make, potentially even how decisions will be made about our health [20]. Modern computer vision systems are now deployed in sensitive contexts such as autonomous vehicles where the consequences can be life or death [45]. Furthermore, researchers are only beginning to understand more subtle downstream consequences and dangers of deploying biased classifiers in production machine learning systems, which has already been identified as potentially reinforcing inequality [64]. Within this context, it becomes clear that effectively deploying these algorithms requires an understanding of how and why they work, their strengths and weaknesses, and when and where they can operate safely. **As the size, capabilities and responsibilities of these models continue to grow, so should our ability to interact with and control them to achieve a desired outcome.** Crucially, this ability should be used to exert fine-grained control over the behavior of the models to ensure that they operate in a manner we deem most beneficial and safe for all.

The ability to specify and shape the output behaviors of these models - bend these models to our will - is not just important from a safety perspective, it represents fundamental progress in the field and opens up avenues for advancements in other domains of science and business as well [4, 19, 63, 73, 97, 113]. Deep learning models have even bridged the gap to creative domains where artists and musicians are already finding new ways of creative expression using these models as their instruments and tools [94]. Most broadly, it is changing the way we interface with, potentially even *program* computers. For that reason, understanding and controlling the emergent capabilities of generative models serve the primary focuses of this work.

1.1 A new type of software

The notion of deep neural networks as "Software 2.0" was first characterized by Karpathy [52] as model of software development that is model and data driven as opposed to code-driven. To summarize, Karpathy contrasts two types of software:

- **Software 1.0**: The software we are all familiar with, written in a language such as C++ or Python, whereby a programmer provides explicit, step-by-step instructions to be carried out by the machine to achieve some desired behavior.

- **Software 2.0**: Exemplified by the modern deep learning training workflow. Instead of providing the inputs to a program and the exact rules for how to operate on that input, programmers provide input data and output data, in essence specifying a goal on the desired behaviour or outcomes by example, not instruction. It is now the task of the machine to learn and execute the rules and behaviors of the program. In practice, software 2.0 programs are written by roughly framing a task as optimization procedure specified by some evaluation criterion and providing a neural network with the right inductive biases to effectively solve the optimization problem. The challenge is curating the right data, designing the right loss objective, and structuring the deep network correctly.

Indeed, this new way of thinking about solving problems with software has brought about significant advancements. This continued success supports the notion that deep

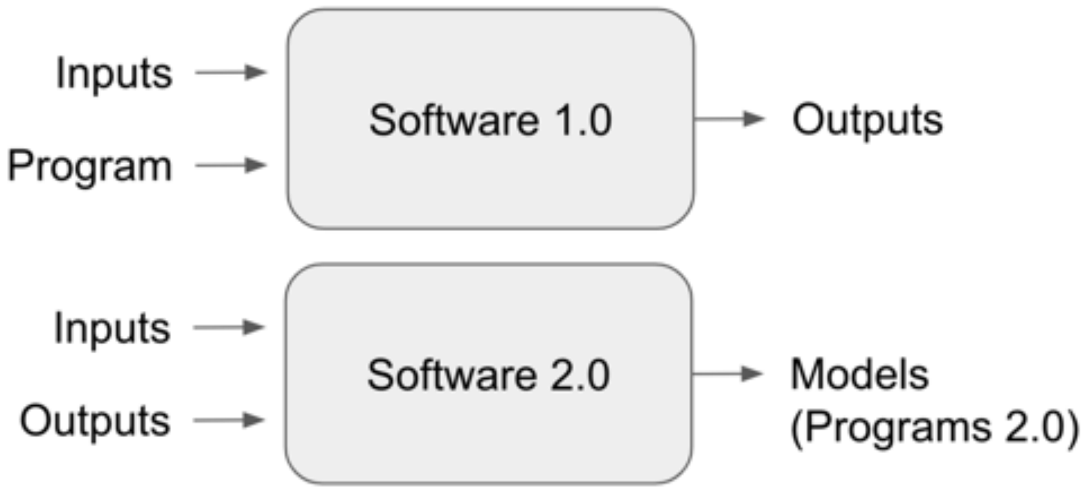


Figure 1-1: Comparing the I/O structure of software 1.0 vs software 2.0. In software 1.0, programmers provide inputs and the explicit instructions to be performed on those inputs to produce outputs. In contrast, software 2.0 consumes examples of inputs and outputs and must produce models that have learned the rules of the desired program.

learning thrives at scale, and this progress has come with a "voracious" appetite for computing power [107]. Currently, *bigger really is better* in the world of deep learning. However, training these large scale models is quickly becoming economically, technically and environmentally unsustainable [107]. Already, the most impressive recent feats of AI are obtained through massive investments from a select few companies to train models on supercomputer-scale hardware, and not all models are shared with the rest of the research community. Given these circumstances, how should AI research proceed?

1.2 Software 3.0

The outcomes of these massive investments to train models larger than ever before suggest that we may already be on the precipice of yet another type of software. Indeed, the rise of such models, ones that are trained on broad data at scale making them adaptable to a wide array of downstream tasks, has been so central to recent progress in AI that these models have earned the contentious name of *foundation models* in a recent comprehensive report [11]. Foundation models continue to demonstrate that interesting and sophisticated behavior can emerge even when these models are trained on relatively straightforward software 2.0 objectives. Most notably, *GPT-3* [15] is a 175 billion parameter autoregressive language model that is trained with the very simple objective of correctly predicting the next word in a sequence given the preceding words yet develops a fairly impressive set of emergent capabilities in the pursuit of doing this prediction problem over massive internet scale text corpora. For example, by providing the right input prompt, it demonstrates an impressive ability to author entire news articles, essays, and other types of literary works with enough fluency to fool the causal reader into thinking it was written by a human. Similarly large models such as DALL-E [87] were trained to align images with their captions on a dataset of over 400 million image-text pairs, and by doing so is able to imagine unusual novel objects such as an "avocado armchair" despite never encountering one during training. Most recently, derivatives of *GPT-3* such as the open source effort GPT-Neo [10], have shown the ability to correctly complete basic coding interview questions [41], suggesting that models like it may be writing code on our behalf in the future.

Harnessing emergent capabilities

In light of these recent advancements, we argue that a version of software 3.0 builds directly on the fruits of software 2.0. ***Software 3.0* is concerned with identifying and harnessing the emergent capabilities of these pretrained *foundation* models.** The software 3.0 programmer will spend very little time training models purely from scratch - this task is left to a select few organizations with the financial and computational resources to do so. For example, the single training run to train GPT-3 reportedly cost a staggering 12 million dollars [98]. Instead, she has access to vast catalog of pretrained foundation models at her disposal - her task is to discover and employ effective ways of interacting with and manipulating these models such that they are "primed" and ready to carry out the desired task at hand. A wealth of initial work on "prompt engineering" is quickly emerging as access to these large scale

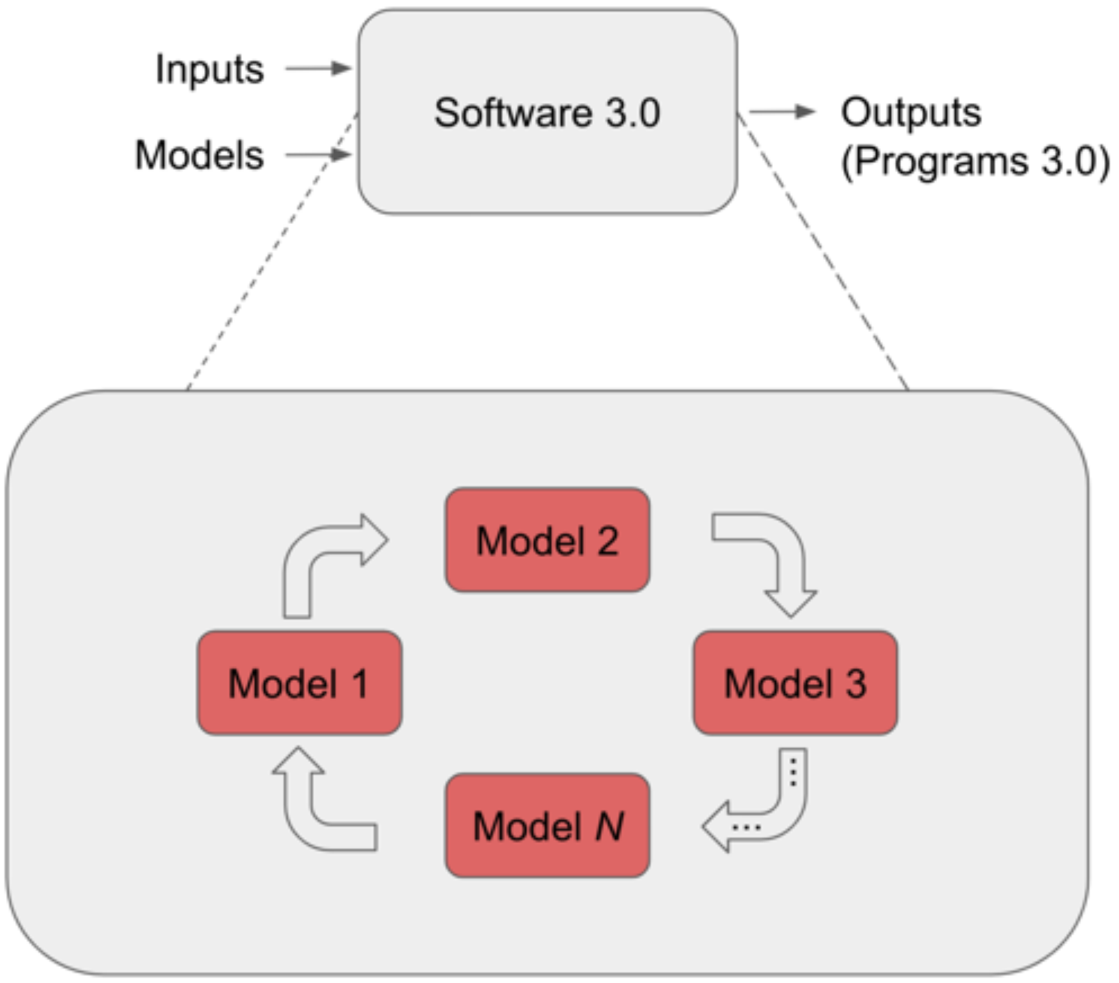


Figure 1-2: **Software 3.0** is concerned with harnessing the emergent capabilities of existing pretrained AI foundation models and orchestrating their interactions to carry out higher-order tasks.

models spreads to more researchers [26, 65, 67, 96, 101].

Software 3.0 will likely extend beyond the study of a single model in isolation. Due to their computationally homogeneous nature, these modules compose and cooperate together gracefully with the ability to "prompt" and provide feedback to one another in an interacting system. In this way, an can be effectively "glued/melded" together to form a cohesive whole in the same way that the human brain combines a jumble of specialized substructures in a way that works.

1.3 A case study: Controllable Neural Image Synthesis

In this work, we broadly forward the notion of "Software 3.0" as both an effective emerging approach to forwarding AI research and a design principle for constructing more sophisticated AI systems. While it will not replace software 2.0 entirely - in fact it depends on it - we hypothesize that it will become a rich and prominent approach to A.I. research in the near-future, particularly for the vast majority of researchers without access to near infinite amounts of compute. We support this hypothesis with a line of research at the intersection of computer vision/graphics, cognitive science and natural language understanding that documents a successful transition from software 2.0 to 3.0 in detail. In particular, we demonstrate the advantages of transitioning to software 3.0 research in the context of learning to control a class of successful generative models for image synthesis called Generative Adversarial Networks (GANs).

In Chapter 2, we report an empirical study on training a large-scale, class conditional image GAN on the Places365 dataset and demonstrate basic techniques for obtaining control over the generated output images. Importantly, we detail the compute resources needed to train among the earliest and smallest foundation models to highlight the growing inaccessibility of software 2.0 research.

In Chapter 3, we introduce an additional pretrained network module which works in cooperation with the GAN to manipulate a particular attribute of the generated output image. Moreover, because this additional network was trained on data collected in a human memory experiment, the combined system allows us to create "visual definitions" of measured cognitive attributes, which can be difficult to explicitly articulate in words (i.e., what does it *look like* when an image is made to be more memorable to a human? A follow-up user study confirmed that the emergent modification patterns do, in fact, have the desired downstream effects during human memory experiments, thus demonstrating how the strengths of these models can be used to surface scientific insights such emergent properties of human visual memory.

Lastly, in Chapter 4, we demonstrate how the release of a single next-generation foundation model immediately paves the way for a new class of powerful image-editing applications further enhanced with a user-friendly natural language interface.

Finally, we conclude with some remarks about the current rate of progress in AI and some ways in which research may rapidly evolve beyond software 3.0.

Chapter 2

Generative Modeling

Despite recent progress in generative image modeling, successfully generating high-resolution, diverse samples from complex scene-centric datasets such as the Places365 dataset remains an open challenge. To this end, we train Generative Adversarial Networks at large scale. Similar to previously published work, we find that our trained generator is amenable to a simple "truncation trick," allowing fine control over the trade-off between sample fidelity and variety by reducing the variance of the Generator's input. Our experiments lead to models that set the new state of the art in class-conditional image synthesis on the Places365 dataset. When trained at 128×128 resolution, our models (BigGANs) achieve an Inception Score (IS) of 46.220 and Frechet Inception Distance (FID) of 2.88, establishing strong baseline performance on scene-centric image synthesis.

2.1 Introduction

Generative Adversarial Networks (GANs) have garnered tremendous interest from the computer vision and machine learning research communities, driving rapid progress in generative image modeling [29], captured the eyes and ears of graphic artists and musicians alike presenting new avenues for artistic expression [25, 94], and have even have acquired the concern of policy makers and governments for their potential for misuse [58]. Prominent work such *BigGAN* [13] has shown that applying recent empirical and theoretical insights at scale can drastically improve performance on the difficult task of generating diverse, high-resolution samples from complex datasets. In addition to maintaining state-of-art performance on ImageNet generation, a growing body of evidence suggests that BigGAN can reliably factor complex visual structure into interpretable low-dimensional representations, a necessary prerequisite to developing methods that reason about these representations. BigGAN's latent space reveals simple factorizations such as color, luminance, rotations [49], as well as complex, cognitive dimensions such as memorability, aesthetics and emotional valance [28].

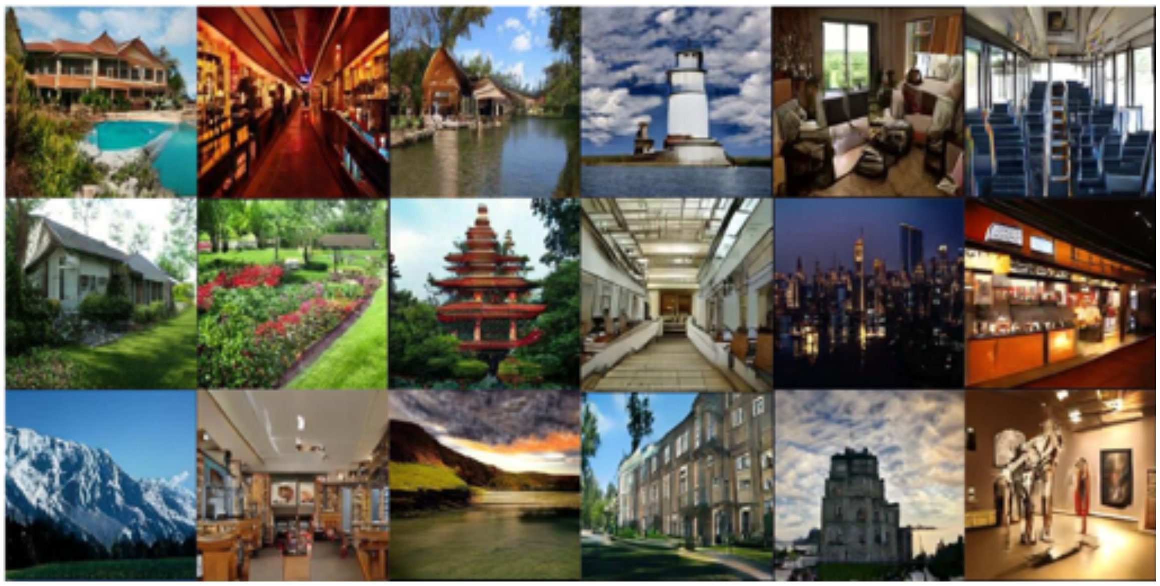


Figure 2-1: Preliminary samples from a 256x256 pixel resolution BigGAN-deep model trained on the Places365-Challenge dataset containing over 8 million annotated scene-centric images. A truncation threshold of 0.5 was used for sampling the random noise vector z. This model was only trained for 100k iterations.

While unconditional GANs have previously been deployed in a wide variety of visual domains, the progress of high-resolution, class-conditional GANs has been measured almost exclusively with respect to object-related, ImageNet-based metrics. **The primary objective of this work is to establish state-of-the-art GAN baselines for scene-centric image synthesis and release the pretrained models to the vision community**. Towards this aim, we train variants of the BigGAN and BigGAN-Deep architectures on the full Places365-Challenge dataset (∼8M images), which is approximately 8x as large as ImageNet. A secondary objective is to test and optimize our open source BigGAN implementation in the distributed setting, which makes achieving the larger resolution variants feasible.

2.2 Related Work

2.2.1 GAN Framework

GANs are a framework for teaching a deep neural network model to capture the training data's distribution so we can generate new data from that same distribution. The training strategy is to define a game between two competing networks: a *generator* a function G that spawns "fake" images by mapping a source of noise to the input space and a *discriminator* a function D that must distinguish between a generated sample and a true data sample. In a zero-sum, non-cooperative game, the generator is trained to fool the discriminator with the following minimax objective:

$$\begin{aligned}\min_G \max_D \; & \mathbb{E}_{x\sim p_{data}(x)}\big[\log D(x)\big] \\ & + \mathop{\mathbb{E}}_{z\sim p_z(z)}\Big[\log(1-D(G(z)))\Big]\end{aligned} \tag{2.1}$$

where p_{data} is the data distribution and P_z is some simple noise distribution, such as a normal distribution. To address a problem of vanishing gradients in G, a *non-saturating* GAN variant was proposed in the original GAN paper and empirically shown to outperform the original minimax formulation where the discriminator maximizes $\mathbb{E}_{x\sim p_{data}(x)}\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log(1-D(G(z))\big]$ and the generator maximizes Generator maximizes: $\mathbb{E}_{x\sim p_z(z)}\big[\log(D(G(z)))\big]$.The intuition for the non-saturating variant is that, in practice, it's easier to optimize the discriminator than the generator, especially early on in training. If the generator is not doing a good job yet, the gradients for the generator diminishes and the generator does not learn.

Convolutional GANs Deep Convolutional GANS (DCGANs) [85] represented a major step forward in the success of GAN image synthesis by modeling G and D with covolutional neural network, and it stands as one of the most popular and successful baseline GAN architectures. DCGAN proposes a fully convolutional network that does away with max pooling and fully connected layers, and introduces several important architectural elements such as strided convolutions and batch normalization.

Conditional GANs are an important extension to the GAN framework, allowing for greater control over the final output of the generator. Here, both the generator and discriminator are conditioned on some data y (class label or data from some other modality).

Researchers have tried various approaches to incorporating conditional information into the discriminator, including concatenating class embeddings at various stages of the network such as the input and hidden layers.

Methods for conditioning the generator have typically mirrored their discriminator counterpart. However, more recently, [21, 24] have conditioned the generator by way of *Class Conditional Batchnorm*. Here, it behaves much like traditional batchnorm except that it learns per-class gain weights γ and bias β to be applied to the output. The running input statistics (mean/var) are not class dependent in conditional batchnorm.

Spectral Normalization (SNGANs) Training GANs is notoriously difficult. Adversarial training is a highly dynamic characterized by instability, particularly in the discriminator network. Models may never converge and mode collapses are common. A growing body of research has focused of improvements to the GAN training procedure to impart stability. Very broadly speaking, this work falls into two camps: modifications to the objective function to encourage convergence and constraints on the discriminator through gradient penalties or normalization to counteract the use of unbounded loss functions. For example, one popular alternative called Wasserstein GAN (WGAN) [3] leverages Wasserstein distance to produce an objective function with better theoretical and empirical properties. In particular, the WGAN objective function results in a discriminator whose gradients with respect to to its inputs are better behaved (smoother) than the original GAN counterparts, which makes optimization of the generator easier. The Wasserstein loss function requires the discriminator to be 1-Lipschitz, which relates to how quickly a function can change. Initial techniques for ensuring the Lipschitzness of D involved constraining the norms of its gradients through simple gradient clipping and more computationally expensive gradient penalties. While empirically effective at stabilizing training, computing gradient norms implies non-trivial running time overhead.

Miyato et al. proposed an exciting and normalization technique called Spectral Normalization for Generative Adversarial Networks (SNGANs) [74] that approximates the Wasserstein loss as a discriminator function. which enforces Lipschitz continuity on the weights of the discriminator and stabilizes training. Spectral normed layers enforce Lipschitz continuity by normalizing its parameters with running estimates of their first singular values. These estimates can be computed efficiently using the power iteration method.

2.2.2 Evaluation Metrics

In early GAN research, samples were compared visually, which introduces several issues (ratings could be highly subjective and biases, and evaluating at scale is difficult). On the other hand, objectively evaluating implicit generative models still remains difficult and an active area of investigation. Despite their notable flaws, researchers have recently settled on two popular metrics, which aim to capture image quality and diversity:

Inception Score (IS) measures how confidently an ImageNet-pretrained InceptionV3 network can classify generated samples and the diversity of its predictions over large collection of samples. If a model produces samples that InceptionV3 can confidently classify, this contributes to a higher IS. A high diversity of classifications also contributes to a higher IS. Concretely, the inception score can be computed with $\texttt{IS} = \exp\left(\mathbb{E}_{x\sim P_g}\left[D_{KL}(p(y|x)\|p(y)\right]\right)$.

Fréchet Distance (FID) measures the distance between two distributions. Here, an Inception Network is used to generate feature representations for both the real images from the dataset of interest and generated samples from the model. These feature distributions are modeled by multivariate Guassian distributions. The shorter the Fréchet Distance between these two distributions, the more closely the fake images resemble the real ones. The FID formula is $\texttt{FID} = \|\mu_x - \mu_y\|_2^2 + Tr(\Sigma_x + \Sigma_y - 2(\Sigma_x\Sigma_y)^{\frac{1}{2}})$, where (μ_x, Σ_x), and (μ_y, Σ_y) are the mean and covariance of the embedded sampled from P_{data} and P_{model}, respectively.

2.3 Methods

In this work, we jointly investigate (i) the feasibility of training large-scale GANs using open-source/publicly released software and comparatively resource constrained hardware, and (ii) how methods driving recent advances in large-scale, high-fidelity image synthesis transfer from object-based (ImageNet) datasets to scene-centric datasets such as Places365.

2.3.1 Network Architectures

We employ the BigGAN and BigGANDeep architectures of Brock et al. [13], which uses the hinge loss [68, 109] GAN objective. We provide class information to `G` with class-conditional BatchNorm [21, 24] and to D with projection [74]. The optimization settings follow Brock et al. (2019) (notably employing Spectral Norm in G) with the modification that we consider 1 and 2 D steps per G step. For evaluation, we employ moving averages of G's weights following [53]; Mescheder et al. (2018); Yazc et al. (2018), with a decay of 0.9999. We use Orthogonal Initialization (Saxe et al., 2014), whereas previous works used $\mathcal{N}(0, 0.02I)$ [85] or Xavier initialization [27].

2.3.2 Optimization and Inference

To optimize our networks, we use the optimizer settings from the original SA-GAN implementation of Zhang et al. [118], by employing higher learning rates for `G` $(1\cdot10^{-4})$, and `D` $(4\cdot10^{-4})$, as well a single `D` optimization step per `G` step. While Brock et al. [13] were able to achieve superior IS/FID metrics using a learning rate of $5\cdot10^{-5}$ for `G`, $2\cdot10^{-4}$ for `D`, and two `D` optimization steps per `G`, these modifications emerged as a natural first step towards saving on training time and compute, and found it produced negligible side-effects in most but all cases (discussed more below).

Contrary to Brock et al., we were not able to use *Cross-Replica BatchNorm* [46] in `G`, also known as *synchronized BatchNorm*, where batch statistics are aggregated across all devices, rather than a single device as in standard implementations. We experimented with three separate implementations including the official PyTorch release in version 1.1.0. Despite identical forward passes, slight differences from the standard, built-in BatchNorm appeared to be sufficient to cripple training.

Accumulate to Optimize: Brock et al. [13] showed that simply increasing the batch size by a factor of 8 improved the state-of-the-art IS by 46%. The authors conjecture that this is a result of each batch covering more modes, providing better gradients for both networks. Unfortunately, *BigGAN* is very expensive in term of memory requirements, hence large batch sizes are difficult to attain. To overcome this problem, our training strategy is centred on accumulating gradients until the neural network has processed the desired number of examples, 2048 in the case of BigGAN. Known as **Gradient accumulation**, this technique allows us to simulate larger batch sizes not attainable due to hardware (typically memory) constraints. Here, each batch is further split into minibatches, which are processed sequentially. Gradients from each minibatch are "accumulated" and the optimization step is only performed after all minibatches have processed. The pseudocode for one iteration involving this process is provided in Algorithm 1. The standard training process is modified such that gradients for both `G` and `D` are accumulated separately for a specified number of accumulations. As a result, the approach achieves an effective batch size equal to `No. accumulation steps` $\times$ `mini batch size`.

2.3.3 Implementation

Models are trained using PyTorch on a variety of computational platforms ranging from single, resource-constrained 4-8 GPU machines to large scale distributed experiments totaling 92 NVIDIA Tesla V100 GPUs. Our code extends the officially unofficial BigGAN implementation, which can be found at https://github.com/alexandonian/BigGAN-PyTorch.

Algorithm 1 Single iteration of Generative Adversarial Training with Gradient Accumulation.

```
1: for step ← 1 to $nDs$ do
2:     $g_D \leftarrow 0$
3:     for accumulation ← 1 to $nDa$ do
4:         Sample $z$ and $y$
5:         $D_{real} \leftarrow D(x, y)$ # Evaluate real
6:         $D_{fake} \leftarrow D(G(z, y))$ # Evaluate fake
7:         $L_{D_{real}}, L_{D_{fake}} \leftarrow$ hinge_loss($D_{real}, D_{fake}$)
8:         $L_D \leftarrow (L_{D_{real}} + L_{D_{fake}})/nDa$ # Sum losses.
9:         $g_D \leftarrow g_D + \nabla_\theta L_D$ # Accumulate gradients
10:    $\theta \leftarrow \theta - \eta \cdot g_D$ # Step optimizer
11: $g_G \leftarrow 0$
12: for accumulation ← 1 to $nGa$ do
13:    Sample $z$ and $y$
14:    $D_{fake} \leftarrow D(G(z, y))$ # Evaluate fake examples
15:    $L_G \leftarrow$ hinge_loss($D_{fake}$)$/nDs$
16:    $g_G \leftarrow g_G + \nabla_\theta^{(a)} L_G$
17: $\theta \leftarrow \theta - \eta \cdot g_G$
```

where:

- `hinge_loss`(): Compute hinge loss function.
- nDs: Number of Discriminator Steps.
- nDa: Number of Discriminator Accumulations.
- nGa: Number of Generator Accumulations.

Provider	Model	Dataset	No. GPUs	G.A.S	BS	min/1k Itr.
Google Cloud	BigGAN (128)	ImageNet	8 x 16Gb V100s	8	256	$\sim$ 145
Core Scientific	BigGAN (128)	Places365	6 x 32Gb V100s	6	360	$\sim$ 145
IBM Cloud	BigGAN (256)	Places365	92 x 16Gb V100s	2	11	200+
MIT-Satori	BigGAN (128)	Places365	16 x 32Gb V100s	2	32	30
	BigGAN (256)	Places365	32 x 32Gb V100s	2	32	100

Table 2.1: Computational configurations employed in this study. The 128x128 models trained on Google Cloud and Core Scientific used a single multi-GPU machine, which allows for low-latency GPU-to-GPU synchronization. The 256x256 model was trained for 100k iterations using 42 2xGPU nodes on IBM Cloud. Despite having an order of magnitude more GPUs and fewer gradient accumulations, the rate of training in this configuration is significantly lower than its single node counterparts, suggesting that the distributed nature of this setup introduces a heavy synchronization time penalty. In contrast, the infiniband network connections in the MIT-Satori cluster mitigated these synchronization penalties.

Model	Res.	Itr.	(min FID)/IS	FID/IS	FID/(max IS)
BigGAN	128	100k	2.86 / 45.85 $\pm$ 0.53	2.95 / 46.29 $\pm$ 0.74	12.69 / 79.78 $\pm$ 1.05
BigGAN-Deep	128	62k	5.06 / 46.60 $\pm$ 1.00	5.06 / 42.99 $\pm$ 0.66	15.06 / 86.76 $\pm$ 0.97
BigGAN	256	150k	2.51 / 94.74 $\pm$ 0.83	2.51 / 94.74 $\pm$ 0.83	13.82 / 129.09 $\pm$ 1.35
BigGAN-Deep	256	100k	4.97 / 78.12 $\pm$ 0.92	4.99 / 77.92 $\pm$ 0.59	22.25 / 144.54 $\pm$ 0.86

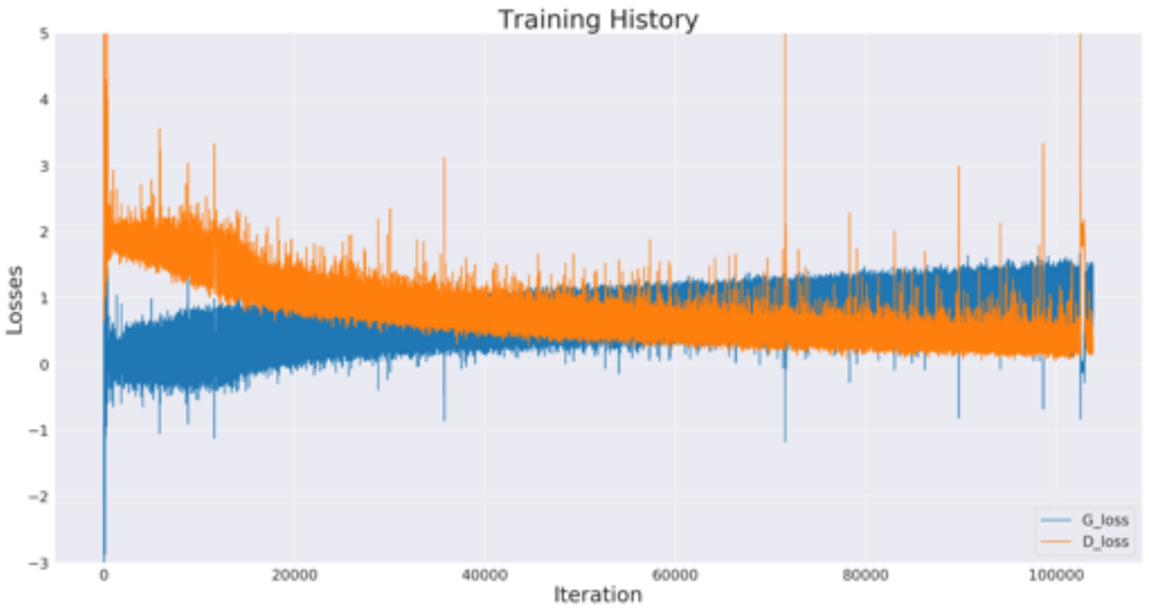
Table 2.2: Evaluation of models for different datasets, resolutions and training iterations via Fréchet Inception Distance (FID, lower is better) and Inception Score (IS, higher is better).

2.4 Experiments

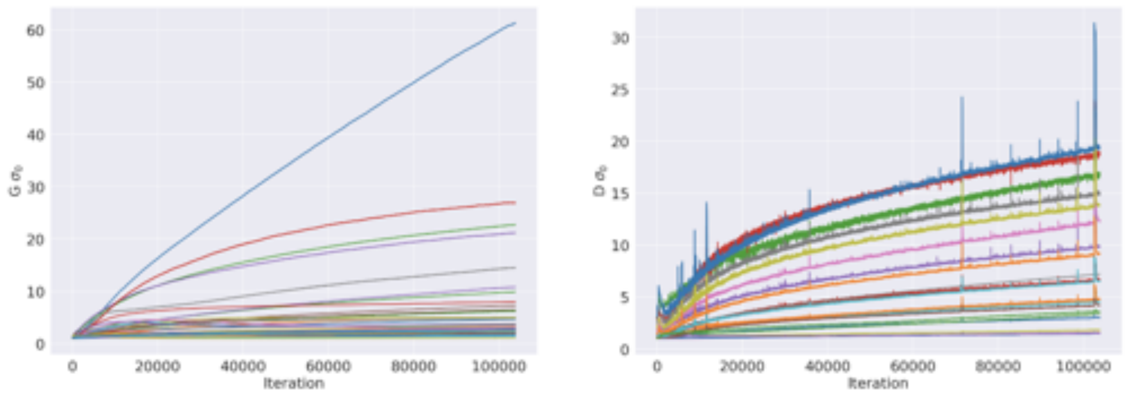
Training: Due to the high computational cost of training large scale GANs, we continuously monitor a suite of standard metrics and values to ensure forward progress during training. Examples of these metrics, which include the generator and discriminator loss, the first singular values of layer weights, and current IS and FID scores are depicted in figure 2-2 for the Places365 BigGAN trained at a resolution of 128 x 128. Combined with manual inspection of examples of generated samples, this monitoring process ensures that compute cycles are not being wasted on a model that has experienced mode collapse.

Evaluation: We evaluate our implementation BigGAN and BigGAN-deep on Places365 at 128x128 resolution and BigGAN-deep on the full Places365-Challenge (8M+ images) dataset at 256x256 resolution, employing the configurations from Table 2.1. The samples generated by our model are presented in Figure 2-3.

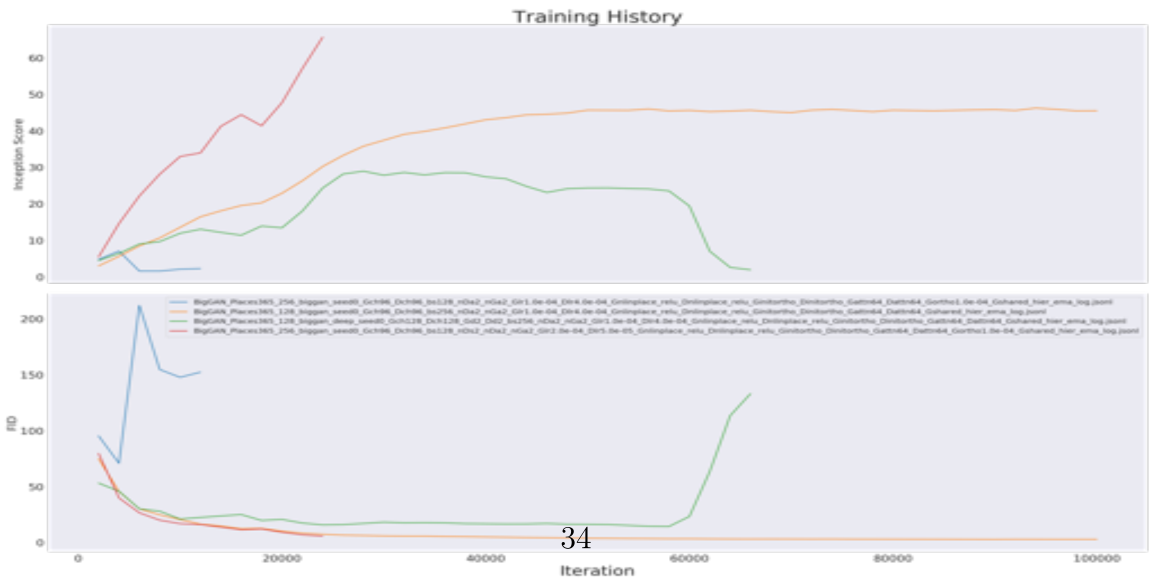
In Table 2.2, we report IS and FID metrics produced with an Places365 trained InceptionV3 classifier. As with [13], our models are able to trade sample variety for quality via the *Truncation Trick*; therefore, the best way to compare against prior work remains unclear. First we report the FID/IS values obtained with a truncation setting that obtains the best (min) FID. Next, we report the FID/IS scores obtained without any truncation. Finally, we list the FID/IS scores at the maximum IS achieved by each model. All Places365 models achieve state of the art performance. **These results will serve as an important benchmark for future researchers to measure their progress**.



(a) Losses



(b) Plot of the first singular value σ_0 in the layers of `G`(left) and `D` right before Spectral Normalization. Most layers in `G` have well-behaved spectra, but a small subset grow through training without constraints. After 150k iterations, the spectra of `G` shows no evidence collapse. `D`'s spectra are slight noisier but otherwise well-behaved. These observations are consistent with published results.



(c) A snapshot of IS/FID training trajectories for several runs.

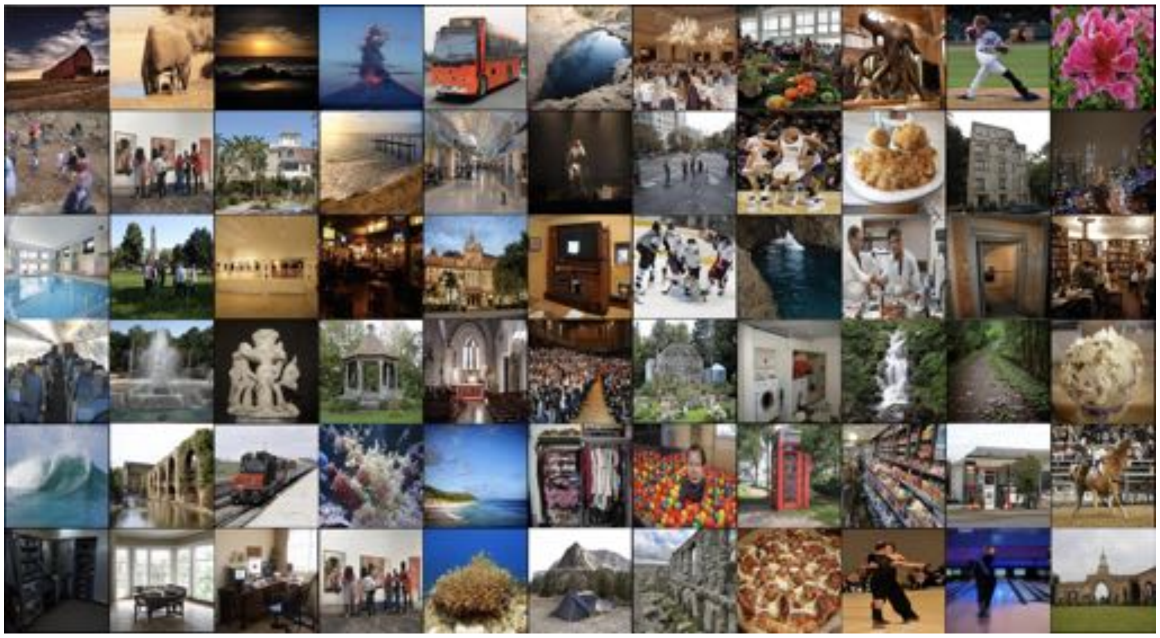


Figure 2-3: A collage of samples from the fully trained 256x256 pixel resolution model highlights the diversity and quality of generated samples.

2.5 Analysis

Truncation Trick: Brock et al. found that using a latent distribution for sampling that differs from that used in training can have beneficial effects on the final performance of the model. Known as the *Truncation Trick*, this technique allows for post-hoc control over the trade-off between sample fidelity and variety.

In Figure 2-4, we vary the amount of truncation during sampling and visualize the corresponding output samples. These qualitative results are consistent with the truncation trick and highlight pathologies that emerge at extreme truncation values. At particularly high truncation values (2+), we observe a breakdown in image quality. On the other side of the spectrum, very small truncation values virtually eliminate all sample diversity, frequently resulting in near identical outputs.

In Figure 2-5, we quantify this observation by showing a plot of how FID changes with respect to IS as the noise variance (variance in z) ranges from 0 to 1 with step size 0.05. Interestingly the 128 pixel BigGAN trained on Places achieves its minimum FID score and IS score simultaneously, suggesting a strong trade-off between the two metrics.

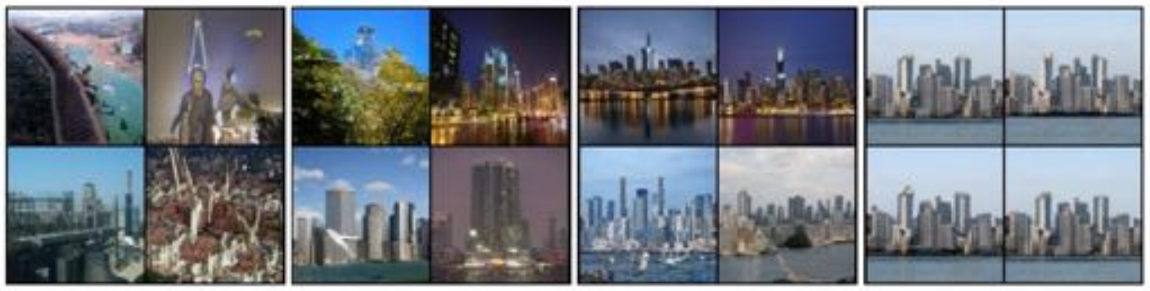


Figure 2-4: The effects of varying the amount of truncation during sampling from the 128 pixel BigGAN model for the “Skyscraper” scene category. From left to right, the threshold is set to 2, 1, 0.5, 0.04.

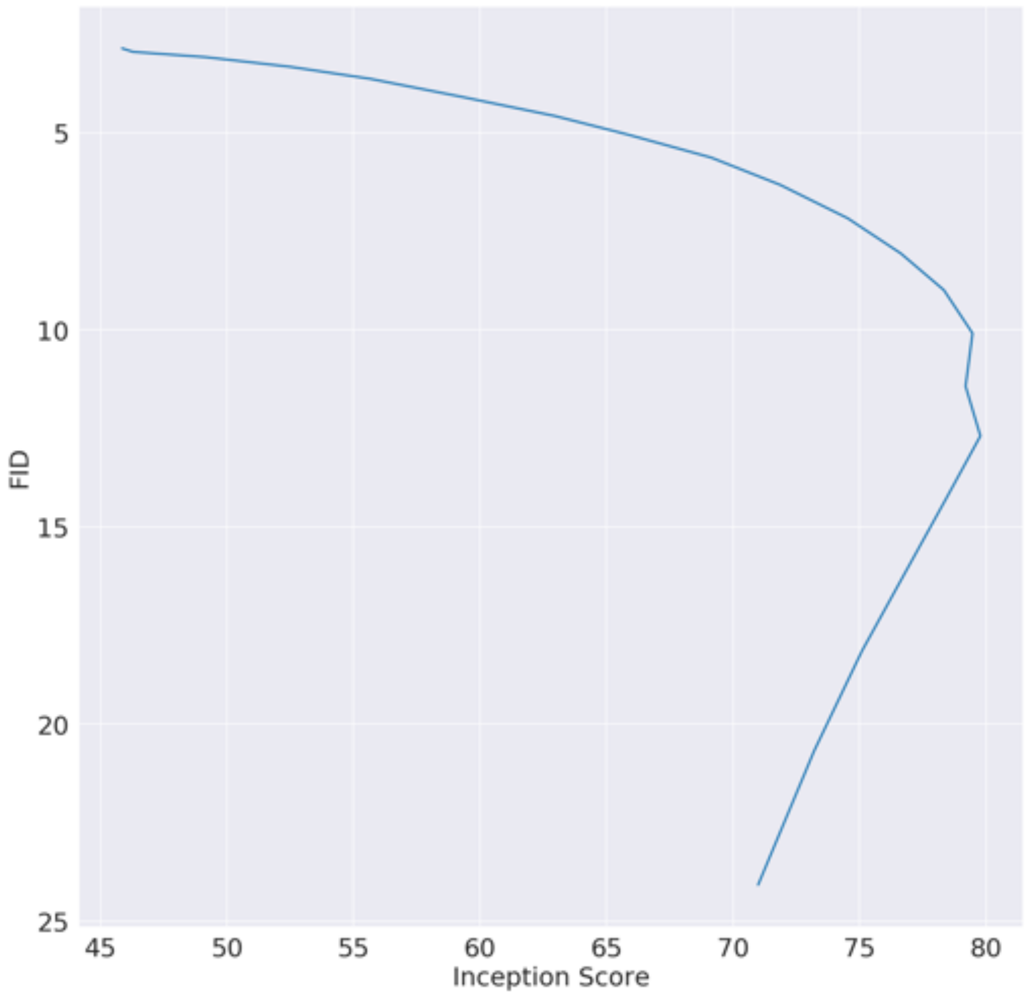


Figure 2-5: FID vs. IS at 128 pixel resolutions.

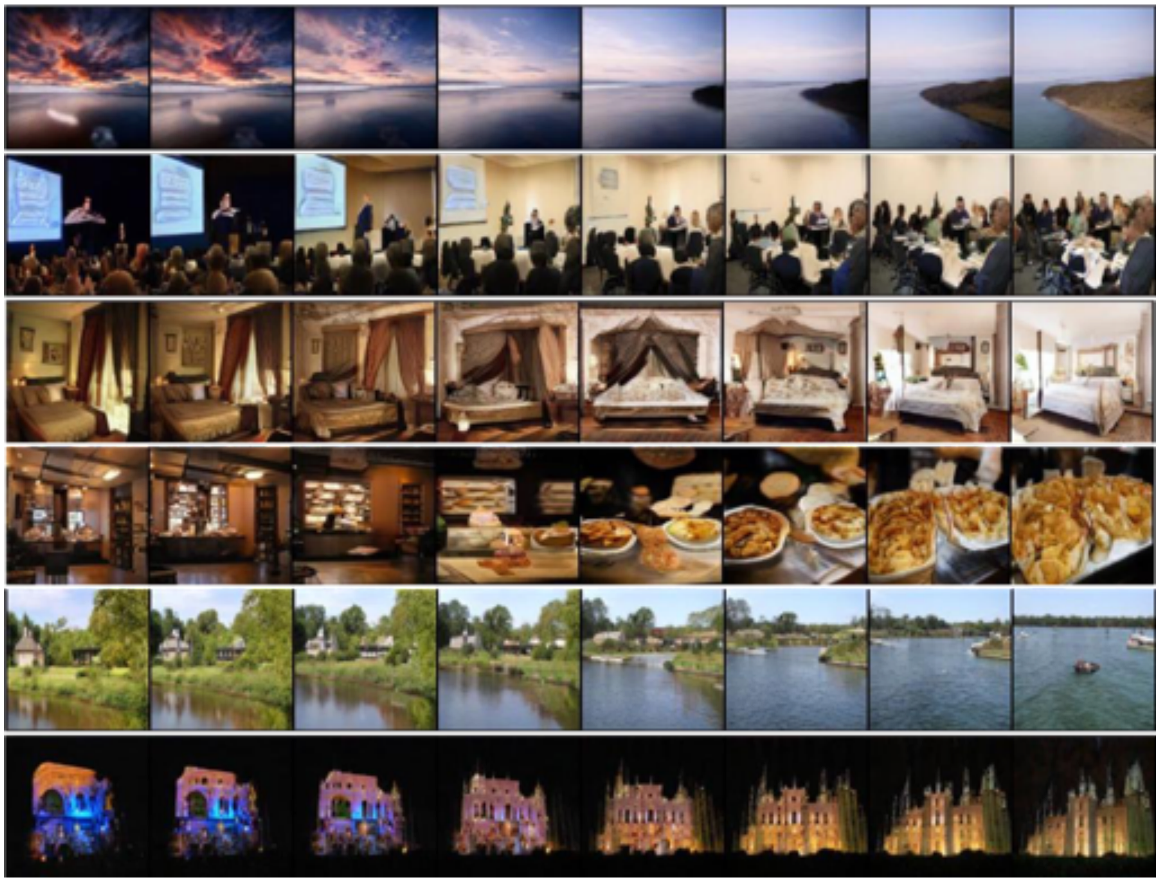


Figure 2-6: Linear interpolations between two randomly sampled coordinates in Z for a given class y shows that the latent space has developed smooth transitions, with each sample appearing as a visually plausible intermediate version of its right and left neighbors.

Walking in the Latent Space A common technique for investigating the structure of a GAN's latent space is to traverse a path along its learned manifold and observe how output samples change. Such a walk is useful for identifying the hierarchical structure of the latent space and hints at memorization, indicated by sharp visual transitions. In Figure 2-6, we show linear interpolations between two randomly samples coordinates in z for several choices of y.

Class-wise interpolation To efficiently facilitate its class-conditional nature, our model uses a single class embedding layer which linearly maps a one-hot encoded class vector c to a shared representation space. By interpolating between two class embeddings in this shared space, we can produce samples that contain a mixture of the components from the classes. Figure 2-7 demonstrates this effect nicely. This

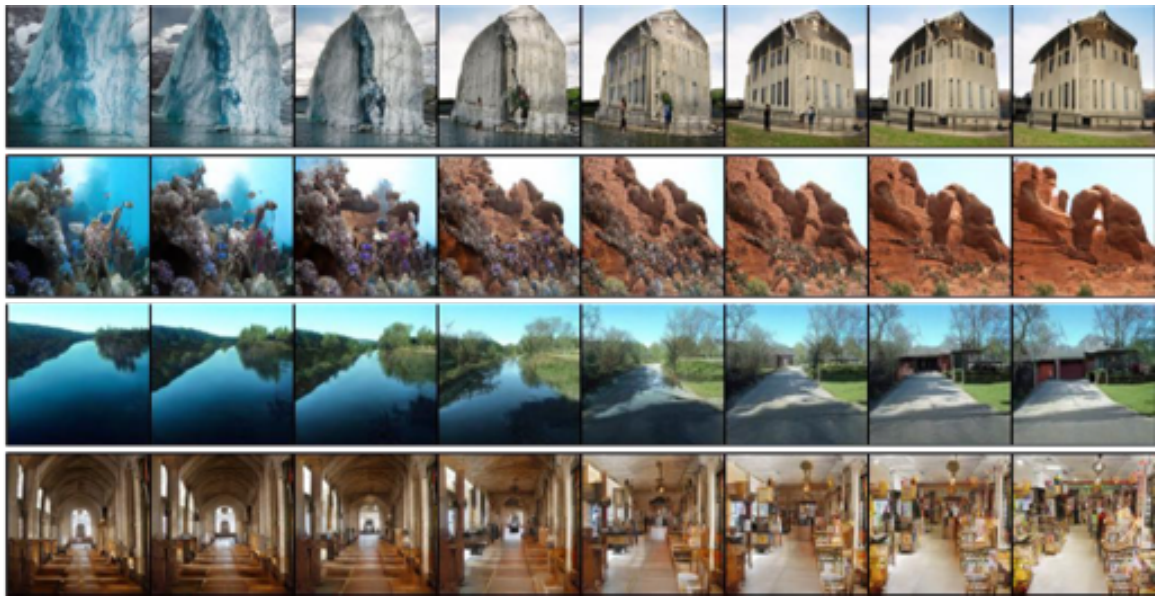


Figure 2-7: Interpolations between classes c with z held constant. Scene semantics and spatial layouts are frequently maintained between endpoints.

technique forms the basis for some artistic GAN applications such as the popular *GANBreeder* visualization app.

2.6 Conclusion

In this project, we have made significant progress towards establishing scene-centric GAN baselines. We anticipate that this initial work will continue to generate downstream value. Providing the community with tested open-source implementations allow researchers to reproduce and improve these results. Public release of the pre-trained model weights provides a new avenue for GAN analysis and GAN driven applications. For example, the following lines of research would directly benefit from incorporating a modern, high-fidelity, scene-centric GAN:

Intepretable Learning: Bau et al. [8] proposed a powerful framework **Gan Dissection** for visualizing and understanding the structure learned by a generative network. A pretrained Places BigGAN is a particularly natural dissection candidate (even compared to an ImageNet-trained BigGAN), as it would allow researchers to understand what the units of a high-fidelity class-conditional GAN have learned.

Graphics & Art: In collaboration with the MIT-IBM Watson AI Lab, Bau et al. released the **GANpaint app**, which works by directly activating and deactivating sets of neurons in a deep network trained to generate images. The app demonstrates that, by learning to draw, the network also learns about objects such as trees and doors and rooftops. Replacing the underlying GAN with a high-fidelity, class conditional generator would significantly increase the quality, applicability and appeal of the GANpaint application.

Human Visual Cognition: In the next chapter, we will present **GANalyze**, framework that uses Generative Adversarial Networks (GANs) to study cognitive properties like memorability, aesthetics, and emotional valence. GANs allow us to generate a manifold of natural-looking images with fine-grained differences in their visual attributes. By navigating this manifold in directions that increase memorability, it is possible visualize what it looks like for a particular generated image to become more or less memorable. The resulting "visual definitions" surface image properties (like "object size") that may underlie memorability. By introducing additional GANs capable of generating more visually diverse images allows us to elucidate emerging factors that otherwise wouldn't be able to be visualized due to the limited visual domains captured by an ImageNet BigGAN.

Chapter 3

GANalyze

In this chapter, we introduce a framework that uses Generative Adversarial Networks (GANs) to study cognitive properties like memorability. These attributes are of interest because we do not have a concrete visual definition of what they entail. What does it look like for a dog to be more memorable? GANs allow us to generate a manifold of natural-looking images with fine-grained differences in their visual attributes. By navigating this manifold in directions that increase memorability, we can visualize what it looks like for a particular generated image to become more memorable. The resulting "visual definitions" surface image properties (like "object size") that may underlie memorability. Through behavioral experiments, we verify that our method indeed discovers image manipulations that causally affect human memory performance. We further demonstrate that the same framework can be used to analyze image aesthetics and emotional valence. ganalyze.csail.mit.edu.

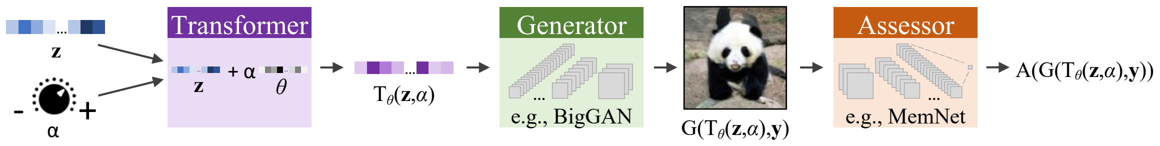

z
Transformer
z + α θ
- α +
$T_\theta(\mathbf{z},\alpha)$
Generator
e.g., BigGAN
$G(T_\theta(\mathbf{z},\alpha),\mathbf{y})$
Assessor
e.g., MemNet
$A(G(T_\theta(\mathbf{z},\alpha),\mathbf{y}))$

Figure 3-1: **Schematic of the model**. The model learns how to transform a $\mathbf{z}$ vector such that when fed to a Generator, the resulting image's property of interest changes. The transformation is achieved by the *Transformer*, who moves the $\mathbf{z}$ vector along a learned direction, θ, in the Generator's latent space. An *Assessor module* (e.g., MemNet) predicts the property of interest (e.g., memorability). Finally, α acts as a knob to set the desired degree of change in the Assessor value (e.g., MemNet score), telling the *Transformer* how far to move along θ.

3.1 Introduction

Why do we remember the things we do? Decades of work have provided numerous explanations: we remember things that are out of context [17, 104], that are emotionally salient [16], that involve people [47], etc. But a picture is, as they say, worth a thousand words. What does it *look like* to make an image more memorable? The same questions can be asked for many cognitive visual properties: what visual changes can take a bland foggy seascape and add just the right colors and tones to make it serenely beautiful.

Attributes like memorability, aesthetics, and emotional valence are of special interest because we do not have concrete definitions of what they entail. This contrasts with attributes like “object size" and “smile". We know exactly what it means to zoom in on a photo, and it’s easy to imagine what a face looks like as it forms a smile. It’s an open question, on the other hand, what exactly do changes in “memorability" look like? Previous work has built powerful predictive models of image memorability [47, 60] but these have fallen short of providing a fine-grained visual explanation of what underlies the predictions.

In this paper, we propose a new framework, GANalyze, based on Generative Adversarial Networks (GAN) [30], to study the visual features and properties that underlie high-level cognitive attributes. We focus on image memorability as a case study, but also show that the same methods can be applied to study image aesthetics and emotional valence.

Our approach leverages the ability of GANs to generate a continuum of images with fine-grained differences in their visual attributes. We can learn how to navigate the GAN’s latent space to produce images that have increasing or decreasing memorability, according to an off-the-shelf memorability predictor [60]. Starting with a seed image, this produces a sequence of images of increasing and decreasing predicted memorability (see Figure 1). By showing this visualization for a diverse range of seed images, we come up with a catalog of different image sequences showcasing a variety of visual effects related to memorability. We call this catalog a *visual definition* of image memorability. GANalyze thereby offers an alternative to the non-parametric approach in which real images are simply sorted on their memorability score to visualize what makes them memorable (example shown in the Supplementary Materials). The parametric, fine-grained visualizations generated by GANalyze provide much clearer visual definitions.

These visualizations surface several correlates of memorability that have been overlooked by prior work, including “object size", “circularity", and “colorfulness". Most past work on modeling image memorability focused on semantic attributes, such as object category (*e.g.*, “people" are more memorable than “trees") [47]. By applying our approach to a class-conditional GAN, BigGAN [14], we can restrict it to only make changes that are orthogonal to object class. This reveals more fine-grained changes that nonetheless have large effects on predicted memorability. For example, consider the cheeseburgers in Figure 3-3. Our model visualizes more memorable cheeseburgers as we move to the right. The apparent changes go well beyond semantic category – the right-most burger is brighter, rounder, more canonical, and, we think, looks tastier.

Since our visualizations are learned based on a *model* of memorability, a critical step is to verify that what we are seeing really has a causal effect on human behavior. We test this by running a behavioral experiment that measures the memorability of images generated by our GAN, and indeed we find that our manipulations have a causal effect: navigating the GAN manifold toward images that are predicted to be more memorable actually results in generating images that are measurably more memorable in the behavioral experiment.

Our contributions include the following:

- Introducing GANalyze, a framework that uses GANs to provide a *visual definition* of image properties, like memorability, aesthetics, and emotional valence, that we can measure but are not easy, in words, to define.

- Showing that this framework surfaces previously overlooked attributes that correlate with memorability.

- Demonstrating that the discovered transformations have a causal effect on memorability.
- Showing that GANalyze can be applied to provide visual definitions for aesthetics and emotional valence.

3.1.1 Related work

Generative Adverserial Networks (GANs). GANs [30] introduced a framework to synthesize natural-looking images [14, 14, 54, 56, 119]. Among the many applications for GANs are style transfer [129], visual prediction [72], and “sim2real" domain adaptation [12]. Concurrent work explores the ability to manipulate images via simple transformations in latent space [49, 99]. Here, we show how they can also be applied to the problem of understanding high-level, cognitive image properties, such as memorability.

Understanding CNN representations. The internal representations of CNNs can be unveiled using methods like network dissection [7, 124, 125] including for a CNN trained on memorability [60]. For instance, Khosla et al. [60] showed that units with strong positive correlations with memorable images specialized for people, faces, body parts, etc., while those with strong negative correlations were more sensitive to large regions in landscapes scenes. Here, GANalyze introduces a new way of defining what memorability, aesthetic, and emotional valence variability look like.

Modifying Memorability. The memorability of an image, like faces, can be manipulated using warping techniques [59]. Concurrent work has also explored using a GAN for this purpose [103]. Another approach is a deep style transfer [102] which taps into more artistic qualities. Now that GANs have reached a quality that is often almost indistinguishable from real images, they offer a powerful tool to synthesize images with different cognitive qualities. As shown here, our GANalyze framework successfully modified GAN-generated images across a wide range of image categories to produce a second generation of GAN realistic photos with different mnemonic qualities.

3.2 Model

3.2.1 Formulation

We start with a pretrained Generator G, who takes a noise vector $\mathbf{z}$ and a one-hot class vector $\mathbf{y}$ as input and generates a photo-realistic image $G(\mathbf{z}, \mathbf{y})$. Assumed is also an Assessor function A that assesses an image property of interest, in this case memorability. Our goal was to learn to transform any given noise vector $\mathbf{z}$ of any class $\mathbf{y}$ such that the memorability of its resulting, generated image increases (or decreases) with a certain amount α. The transformation is achieved by a Transformer function, who moves the input $\mathbf{z}$ along a certain direction θ in the latent space. We express the objective as:

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{z},\mathbf{y},\alpha}[(A(G(T_\theta(\mathbf{z}, \alpha), \mathbf{y}))\\ -(A(G(\mathbf{z}, \mathbf{y})) + \alpha))^2]\end{aligned} \tag{3.1}$$

Note that this is simply the MSE loss between the target memorability score, i.e. the seed image's score $A(G(\mathbf{z}, \mathbf{y}))$ increased by α, and the memorability score of the transformed clone image $A(G(T_\theta(\mathbf{z}, \alpha), \mathbf{y}))$. The scalar α acts as a metaphorical knob which one can use to turn up or turn down memorability. The optimizing problem is $\theta^* = \operatorname{argmin}_\theta \mathcal{L}(\theta)$. The Transformer T is defined as:

$$T(\mathbf{z}, \alpha) = \mathbf{z} + \alpha\theta \tag{3.2}$$

Figure 3-1 presents a schematic of the model. Finally, note that when $\alpha = 0$, T becomes a null operation and $G(T_\theta(\mathbf{z}, \alpha), \mathbf{y})$ then equals $G(\mathbf{z}, \mathbf{y})$.

3.2.2 Implementation

For the results presented here, we used the Generator of BigGAN [14], which generates state-of-the art GAN images and is pretrained on ImageNet [92]. The Assessor was implemented as MemNet [60], a CNN predicting image memorability. Note, however, that training our model with different Generators or different Assessors can easily be achieved by substituting the respective modules. We discuss other Assessors in Section 3.4.

To train our model and find θ^*, we built a training set by randomly sampling 400K $\mathbf{z}$ vectors from a standard normal distribution truncated to the range $[-2, 2]$. Each $\mathbf{z}$ was accompanied by an α value, randomly drawn from a uniform distribution between -0.5 and 0.5, and a randomly chosen $\mathbf{y}$. We used a batch size of 4 and an Adam optimizer.

In view of the behavioral experiments (see Section 3.3), we restricted the test set to 750 randomly chosen Imagenet classes and two $\mathbf{z}$ vectors per class. Each $\mathbf{z}$ vector was then paired with five different α values: $[-0.2, -0.1, 0, 0.1, 0.2]$. Note that this includes an α of 0, representing the original image $G(\mathbf{z}, \mathbf{y})$. Finally, the test set consisted of 1.5K sets of five images, or 7.5K test images in total.

3.3 Experiments

3.3.1 Model validation

Did our model learn to navigate the latent space such that it can increase (or decrease) the Assessor score of the generated image with positive (or negative) α values?

Figure 3-2.A suggests that it did. The mean MemNet score of test set images increases with every increment of α. To test this formally, we fitted a linear mixed-effects regression model to the data and found a (unstandardized) slope (β) of 0.68 ($95\% CI = [0.66, 0.70], p < 0.001$), confirming that the Memnet score increases significantly with α.

3.3.2 Emerging factors

We observe that the model can successfully change the memorability of an image, given its $\mathbf{z}$ vector. Next, we ask which image factors it altered to achieve this. The answer to this question can provide further insight into what the Assessor has learned

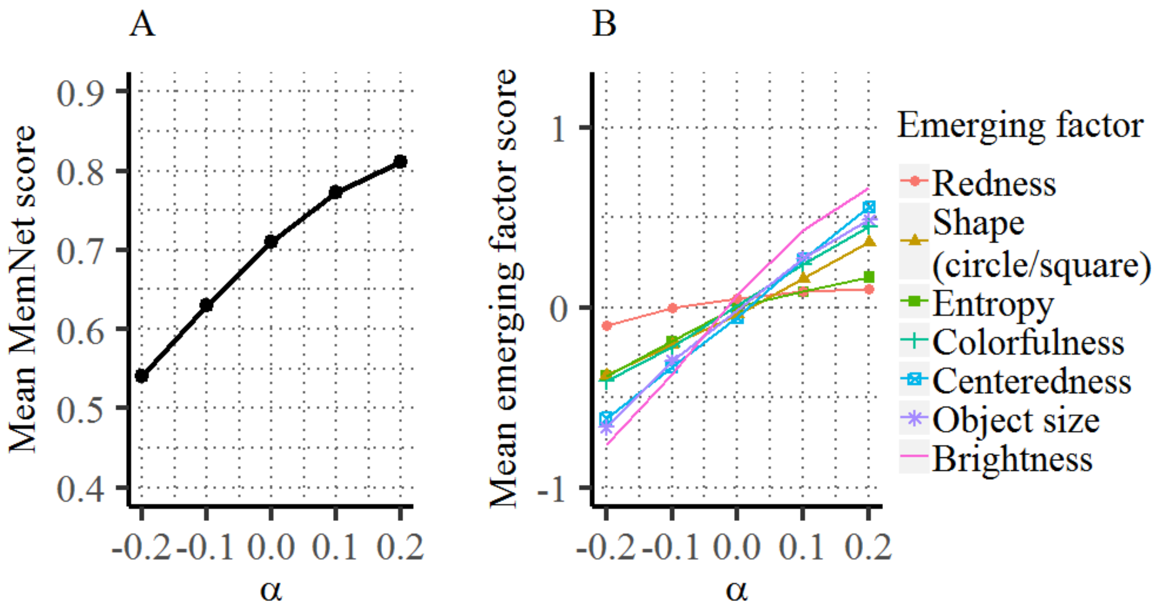


Figure 3-2: **Model results**. A) Graph shows the mean MemNet score across the images in every α condition. Our model successfully learned how to modify a GAN image to decrease (negative α) or increase (positive α) its MemNet score. B) List of emerging factors potentially underlying the effect observed in (A), and graph of how they change in function of α. The factors emerged from visualizations generated by the GANalyze framework (examples shown in Figures 3-3 and in the Supplementary Materials).

about the to-be-assessed image property, in this case what MemNet has learned about memorability. From a qualitative analysis of the test set (examples shown in Figure 3-3), a number of candidate factors stand out.

First, MemNet assigns higher memorability scores when the **size of the object** (or animal) in the image is larger, as our model is in many cases zooming in further on the object with every increase of α.

Second it is **centering** the subject in the image frame.

Third, it seems to strive for **square** or **circular** shapes in classes where it is realistic to do so (e.g., snake, cheeseburger, necklace, and espresso in Figure 3-3).

Fourth, it is often **simplifying** the image from low to high α, by reducing the clutter and/or number of objects, such as in the cheeseburger or flamingo, or by making the background more homogeneous, as in the snake example.

A fifth observation is that the subject's **eyes** sometimes become more pronounced and expressive, in particular in the dog classes (see Figure 2-1).

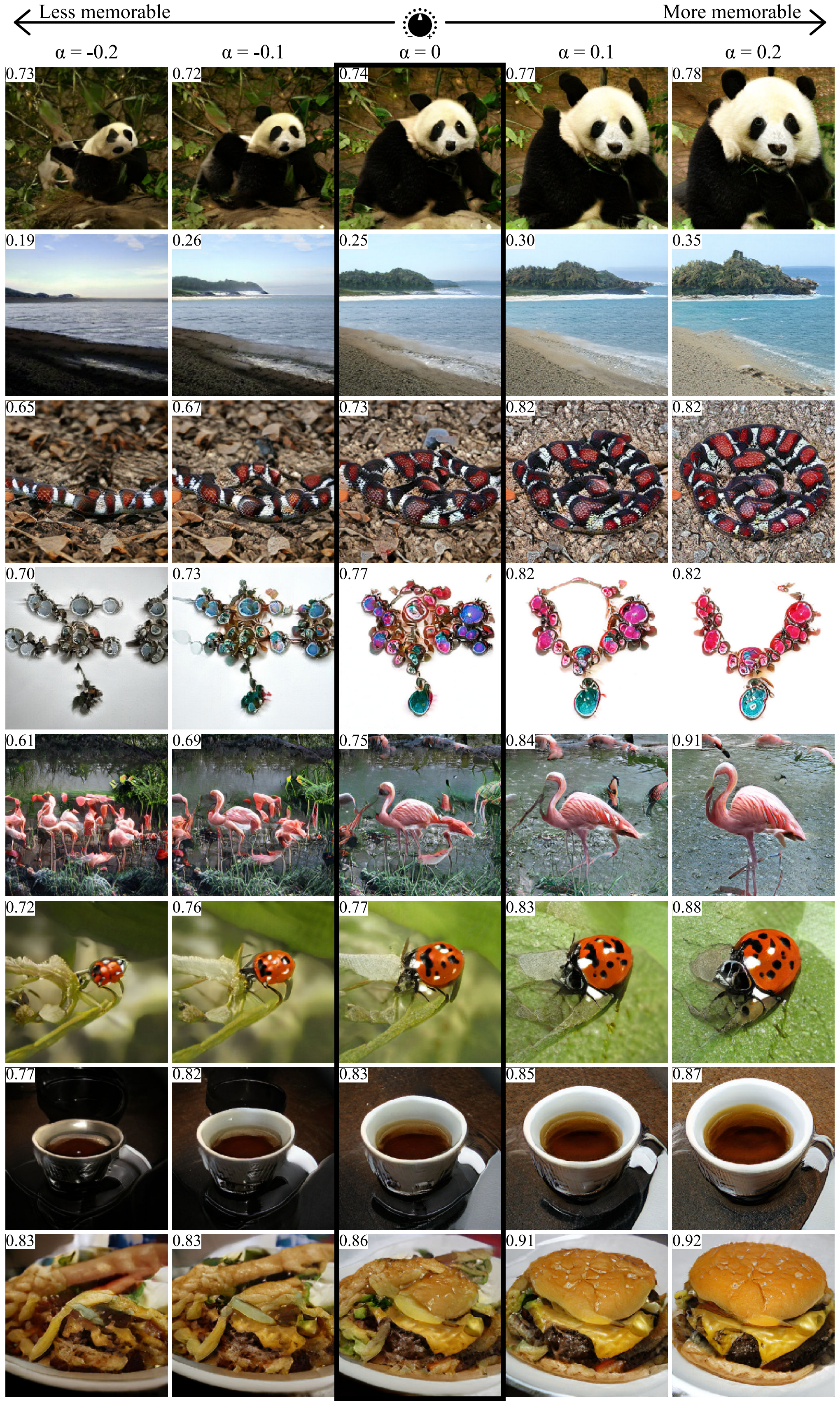


Figure 3-3: **Examples of generated images** along the memorability dimension. The middle column represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed to create a series of clone images more or less memorable.

Sixth, one can also detect color changes between the different α conditions. Positive $\alpha's$ often produce **brighter** and more **colorful** images, and negative $\alpha's$ often produce darker images with dull colors. Finally, for those classes where multiple object hues can be considered realistic (e.g., the bell pepper and the necklace in Figure 2-1 and Figure 3-3), the model seems to prefer a **red** hue.

To verify our observations, we quantified the factors listed above for the images in the test set (except for "expressive eyes", which is more subjective and harder to quantify). Brightness was measured as the average pixel value after transforming the image to grayscale. For colorfulness, we used the metric proposed by [38], and for redness we computed the normalized number of red pixels. Finally, the entropy of the pixel intensity histogram was taken as proxy for simplicity. For the other three factors, a pretrained Mask R-CNN [39, 71] was used to generate an instance-level segmentation mask of the subject. To capture object size, we calculated the mask's area (normalized number of pixels). To measure centeredness, we computed the deviation of the mask's centroid from the center of the frame. Finally, we calculated the length of minor and major axes of an ellipse that has the same normalized second central moments as the mask, and used their ratio as a metric of squareness. Figure 3-2.B shows that the emerging factor scores increase with α.

3.3.3 Realness

While BigGAN achieves state-of-the-art to generate highly realistic images, there remains some variability in the "realness" of the output. How best to evaluate the realness of a set of GAN-images is still an open question. Below, we discuss two automatically computed realness measures and a human measure in relation to our data.

Automatic measures

In Figure 3-4.A, we plot two popular automatic measures in function of α: the Frechet Inception Distance (FID) [42] and the Inception Score (IS) [93]. A first observation

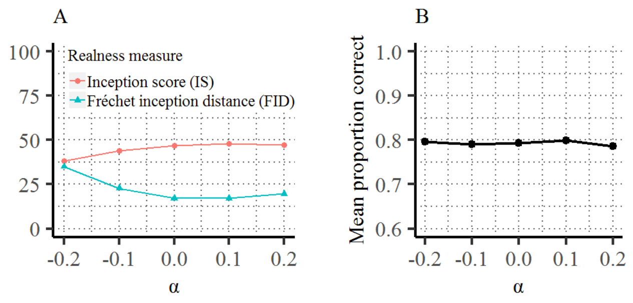


Figure 3-4: **Realness measures as a function of** α. A) Two popular automatic measures for evaluating the realness of a set of GAN images. Note that lower FID values indicate higher realness. B) Human fakeness discriminability, measured as the mean proportion correct in a 2AFC-task in which AMT workers had to discriminate GAN-images (fake) from real photographs.

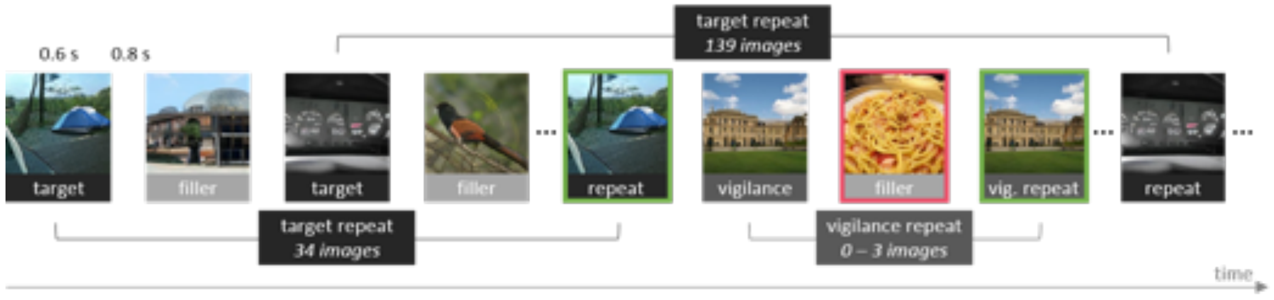


Figure 3-5: **Schematic of the visual memory game**. Each image is shown for 600 ms, with a blank interstimulus interval of 800 ms. Workers are asked to respond whenever they recognize a repeat of a previously shown image. For a correct response, the frame around the image briefly turns green. A red frame, on the other hand, indicates a mistake.

is that the FID is below 40 in all α conditions. An FID as low as 40 already corresponds to reasonably realistic images. Thus the effects of our model's modifications on memorability are not explained by making the images unrealistic. But we do observe interesting differences in FID- and IS-differences related to α, suggesting that more memorable images have more interpretable semantics.

Human measure

In addition to the two automatic measures, we conducted an experiment to collect human realness scores. The experiment consisted of a two-alternative forced choice (2AFC) task, hosted on Amazon Mechanical Turk (AMT), in which workers had to discriminate GAN-images from real ones. Workers were shown a series of pairs, consisting of one GAN-image and one real image. They were presented side by side for a duration of 1.6 s. Once a pair had disappeared off the screen, workers pressed the j-key when they thought the GAN-image was shown on the right, or the f-key when they thought it was shown on the left. The position of the GAN-image was randomized across trials. The set of real images used in this experiment was constructed by randomly sampling 10 real ImageNet exemplars per GAN-image class. The set of GAN-images was the same as the one quantified on memorability in Section 3.3.4. A GAN-image was randomly paired with one of the 10 real images belonging to the same class. Each series consisted of 100 trials, of which 20 were vigilance trials. For the vigilance trials, we generated GAN-images from $\mathbf{z}$ vectors that were sampled from the tails of a normal distribution (to make them look less real). For a worker's first series, we prepended 20 trials with feedback as practice (not included in the analyses). Workers could complete up to 17 series, but were blocked if they scored less than 65% correct on the vigilance trials. Series that failed this criterion were also excluded from the analyses. The pay rate equaled \$0.50 per completed series. On average, each of our test images was seen by 2.76 workers, meaning 4137 data points per α condition.

We did not observe differences in task performance between different α (see Figure 3-4.B). Indeed, a logistic mixed-effects regression fitted to the raw, binary data (correct/incorrect) did not reveal a statistically significant regression weight for α ($\beta = -0.08, 95\% CI = [-0.33, 0.18], p = 0.55$). In other words, the model's image modifications did not affect workers' ability to correctly identify the fake image, indicating that perceptually, the image clones of a seed image did not differ in realness.

3.3.4 Do our changes causally affect memory?

In addition to the MemNet scores, is our model also successful at changing the probability of an image being recognized by participants in an actual memory experiment? We tested people's memory for the images of a test set (see Section 3.2.2) using a repeat-detection visual memory game hosted on AMT (see Figure 3-5) [47,60]. AMT workers watched a series of one image at a time and had to press a key whenever they saw a repeat of a previously shown image. Each series comprised 215 images, shown each for 600 ms with a blank interval of 800 ms in-between. Sixty images were targets, sampled from our test set, and repeated after 34–139 intervening images. The remaining images were either filler or vigilance images and were sampled from a separate set. This set was created with 10 $\mathbf{z}$ vectors per class and the same five α values as the test set: $[-0.2, -0.1, 0, 0.1, 0.2]$, making a total of 37.5K images. Filler images were only presented once and ensured spacing between a target and its repeat. Vigilance images were presented twice, with 0–3 intervening images in-between the two presentations. The vigilance repeats constituted easy trials to keep workers attentive. Care was taken to ensure that a worker never saw more than one $G(T_\theta(\mathbf{z}, \alpha), \mathbf{y})$ for a given $\mathbf{z}$. They could complete up to 25 series, but were blocked if they missed more than 55% of the vigilance repeats in a series or made more than 30% false alarms. Series that failed this were not analyzed. The pay rate was $0.50 per completed series. On average, a test image was seen by 3.16 workers, with 4740 data points per α condition.

Workers could either recognize a repeated test image (hit, 1), or miss it (miss, 0). Figure 3-6.A shows the hit rate across all images and workers. The hit rate increases with every step of α. Fitting a logistic mixed-effects regression model to the raw, binary data (hit/miss), we found that the predicted log odds of image being recognized increase with 0.19 for an increase in α of 0.01 ($\beta = 1.92, 95\%CI = [1.71 - 2.12], p < 0.001$). This shows that our model can successfully navigate the BigGAN latent space in order to make an image more (or less) memorable to humans.

Emerging factors

Given human memory data for images modified for memorability, we evaluate how the images' emerging factor scores relate to their likelihood of being recognized. We fitted mixed-effects logistic regression models, each with a different emerging factor as the predictor, see Table 3.1. Except for entropy, all the emerging factors show a significant, positive relation to the likelihood of a hit in the memory game, but none fit the data as well as the model's α. This indicates that a single emerging factor is not enough to fully explain the effect observed in Figure 3-6.A. Note that the

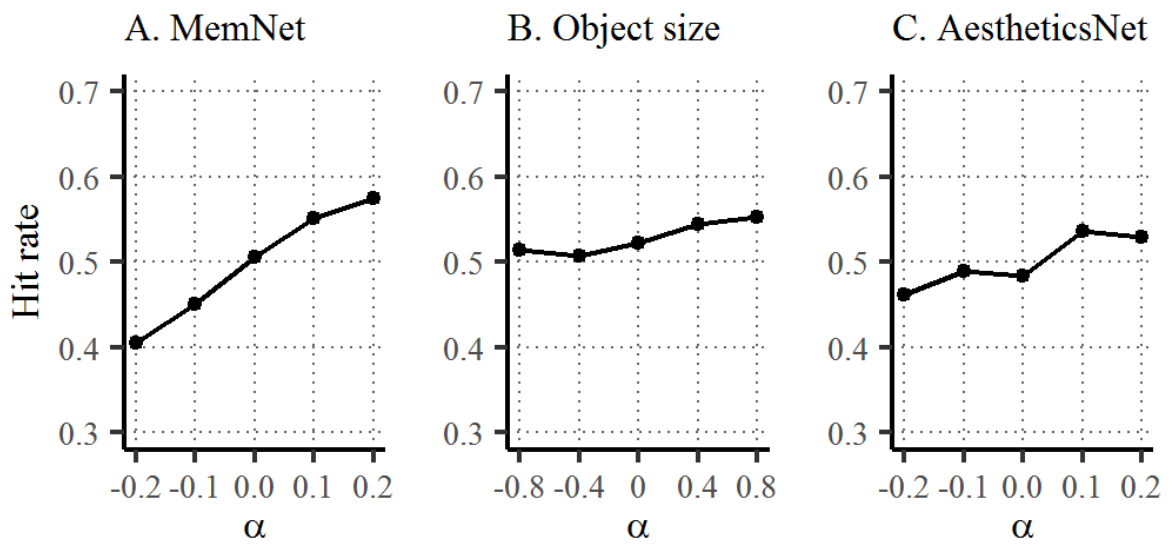


Figure 3-6: **Human memory performance** for images modified according to different Assessors: A) MemNet, B) Object size and C) AestheticsNet. Performance is measured as the hit rate across all images and workers in the memory game for each property.

emerging factor results are correlational and the factors are intercorrelated, hampering conclusions about which individual factors truly causally affect human memory performance. As an example of how this can be addressed within the GANalyze framework, we conducted an experiment focusing on the effect of one salient emerging factor: **object size**. As seen in Figure 3-3, more memorable images tend to center and enlarge the object class.

We trained a version of our model with an Object size Assessor, instead of the MemNet Assessor. This is the same Object size Assessor used to quantify the object size in the images modified according to MemNet (e.g., for the results in Figure 3-2.B), now teaching the Transformer to perform "enlarging" modifications. After training with 161,750 **z** vectors, we generated a test set as described in Section 3.2.2, except with a different set of α's: $[-0.8, -0.4, 0, 0.4, 0.8]$. We chose these values to qualitatively match the degree of object size changes achieved by the MemNet version of the model. Figure 3-7.A visualizes the results achieved on the test set. The model successfully enlarges the object with increasing alpha's, as confirmed by a linear mixed-effects regression analysis ($\beta = 0.07, 95\% CI = [0.06, 0.07], p < 0.001$). Figure 3-9 shows example images generated by that model. A comparison with images

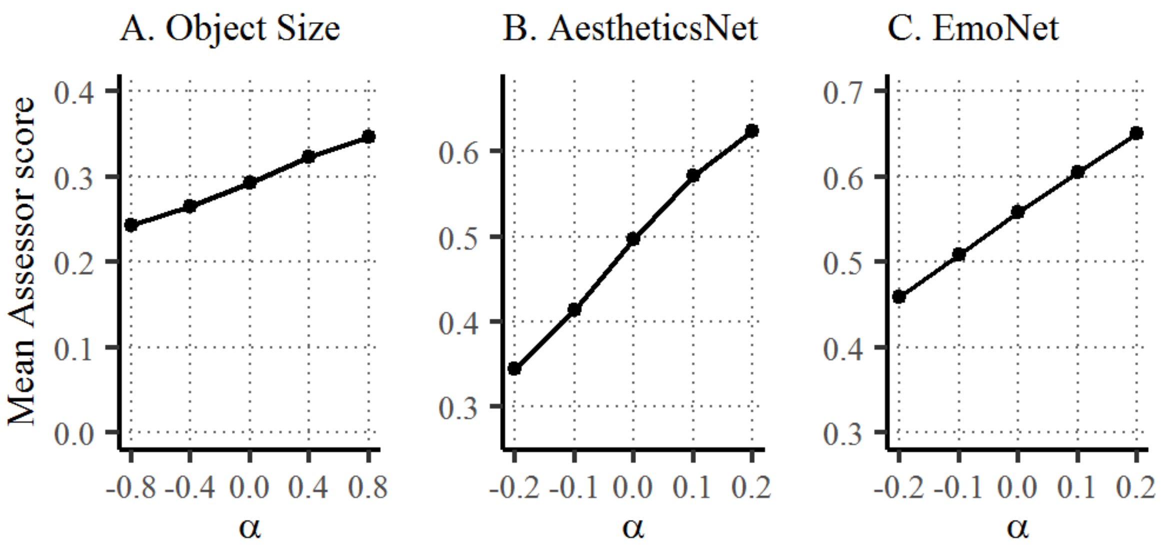


Figure 3-7: **Model results for additional Assessors**. Graphs show the mean Assessor (A: Object size, B: AestheticsNet, C: EmoNet) score across images in every α condition.

modified according to MemNet suggests that the latter model was doing more than just enlarging the object.

To study how the new size modifications affect memorability, we generated a new set of images (7.5K targets, 37.5K fillers) with α's $[-0.8, -0.4, 0, 0.4, 0.8]$. We chose these values to qualitatively match the degree of object size changes achieved by the MemNet version of the model. The new images were then quantified using the visual memory game (on average 2.36 data points per image and 3540 per α condition). Figure 3-6.B shows the results. Memory performance increases with α, as confirmed by a logistic mixed-effects analysis ($\beta = 0.11, 95\%CI = [0.06, 0.18], p < 0.001$, although mostly for positive α values.

3.4 Other image properties

As mentioned in Section 3.2.2, the proposed method can be applied to other image properties, simply by substituting the Assessor module. To show our framework can generalize, we trained a model for aesthetics and emotional valence. Emotional valence refers to how positive (or negative) the emotions evoked by an image are experienced. The respective Assessors were AestheticsNet [62] and our own EmoNet

Factor	Log Odds	CI	p	Tjur's D
Brightness	0.28	$[0.24, 0.32]$	<0.001	0.066
Centeredness	0.24	$[0.19, 0.29]$	<0.001	0.059
Colorfulness	0.17	$[0.14, 0.21]$	<0.001	0.054
Entropy	0.03	$[-0.04, 0.10]$	0.441	0.062
Redness	0.06	$[0.00, 0.12]$	0.042	0.055
Shape	0.19	$[0.14, 0.24]$	<0.001	0.060
Object size	0.32	$[0.27, 0.37]$	<0.001	0.050
α	1.92	$[1.71, 2.12]$	<0.001	0.074

Table 3.1: **Relation between emerging factors and human memory performance.** We show the output of logistic mixed-effects regressions. From left to right: the regression weight, the confidence interval (CI) for that weight, the p-value for statistical significance, and Tjur's coefficient of discrimination (D), being the regression model's goodness of fit [108]. The emerging factor values were normalized before running the regression models.

(a ResNet50 model [40], pretrained on Moments [75], fine-tuned to Cornell Emotion6 [82]). Figure 3-7.B shows the average AestheticsNet scores per α condition of a test set. The scores significantly increase with α, as evidenced by the results of a linear mixed-effects regression ($\beta = 0.72, 95\%CI = [0.70, 0.74], p < 0.001$). Similarly, the EmoNet scores significantly increase with α in a test set ($\beta = 0.44, 95\%CI = [0.43, 0.45], p < 0.001$, see Figure 3-7.C). Example visualizations are presented in Figure 3-8 and in the Supplementary Materials.

Based on a qualitative inspection of such visualizations, we observed that the aesthetics model is modifying factors like depth of field, color palette, and lighting, suggesting that the AestheticsNet is sensitive to those factors. Indeed, the architecture of the AestheticsNet includes attribute-adaptive layers to predict these factors, now highlighted by our visualizations. The emotional valence model often averts the subject's gaze away from the "camera" when decreasing valence. To increase valence, it often makes images more colorful, introduces bokeh, and makes the skies more blue in landscape images. Finally, the teddy bear in Figure 2-1 (right) seems to smile more. Interestingly, the model makes different modifications for every property (see Figure 3-9), suggesting that what makes an image memorable is different from what makes it aesthetically pleasing or more positive in its emotional valence.

A final question we asked is whether an image modified to become more (less) aesthetic also becomes more (less) memorable? To test this, we quantified the images of the aesthetic test set on memorability by presenting them to workers in the visual memory game (we collected 1.54 data points per image and 2306 data points per α condition). Figure 3-6.C shows the human memory performance in function of an α that is tuning aesthetics. A logistic mixed-effects regression revealed that with an 0.1 increase in the aesthetics α, the predicted log odds of an image being recognized increase with 0.07 ($\beta = 0.72, 95\% CI = [0.44, 1.00], p < 0.001$). While modifying an image to make it more aesthetic does increase its memorability, the effect is rather small, suggesting that memorability is more than only aesthetics and that our model was right to modify memorability and aesthetics in different ways.

3.5 Conclusion

We introduce GANalyze, a framework that shows how a GAN-based model can be used to visualize what another model (i.e. CNN as an Assessor) has learned about its target image property. Here we applied it to memorability, yielding a kind of "visual definition" of this high-level cognitive property, where we visualize what it looks like for an image to become more or less memorable. These visualizations surface multiple candidate features that may help explain why we remember what we do. Importantly, our framework can also be generalized to other image properties, such as aesthetics or emotional valence: by replacing the Assessor module, the framework allows us to explore the visual definition for any property we can model as a differentiable function of the image. We validated that our model successfully modified GAN images to become more (or less) memorable via a behavioral human memory experiment on manipulated images.

GANalyze's intended use is to contribute to the scientific understanding of otherwise hard to define cognitive properties. Note that this was achieved by modifying images for which the encoding into the latent space of the GAN was given. In other words, it is currently only possible to modify seed images that are GAN-images them-

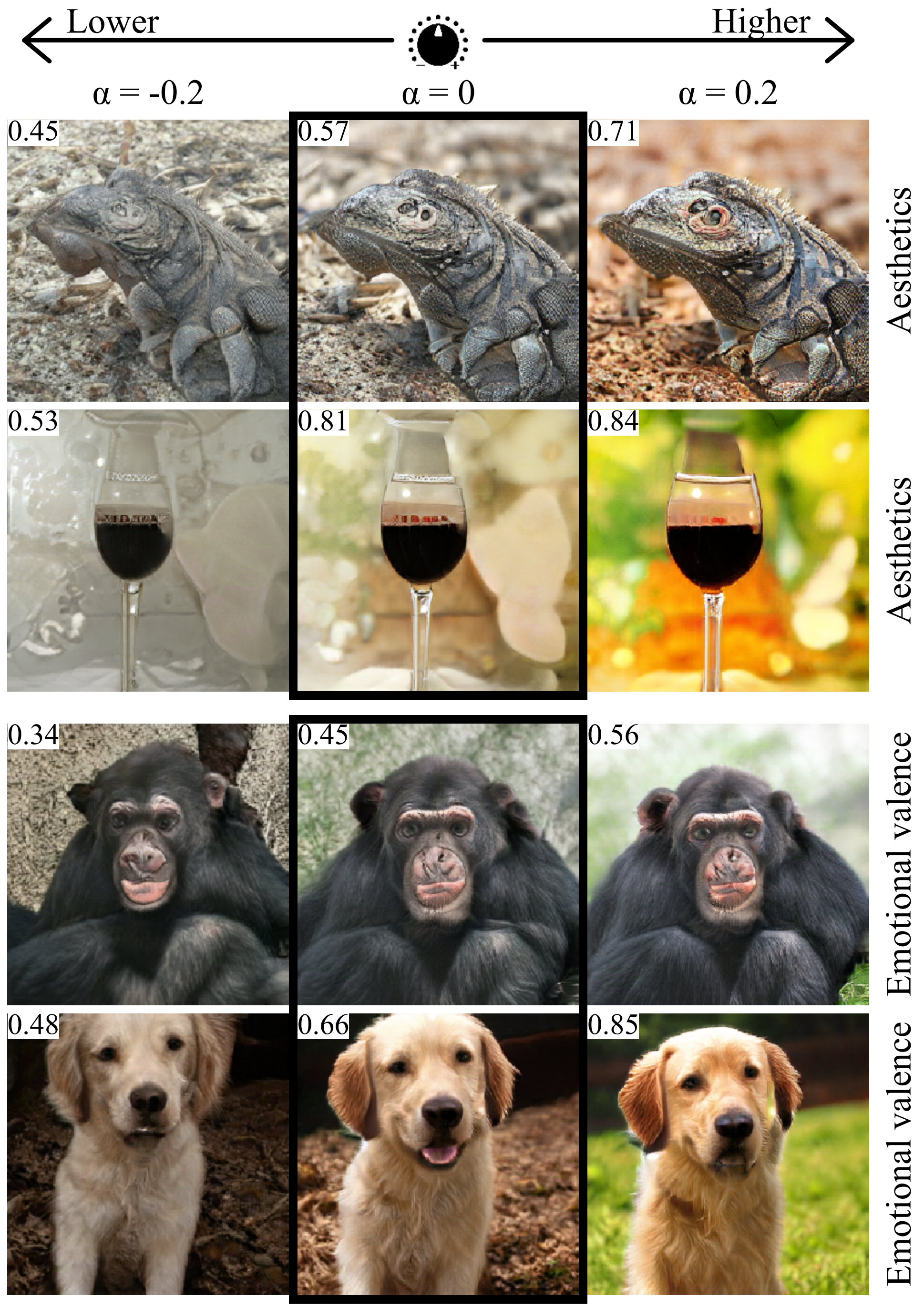


Figure 3-8: **Examples of generated images** along the aesthetics (top) and emotional valence (bottom) dimension. The middle column represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed. An images' Assessor score is shown in its top left corner. More examples are included in the Supplementary Materials.

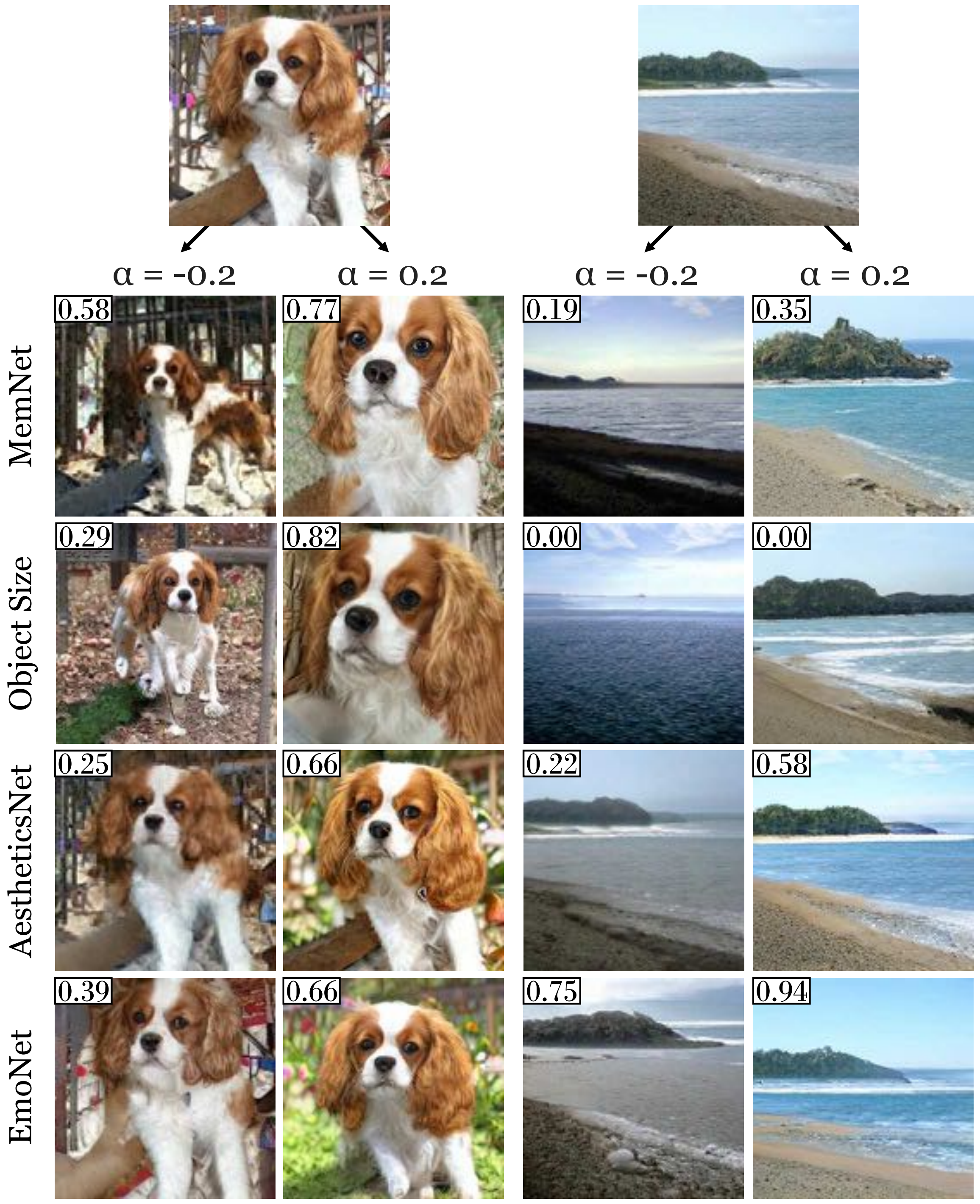


Figure 3-9: **Comparison of examples generated according to different Assessors.** The top row represents $G(\mathbf{z}, \mathbf{y})$, the generated image serving as the original seed to create a series of images with a higher or lower Assessor value. The respective Assessor values are indicated in the top left corner. Note that for object size, we used a different α range: {-0.8,0.8}.

selves, not user-supplied, real images. However, should advances in the field lead to an encoder network, this would become possible and it would open applications in graphics and education, for example, where selected images can be made more memorable. One should also be wary, though, of potential misuse, especially when applied to images of people or faces. Note that the BigGAN [14] generator used here was trained on ImageNet categories [92] which only occasionally include people, and that it does not allow to render realistically looking people. Nevertheless, with generative models yielding ever more realistic output, an increasingly important challenge in the field is to develop powerful detection methods to allow us to reliably distinguish generated, fake images from real ones [36] [35] [37].

Acknowledgments This work was partly funded by NSF award 1532591 in Neural and Cognitive Systems (to A.O), and by a fellowship (1108116N) and travel grant (V4.085.18N) awarded to Lore Goetschalckx by the Research Foundation - Flanders (FWO).

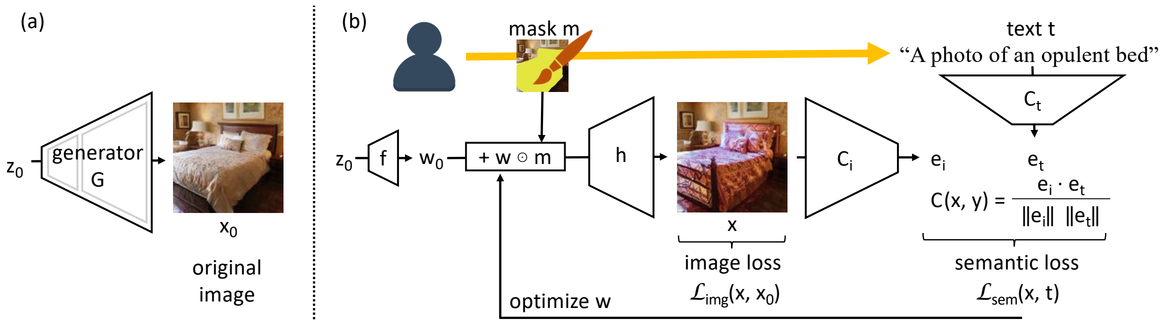
Chapter 4

Paint By Word

In this chapter, we investigate the problem of zero-shot semantic image painting. Instead of painting modifications into an image using only concrete colors or a finite set of semantic concepts, we ask how to create semantic paint based on open full-text descriptions: our goal is to be able to point to a location in a synthesized image and apply an arbitrary new concept such as “rustic” or “opulent” or “happy dog.” To do this, our method combines a state-of-the art generative model of realistic images with a state-of-the-art text-image semantic similarity network. We find that, to make large changes, it is important to use non-gradient methods to explore latent space, and it is important to relax the computations of the GAN to target changes to a specific region. We conduct user studies to compare our methods to several baselines.

4.1 Introduction

A writer can create vivid verbal imagery using just a few carefully-chosen words, but written scenes are abstract, without any specific physical form. In this paper, we ask how to enable an artist to use words to build an image concretely, painting textually described visual concepts at specific locations in a scene. Our work is inspired and enabled by recent dramatic progress in text-to-image generation [84, 88]. However, rather than having the AI compose the whole image, we ask how large models can be used collaboratively to let a person apply words as a paintbrush. We wish to enable



(a)
z_0
generator G
x_0
original image
(b)
mask m
text t
"A photo of an opulent bed"
C_t
z_0
f
w_0
+ w ⊙ m
h
C_i
e_i
e_t
x
$C(x, y) = \frac{e_i \cdot e_t}{\|e_i\| \ \|e_t\|}$
image loss
$\mathcal{L}_{img}(x, x_0)$
semantic loss
$\mathcal{L}_{sem}(x, t)$
optimize w

Figure 4-1: An overview of our method. (a) we begin with an image x_0 that is decoded from a latent z_0 by a pretranied generative model of realistic images $x_0 = G(z_0)$. (b) To edit the image, the user selects a region m and gives an arbitrary run of descriptive text t. The image generator $G(z) = h(f(z))$ is decomposed to expose a image representation w that can be split spatially between the painted and unpainted region. Then a text-image semantic similarity model $C(x, t)$ is used to define a semantic consistency loss that is used together with an image loss (minimizing changes outside the mask) to optimize w inside the region to make the synthesized region match the given text.

a user to paint brushstrokes on specific objects and regions within a generated image, then restyle, alter, or insert new objects by describing the desired visual concept with words.

The current work introduces the problem of zero-shot semantic image manipulation, in which the words and painting gestures available to the user are unconstrained and unknown ahead of time during model training. Our goal is to enable a user to point at an image and apply an arbitrary new concept that may include visual concepts as wide-ranging as "rustic" or "opulent" or "happy dog."

To paint with words, we pair large-scale generative adversarial networks (GANs [31]) that are trained unconditionally or with simple class conditioning [13, 57], together with a full-text image retrieval network trained to measure the semantic similarity between text and an image [84]. We ask whether such a semantic similarity model can provide enough information to drive the image synthesis process to match a specific description within or beyond the domain on which the GAN is trained. Then we ask how the structure of a GAN can be exploited to generate realistic images where a specific part of the image is modified to match a textual description. We propose a simple architecture to enable painting with words, and we apply it using CLIP [88] paired with StyleGAN2 [57] and BigGAN [13].

We conduct user studies to quantify the realism and accuracy of the changes made when using our method to edit a part of a generated image; we compare our method to several baselines, and we ask which types of visual concepts are easier or harder to paint by word. Finally we investigate several new semantic image manipulations that are enabled using our method. Code and data will be available upon publication.

4.2 Related Work

Our application is inspired by recent progress in the text-to-image synthesis problem [84, 87] as well as paint-based semantic image manipulation methods [6, 80].

Text-to-image synthesis. Synthesizing an image based on a text description is an ambitious problem that has attracted a variety of proposed solutions: initial RNN-based generators [32, 70] have been followed a series of by GAN-based [31] methods that have produced increasingly plausible images. The GAN methods have adopted two distinct approaches to generating realistic images from text. One is to generate images corresponding to text by sampling GAN latents that match semantics according to a separately trained image-text matching model [89]; this idea can be refined by viewing the sampling as an energy-based modeling procedure [79]. The second approach is to train a conditional generator that explicitly takes a language embedding as input [89]; the image quality of this approach can be improved using multi-stage generators [120, 121, 123], and semantic consistency can be improved through careful design of architectures and training losses [66, 83, 106, 115, 116, 130]. As an alternative approach, generating images from scene graphs instead of sentences [51] can allow a generator to exploit logical structure explicitly. Recent work stands apart by eschewing the use of GANs: the DALL-E [87] model generates images from text autoregressively using a transformer [110] based on GPT-3 [15] to jointly generate natural text and image tokens using a VQ-VAE encoding [88]; outputs are sampled to maximize semantic consistency via a state-of-the-art image-text matching model CLIP [84]. Artist Murdock has observed that CLIP can be used as a source of gradients to guide a generator [76–78]. The current paper is different from prior text-to-image work, because our goal is not to generate an image from text, but to define a paintbrush using text that enables a user to manipulate a semantics of a generated image at a specific painted location.

Semantic image painting. Our work is also inspired by a series of methods that have enabled users to construct realistic images by painting a finite selection of chosen concepts at user-specified locations; these can be thought of as "paint by number." In this setting, there have again been two approaches enabled by GANs. One approach is to find GAN latents to generate images that match a user's painted intention: this can be done to match color [128] or a finite vocabulary of visual concepts [6, 8] at a given location. The second approach is to train an image-conditional Pix2Pix [48, 112] model on the task of generating a realistic image with a semantic segmentation that matches a given painted input; the SPADE method [80] refines this approach to improve image quality given the limited information available in flat segmentation inputs. While these methods all enable a user to paint images using a finite vocabulary of semantic concepts, the goal of the current work is to enable a user to paint an unlimited vocabulary of visual concepts, specified by free text.

GAN latent space methods. Semantic manipulations in GAN latent spaces have been explored from several other perspectives. Early work on GANs observed the presence of some latent vector directions corresponding to semantic concepts [86], and further explorations have developed methods to identify interesting vector directions that steer latent space by using reconstruction losses [49, 100] or by learning from classification losses [23, 28]. One disadvantage of this approach is that a semantic direction can only be found if we train a model to look for it in advance. On the other hand, it has been observed that interior latents in GANs are partially disentangled [8]. So recent work has developed unsupervised methods for identifying disentangled interpretable directions in these interior latents [44, 114]. Our approach differs from the above because our method for identifying latent manipulations is supervised neither by a set of finite classes nor does it rely on disentanglement; rather it identifies semantic latent changes in a zero-shot manner using full-text image semantic similarity.

GAN inversion methods. We note that the leading state-of-the-art generative models are not trained with an encoder. Therefore, in order to apply GAN manipulation methods on real user-provided images, one must solve GAN inversion. That is, we must be able to encode a given image in the latent space of the GAN. Because several powerful editing methods are enabled by GAN latent manipulation, the problem of inverting a GAN is of widespread interest and is the subject of an active line of ongoing research [1, 2, 9, 33, 43, 69, 90, 127, 128, 131]. The GAN inversion problem is complementary to our work. In this paper, we shall assume that the image to be edited is represented in the latent space of the generator.

4.3 Method

There are two challenging problems that must be simultaneously solved in order to implement paint-by-word: first, we must achieve *semantic consistency*, altering the painted part of the image to match the text description given by the user; and second, we must achieve *realism*: the altered part of the image should have a plausible appearance. In particular, the newly painted content should be consistent with the context of the rest of the image in terms of pose, color, lighting, and style.

We adopt a framework that addresses these two concerns using two separately trained networks: the first is a semantic similarity network $C(x, t)$ that scores the semantic consistency between an image x and a text description t. This network need not be concerned with the overall realism of the image. The second is a convolutional generative network $G(z)$ that is trained to synthesize realistic images given a random z; this network enforces realism. Given G and C, we can formulate the following optimization to generate a realistic image $G(z^*)$ that matches descriptive text t:

$$z^* = \arg\max_z C(G(z), t) \tag{4.1}$$

This simple approach factors the problem into two models that can be trained at large

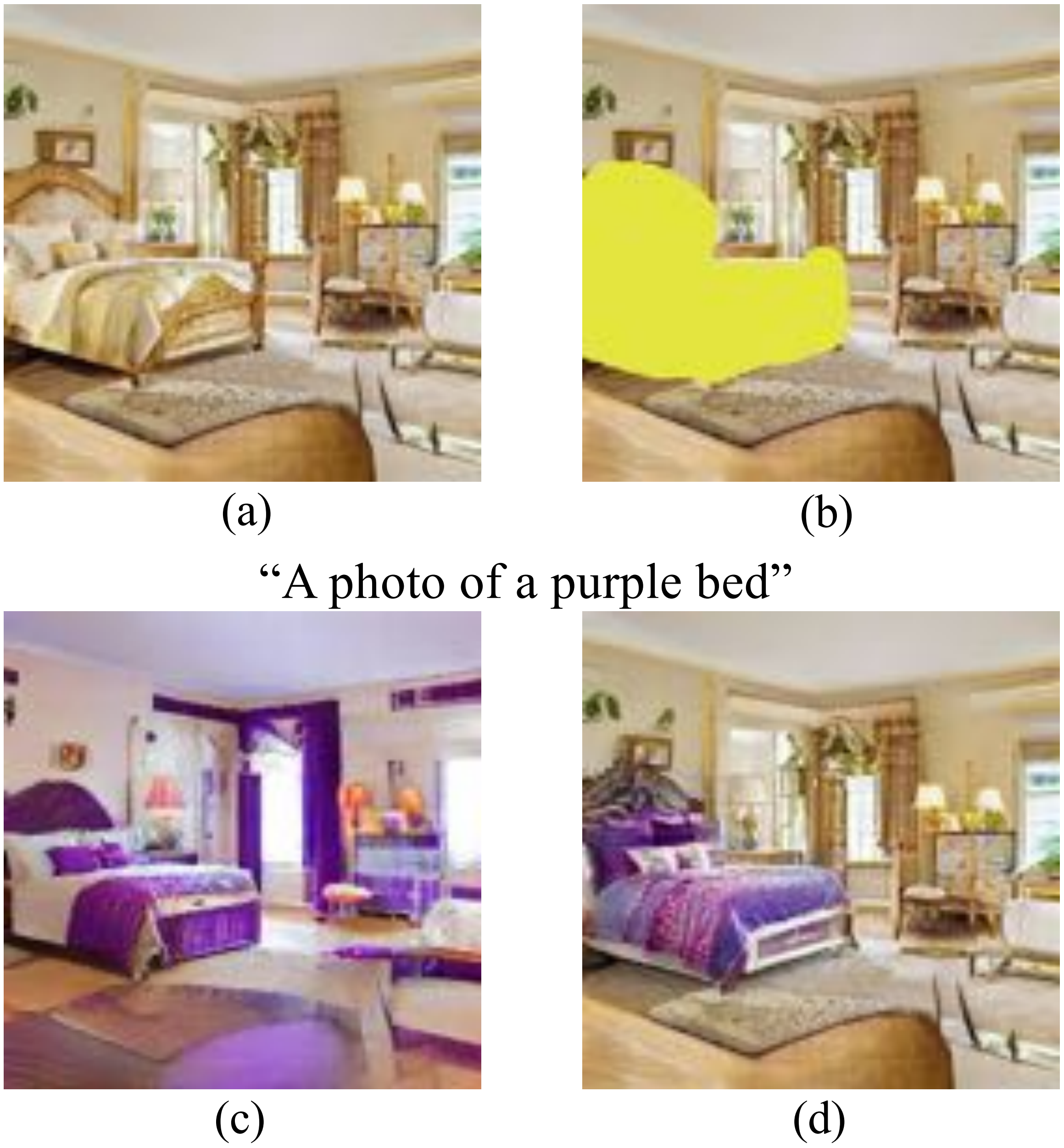


Figure 4-2: Comparing the effect of optimizing with and without a spatially split generator. The masked loss $C_{t,m}(x)$ is used to generate modifications of a generated image (a) that matches the description “A photo of a purple bed.” where the loss is masked to the user-painted region m as shown in (b). The modification is done in two ways. In (c), the original unmodified StyleGAN2 [57] model (trained on LSUN [117] bedrooms) is used and an optimizer finds a $G(z)$ that minimizes the masked loss. In (d), the optimization is done over a spatially-split generator $G_{z,m}(w)$. Although in both cases, the semantic loss examines only the region of the bed, only the split generator allows changes to be made in the user-specified region without altering the rest of the scene.

scale without any awareness of each other, and we shall use it as a starting point. It allows us to take advantage of recent progress in state-of-the-art models that can be used for G and C (Such as StyleGAN [55, 57], BigGAN [13], CLIP [84], and ALIGN [50]). Because our method allows the direct use of such large-scale models, this simple architecture is a promising approach for generating images from a text description even without any spatial conditions. We study this approach empirically in Section 4.4.1.

4.3.1 Semantic consistency in a region

When providing semantic paint, the method of Equation 4.1 is not sufficient, because it does not direct changes specifically towards a user's chosen painted area. To focus effects on to one area, we direct the matching network C to attend only to the region of the user's brushstroke instead of the whole image. This can be done in a straightforward way: given a user-supplied mask m, we define a masked semantic similarity model

$$C_{t,m}(x) = C(x \odot m, t) \tag{4.2}$$

Here $x \odot m$ is a projection that zeroes the components of x outside the region of the mask m.

By hiding the regions outside the mask from the semantic similarity model, the masked model $C_{t,m}(x)$ focuses the optimization on the selected region. However, in practice we find that this is not enough to direct changes to only a single object. Figure 4-2(c) shows the effect of optimizing $\arg\max_z C_{t,m}(G(z))$ to match a text description in a StyleGAN2 [57] model. Although no gradients pass from C to G outside the masked region, G has a computational structure that links the appearance of objects outside the mask to objects within the mask.

In order to allow a user to edit a single object, we must obtain a generative model that allows the region inside of the mask to be unlinked. Interestingly, this can be done without training a new large-scale model from scratch.

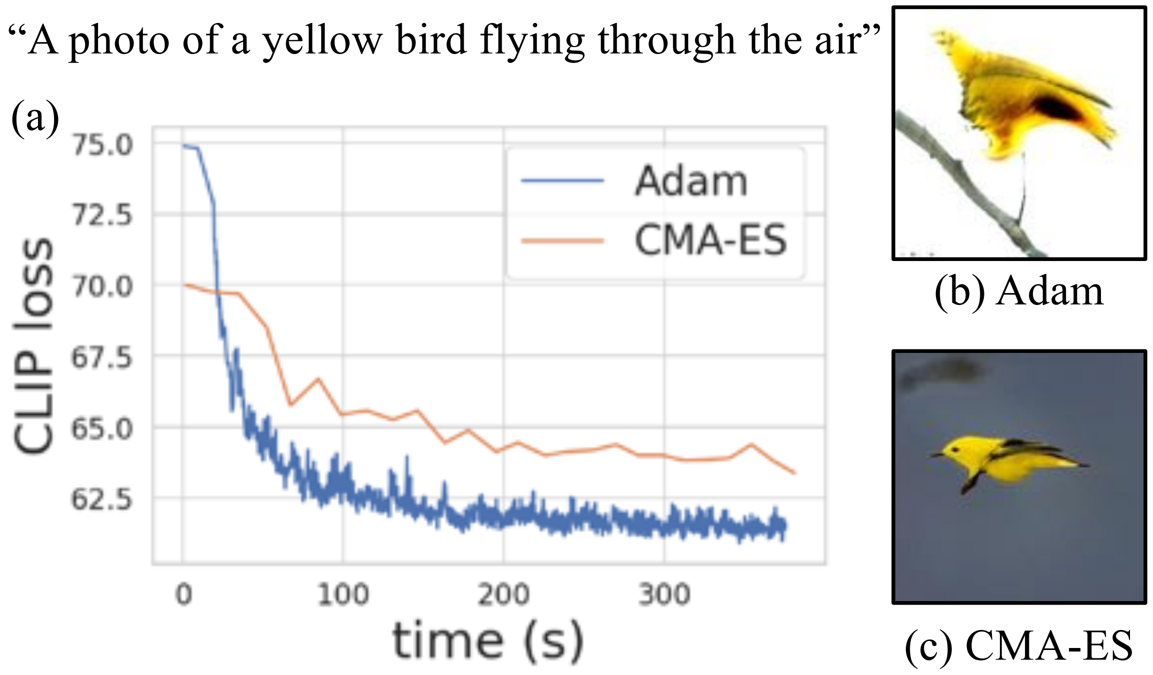


Figure 4-3: Avoiding adversarial examples when optimizing an image to match the description "A photo of a yellow bird flying through the air." The gradient descent method (Adam) obtains good scores while yielding an unrealistic image that looks neither like it is flying, nor like a real bird. In practice, we obtain better results by using the non-gradient sampling method (CMA-ES). Although samples do not achieve scores that are as good, the generated samples avoid becoming adversarial, and user studies (Section 4.4.1) show that results are both more realistic and accurate in practice.

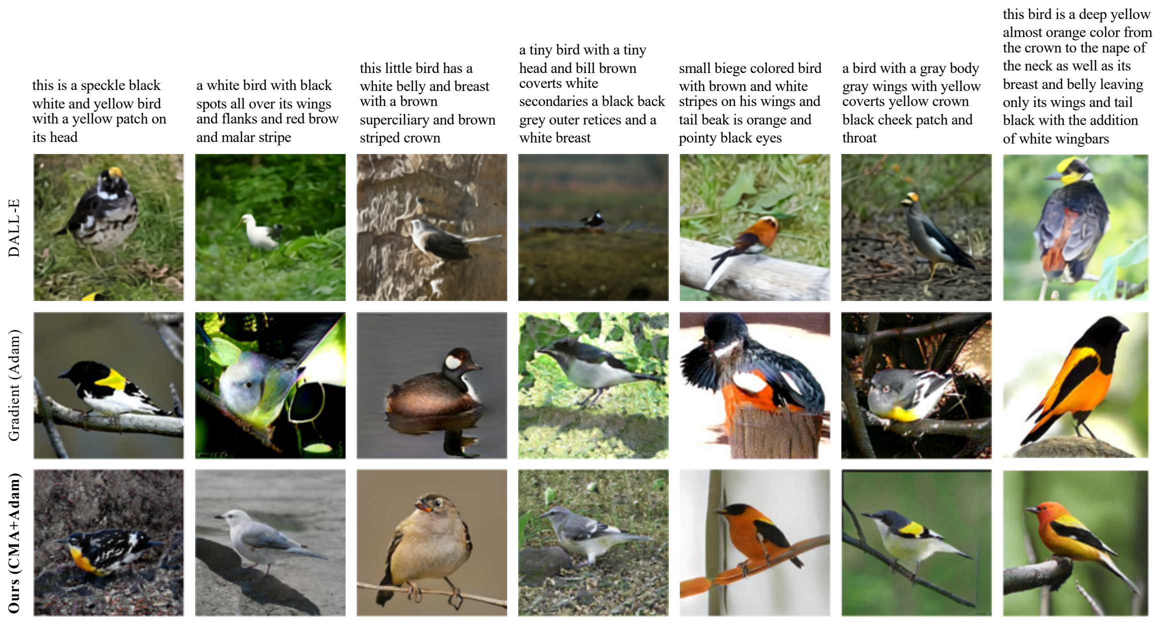

this is a speckle black white and yellow bird with a yellow patch on its head
a white bird with black spots all over its wings and flanks and red brow and malar stripe
this little bird has a white belly and breast with a brown superciliary and brown striped crown
a tiny bird with a tiny head and bill brown coverts white secondaries a black back grey outer retices and a white breast
small biege colored bird with brown and white stripes on his wings and tail beak is orange and pointy black eyes
a bird with a gray body gray wings with yellow coverts yellow crown black cheek patch and throat
this bird is a deep yellow almost orange color from the crown to the nape of the neck as well as its breast and belly leaving only its wings and tail black with the addition of white wingbars
DALL-E
Gradient (Adam)
Ours (CMA+Adam)

Figure 4-4: Qualitative comparison of a handful of birds from the user study conducted in Section 4.4.1. Our method (third row) is applied to synthesizing full images based on description of birds. It is compared to an ablation where the Adam optimizer is used alone instead of CMA-ES; and we also compare our method to DALL-E [88]. Note that images from our method tend to be more realistic more often when compared to the other implementations.

4.3.2 Generative modeling in a region

To create a generative model that decouples the region inside and outside the mask, we exploit the convolutional structure of G. We work with an intermediate latent representation of the image w that has a spatial structure over the field of the image. To access this structure, we express G as two steps:

$$x = G(z) = h(f(z)) \tag{4.3}$$

$$w = f(z) \tag{4.4}$$

Given a mask m provided by the user's brushstrokes, we then decompose w into a portion outside the mask w_0 and a portion inside the mask w_1, as follows:

$$w = w_0 + w_1 \tag{4.5}$$

$$w_1 = w \odot m \tag{4.6}$$

Here $w \odot m$ is a projection that zeroes the components of w outside the region of the mask m. In practice, we downsample m to the relevant feature map resolution(s) of w, and use the Hadamard product to zero the outside components.

For a given original image $G(z)$ we can fix the original $w_0 = w - w \odot m$ while allowing w_1 to vary. This split gives us a new generative model, where the representation outside the mask m is fixed as w_0, while the representation inside the mask is parameterized by w:

$$G_{z,m}(w) = h(w_0 + w \odot m) \tag{4.7}$$

Splitting w_0 and w in this way relaxes the problem, and allows $G_{z,m}(w)$ to generate images that the original model $G(z)$ could not generate.

In BigGAN, we use as w the featuremap output of a layer of the network. This featuremap has a direct natural spatial structure, and $w \odot m$ simply zeros the featuremap locations outside the mask. In our BigGAN experiments we split the featuremap output of the first convolutional block of the generative network.

In StyleGAN, the natural interior latent, the w vector, modulates featuremaps by changing channel normalizations uniformly across the spatial extent of the featuremap at all layers. In our formulation, we split the w style latent vector spatially by applying a one style modulation outside the user-specified mask, fixed as w_0, and another style modulation w inside the mask. In StyleGAN, the style modulation is applied across all layers, so we apply this split at the appropriate resolution at every layer of the model: the effect is to have a model that has two style vectors w_0 and w, instead of one.

This additional flexibility is instrumental for allowing the generator to create user-specified attributes in the region of interest without changing attributes of unrelated parts of the image. Figure 4-2(d) shows the effect of splitting the model: the appearance of one object can be changed without changing the appearance of other objects in the scene. Combining (4.2) with (4.7) allows us to define the following masked semantic consistency loss in the region

$$\mathcal{L}_{\text{sem}}(w) = -C_{t,m}(G_{z,m}(w)) \tag{4.8}$$

In order to explicitly limit changes in the synthesized image outside the painited region, we apply the following image consistency loss:

$$\mathcal{L}_{\text{img}}(w) = d(x \odot (1-m), G_{z,m}(w) \odot (1-m)) \tag{4.9}$$

Here d denotes an image similarity metric: in our experiments, for d we use a sum of an L2 pixel difference and the LPIPS [122] perceptual similarity. In $\mathcal{L}_{\text{img}}$, the inverse of the user's masked region is applied, so this loss term does not limit changes inside the painted region.

The full loss adds the two loss terms:

$$\mathcal{L}(w) = \mathcal{L}_{\text{sem}}(w) + \lambda_{img}\mathcal{L}_{\text{img}}(w) \tag{4.10}$$

$$w^* = \arg\min_{w} \mathcal{L}(w) \tag{4.11}$$

4.3.3 Avoiding adversarial attack using CMA

When using gradients to optimize a single image to a deep network objective, it is well-understood that it is easy to obtain an adversarial example [105] that fools the network, achieving a strong score without being a typical representative of the distribution modeled by the network. For example, it is easy to obtain an image that is classified as one class while having the appearance of an unrelated class.

The problem of adversarial attack also arises in our application. Figure 4-3(a) plots the loss when optimizing Equation 4.1 using the Adam [61] optimizer, where G is a StyleGAN2 [57] trained on LSUN birds [117], and where C is the pretrained CLIP `ViT-B/32` network [84]. The convergence plot shows a rapid and stable-looking improvement as the image is modified to better match the phrase "A photo of a yellow bird flying through the air," but the longer the optimization runs, the less like a flying bird the image becomes in practice. Figure 4-3(b) shows the image that results after several minutes of optimization: it achieves a strong score and yet has many artifacts that look obviously synthetic. It does not resemble a photo of a bird.

One way to think about the problem is that our networks C and G were trained for optimal expected behavior over a distribution, not worst-case behavior for every individual. So when optimizing a single instance it is not hard to find an individual point at which both models misbehave.

We solve this problem by switching optimization strategies. Instead of seeking out a single optimal image, we use the Covariance Matrix Adaptation evolution strategy (CMA-ES) [34], which is a non-gradient method that aims to optimize a Gaussian distribution to have minimal loss when random samples are drawn from the distribution. Although CMA-ES may not achieve individual point losses that are as low

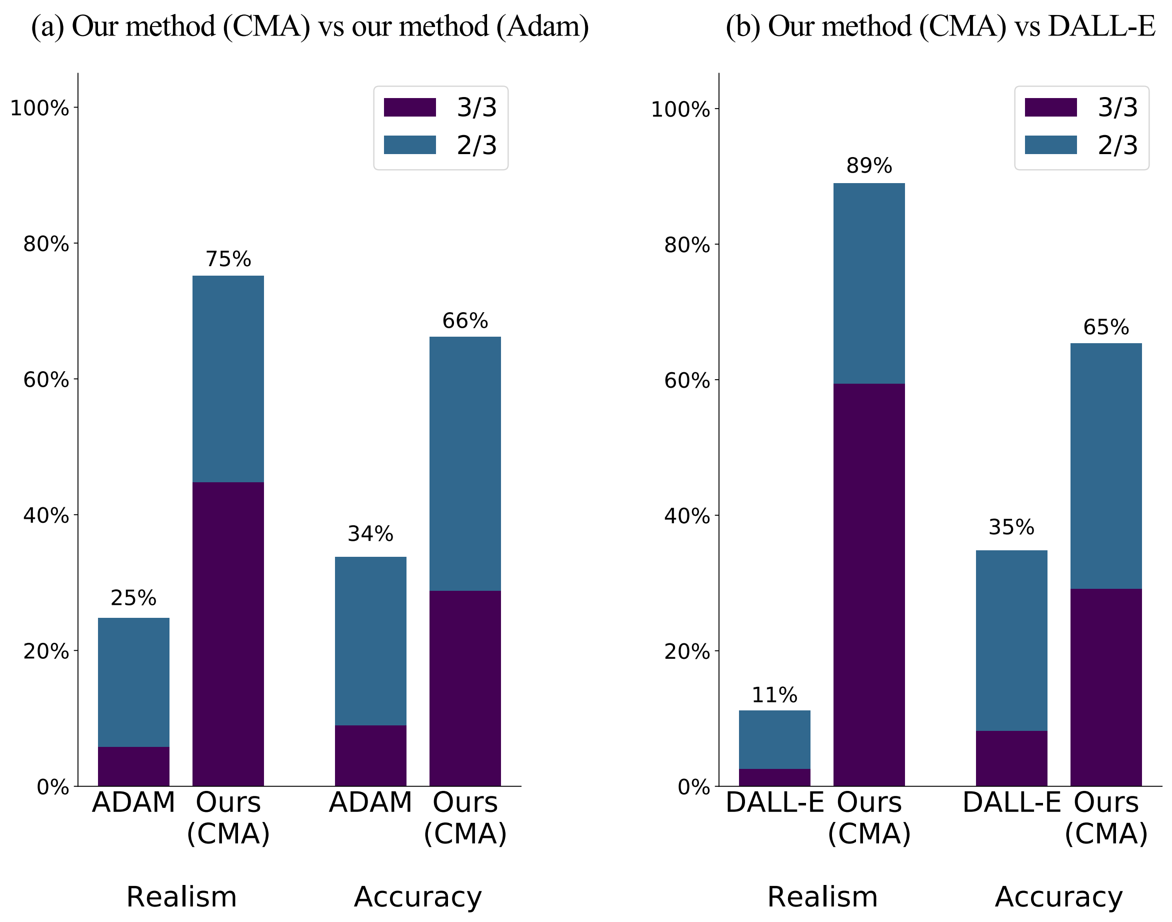


Figure 4-5: User study comparing the semantic accuracy and realism of our basic method (Equation 4.1). We ask users to compare images generated by our method to two baselines; three workers examined each pair of images to make each assessment. We compared our method as implemented using CMA [34] versus to our method using the Adam optimizer [61]. Our CMA method outperforms our Adam method in both accuracy and realism. To compare to a state-of-the-art baseline, we also compare our method to DALL-E [88], which is text-to-image system that was trained on a much broader and more difficult domain. Our method outperforms DALL-E in this setting, possibly because here our generator is trained to specialize on the narrow domain of birds. In this test, our method pairs a 256-pixel StyleGAN2 [55, 57] trained on CUB birds [111] with the CLIP [84] `ViT/32B` text-image semantic similarity network.

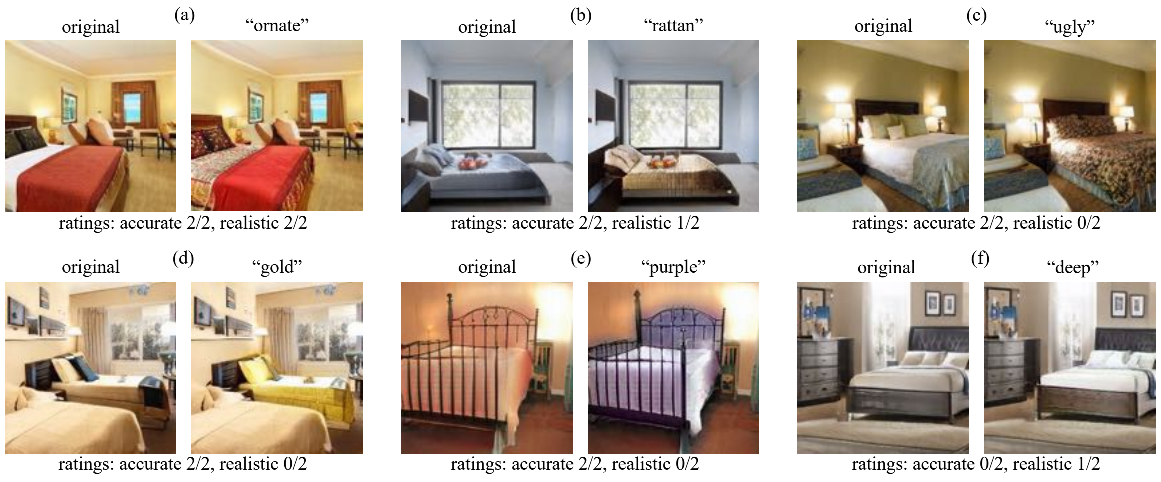

original
(a)
"ornate"
ratings: accurate 2/2, realistic 2/2
original
(b)
"rattan"
ratings: accurate 2/2, realistic 1/2
original
(c)
"ugly"
ratings: accurate 2/2, realistic 0/2
original
(d)
"gold"
ratings: accurate 2/2, realistic 0/2
original
(e)
"purple"
ratings: accurate 2/2, realistic 0/2
original
(f)
"deep"
ratings: accurate 0/2, realistic 1/2

Figure 4-6: Representative successes and failures in a large-scale user study of image edits. 3000 edits were made on 300 randomly sampled StyleGAN2 bedroom outputs, and the edited image was compared to the original by crowdsourced workers. In cases a-e, 2/2 workers rated the modified image as a better match to the word than the original, and in case (f), both workers rated the modified image as a worse match to the word. In case (a) 2/2 workers rated the edited image as more realistic than the original, in cases (b) and (f) the two workers split on realism, and in cases c-e, both workers rated the edited image as less realistic than the original.

as a gradient method like Adam, we find that in practice, the images that it obtains match modeled semantics very well, while remaining more realistic than the images produced by gradient descent: Figure 4-3(c) shows a typical image from the distribution after running CMA optimization for an equivalent amount of time as Adam. In Section 4.4.1 we compare CMA-ES to Adam in a human evaluation.

4.4 Results

To understand the strengths and weaknesses of our method, we conduct user studies in two problem settings. Then we explore the types of new image manipulations that are enabled by paint-by-word.

4.4.1 Testing full image generation

Without a user-provided mask, our method is a simple factorization of the text-conditional image generation problem (Equation 4.1). We wish to understand if this

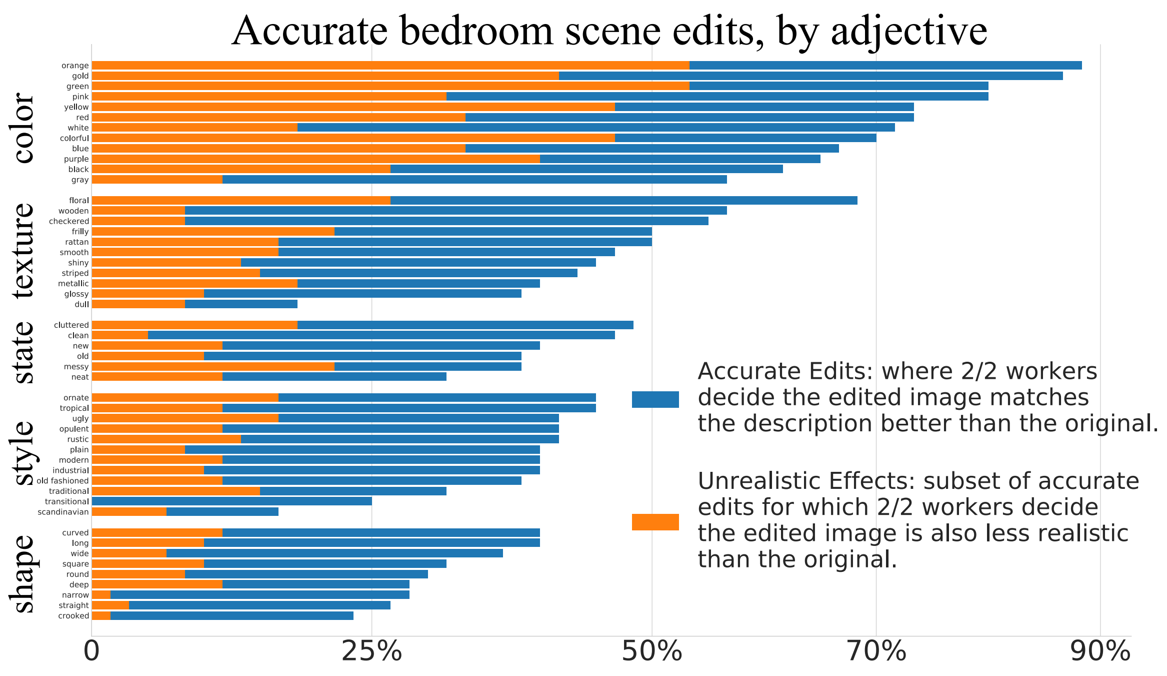


Figure 4-7: A detailed breakdown of user edits that were considered accurate by 2/2 workers, according to individual word. The most accurate edits among those we tested are for the color orange; and the least accurate are for the word "Scandinavian". Edits that were considered less realistic than the original are shown in orange.

straightforward split of the problem has disadvantages compared to other approaches that explicitly condition image generation, so we begin by evaluating this architecture in comparison to a state-of-the-art model.

For G we train a 256-pixel StyleGAN2 [57] with Adaptive Data Augmentation [55] on the CUB birds data set [111]. Standard training settings are used: the generator is unconditional, so no bird class labels nor text descriptions are revealed to the network during training. For C we use an off-the-shelf CLIP [84] model, using the pretrained `ViT-B/32` weights; this model was trained on an extensive proprietary data set of 400 million image/caption pairs. We generate images using two different techniques to maximize $C(G(z), t)$. As a simple baseline we optimize z using first-order gradient descent (Adam [61]). Then we also optimize z using CMA [34] followed by Adam.

To evaluate our method, we generate images from 500 descriptions of birds from a test dataset also used for evaluation of DALL-E in [88]. To compare the images generated by our method to images generated by DALL-E, we conduct a user study on Amazon Mechanical Turk [5] in which workers are asked to choose which of a pair

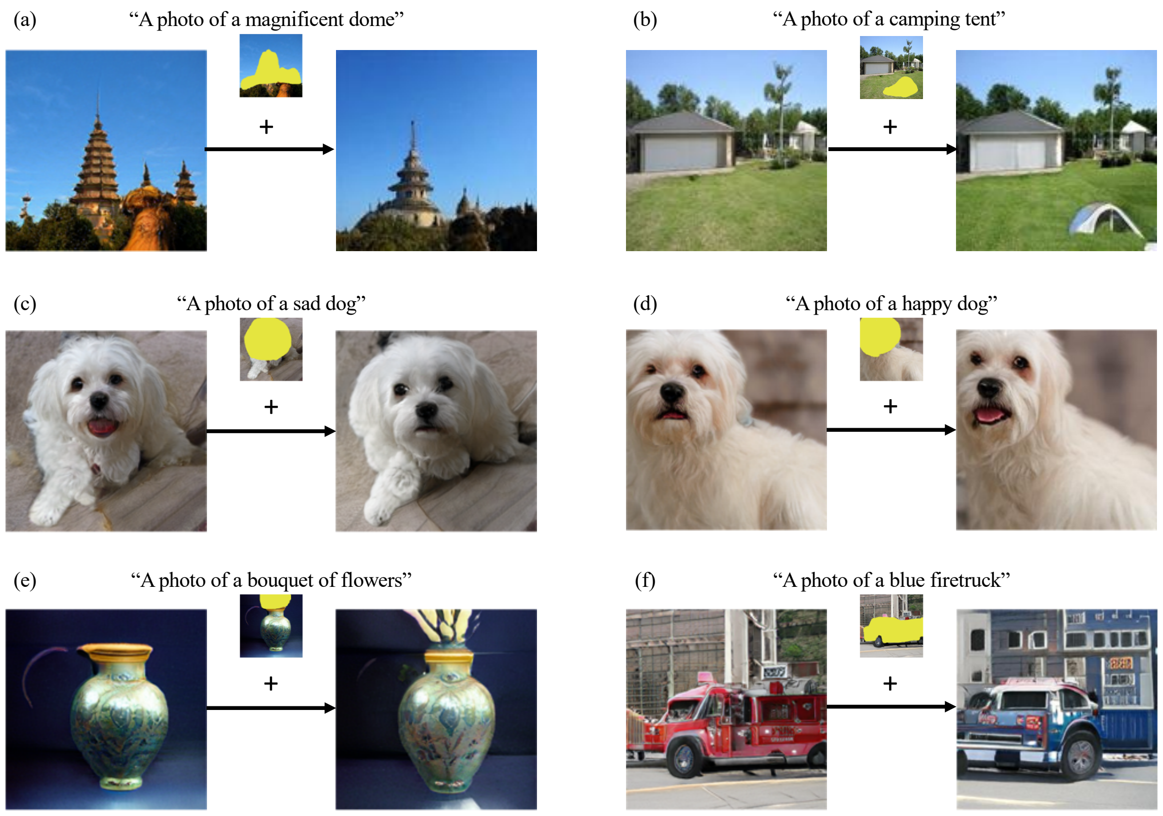

(a)
"A photo of a magnificent dome"
+
(b)
"A photo of a camping tent"
+
(c)
"A photo of a sad dog"
+
(d)
"A photo of a happy dog"
+
(e)
"A photo of a bouquet of flowers"
+
(f)
"A photo of a blue firetruck"
+

Figure 4-8: Exploring edits using our method with BigGAN. Because BigGAN models have been pretrained on very broad distributions, they provide an opportunity to experiment with a broad range of types of edits.

	color	texture	state	style	shape
accurate (2/2)	72.8	46.5	40.6	37.2	31.7
not accurate (0/2)	3.9	13.2	20.0	20.4	22.8
realistic (2/2)	13.1	18.2	21.7	26.1	25.4
not realistic (0/2)	48.8	34.1	30.0	27.8	27.2

Table 4.1: User study of edits, by semantic category. Percentages shown include only cases where both evaluations agree. "Accurate" counts cases where the edited image is a better match for the text. "Realistic" counts cases where the edited image is more realistic than the original. In both cases the opposite counts are also shown.

of images are more realistic, and in a separately asked question, which of a pair more closely resemble a given text description.

The results are shown in Figure 4-5. We find that, for this test setting, our method gives competitive results. The CMA optimization process is able to steer G to generate an image that is found to be a better fit for the description than the DALL-E method 65.4% of the time, and more realistic 89% of the time. The CMA method is also better than Adam alone, more accurate in 66.2%; more realistic in 75.2%.

This experiment is not intended to show that our simple method is superior to DALL-E: it is not. Our network is trained only on birds; it utterly fails to draw any other type of subject. Because of this narrow focus, it is unsurprising that it might be better at drawing realistic bird images than the DALL-E model, which is trained on a far broader variety of unconstrained images. Nevertheless, this experiment demonstrates that it is possible to obtain state-of-the-art semantic consistency, at least within a narrow image domain, without explicitly training the generator to take information about the textual concept as input.

4.4.2 A large scale user study of bedroom image edits

Next, we conduct a large-scale user study on localized editing of objects in bedroom scenes. For this study, we sample 300 images generated by StyleGAN2 [57] trained on LSUN bedrooms [117] and manually paint a single bed in each image. Then for each image we perform 10 paint-by-word tests by applying one of a set of 50 text descriptions of colors, textures, styles, states, and shapes. Humans compare each of the 3000 edited images to the corresponding original image: they evaluate whether one image is better described by the text description than the other, and separately whether one image is more realistic than the other. Each comparison is done without revealing which image is the original. Each comparison is evaluated by two people.

Results are tabulated by category in Table 4.1, and charted by word in Figure 4-7. Our method can decisively edit a range of colors and textures, but it is also effective for a range of other visual classes to a lesser degree. In all tested categories, most visual words can have some positive effect discerned by raters, however our method is weakest at being able to alter the shapes of objects. Examples of success and failure cases from the test are shown in Figure 4-6.

In Table 4.1, we note that ratings of realism of edited images are worst in the same categories where the strength of the semantic consistency is best. Some of the loss of realism is due to visible artifacts such as the blurred colors seen in Figure 4-6(e). However, other cases without noticeable visual artifacts are rated as less realistic by raters as in Figure 4-6(c,d), including many of the most interesting edits in the study. In these cases, the new painted styles are noticeable and stand out as less realistic than the original because the painted object now has an atypical style for the context - such as a gold bed in a cream-colored room, or an ugly bed in a hotel room. The portion of effective edits that are rated as unrealistic is shown as orange bars in Figure 4-6.

4.4.3 Demonstrating paint-by-word on a broader diversity of image domains

To demonstrate the usefulness of our method beyond the narrow domains of birds and bedrooms, we apply our method on BigGAN models [13]. We apply two BigGAN models. First, we apply a BigGAN model trained on MIT Places [126]; and then we apply a BigGAN model trained on Imagenet [22].

Because the BigGAN models are trained on broad distributions of images, these generative models allow us to experiment with a wide variety of different editing operations. In Figure 4-8, We demonstrate our method's ability to alter the style of buildings outdoors (a); we also demonstrate the ability to add new objects that did not previously exist in the scene (b). Both (a) and (b) demonstrate the potential of of BigGAN trained on Places to be used to manipulate general scene images. On BigGAN Imagenet, we discover the ability to change whether a dog is happy or sad (c-d). We can also add new objects to an appropriate context (e). A challenging task is to force the model to synthesize objects outside its training domain. Although the Imagenet training data contains no blue firetrucks, we can use language to specify that we wish for the model to render a firetruck as blue (f). Using our method, it can create blue parts of a vehicle, but the out-of-domain task of creating a whole blue firetruck remains difficult and introduces distortions.

4.5 Discussion

We have introduced the problem of paint-by-word, a human-AI collaborative task which enables a user to edit a generated image by painting a semantic modification specified by any text description, to any location of the image. Our work is enabled by the recent development of large-scale high-performance networks for realistic image generation and accurate text-image similarity matching. We have shown that a natural combination of these powerful components can work well enough to enable paint-by-word. Our user studies and demonstrations of effective edits have taken a first step in characterizing the opportunities, challenges and potential utility in this new application.

Acknowledgements. We thank Aditya Ramesh at OpenAI for assistance with DALL-E evaluation data. We thank IBM for the donation to MIT of the Satori supercomputer that enabled training BigGAN on MIT Places, and we thank OpenAI, Google, and Nvidia for publishing weights for pretrained large-scale CLIP, BigGAN and StyleGAN models that make this work possible. We thank Hendrik Strobelt and Daksha Yadav for their insights, encouragement, and valuable discussions, and we are grateful for the support of DARPA XAI (FA8750-18-C-0004), the Vannevar Bush Faculty Fellowship Program of the ONR (N00014-16-1-3116 to A.O.), and Signify Lighting Research.

Chapter 5

Conclusion

5.1 Contributions

The overarching goal of this work is to promote the proposed “software 3.0” as both an effective emerging approach for forwarding AI research and a design principle for constructing more sophisticated AI systems. We support these claims with a concrete case study on controllable neural image synthesis with Generative Adversarial Networks (GANs). We further demonstrate how insights in other scientific domains can be made using this approach to AI research as well as downstream user applications.

5.2 Future directions

The rate of progress in the field currently is staggering. Foundation models demonstrate raw potential and are rapidly gaining interest in the community due to their leading performance and surprising emergent capabilities. However, the name was specifically chosen in part to capture their *incomplete* nature and need for further adaptation and study. While foundation models present immediate opportunities for significant advances in AI, the most important reason for studying how they work and how they can be controlled is that these models are just as quickly being deployed into the real world with far-reaching consequences.

For example’s GitHub’s Copilot based on OpenAI’s Codex model [18] is in the technical preview phase where it is already assisting developers in write code in meaningful ways. Yet, many questions remain about whether the model can be safely deployed, who will be able to claim ownership of the code generated by the model as it continues to improve, whether or not proper licensing protocols will be followed and enforced by the model, among many others. A recent cybersecurity evaluation of Copilot's code contributions suggest that its current suggestions may introduce bugs and security vulnerabilities about 40% of the time [81]. The Free Software Foundation has gone as far as to say that Copilot is "unacceptable and unjust," [91] further reinforcing the need to study these models from different persepctives. In general, powerful but partially opaque machine learning models are being deployed at a rate that likely exceeds our growth in understanding, hence warranting increased investigation in this area.

Beyond Software 3.0. Language models that learn to produce source code from natural language prompting are of particular interest to developers whose primary concrete output is source code to produce apps, websites, etc,. In a similar fashion, AI researchers and practitioners spend a significant amount of time reading, writing and debugging the source code to run machine learning experiments. Exciting early evidence shows that these models not only have knowledge of generic programmatic concepts and tasks, but even of machine learning model and training code, which potentially includes knowledge of code that resembles the code used to produce the model itself. Moreover, these models demonstrate elementary abilities to produce terminal commands and shell scripts that would required to navigate a command line and execute generated code. In other words, models that are quite literally *self-replicating* and *self-improving* with respect to their own source code appear to be on the imminent horizon. These observations are particularly motivating and hint at yet the next version of software, whereby AI models are responsible for producing (or assist in producing) the source code to train new machine learning models. A major breakthrough would be to instill these models with *agency* so that they can write and carry out machine learning experiments on their own with simple natural language instructions from the programmer, perhaps even with the explicit intent to train better performing versions of itself. Once models reach this level of proficiency, the bottleneck on experimental throughput and the ability to investigate new ideas will no longer be the rate at which a human programmer can implement it, but on our far more scalable ability to increase computational throughput.

Bibliography

[1] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan: How to embed images into the stylegan latent space? In *ICCV*, 2019. 65

[2] Rameen Abdal, Yipeng Qin, and Peter Wonka. Image2stylegan++: How to edit the embedded images? In *CVPR*, 2020. 65

[3] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017. 28

[4] Dalya Baron. Machine learning in astronomy: A practical overview. *arXiv preprint arXiv:1904.07248*, 2019. 18

[5] Jeff Barr and Luis Felipe Cabrera. Ai gets a brain: New technology allows software to tap real human intelligence. *Queue*, 4(4):24–29, 2006. 75

[6] David Bau, Hendrik Strobelt, William Peebles, Jonas Wulff, Bolei Zhou, Jun-Yan Zhu, and Antonio Torralba. Semantic photo manipulation with a generative image prior. *ACM Transactions on Graphics*, 38(4):1–11, 2019. 63, 64

[7] David Bau, Bolei Zhou, Aditya Khosla, Aude Oliva, and Antonio Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6541–6549, 2017. 44

[8] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Zhou Bolei, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. Gan dissection: Visualizing and understanding generative adversarial networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 38, 64

[9] David Bau, Jun-Yan Zhu, Jonas Wulff, William Peebles, Hendrik Strobelt, Bolei Zhou, and Antonio Torralba. Inverting layers of a large generator. In *ICLR Debugging Machine Learning Models Workshop*, 2019. 65

[10] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow, 2021. 20

[11] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut,

Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021. 20

[12] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *CoRR*, abs/1709.07857, 2017. 44

[13] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018. 25, 30, 31, 33, 62, 63, 67, 78

[14] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. *CoRR*, abs/1809.11096, 2018. 43, 44, 45, 59

[15] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020. 20, 63

[16] Tony W Buchanan and Ralph Adolphs. The role of the human amygdala in emotional modulation of long-term declarative memory. *Advances in Consciousness Research*, 44:9–34, 2002. 42

[17] Zoya Bylinskii, Phillip Isola, Constance Bainbridge, Antonio Torralba, and Aude Oliva. Intrinsic and extrinsic effects on image memorability. *Vision research*, 116:165–178, 2015. 42

[18] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. 81

[19] Radoslaw Martin Cichy, Kshitij Dwivedi, Benjamin Lahner, Alex Lascelles, Polina Iamshchinina, M Graumann, A Andonian, NAR Murty, K Kay, G Roig, et al. The algonauts project 2021 challenge: How the human brain makes sense of a world in motion. *arXiv preprint arXiv:2104.13714*, 2021. 18

[20] Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future healthcare journal*, 6(2):94, 2019. 17

[21] Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron C Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017. 28, 30

[22] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 78

[23] Emily Denton, Ben Hutchinson, Margaret Mitchell, and Timnit Gebru. Detecting bias with generative counterfactual face attribute augmentation. In *Fairness, Accountability, Transparency and Ethics in Computer Vision Workshop (in conjunction with CVPR)*, 2019. 64

[24] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *arXiv preprint arXiv:1610.07629*, 2016. 28, 30

[25] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. Gansynth: Adversarial neural audio synthesis. *arXiv preprint arXiv:1902.08710*, 2019. 25

[26] Tianyu Gao, Adam Fisch, and Danqi Chen. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723*, 2020. 21

[27] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 30

[28] Lore Goetschalckx, Alex Andonian, Aude Oliva, and Phillip Isola. Ganalyze: Toward visual definitions of cognitive image properties. *arXiv preprint arXiv:1906.10112*, 2019. 26, 64

[29] Ian Goodfellow. 4 years of gan progress, 2018 (accessed July 31, 2019). `https://twitter.com/goodfellow_ian/status/969776035649675265`. 25

[30] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014. 42, 44

[31] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information Processing Systems*, 2014. 62, 63

[32] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. In *International Conference on Machine Learning*, pages 1462–1471. PMLR, 2015. 63

[33] Shanyan Guan, Ying Tai, Bingbing Ni, Feida Zhu, Feiyue Huang, and Xiaokang Yang. Collaborative learning for faster stylegan embedding. *arXiv preprint arXiv:2007.01758*, 2020. 65

[34] Nikolaus Hansen. The cma evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772*, 2016. 13, 72, 73, 75

[35] Karen Hao. Deepfakes have got congress panicking. this is what it needs to do. *MIT Technology Review*, Jun 2019. 59

[36] Drew Harwell. Top ai researchers race to detect 'deepfake' videos: 'we are outgunned'. *The Washington Post*, Jun 2019. 59

[37] Drew Harwell. Top ai researchers race to detect 'deepfake' videos: 'we are outgunned'. *The Washington Post*, Jun 2019. 59

[38] David Hasler and Sabine E Suesstrunk. Measuring colorfulness in natural images. In *Human vision and electronic imaging VIII*, volume 5007, pages 87–96. International Society for Optics and Photonics, 2003. 49

[39] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 49

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 55

[41] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020. 20

[42] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, pages 6626–6637, 2017. 49

[43] Minyoung Huh, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, and Aaron Hertzmann. Transforming and projecting images to class-conditional generative networks. In *ECCV*, 2020. 65

[44] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *NeurIPS*, 2020. 64

[45] Shantanu Ingle and Madhuri Phute. Tesla autopilot: semi autonomous driving, an uptick for future autonomy. *International Research Journal of Engineering and Technology*, 3(9):369–372, 2016. 17

[46] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 31

[47] Phillip Isola, Jianxiong Xiao, Devi Parikh, Antonio Torralba, and Aude Oliva. What makes a photograph memorable? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1469–1482, jul 2014. 42, 43, 51

[48] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *CVPR*, pages 1125–1134, 2017. 64

[49] Ali Jahanian, Lucy Chai, and Phillip Isola. On the"steerability" of generative adversarial networks. *arXiv preprint arXiv:1907.07171*, 2019. 26, 44, 64

[50] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V Le, Yunhsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. *arXiv preprint arXiv:2102.05918*, 2021. 67

[51] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, pages 1219–1228, 2018. 63

[52] Andrej Karpathy. Software 2.0, 2017 (accessed May, 2021. `https://karpathy.medium.com/software-2-0-a64152b37c35`. 18

[53] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017. 30

[54] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *CoRR*, abs/1710.10196, 2017. 44

[55] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Training generative adversarial networks with limited data. In *NeurIPS*, 2020. 13, 67, 73, 75

[56] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. *CoRR*, abs/1812.04948, 2018. 44

[57] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. *CVPR*, 2020. 12, 13, 62, 63, 66, 67, 72, 73, 75, 77

[58] Makena Kelly. Congress grapples with how to regulate deepfakes, 2019 (accessed July 31, 2019. `https://www.theverge.com/2019/6/13/18677847/deep-fakes-regulation-facebook-adam-schiff-congress-artificial-intelligence`. 25

[59] Aditya Khosla, Wilma A. Bainbridge, Antonio Torralba, and Aude Oliva. Modifying the memorability of face photographs. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2013. 44

[60] Aditya Khosla, Akhil S. Raju, Antonio Torralba, and Aude Oliva. Understanding and predicting image memorability at a large scale. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015. 42, 44, 45, 51

[61] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 13, 72, 73, 75

[62] Shu Kong, Xiaohui Shen, Zhe Lin, Radomir Mech, and Charless Fowlkes. Photo aesthetics ranking network with attributes and content adaptation. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 662–679, Cham, 2016. Springer International Publishing. 54

[63] Mathias Kraus, Stefan Feuerriegel, and Asil Oztekin. Deep learning in business analytics and operations research: Models, applications and managerial implications. *European Journal of Operational Research*, 281(3):628–641, 2020. 18

[64] Susan Leavy, Barry O'Sullivan, and Eugenia Siapera. Data, power and bias in artificial intelligence. *arXiv preprint arXiv:2008.07341*, 2020. 17

[65] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021. 21

[66] Wenbo Li, Pengchuan Zhang, Lei Zhang, Qiuyuan Huang, Xiaodong He, Siwei Lyu, and Jianfeng Gao. Object-driven text-to-image synthesis via adversarial training. In *CVPR*, pages 12174–12182, 2019. 63

[67] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021. 21

[68] Jae Hyun Lim and Jong Chul Ye. Geometric gan. *arXiv preprint arXiv:1705.02894*, 2017. 30

[69] Zachary C Lipton and Subarna Tripathi. Precise recovery of latent vectors from generative adversarial networks. *arXiv preprint arXiv:1702.04782*, 2017. 65

[70] Elman Mansimov, Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Generating images from captions with attention. In *ICLR*, 2016. 63

[71] Francisco Massa and Ross Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch. https://github.com/facebookresearch/maskrcnn-benchmark, 2018. Accessed: Mar 2019. 49

[72] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015. 44

[73] Bryce Meredig. Five high-impact research areas in machine learning for materials science, 2019. 18

[74] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv:1802.05957*, 2018. 29, 30

[75] Mathew Monfort, Alex Andonian, Bolei Zhou, Kandan Ramakrishnan, Sarah Adel Bargal, Yan Yan, Lisa Brown, Quanfu Fan, Dan Gutfreund, Carl Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE transactions on pattern analysis and machine intelligence*, 2019. 55

[76] Ryan Murdock. Aleph2Image. https://colab.research.google.com/drive/1Q-TbYvASMPRMXCOQjkxxf72CXYjR_8Vp, February 2021. 64

[77] Ryan Murdock. The Big Sleep. https://colab.research.google.com/drive/1NCceX2mbiKOSlAd_o7IU7nA9UskKN5WR, January 2021. 64

[78] Ryan Murdock. Thoughts on deepdaze, bigsleep, and aleph2image. https://rynmurdock.github.io/2021/02/26/Aleph2Image.html, January 2021. 64

[79] Anh Nguyen, Jeff Clune, Yoshua Bengio, Alexey Dosovitskiy, and Jason Yosinski. Plug & play generative networks: Conditional iterative generation of images in latent space. In *CVPR*, pages 4467–4477, 2017. 63

[80] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pages 2337–2346, 2019. 63, 64

[81] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. An empirical cybersecurity evaluation of github copilot's code contributions, 2021. 82

[82] Kuan-Chuan Peng, Tsuhan Chen, Amir Sadovnik, and Andrew C Gallagher. A mixed bag of emotions: Model, predict, and transfer emotion distributions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 860–868, 2015. 55

[83] Tingting Qiao, Jing Zhang, Duanqing Xu, and Dacheng Tao. Mirrorgan: Learning text-to-image generation by redescription. In *CVPR*, pages 1505–1514, 2019. 63

[84] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 13, 61, 62, 63, 67, 72, 73, 75

[85] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015. 27, 30

[86] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. 64

[87] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. *arXiv preprint arXiv:2102.12092*, 2021. 20, 63

[88] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *NeurIPS*, 2019. 13, 61, 62, 63, 69, 73, 75

[89] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis. In *ICML*, pages 1060–1069. PMLR, 2016. 63

[90] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. *arXiv preprint arXiv:2008.00951*, 2020. 65

[91] Donald Robertson. Fsf-funded call for white papers on philosophical and legal questions around copilot, 2021. 82

[92] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *Int. J. Comput. Vision*, 115(3):211–252, December 2015. 45, 59

[93] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016. 49

[94] Helena Sarin. Playing a game of ganstruction, 2018 (accessed July 31, 2019). `https://thegradient.pub/playing-a-game-of-ganstruction/`. 18, 25

[95] Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21, 2021. 17

[96] Timo Schick, Sahana Udupa, and Hinrich Schütze. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *arXiv preprint arXiv:2103.00453*, 2021. 21

[97] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020. 18

[98] Ala Shaabana. The future of ai is decentralized, 2021 (accessed May, 2021). `https://twitter.com/goodfellow_ian/status/969776035649675265`. 20

[99] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. *arXiv preprint arXiv:1907.10786*, 2019. 44

[100] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, pages 9243–9252, 2020. 64

[101] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. *arXiv preprint arXiv:2010.15980*, 2020. 21

[102] Aliaksandr Siarohin, Gloria Zen, Cveta Majtanovic, Xavier Alameda-Pineda, Elisa Ricci, and Nicu Sebe. How to make an image more memorable?: A deep style transfer approach. In *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, ICMR 2017, Bucharest, Romania, June 6-9, 2017*, pages 322–329, 2017. 44

[103] Oleksii Sidorov. Changing the image memorability: From basic photo editing to gans. *CoRR*, abs/1811.03825, 2018. 44

[104] Lionel Standing. Learning 10000 pictures. *The Quarterly journal of experimental psychology*, 25(2):207–222, 1973. 42

[105] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 72

[106] Ming Tao, Hao Tang, Songsong Wu, Nicu Sebe, Fei Wu, and Xiao-Yuan Jing. DF-GAN: Deep fusion generative adversarial networks for text-to-image synthesis. *arXiv preprint arXiv:2008.05865*, 2020. 63

[107] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020. 19

[108] Tue Tjur. Coefficients of determination in logistic regression models—a new proposal: The coefficient of discrimination. *The American Statistician*, 63(4):366–372, 2009. 15, 55

[109] Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017. 30

[110] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017. 63

[111] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Takeshi Mita. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 13, 73, 75

[112] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *CVPR*, pages 8798–8807, 2018. 64

[113] Sarah Webb. Deep learning for biology. *Nature*, 554(7693), 2018. 18

[114] Zongze Wu, Dani Lischinski, and Eli Shechtman. Stylespace analysis: Disentangled controls for stylegan image generation. *arXiv preprint arXiv:2011.12799*, 2020. 64

[115] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. Attngan: Fine-grained text to image generation with attentional generative adversarial networks. In *CVPR*, pages 1316–1324, 2018. 63

[116] Guojun Yin, Bin Liu, Lu Sheng, Nenghai Yu, Xiaogang Wang, and Jing Shao. Semantics disentangling for text-to-image generation. In *CVPR*, pages 2327–2336, 2019. 63

[117] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 12, 66, 72, 77

[118] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 30

[119] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018. 44

[120] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *CVPR*, pages 5907–5915, 2017. 63

[121] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. Stackgan++: Realistic image synthesis with stacked generative adversarial networks. *PAMI*, 41(8):1947–1962, 2018. 63

[122] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 71

[123] Zizhao Zhang, Yuanpu Xie, and Lin Yang. Photographic text-to-image synthesis with a hierarchically-nested adversarial network. In *CVPR*, pages 6199–6208, 2018. 63

[124] Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 2018. 44

[125] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *Proceedings of the 2015 International Conference on Learning Representations (ICLR)*, May 2015. 44

[126] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *Neural Information Processing Systems*, 2014. 78

[127] Jiapeng Zhu, Yujun Shen, Deli Zhao, and Bolei Zhou. In-domain gan inversion for real image editing. In *ECCV*, 2020. 65

[128] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, pages 597–613. Springer, 2016. 64, 65

[129] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 44

[130] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. In *CVPR*, pages 5802–5810, 2019. 63

[131] Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. Improved stylegan embedding: Where are the good latents? *arXiv preprint arXiv:2012.09036*, 2020. 65