MACHINE LEARNING
DEPARTMENT

Deep Generative Models II (GANs & Friends, Diffusion Models, …)

Zachary Lipton & Henry Chai

10701 — November 6th

US AI Insight Forum

US AI Insight Forum Topics

- Forum 1: Opening
- Forum 2: Innovation
- Forum 3: Workforce
- Forum 4: High Impact AI
- Forum 5: Democracy & Elections
- Forum 6: Privacy and Liability

Some Attendees

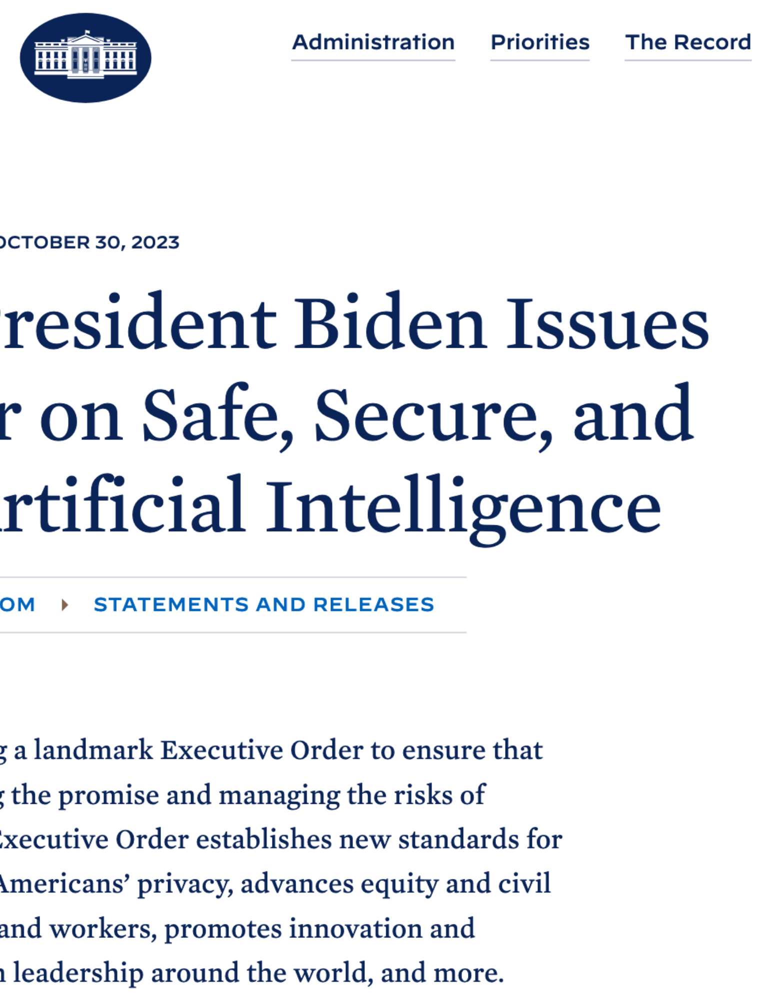
Session 1:

- Elon Musk
- Mark Zuckerberg
- Sundar Pichai
- Satya Nadella
- Eric Schmidt
- Jensen Huang
- Bill Gates
- Sam Altman
- Deborah Raji

Session 6

- Mark Surman – Mozilla Foundation (President)
- Bernard Kim — Match Group (CEO)
- Arthur Evans Jr – American Psychological Association (CEO)
- Mutale Nkonde — AI for the People (CEO)
- Ganesh Sitaraman – Vanderbilt Law
- Gary Shapiro — Consumer Technology Association (CEO)
- Tracy Pizzo Frey—Common Sense Media
- (ZL—10701 Co-Instructor)

Executive Order



Administration Priorities The Record

OCTOBER 30, 2023

FACT SHEET: President Biden Issues Executive Order on Safe, Secure, and Trustworthy Artificial Intelligence

BRIEFING ROOM ▸ STATEMENTS AND RELEASES

Today, President Biden is issuing a landmark Executive Order to ensure that America leads the way in seizing the promise and managing the risks of artificial intelligence (AI). The Executive Order establishes new standards for AI safety and security, protects Americans' privacy, advances equity and civil rights, stands up for consumers and workers, promotes innovation and competition, advances American leadership around the world, and more.

Blueprint for an AI Bill of Rights

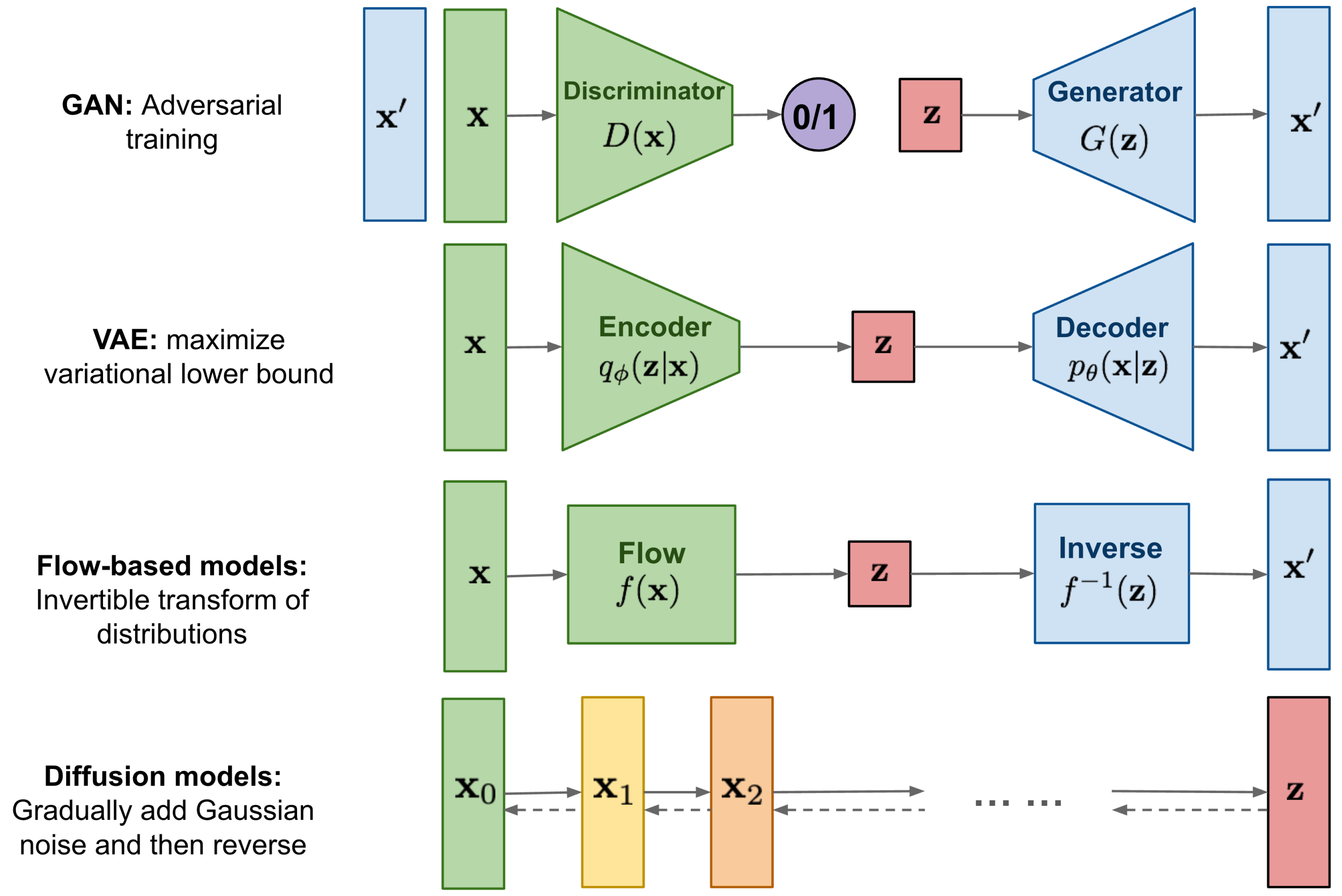
BLUEPRINT FOR AN AI BILL OF RIGHTS

MAKING AUTOMATED SYSTEMS WORK FOR THE AMERICAN PEOPLE

OSTP

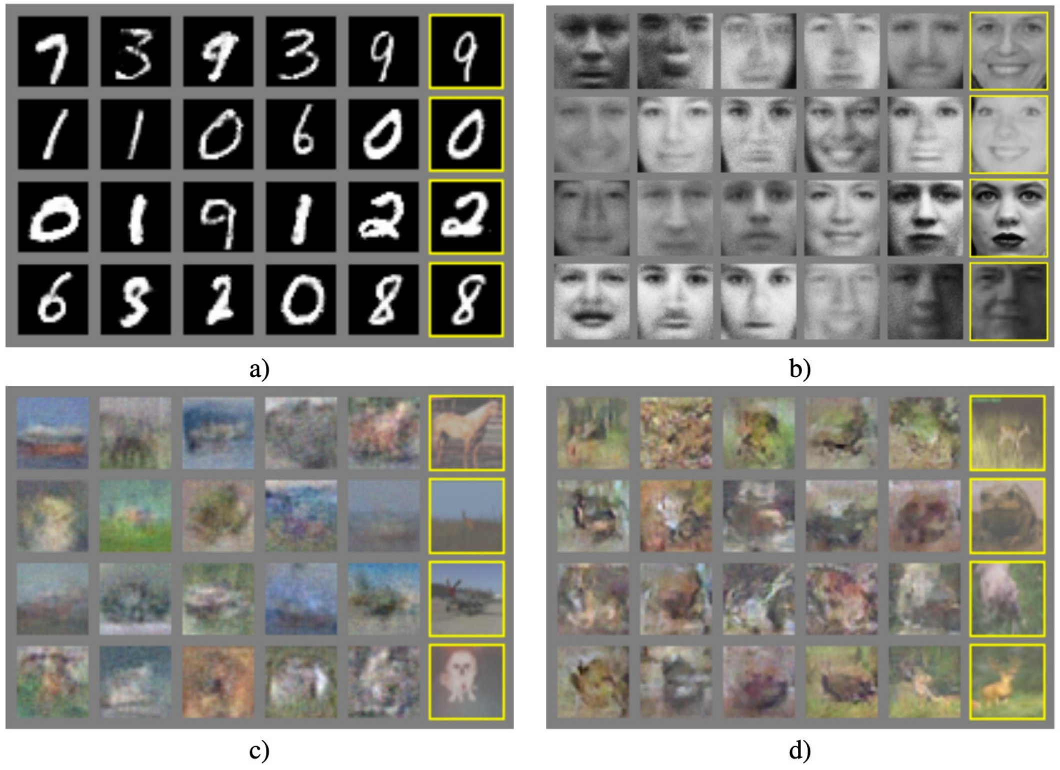
Among the great challenges posed to democracy today is the use of technology, data, and automated systems in ways that threaten the rights of the American public. Too often, these tools are used to limit our opportunities and prevent our access to critical resources or

Back to Deep Generative Models

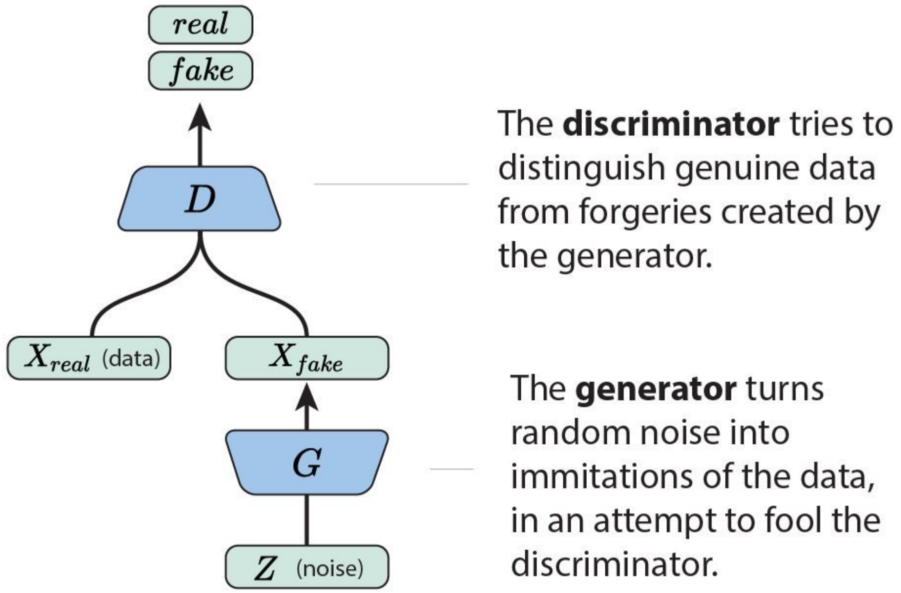

GAN: Adversarial training
$\mathbf{x}'$
$\mathbf{x}$
Discriminator $D(\mathbf{x})$
0/1
$\mathbf{z}$
Generator $G(\mathbf{z})$
$\mathbf{x}'$
VAE: maximize variational lower bound
$\mathbf{x}$
Encoder $q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{z}$
Decoder $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}'$
Flow-based models: Invertible transform of distributions
$\mathbf{x}$
Flow $f(\mathbf{x})$
$\mathbf{z}$
Inverse $f^{-1}(\mathbf{z})$
$\mathbf{x}'$
Diffusion models: Gradually add Gaussian noise and then reverse
$\mathbf{x}_0$
$\mathbf{x}_1$
$\mathbf{x}_2$
... ...
$\mathbf{z}$

Img source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

2014 Generative Adversarial Networks



Generative Adversarial Networks (GANs) are a way to make a generative model by having two neural networks compete with each other.



The **discriminator** tries to distinguish genuine data from forgeries created by the generator.

The **generator** turns random noise into immitations of the data, in an attempt to fool the discriminator.

Figure credit: Chris Olah

Generative Adversarial Networks (Goodfellow et al. NeuRIPS 2014)

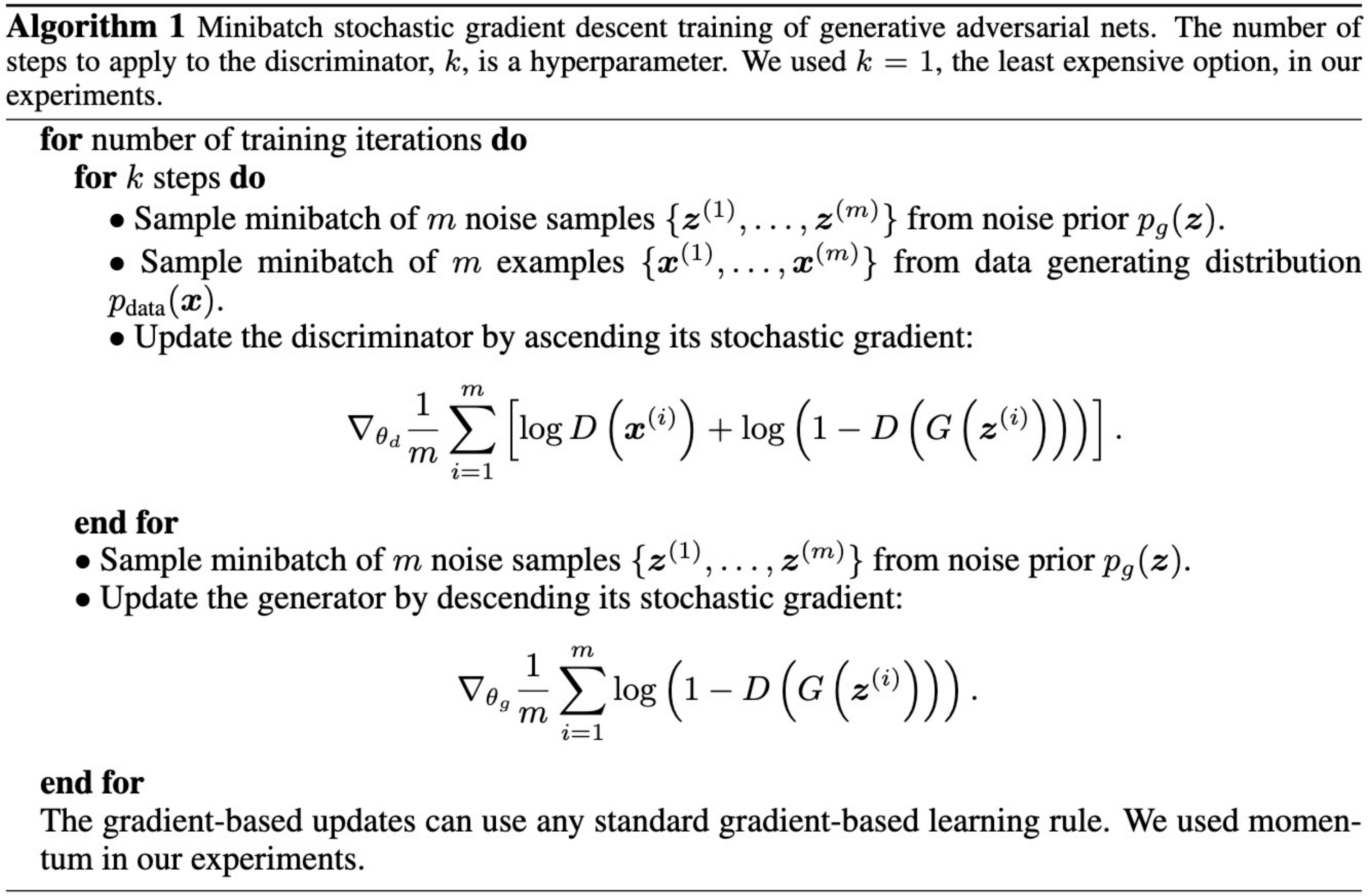
GANs: original training setup

- Dispense with representing likelihood, settle for implicit generation
- Two-player game:
  - Generator (w parameters θ_g)
  - Discriminator (w parameters θ_d)
- Minmax objective

$$\min_{\theta_g} \max_{\theta_d} \quad \mathbb{E}_{x\sim\mathrm{data}} \log D_{\theta_g}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

- Discriminator works to classify real data as real, fake data as fake
- Generator works to make discriminator misclassify fake data as real

Pseudocode



Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
        • Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
        • Update the discriminator by ascending its stochastic gradient:
            $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$
    end for
    • Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \dots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
    • Update the generator by descending its stochastic gradient:
        $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$
end for
```

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Iterative training, modified objective

1. Steps of gradient ascent to improve discriminator on

$$\max_{\theta_d} \quad \mathbb{E}_{x\sim\mathrm{data}} \log D_{\theta_g}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

2. Steps of gradient ascent to improve generator on different objective

$$\max_{\theta_g} \quad \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

Optimizing GANs

- Ideally, at equilibrium:
  1. Generator exactly matches the real data distribution
  2. Discriminator cannot distinguish real from fake $D_{\theta_d}(x) = .5, \forall x$

- In reality:
  - Training is difficult and unstable
  - Balance between discriminator and generator hard to maintain
  - Susceptible to mode collapse: G(x) lacks diversity, D(x) super accurate

Mode Collapse

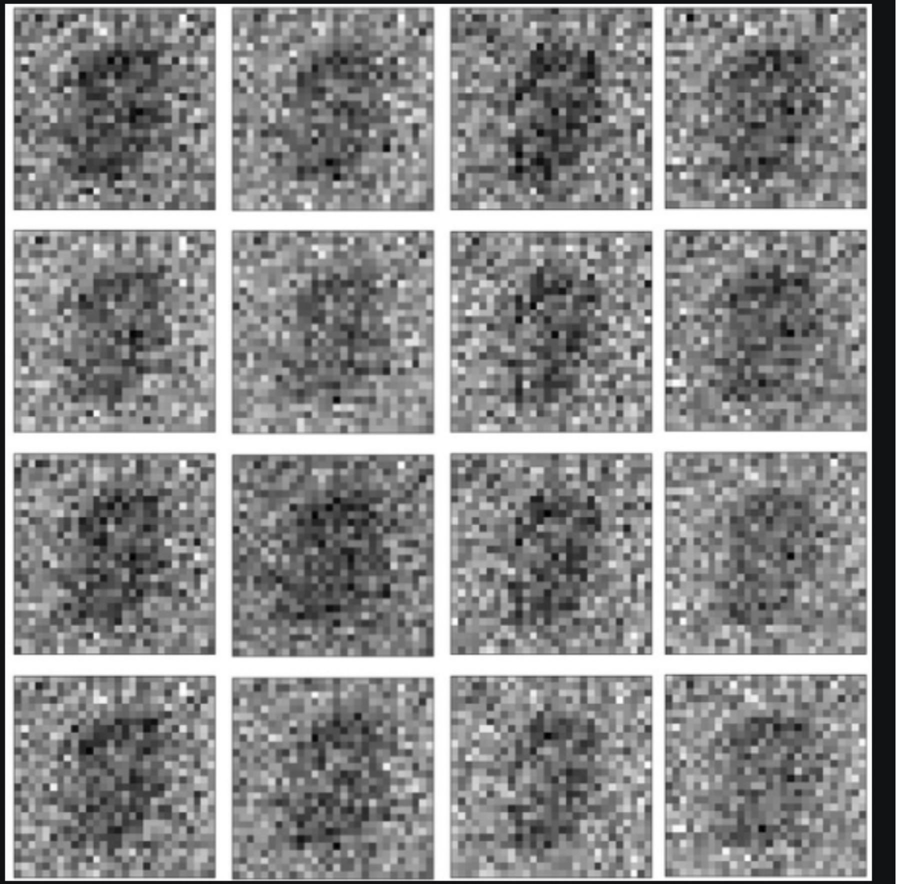


Fig. 4 – These are the output from one of the unstable training. This is on the same training code as above and slightly tweaked hyperparameters, but even after 300 epochs, you can see how bad our images are – an example of convergence failure | Source: Author

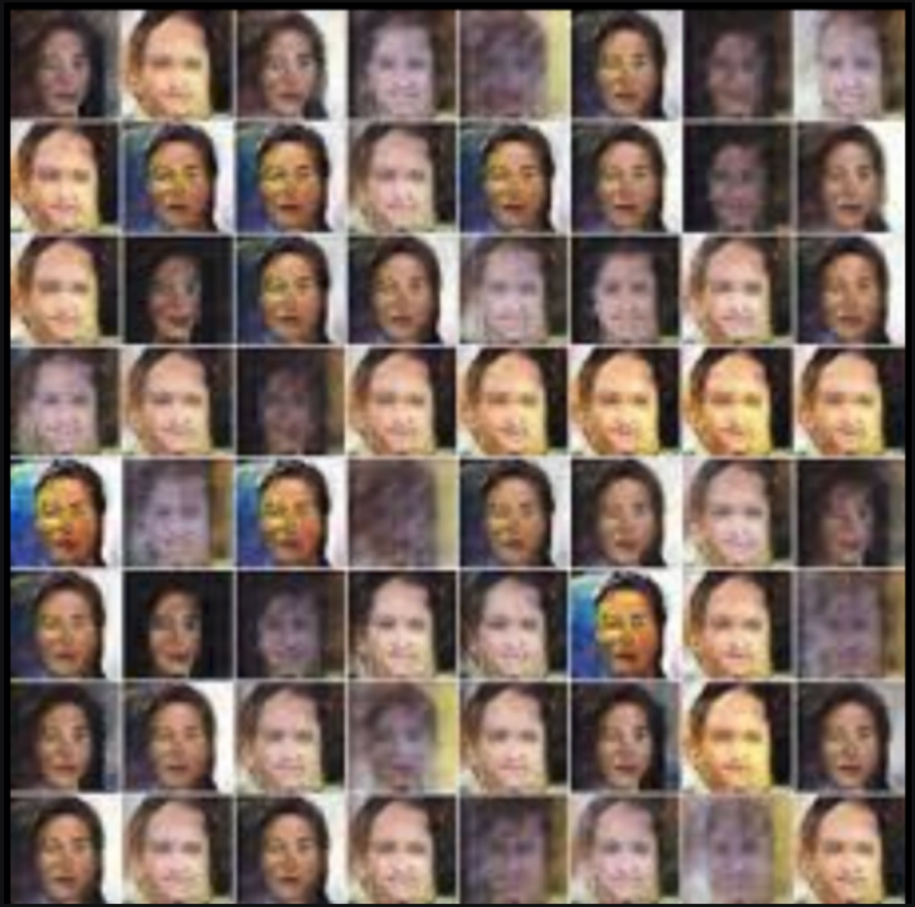
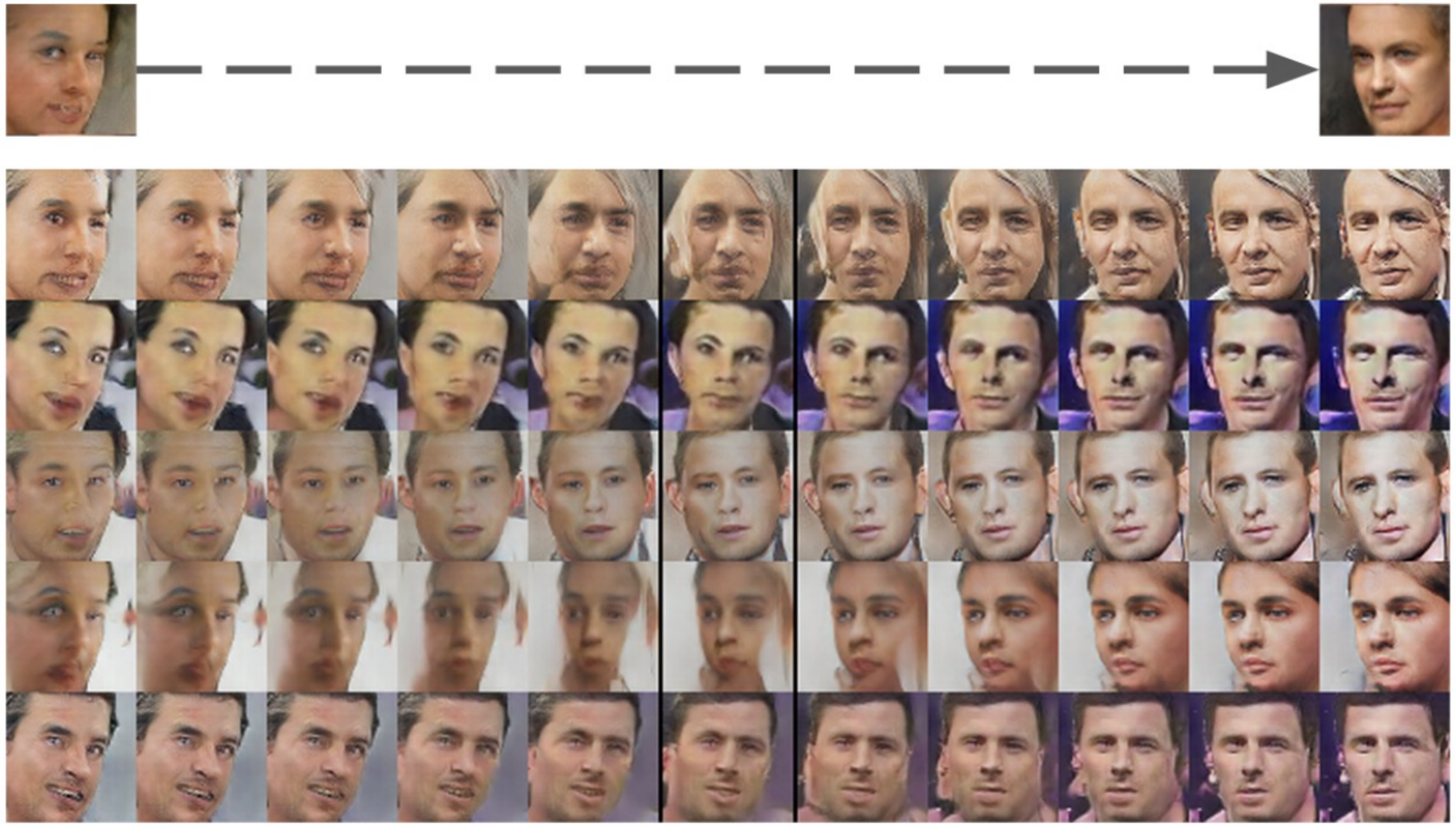


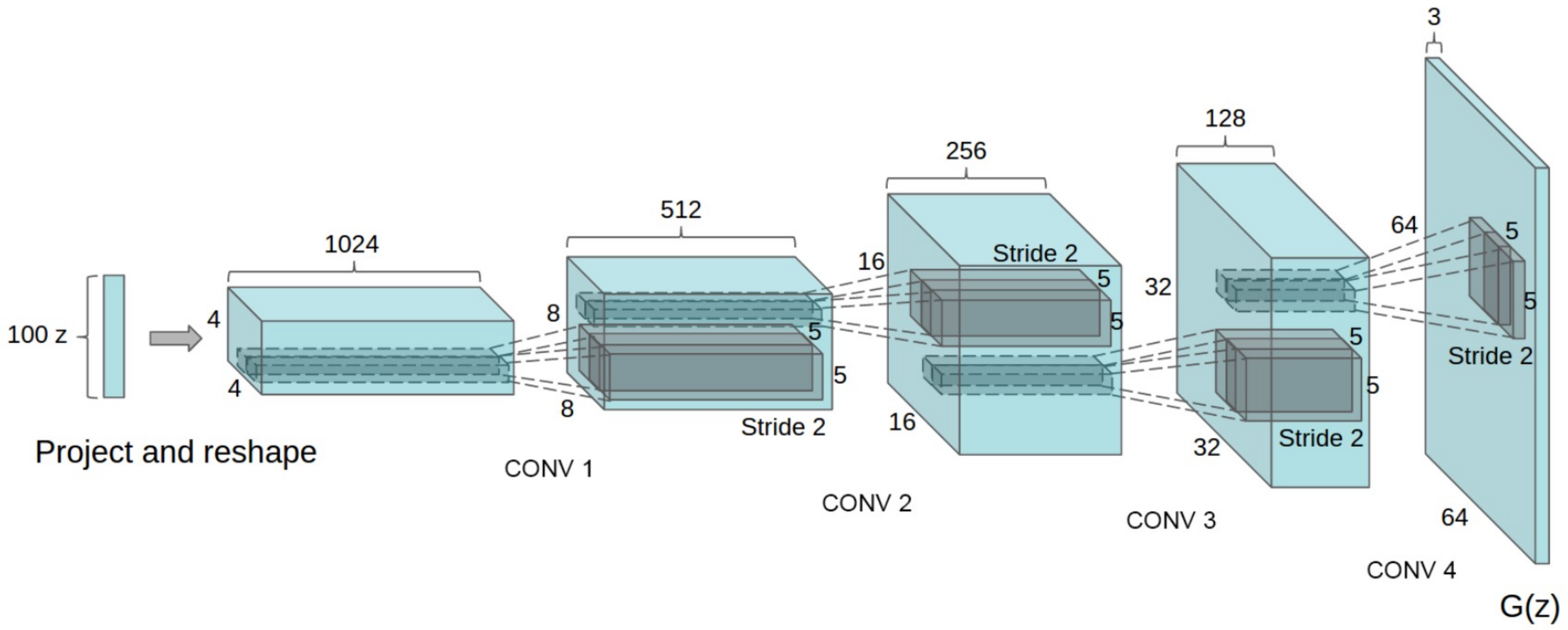
Fig. 5 – This is another example, you can see the same kind of images generated indicating Mode Collapse | Source

Deep Convolutional GANs



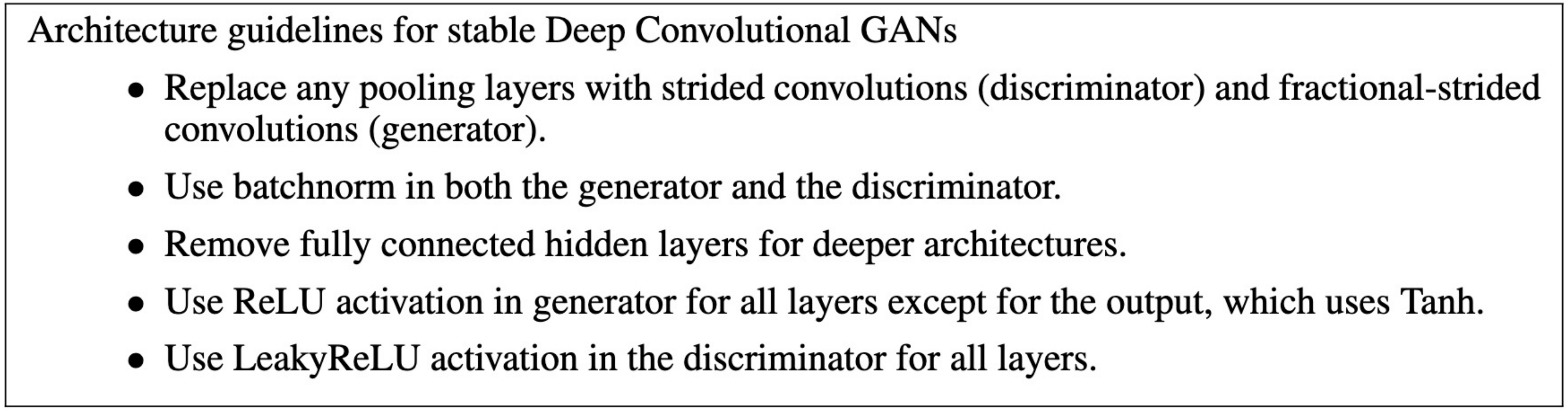
Radford et al. (ICLR 2015)

UpConvolutional Architecture



Radford et al. (ICLR 2015)

Tips & Tricks (DC-GANs)

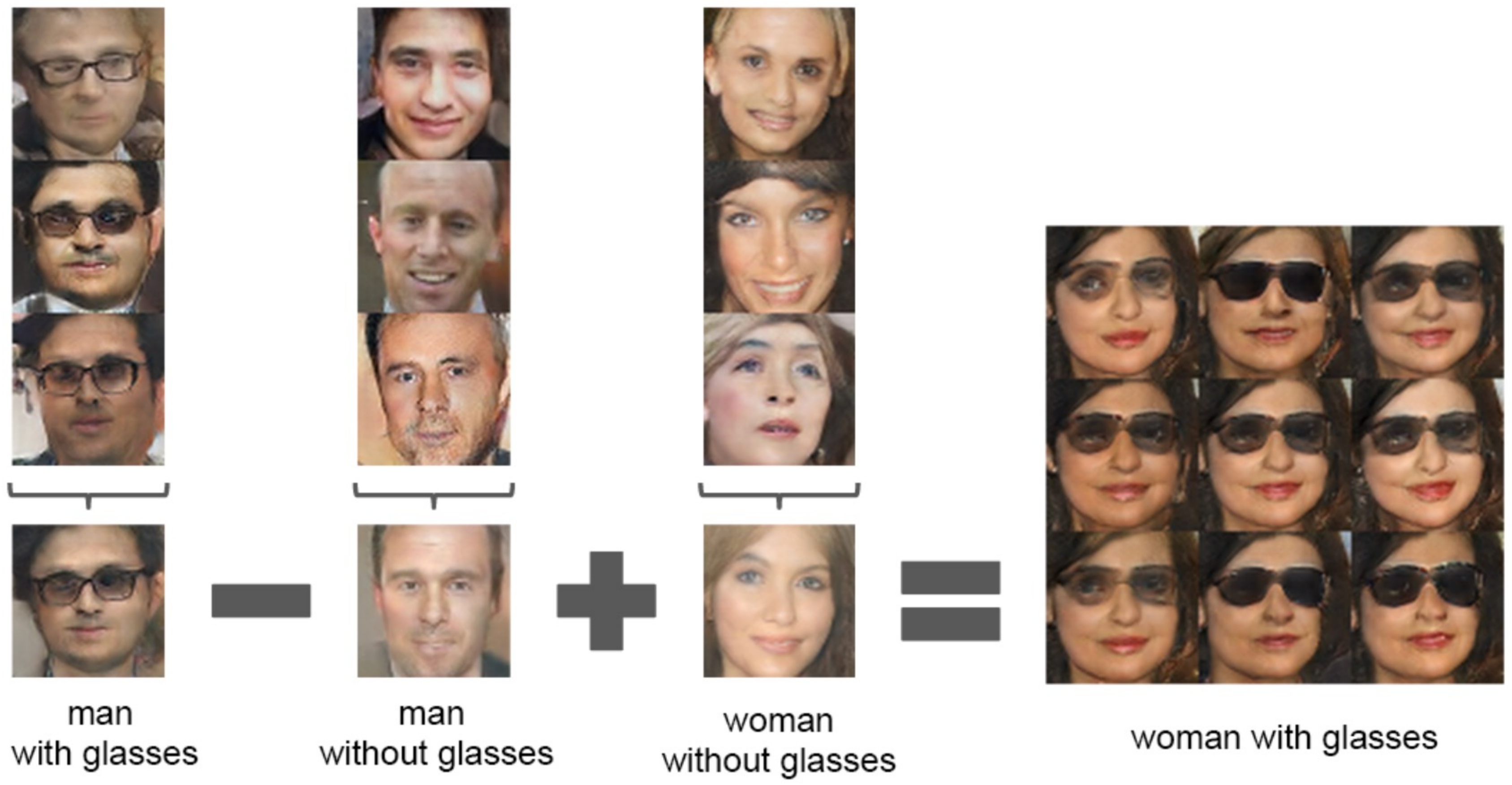


Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

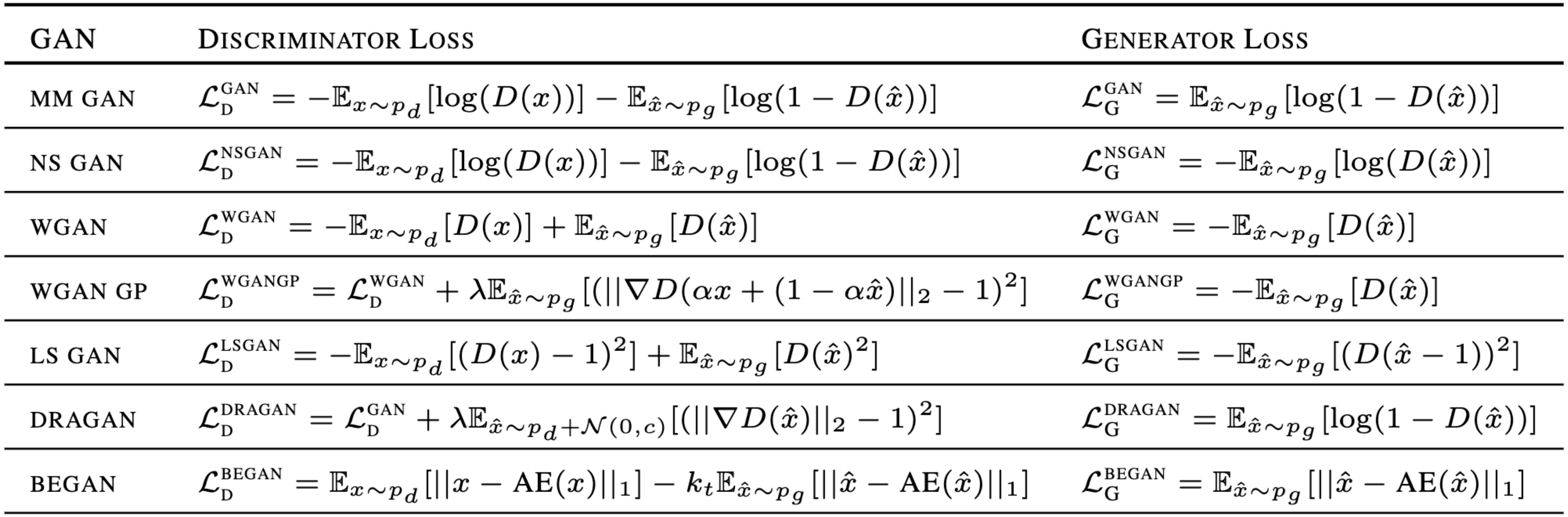
Radford et al. (ICLR 2015)

Latent Vector Arithmetic



Radford et al. (ICLR 2015)

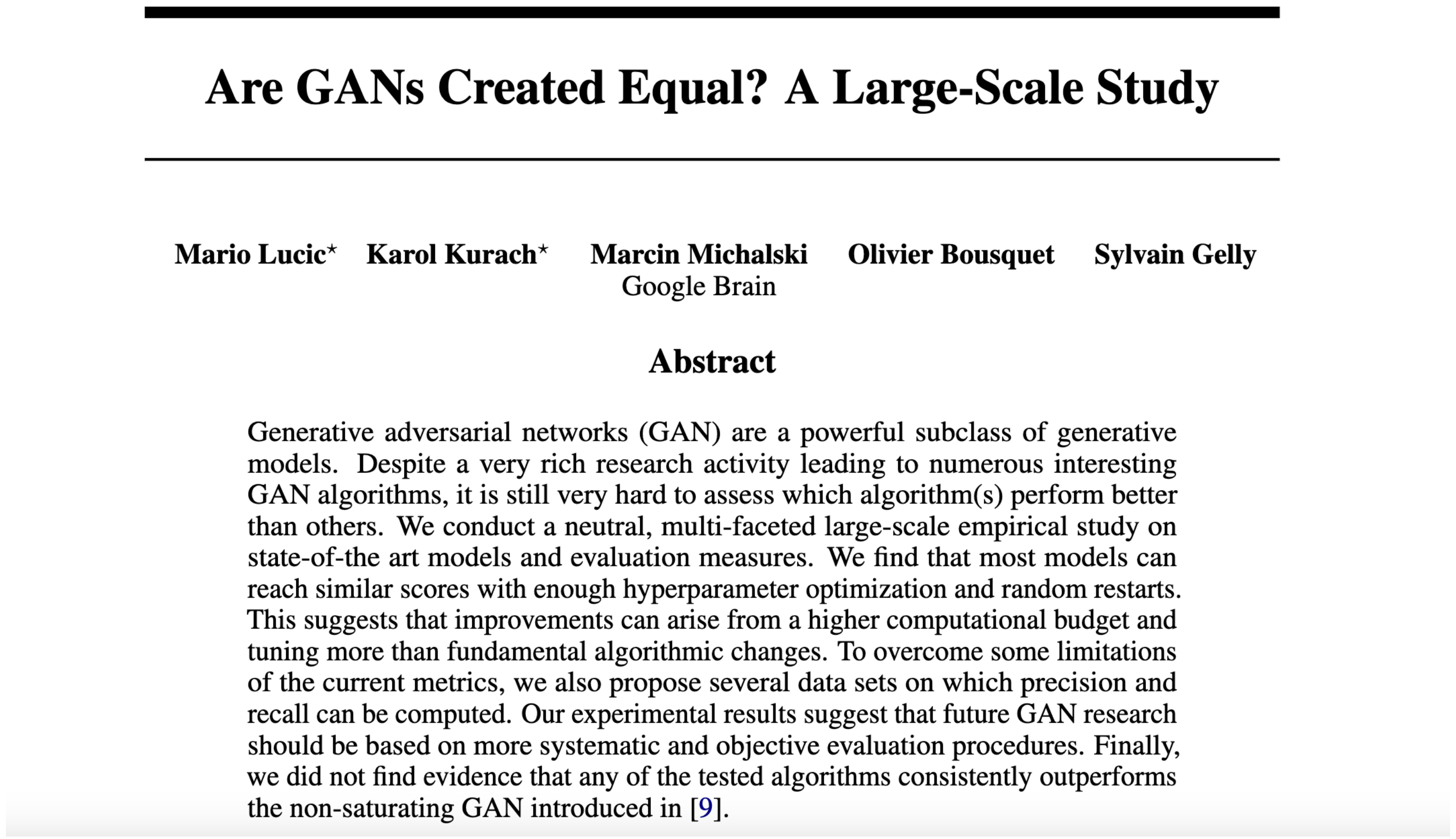
Many Variants of GAN Objectives



GAN	Discriminator Loss	Generator Loss
MM GAN	$\mathcal{L}_{\text{D}}^{\text{GAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{GAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
NS GAN	$\mathcal{L}_{\text{D}}^{\text{NSGAN}} = -\mathbb{E}_{x\sim p_d}[\log(D(x))] - \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$	$\mathcal{L}_{\text{G}}^{\text{NSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[\log(D(\hat{x}))]$
WGAN	$\mathcal{L}_{\text{D}}^{\text{WGAN}} = -\mathbb{E}_{x\sim p_d}[D(x)] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$	$\mathcal{L}_{\text{G}}^{\text{WGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
WGAN GP	$\mathcal{L}_{\text{D}}^{\text{WGANGP}} = \mathcal{L}_{\text{D}}^{\text{WGAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_g}[(\|\|\nabla D(\alpha x + (1-\alpha\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{WGANGP}} = -\mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})]$
LS GAN	$\mathcal{L}_{\text{D}}^{\text{LSGAN}} = -\mathbb{E}_{x\sim p_d}[(D(x)-1)^2] + \mathbb{E}_{\hat{x}\sim p_g}[D(\hat{x})^2]$	$\mathcal{L}_{\text{G}}^{\text{LSGAN}} = -\mathbb{E}_{\hat{x}\sim p_g}[(D(\hat{x}-1))^2]$
DRAGAN	$\mathcal{L}_{\text{D}}^{\text{DRAGAN}} = \mathcal{L}_{\text{D}}^{\text{GAN}} + \lambda\mathbb{E}_{\hat{x}\sim p_d+\mathcal{N}(0,c)}[(\|\|\nabla D(\hat{x})\|\|_2 - 1)^2]$	$\mathcal{L}_{\text{G}}^{\text{DRAGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\log(1-D(\hat{x}))]$
BEGAN	$\mathcal{L}_{\text{D}}^{\text{BEGAN}} = \mathbb{E}_{x\sim p_d}[\|\|x - \text{AE}(x)\|\|_1] - k_t\mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$	$\mathcal{L}_{\text{G}}^{\text{BEGAN}} = \mathbb{E}_{\hat{x}\sim p_g}[\|\|\hat{x} - \text{AE}(\hat{x})\|\|_1]$

figure credit: "Are GANs Created Equal? A Large-Scale Study" Lucic et al. 2018

Some Confusion About What Really Matters



Are GANs Created Equal? A Large-Scale Study

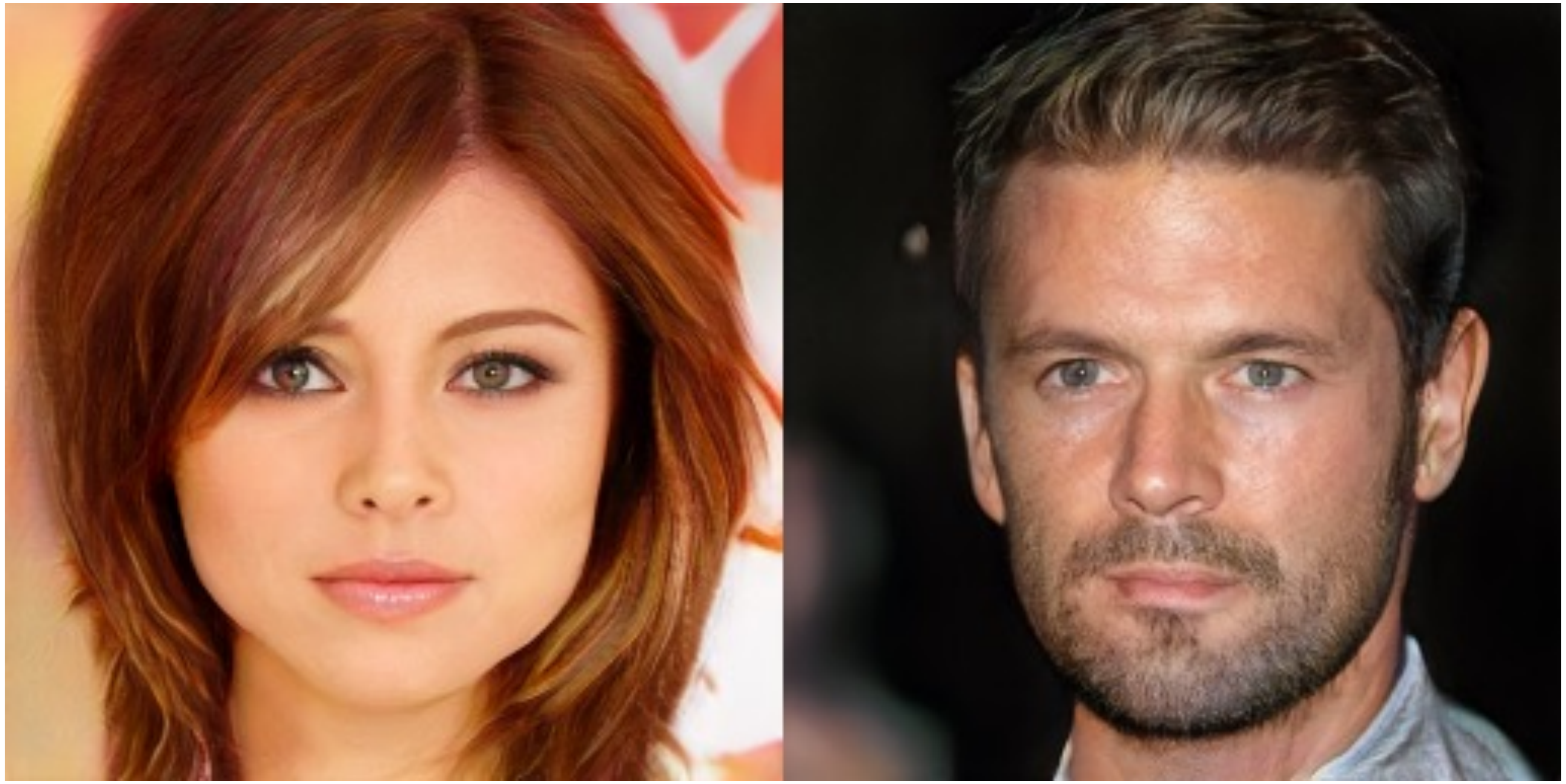
Mario Lucic* **Karol Kurach*** **Marcin Michalski** **Olivier Bousquet** **Sylvain Gelly**
Google Brain

Abstract

Generative adversarial networks (GAN) are a powerful subclass of generative models. Despite a very rich research activity leading to numerous interesting GAN algorithms, it is still very hard to assess which algorithm(s) perform better than others. We conduct a neutral, multi-faceted large-scale empirical study on state-of-the art models and evaluation measures. We find that most models can reach similar scores with enough hyperparameter optimization and random restarts. This suggests that improvements can arise from a higher computational budget and tuning more than fundamental algorithmic changes. To overcome some limitations of the current metrics, we also propose several data sets on which precision and recall can be computed. Our experimental results suggest that future GAN research should be based on more systematic and objective evaluation procedures. Finally, we did not find evidence that any of the tested algorithms consistently outperforms the non-saturating GAN introduced in [9].

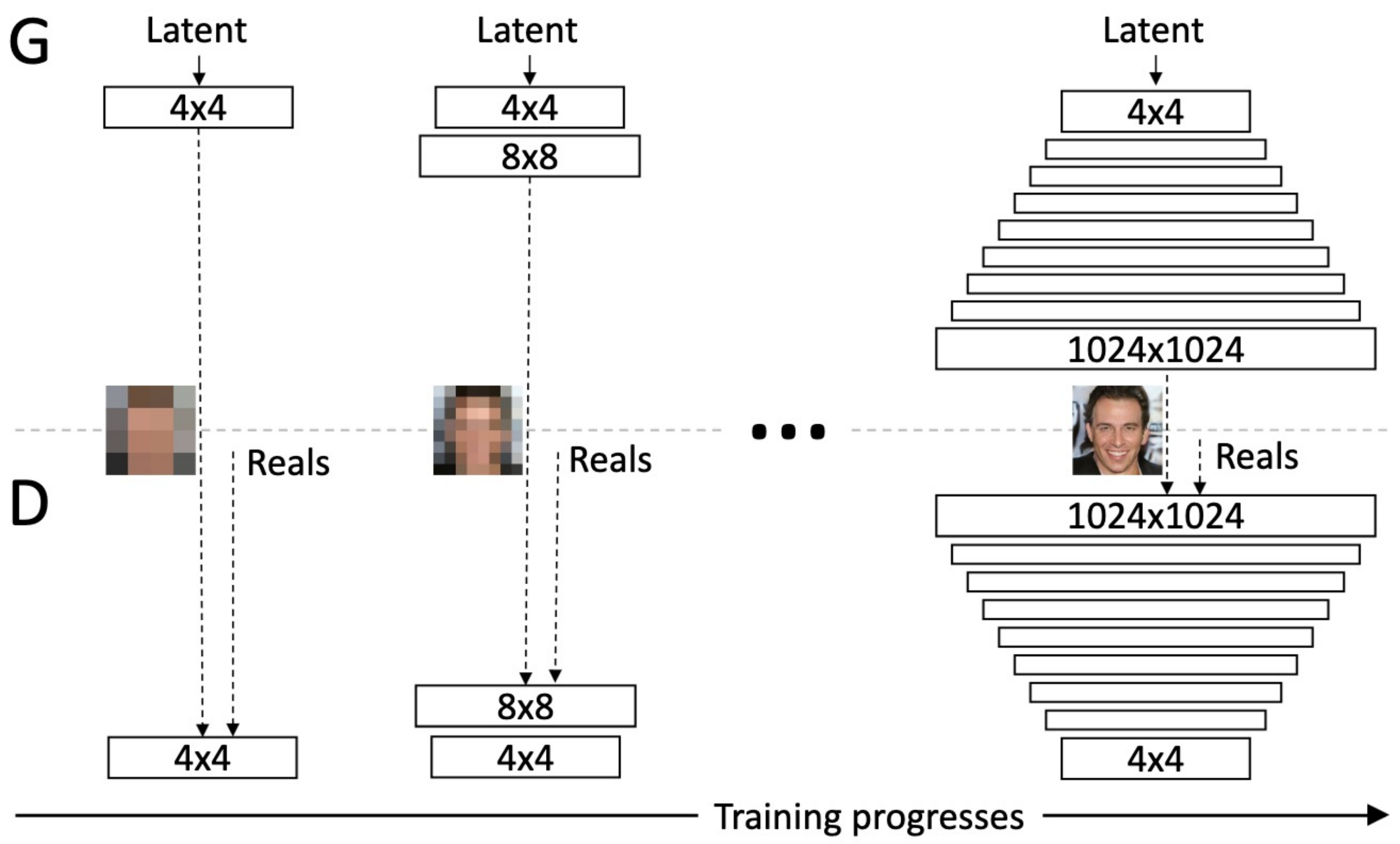
“Are GANs Created Equal? A Large-Scale Study” Lucic et al. 2018

Progressive Growing of GANS



https://www.youtube.com/watch?v=XOxxPcy5Gr4
Progressive Growing (Karras et al., ICLR 2018)

Progressive Growing



Progressive Growing (Karras et al., ICLR 2018)

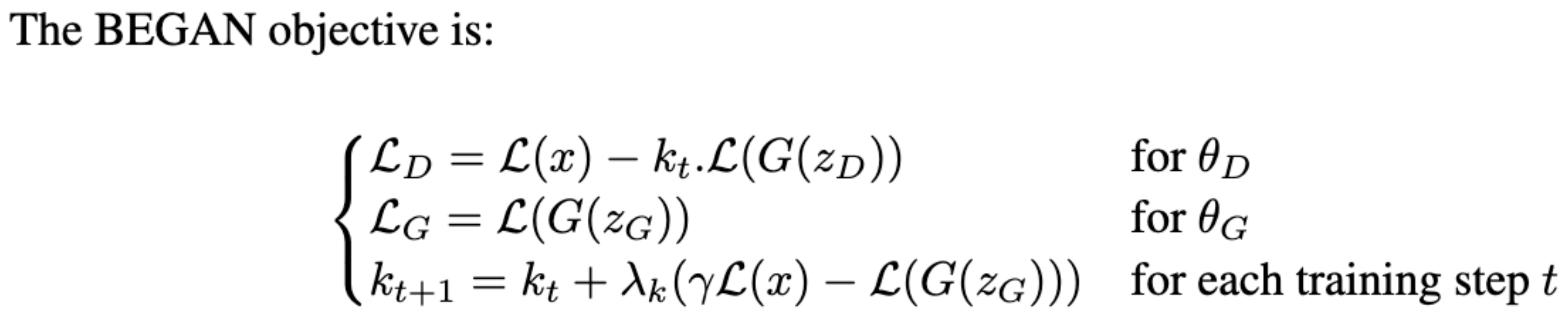
Progressive Growing Details

- Start with 4x4 special resolution for both G and D
- As training advances, incrementally add layers, increasing resolution
- All layers remain trainable throughout the process
- Findings
  - More stable synthesis in high resolutions
  - Speeds up training considerably

Progressive Growing (Karras et al., ICLR 2018)

Maintaining Equilibrium (BEGAN)

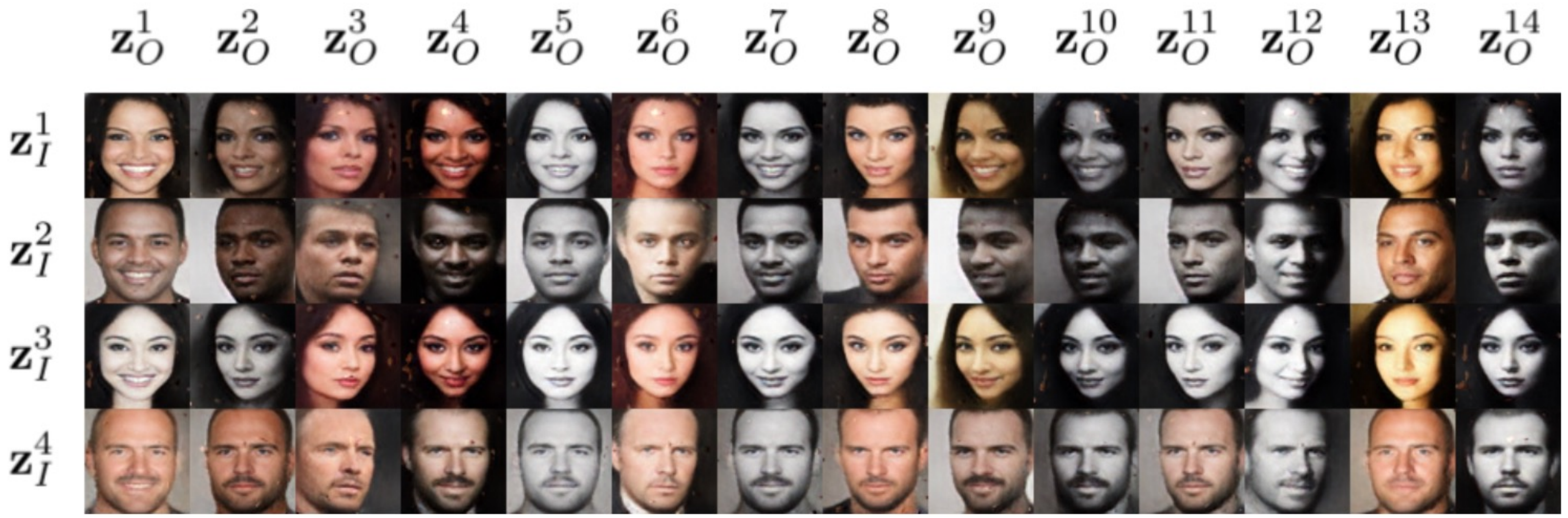
- Make the discriminator an autoencoder
- Idea—low loss for autoencoding real data, but high loss for autoencoding fake examples
- Adversarial game aims to match distribution of losses (vs pixels)
- Tune loss weighting dynamically to maintain parity



The BEGAN objective is:

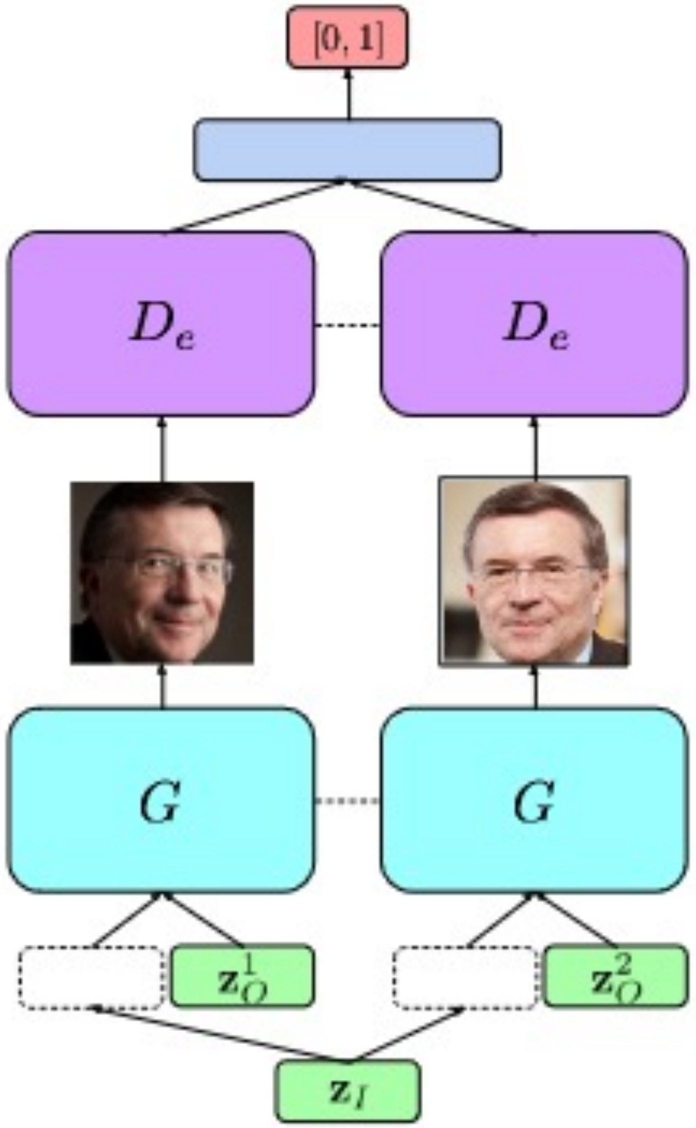
$$\begin{cases} \mathcal{L}_D = \mathcal{L}(x) - k_t.\mathcal{L}(G(z_D)) & \text{for } \theta_D \\ \mathcal{L}_G = \mathcal{L}(G(z_G)) & \text{for } \theta_G \\ k_{t+1} = k_t + \lambda_k(\gamma\mathcal{L}(x) - \mathcal{L}(G(z_G))) & \text{for each training step } t \end{cases}$$

Semantically Decomposing Latents (SD–GAN)



[Semantically Decomposing the Latent Spaces of Generative Adversarial Networks (Donahue, Lipton et al. ICLR 2017)]

The Key Idea

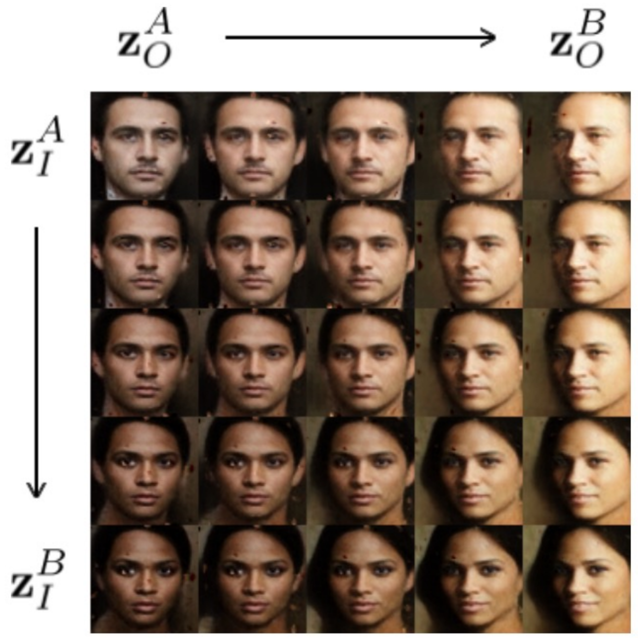


SD-GAN Pseudo-Code

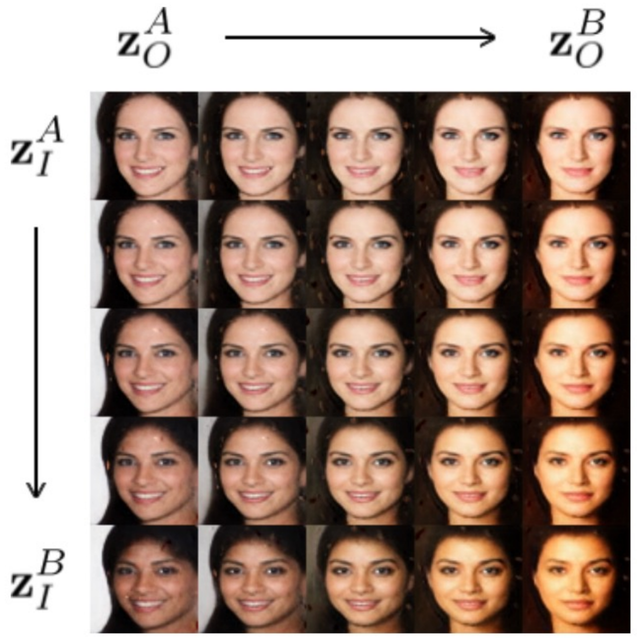
Algorithm 1 Semantically Decomposed GAN Training

```
 1: for n in 1:NumberOfIterations do
 2:     for m in 1:MinibatchSize do
 3:         Sample one identity vector $\mathbf{z}_I \sim \text{Uniform}([-1,1]^{d_I})$.
 4:         Sample two observation vectors $\mathbf{z}_O^1, \mathbf{z}_O^2 \sim \text{Uniform}([-1,1]^{d_O})$.
 5:         $\mathbf{z}^1 \leftarrow [\mathbf{z}_I; \mathbf{z}_O^1]$, $\mathbf{z}^2 \leftarrow [\mathbf{z}_I; \mathbf{z}_O^2]$.
 6:         Generate pair of images $G(\mathbf{z}^1), G(\mathbf{z}^2)$, adding them to the minibatch with label $0$ (fake).
 7:     for m in 1:MinibatchSize do
 8:         Sample one identity $i \in \mathcal{I}$ uniformly at random from the real data set.
 9:         Sample two images of $i$ without replacement $\mathbf{x}_1, \mathbf{x}_2 \sim P_R(\mathbf{x}|I=i)$.
10:         Add the pair to the minibatch, assigning label 1 (real).
11:     Update discriminator weights by $\theta_D \leftarrow \theta_D + \nabla_{\theta_D} V(G,D)$ using its stochastic gradient.
12:     Sample another minibatch of identity-matched latent vectors $\mathbf{z}^1, \mathbf{z}^2$.
13:     Update generator weights by stochastic gradient descent $\theta_G \leftarrow \theta_G - \nabla_{\theta_G} V(G,D)$.
```

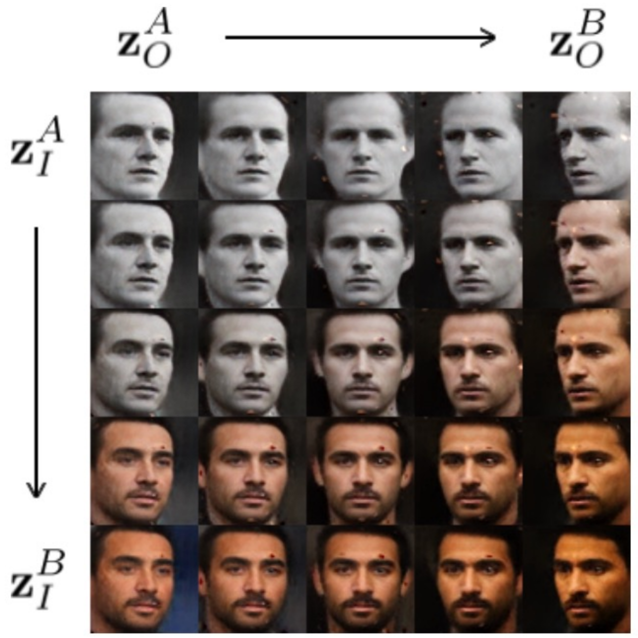
Examples



$\mathbf{z}_O^A$
$\mathbf{z}_O^B$
$\mathbf{z}_I^A$
$\mathbf{z}_I^B$



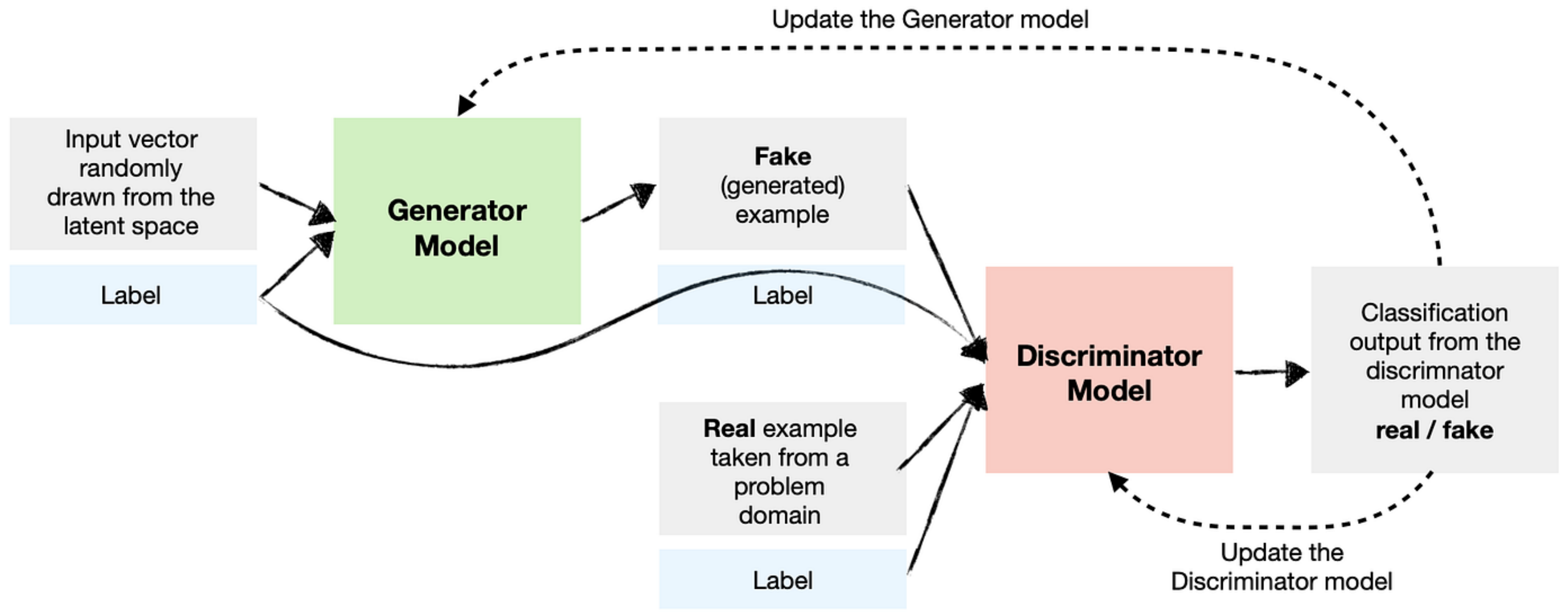
$\mathbf{z}_O^A$
$\mathbf{z}_O^B$
$\mathbf{z}_I^A$
$\mathbf{z}_I^B$



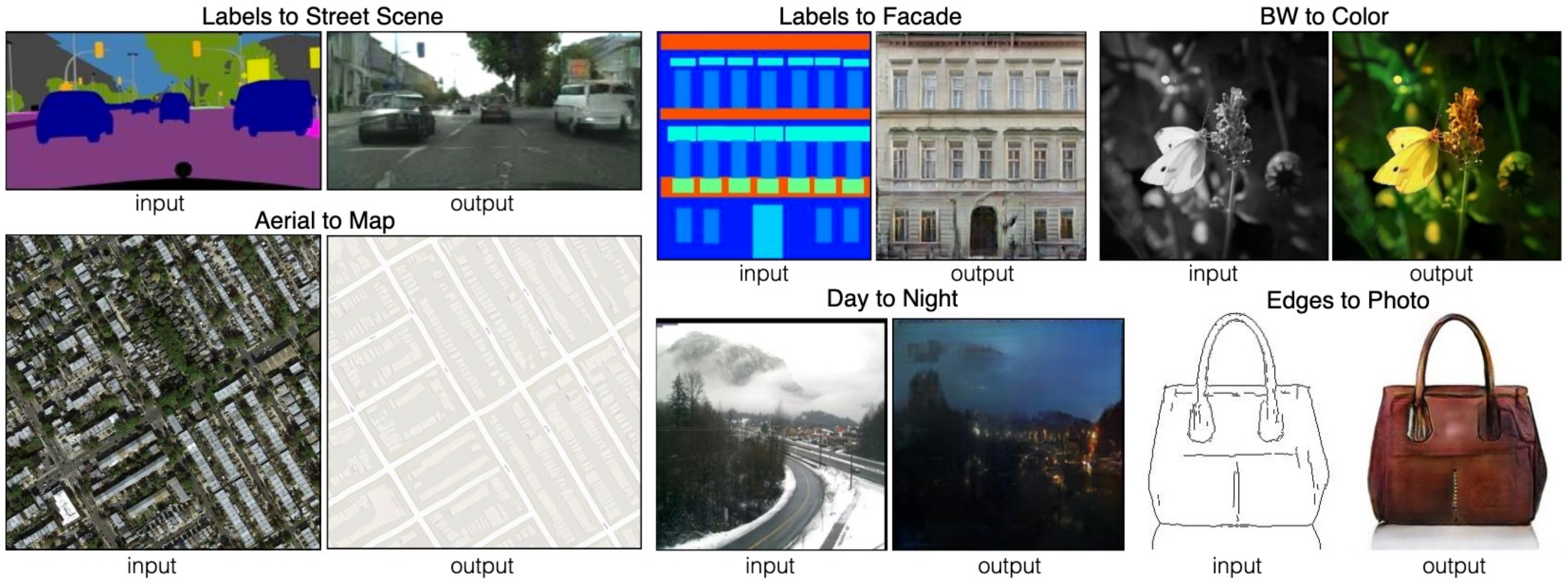
$\mathbf{z}_O^A$
$\mathbf{z}_O^B$
$\mathbf{z}_I^A$
$\mathbf{z}_I^B$

Conditional Generation

- Input Generator gets context *x, noise z*, produces output *y* = G(*x, z*)
- Discriminator tries to distinguish fake pairs $\{(G(x_i, z), \hat{y}_i)\}$ from real input, target pairs $\{(x_i, y_i)\}$

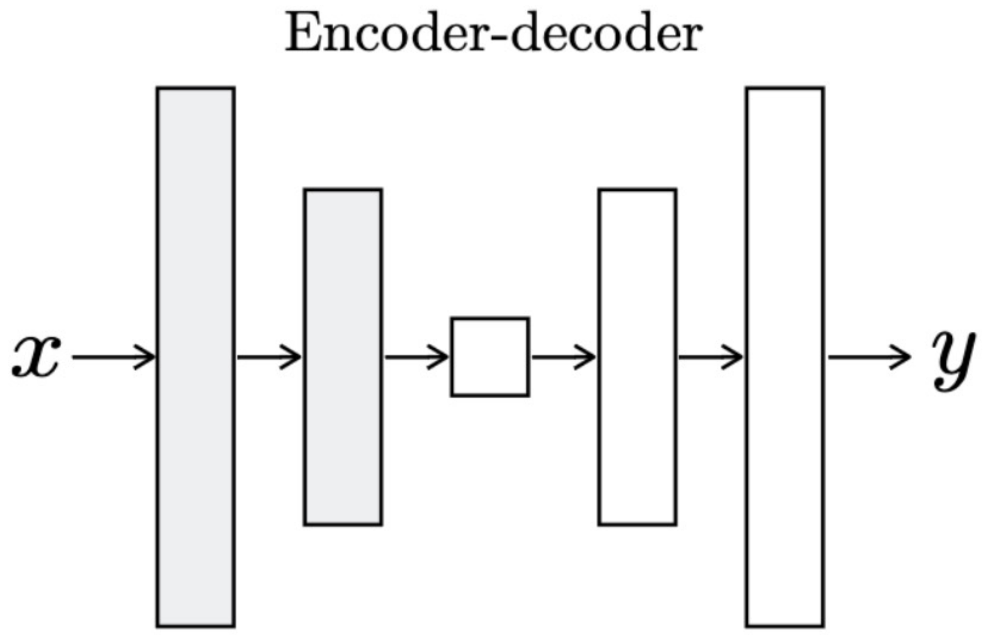

Update the Generator model
Input vector randomly drawn from the latent space
Label
Generator Model
Fake (generated) example
Label
Discriminator Model
Classification output from the discrimnator model real / fake
Real example taken from a problem domain
Label
Update the Discriminator model

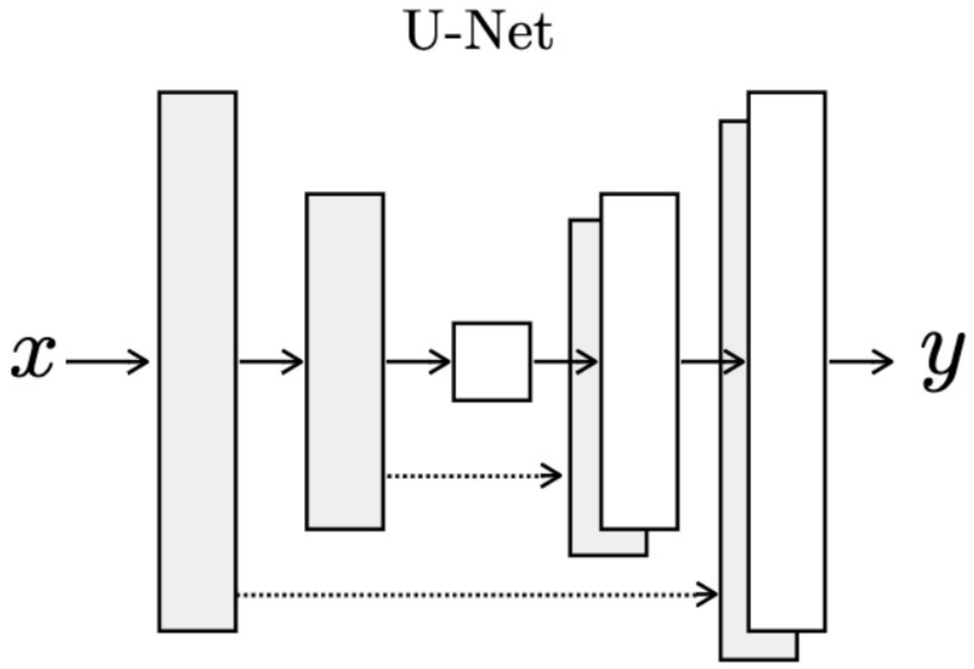
Pix2Pix



Pix2Pix Isola et al. 2017

The U-Net Architecture



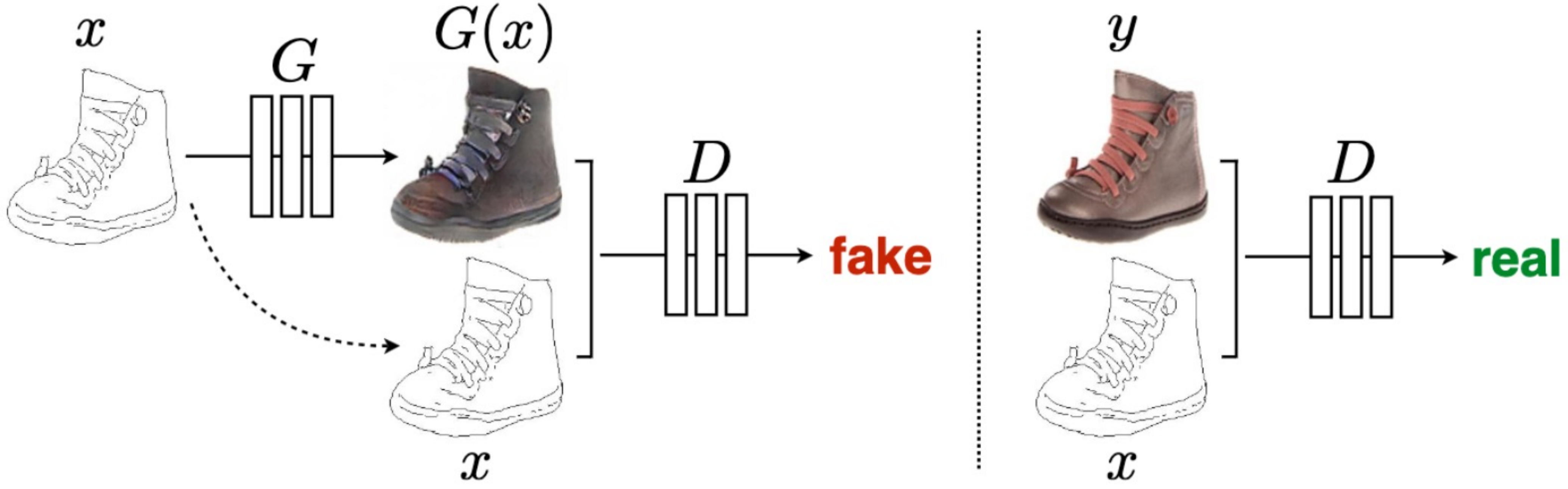


Pix2Pix Learning Setup Adversarial Learning

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G).$$

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned}$$

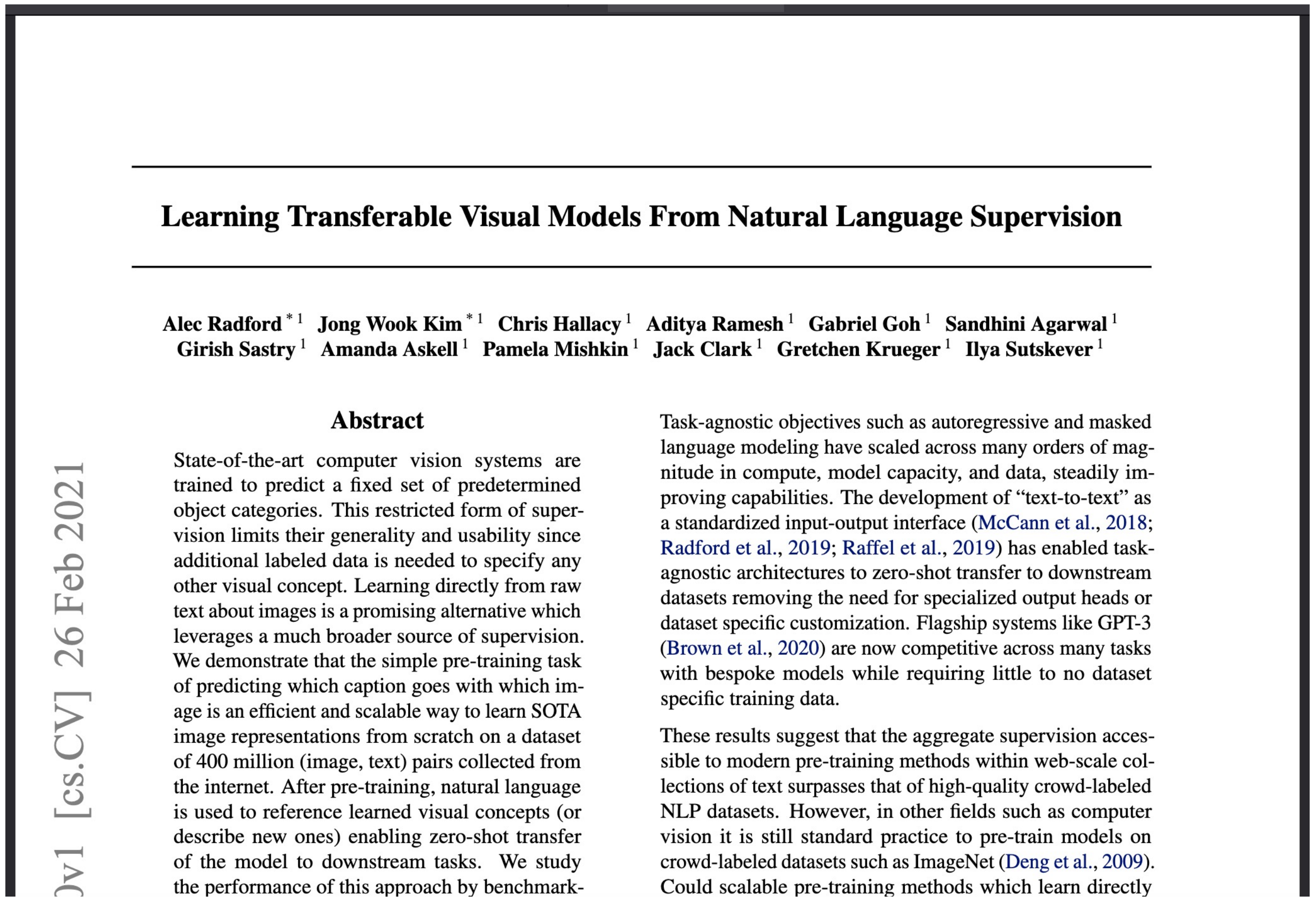
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1].$$



Measuring Sample Quality

- **Inception Score:** feed generated images through Inception v3 model, measure entropy of outputs (lower better), diversity of labels. But what if training data for generative model looks nothing like ImageNet?
- **Frechet Inception Distance (FID):** standard score, measures (mean & variance) of nodes in deepest representation layer of Inception v3
- **Discriminator-based metrics:** train classifier to distinguish real from fake, how easily can it do it (on frozen generator)
- **Human-in-the-loop Evaluation:** human assessments of quality are standard in many papers on image and music generation. Useful for assessing aesthetic properties but may not capture whether the model “learned the distribution”

Contrastive Language-Image Pre-training



Learning Transferable Visual Models From Natural Language Supervision

Alec Radford * 1 Jong Wook Kim * 1 Chris Hallacy 1 Aditya Ramesh 1 Gabriel Goh 1 Sandhini Agarwal 1 Girish Sastry 1 Amanda Askell 1 Pamela Mishkin 1 Jack Clark 1 Gretchen Krueger 1 Ilya Sutskever 1

0v1 [cs.CV] 26 Feb 2021

Abstract

State-of-the-art computer vision systems are trained to predict a fixed set of predetermined object categories. This restricted form of supervision limits their generality and usability since additional labeled data is needed to specify any other visual concept. Learning directly from raw text about images is a promising alternative which leverages a much broader source of supervision. We demonstrate that the simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet. After pre-training, natural language is used to reference learned visual concepts (or describe new ones) enabling zero-shot transfer of the model to downstream tasks. We study the performance of this approach by benchmark-

Task-agnostic objectives such as autoregressive and masked language modeling have scaled across many orders of magnitude in compute, model capacity, and data, steadily improving capabilities. The development of "text-to-text" as a standardized input-output interface (McCann et al., 2018; Radford et al., 2019; Raffel et al., 2019) has enabled task-agnostic architectures to zero-shot transfer to downstream datasets removing the need for specialized output heads or dataset specific customization. Flagship systems like GPT-3 (Brown et al., 2020) are now competitive across many tasks with bespoke models while requiring little to no dataset specific training data.

These results suggest that the aggregate supervision accessible to modern pre-training methods within web-scale collections of text surpasses that of high-quality crowd-labeled NLP datasets. However, in other fields such as computer vision it is still standard practice to pre-train models on crowd-labeled datasets such as ImageNet (Deng et al., 2009). Could scalable pre-training methods which learn directly

CLIP Fast Facts

- Key idea: learn from natural language supervision (appealing but hasn't worked out in past)
- Source of data: "the internet" 400 million image-caption pairs from "a variety of sources"
- Rather than predicting captions, learn joint **embeddings**
- Model:
  - Two encoders, one for images, one for text
  - L2 normalized outputs
- Training
  - Sample a batch
  - Take all dot-products between examples
  - Calculate softmax + cross entropy along each row and column
  - Goal – predict the right caption for each image, right image for each caption
  - Add the two losses loss_i, loss_t, updated by gradient descent

CLIP Overview

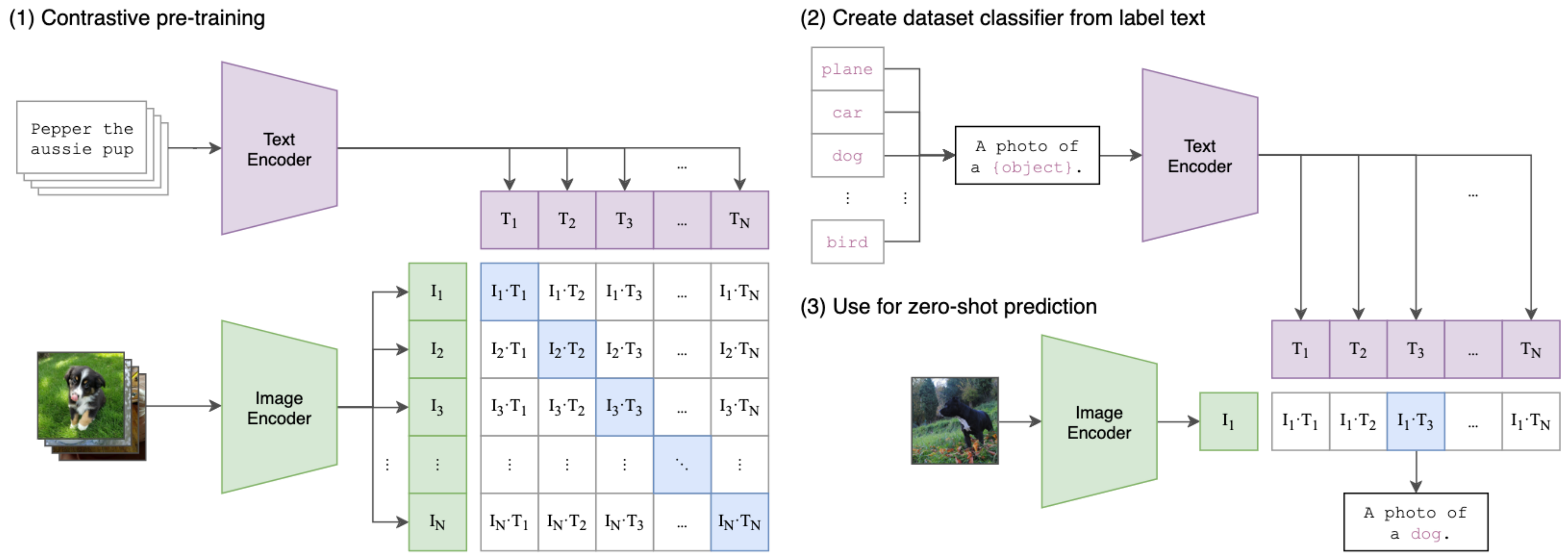
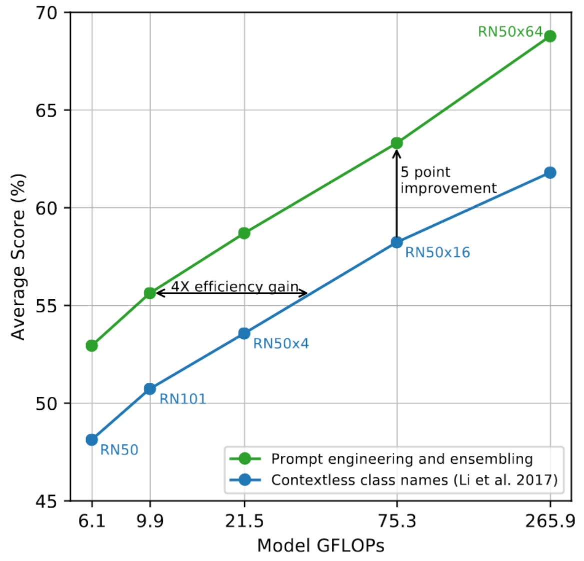
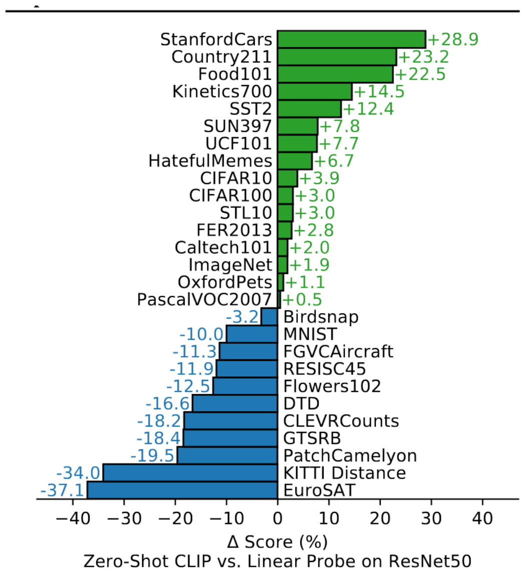


Figure 1. Summary of our approach. While standard image models jointly train an image feature extractor and a linear classifier to predict some label, CLIP jointly trains an image encoder and a text encoder to predict the correct pairings of a batch of (image, text) training examples. At test time the learned text encoder synthesizes a zero-shot linear classifier by embedding the names or descriptions of the target dataset's classes.

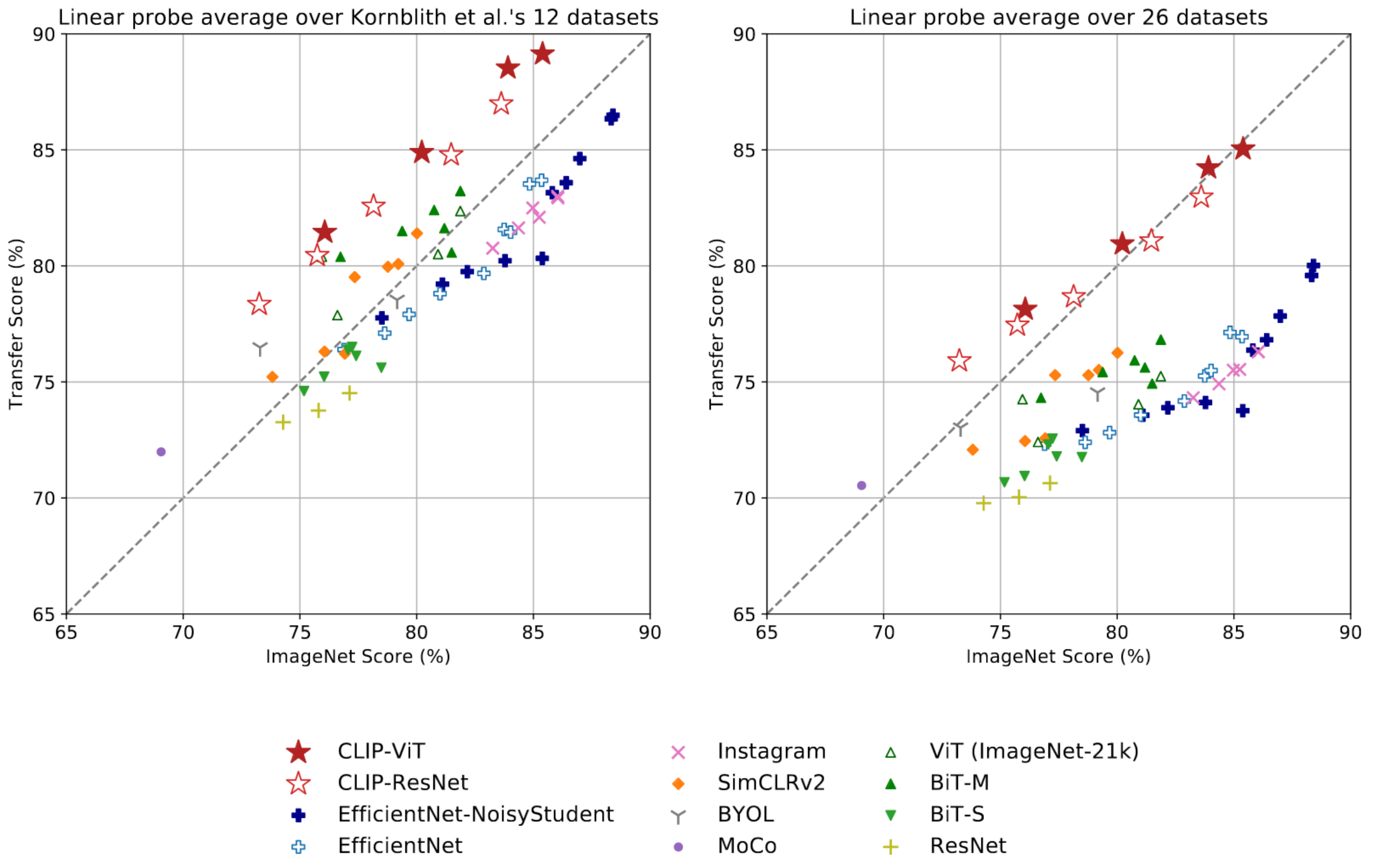
Key Results

- Remarkable zero-shot acc. on new tasks
- Simply encode the label names as caption *“an image of a ____”*
- Prompt engineering on template helps
- Linear probes on CLIP outperform fully supervised pretrained!

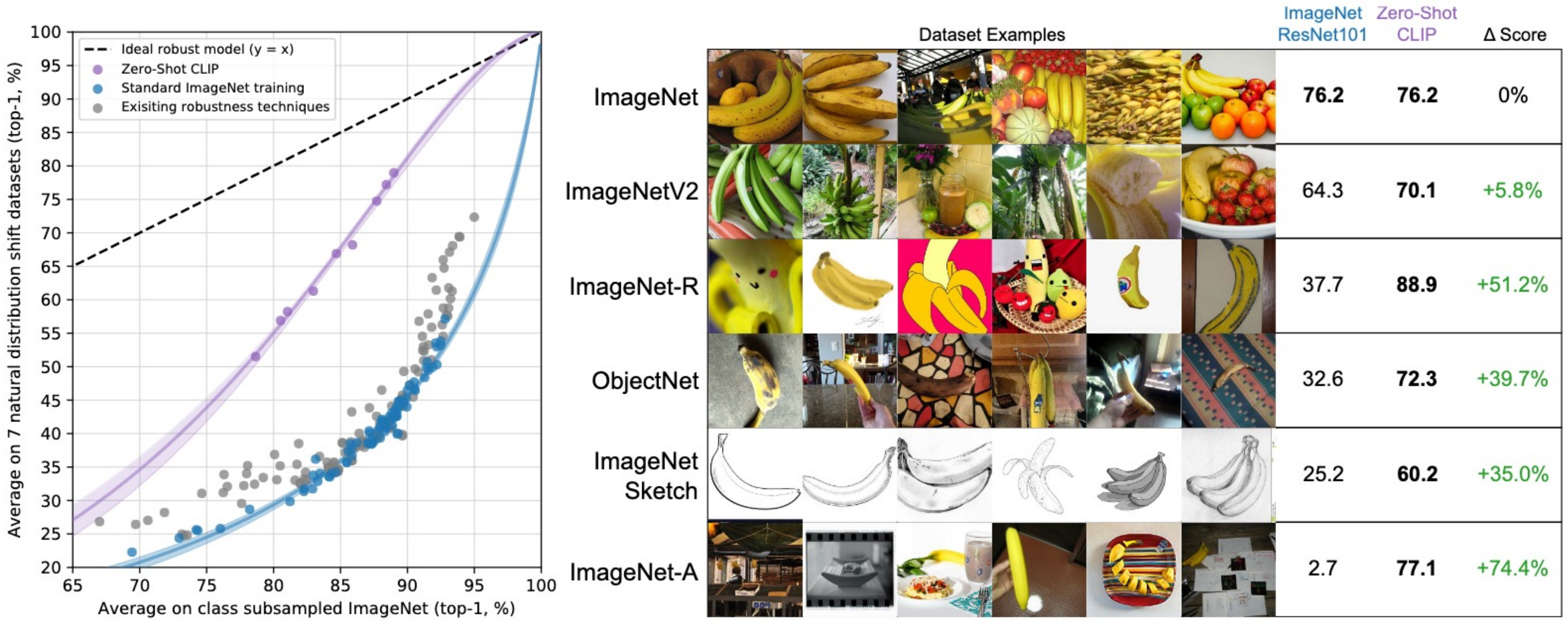




Greater robustness under task shift



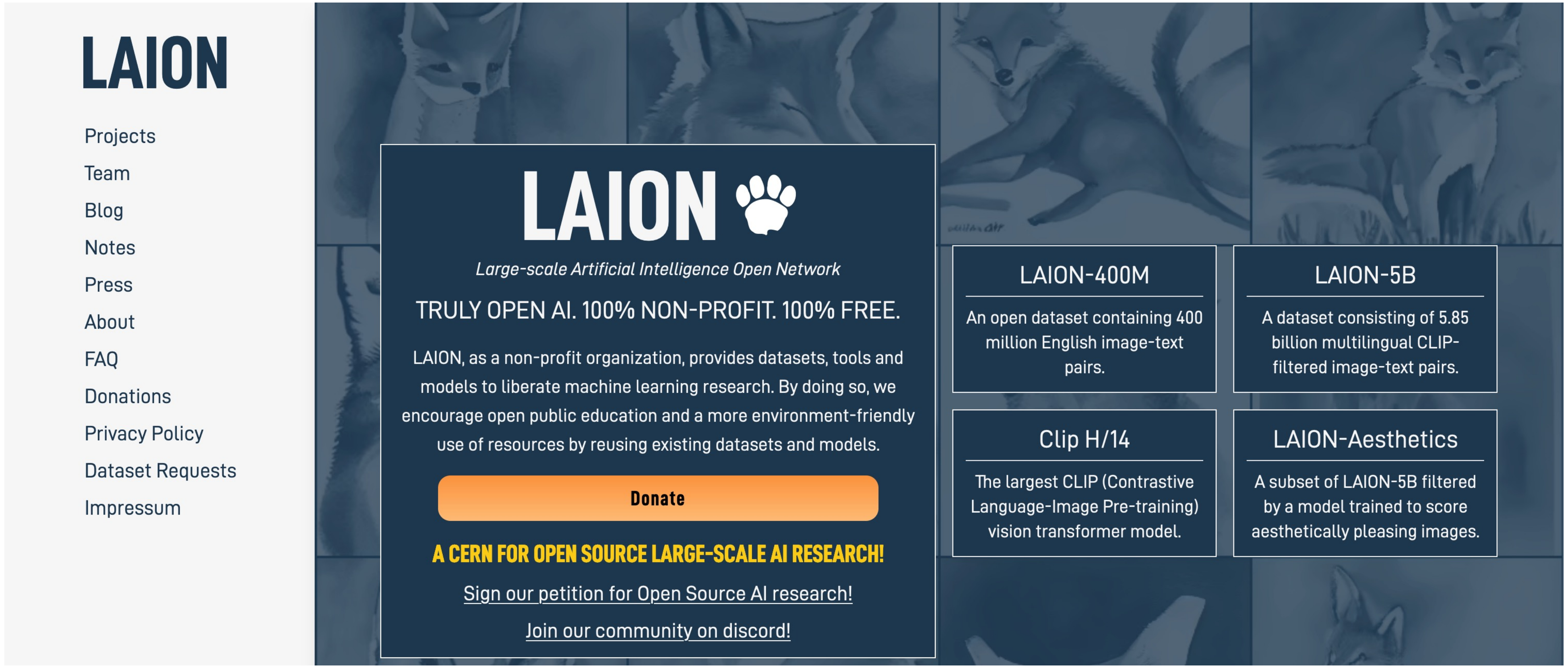
Robustness Under “Natural” Distribution Shifts



	ImageNet ResNet101	Zero-Shot CLIP	Δ Score
ImageNet	**76.2**	**76.2**	0%
ImageNetV2	64.3	**70.1**	+5.8%
ImageNet-R	37.7	**88.9**	+51.2%
ObjectNet	32.6	**72.3**	+39.7%
ImageNet Sketch	25.2	**60.2**	+35.0%
ImageNet-A	2.7	**77.1**	+74.4%

Figure 13. **Zero-shot CLIP is much more robust to distribution shift than standard ImageNet models.** (Left) An ideal robust model (dashed line) performs equally well on the ImageNet distribution and on other natural image distributions. Zero-shot CLIP models shrink this “robustness gap” by up to 75%. Linear fits on logit transformed values are shown with bootstrap estimated 95% confidence intervals. (Right) Visualizing distribution shift for bananas, a class shared across 5 of the 7 natural distribution shift datasets. The performance of the best zero-shot CLIP model, ViT-L/14@336px, is compared with a model that has the same performance on the ImageNet validation set, ResNet-101.

LAION (5B image open dataset)



Large-scale Artificial Intelligence Open Network

TRULY OPEN AI. 100% NON-PROFIT. 100% FREE.

LAION, as a non-profit organization, provides datasets, tools and models to liberate machine learning research. By doing so, we encourage open public education and a more environment-friendly use of resources by reusing existing datasets and models.

A CERN FOR OPEN SOURCE LARGE-SCALE AI RESEARCH!

Sign our petition for Open Source AI research!

Join our community on discord!

LAION-400M

An open dataset containing 400 million English image-text pairs.

LAION-5B

A dataset consisting of 5.85 billion multilingual CLIP-filtered image-text pairs.

Clip H/14

The largest CLIP (Contrastive Language-Image Pre-training) vision transformer model.

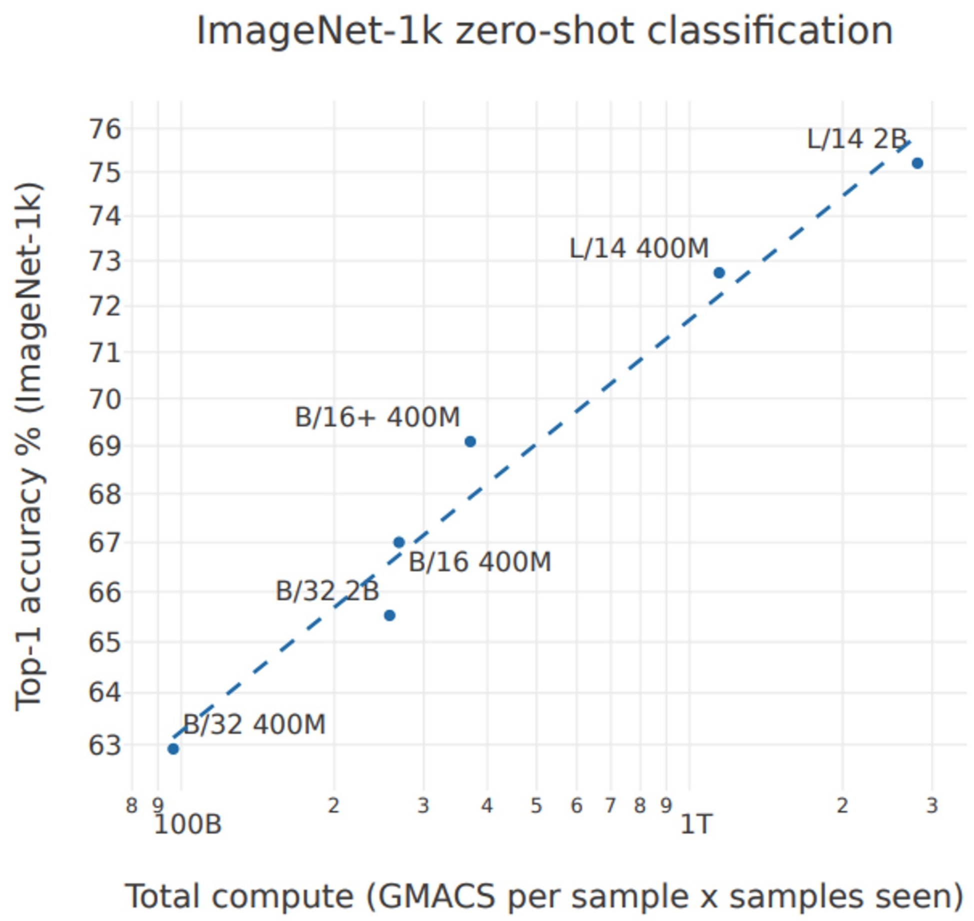
LAION-Aesthetics

A subset of LAION-5B filtered by a model trained to score aesthetically pleasing images.

Scaling Up CLIP Training Helps

- Increase the amount of training data: from 400M to 5B (along with compute)

- Can we filter this raw common-crawl data to get further gains at similar compute



Plot Credits: LAION5B paper

Task: Given a fixed training budget, filter out the best training subset

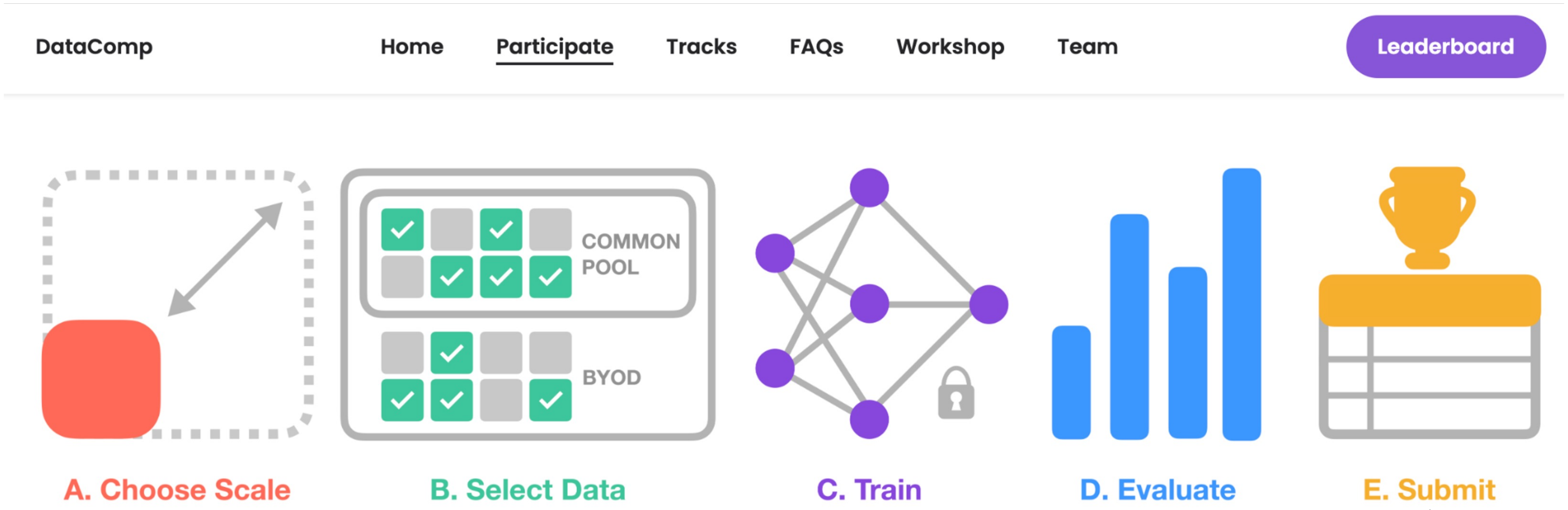


Image Credits: Datacomp

Our Goal: Release a model whose vision embeddings become the de-facto standard across domains

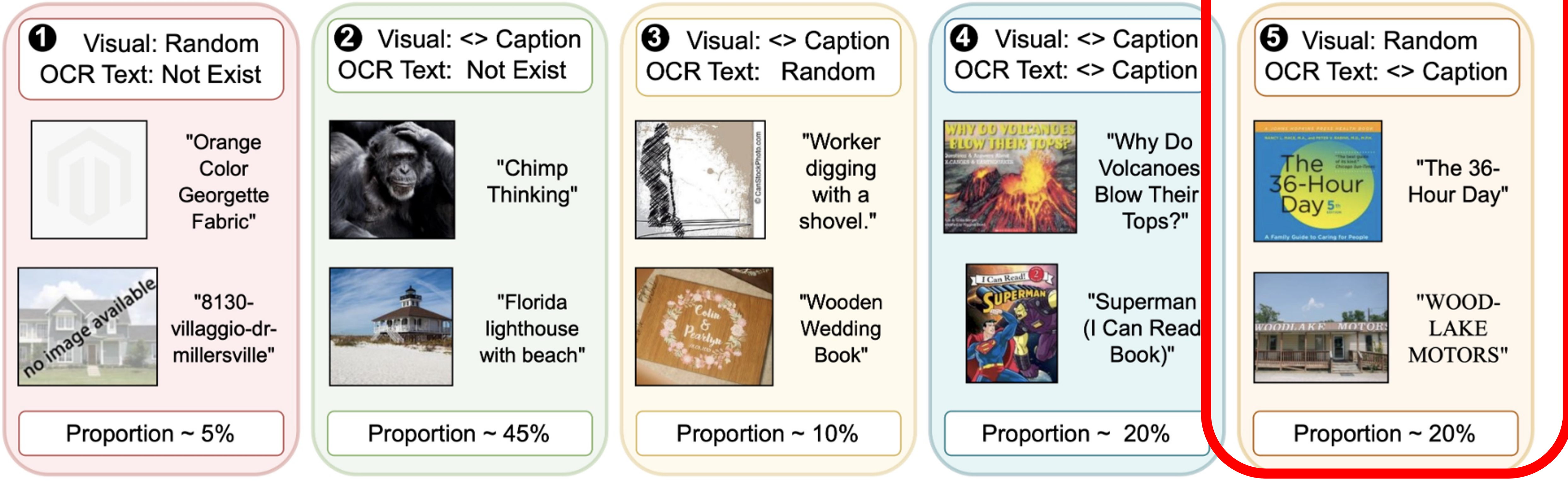
Current Filtering Strategy : CLIP Filtering

CLIP Filtering (previous SOTA)

- Use pre-trained CLIP to calculate similarity between image-caption pairs
- Pretrained CLIP has OCR capabilities, so samples without any visual features are NOT discarded

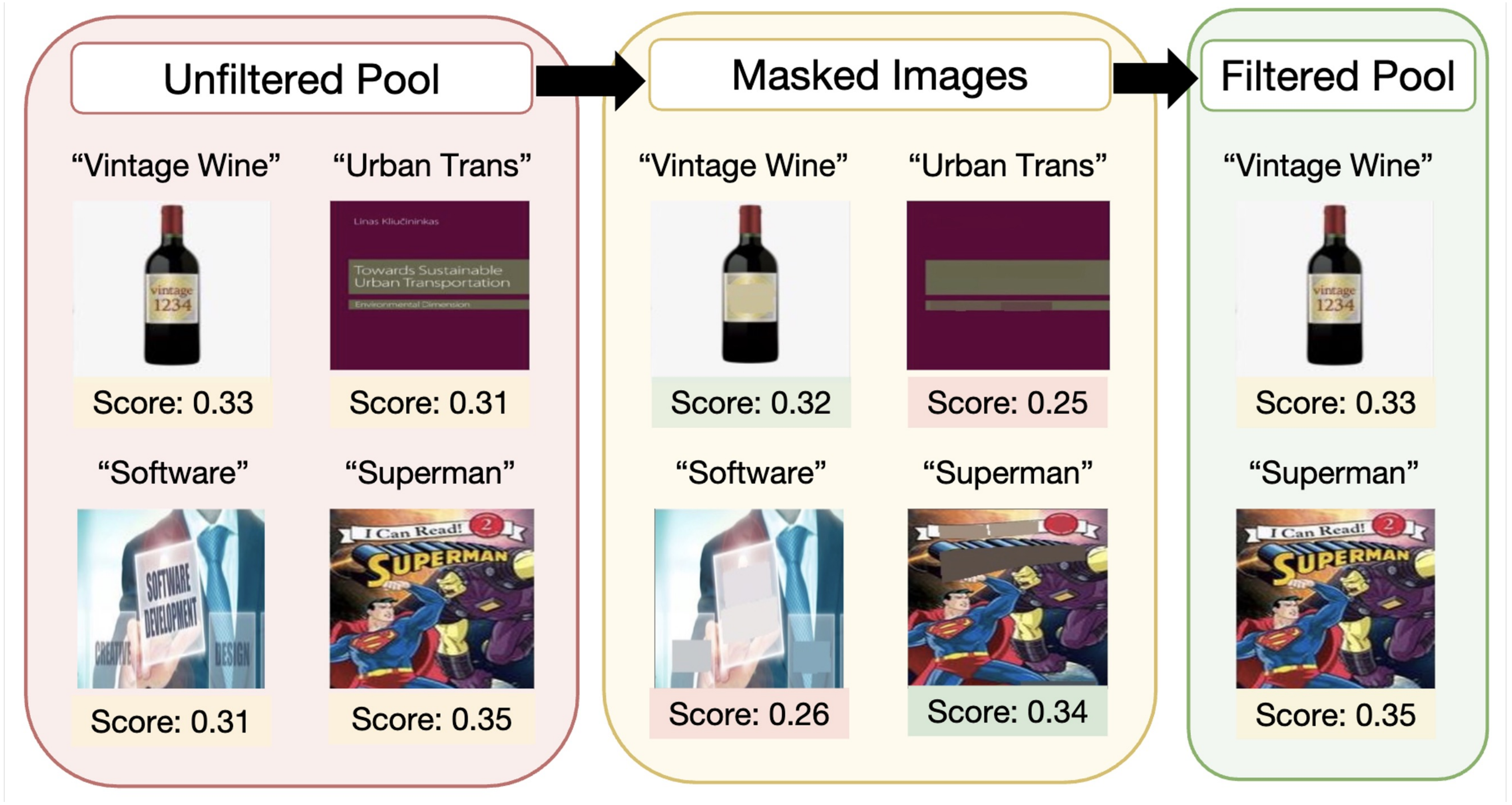
Let’s dig deeper into Common Crawl

- For some image-caption pairs, **caption don’t describe any visual features**
- Model is asked to perform OCR



Method: Text Masking and Re-Scoring

Text Detection → Text Masking → CLIP Score based Filtering



T-MARS (Maini, Goyal et al)

Result: Samples removed by both text-matching and T-MARS

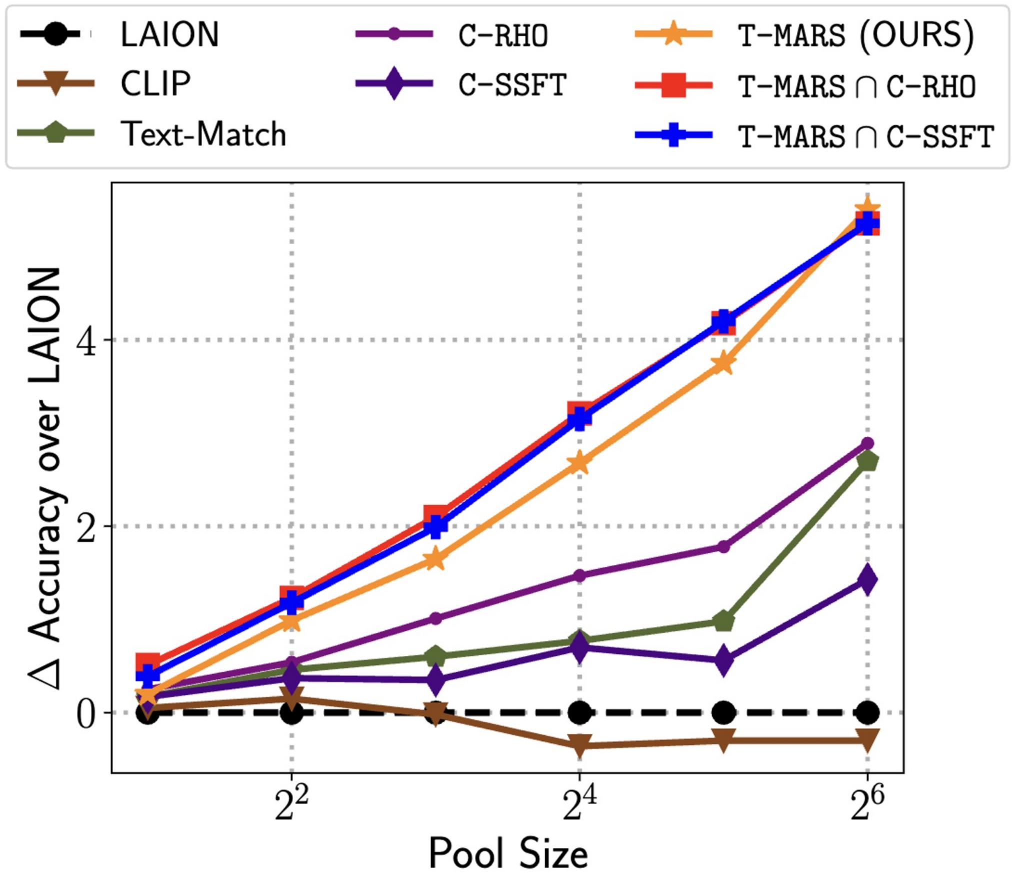
Le second livre de l
a jungle - Kipling

WEB ART DESIGN GOURM
ANDISE CHOCOLAT TENT
ATION PLAISIR 100

Entranement intensif
aux tests d&#039;ap
titude IFSI : Nombre
s, Lettres, Formes,
Dominos, Cartes

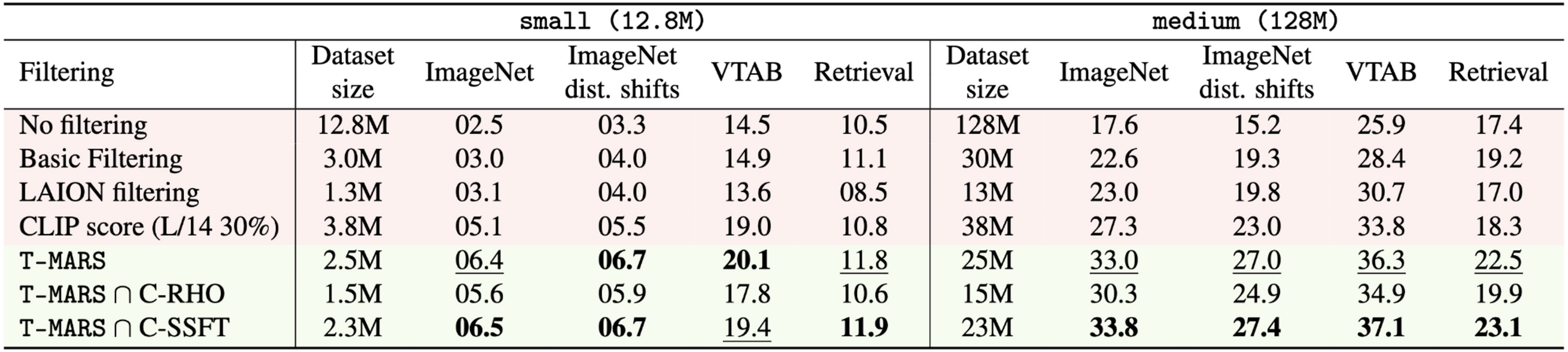
Quilting Internation
al Magazine, 1990 -
QM

Result: Gains over training on LAION increase logarithmically



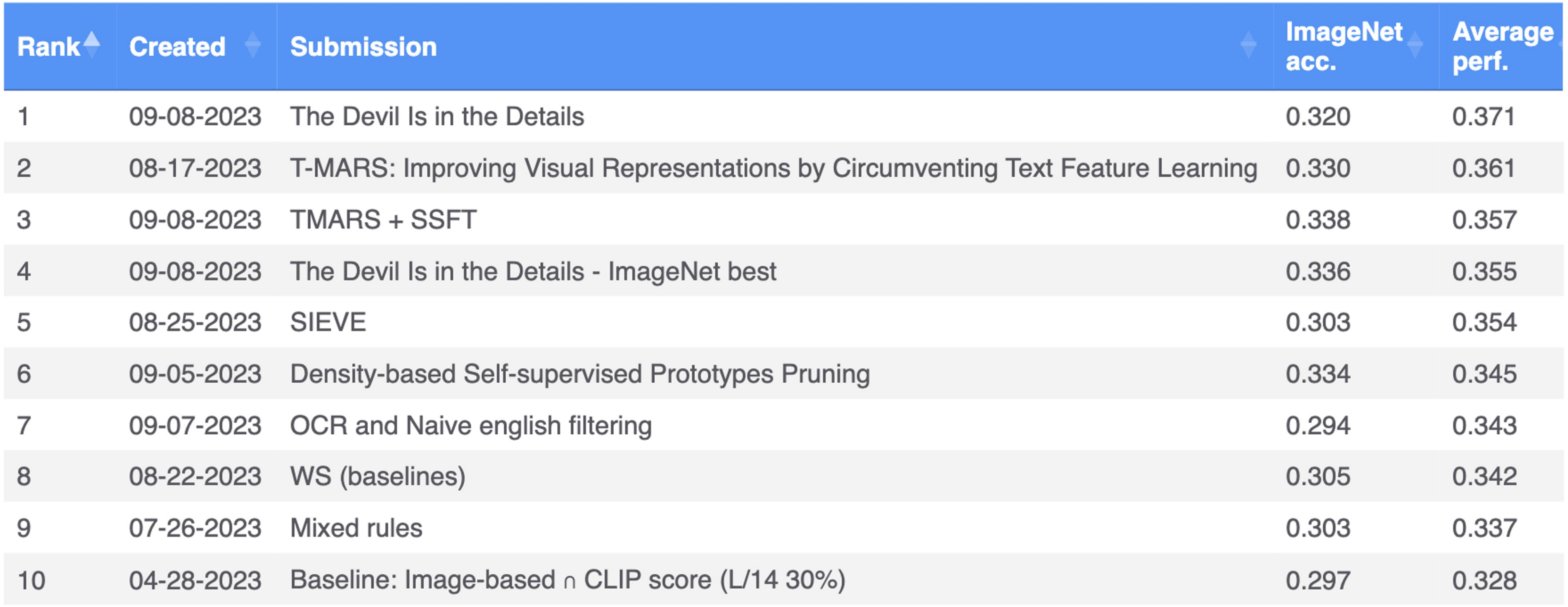
Result: State of Art on DataComp

Table 2: Zero-shot accuracies for various filtering strategies on the `small` and `medium` pools of the DataComp benchmark. ∩ denotes the intersection between two filtering strategies. `T-MARS` outperforms the state-of-art on DataComp by a margin of 5% on the medium scale (ImageNet).



	`small (12.8M)`					`medium (128M)`				
Filtering	Dataset size	ImageNet	ImageNet dist. shifts	VTAB	Retrieval	Dataset size	ImageNet	ImageNet dist. shifts	VTAB	Retrieval
No filtering	12.8M	02.5	03.3	14.5	10.5	128M	17.6	15.2	25.9	17.4
Basic Filtering	3.0M	03.0	04.0	14.9	11.1	30M	22.6	19.3	28.4	19.2
LAION filtering	1.3M	03.1	04.0	13.6	08.5	13M	23.0	19.8	30.7	17.0
CLIP score (L/14 30%)	3.8M	05.1	05.5	19.0	10.8	38M	27.3	23.0	33.8	18.3
`T-MARS`	2.5M	06.4	**06.7**	**20.1**	11.8	25M	33.0	27.0	36.3	22.5
`T-MARS` ∩ C-RHO	1.5M	05.6	05.9	17.8	10.6	15M	30.3	24.9	34.9	19.9
`T-MARS` ∩ C-SSFT	2.3M	**06.5**	**06.7**	19.4	**11.9**	23M	**33.8**	**27.4**	**37.1**	**23.1**

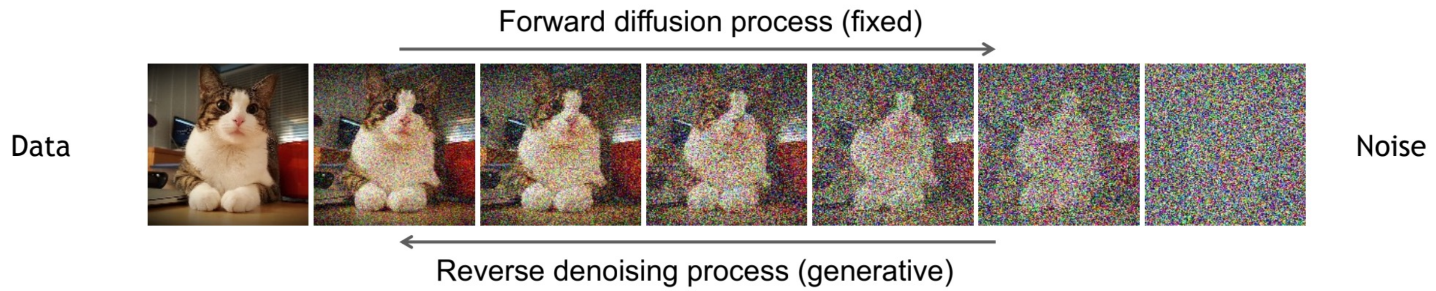
Datacomp



Rank	Created	Submission	ImageNet acc.	Average perf.
1	09-08-2023	The Devil Is in the Details	0.320	0.371
2	08-17-2023	T-MARS: Improving Visual Representations by Circumventing Text Feature Learning	0.330	0.361
3	09-08-2023	TMARS + SSFT	0.338	0.357
4	09-08-2023	The Devil Is in the Details - ImageNet best	0.336	0.355
5	08-25-2023	SIEVE	0.303	0.354
6	09-05-2023	Density-based Self-supervised Prototypes Pruning	0.334	0.345
7	09-07-2023	OCR and Naive english filtering	0.294	0.343
8	08-22-2023	WS (baselines)	0.305	0.342
9	07-26-2023	Mixed rules	0.303	0.337
10	04-28-2023	Baseline: Image-based ∩ CLIP score (L/14 30%)	0.297	0.328

Diffusion Models

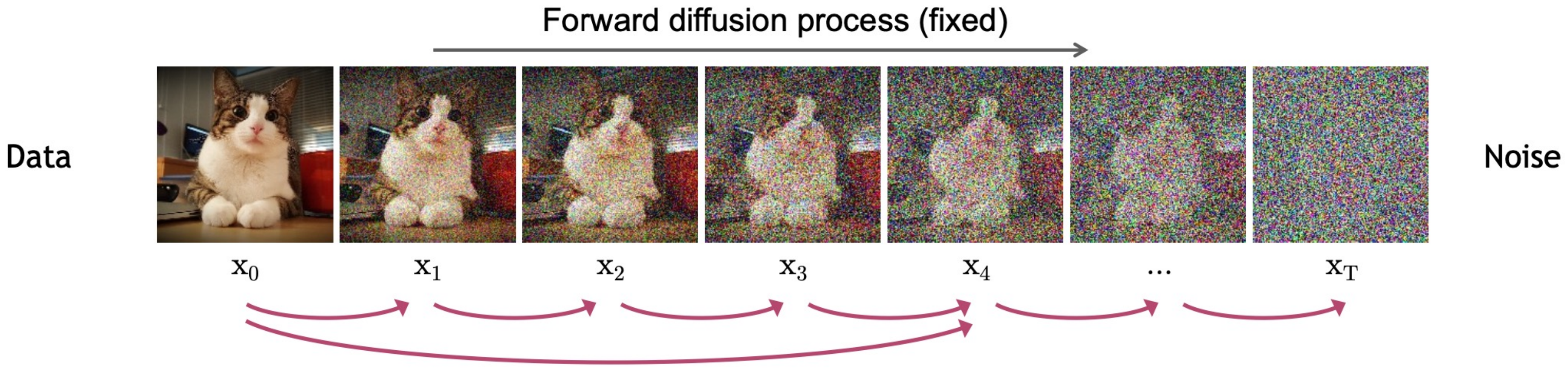
- Diffusion models consist of two processes
- Forward process where noise is iteratively added to an input
- Reverse (denoising) process that produces data given noise


Forward diffusion process (fixed)
Data
Noise
Reverse denoising process (generative)

- Training idea: run forward process, learn to predict noise
- Start with random noise, iteratively apply de-noising model

slide credit: Kreis, Gao, Vahdat tutorial

Diffusion Kernel



Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \to 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

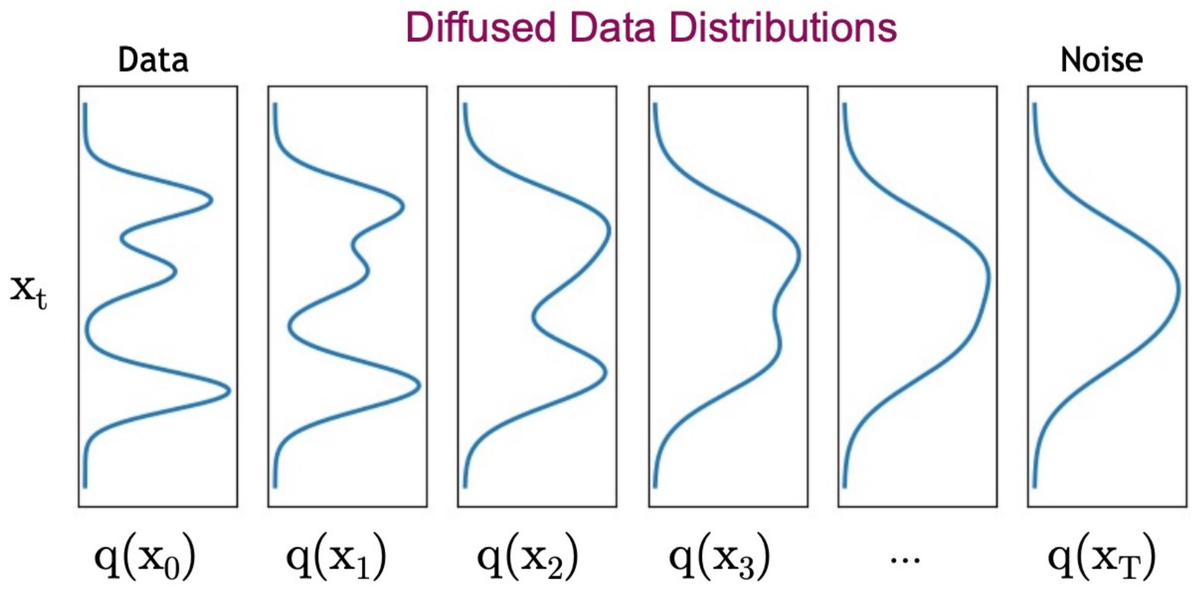
slide credit: Kreis, Gao, Vahdat tutorial

What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

$$\underbrace{q(\mathbf{x}_t)}_{\text{Diffused data dist.}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\text{Joint dist.}} \, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\text{Input data dist.}} \, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\text{Diffusion kernel}} \, d\mathbf{x}_0$$

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

slide credit: Kreis, Gao, Vahdat tutorial

Generative Learning by Denoising

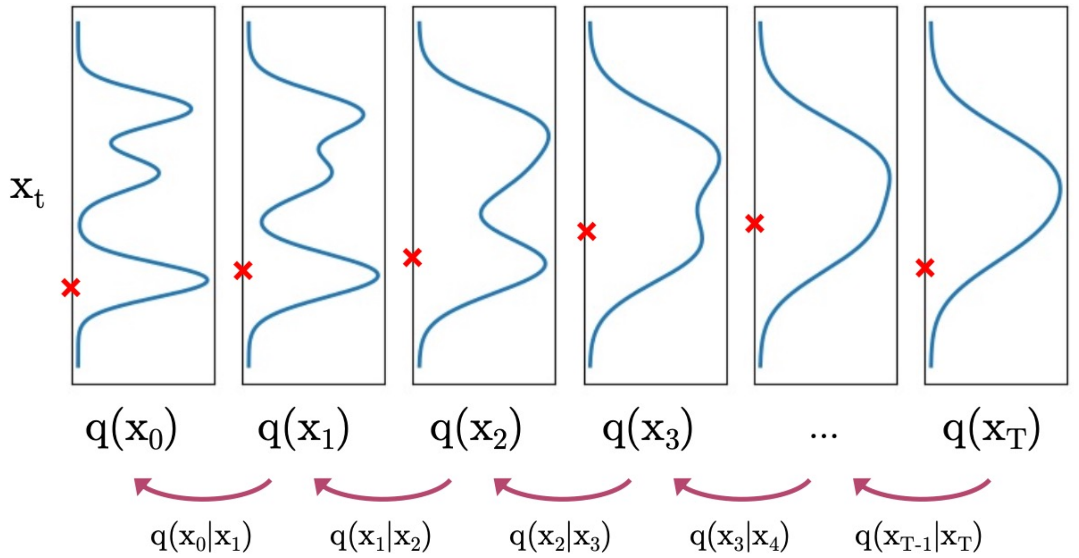
Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Generation:

Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$

Diffused Data Distributions



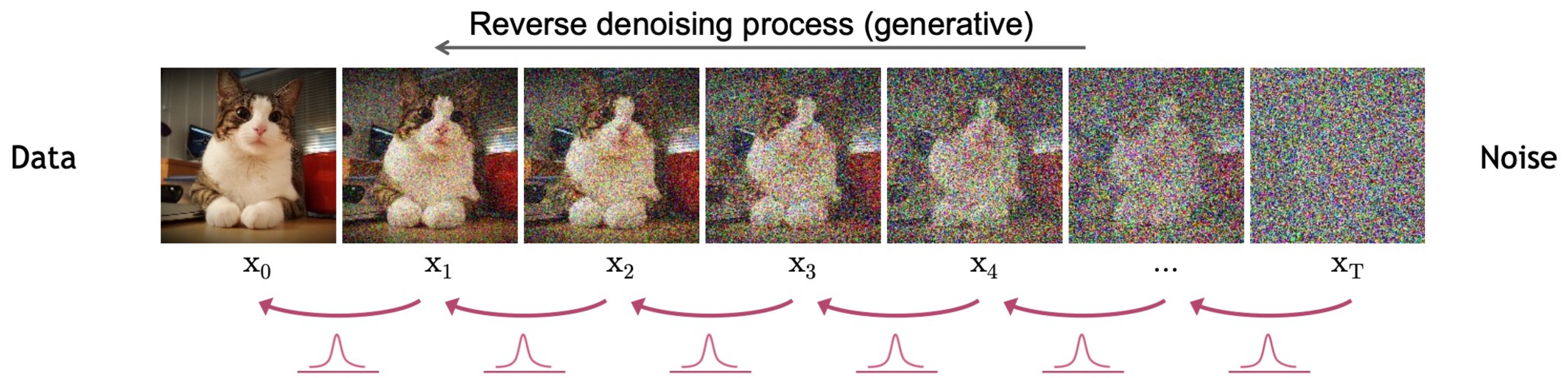
In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

slide credit: Kreis, Gao, Vahdat tutorial

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\text{Trainable network (U-net, Denoising Autoencoder)}}, \sigma_t^2\mathbf{I})$$

$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

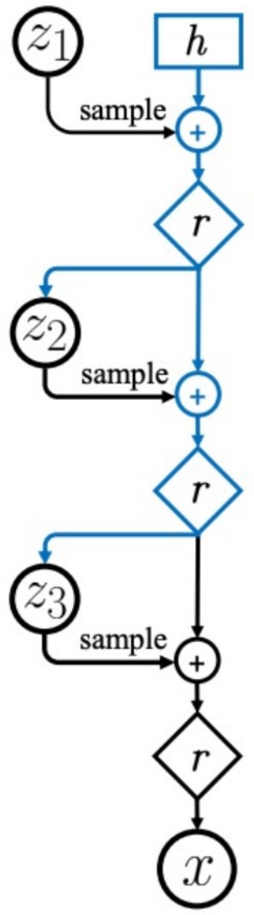
slide credit: Kreis, Gao, Vahdat tutorial

Connection to VAEs

Diffusion models can be considered as a special form of hierarchical VAEs.

However, in diffusion models:

- The encoder is fixed
- The latent variables have the same dimension as the data
- The denoising model is shared across different timestep
- The model is trained with some reweighting of the variational bound.



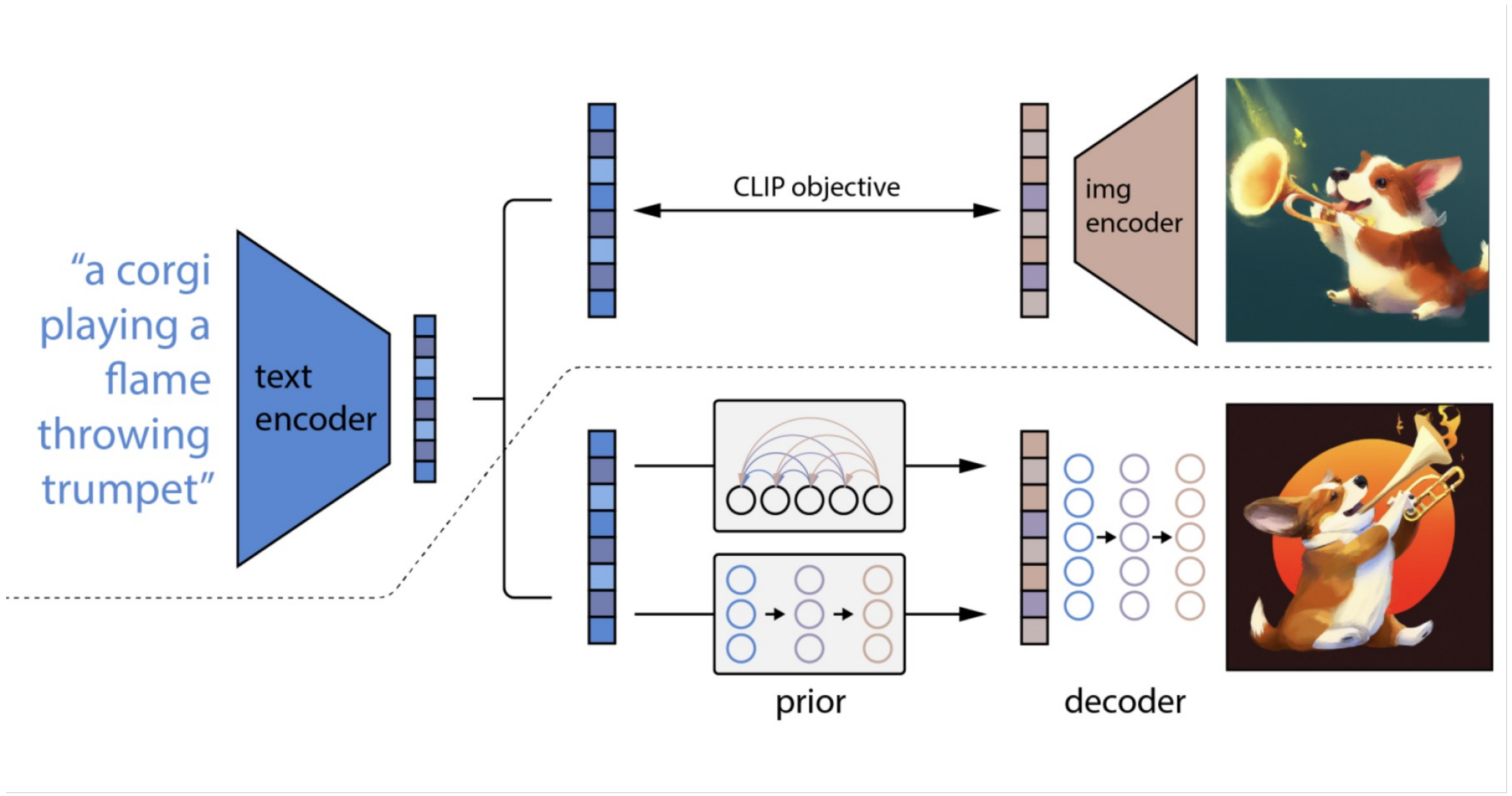
Vahdat and Kautz, NVAE: A Deep Hierarchical Variational Autoencoder, NeurIPS 2020
Sønderby, et al.. Ladder variational autoencoders, NeurIPS 2016.

slide credit: Kreis, Gao, Vahdat tutorial

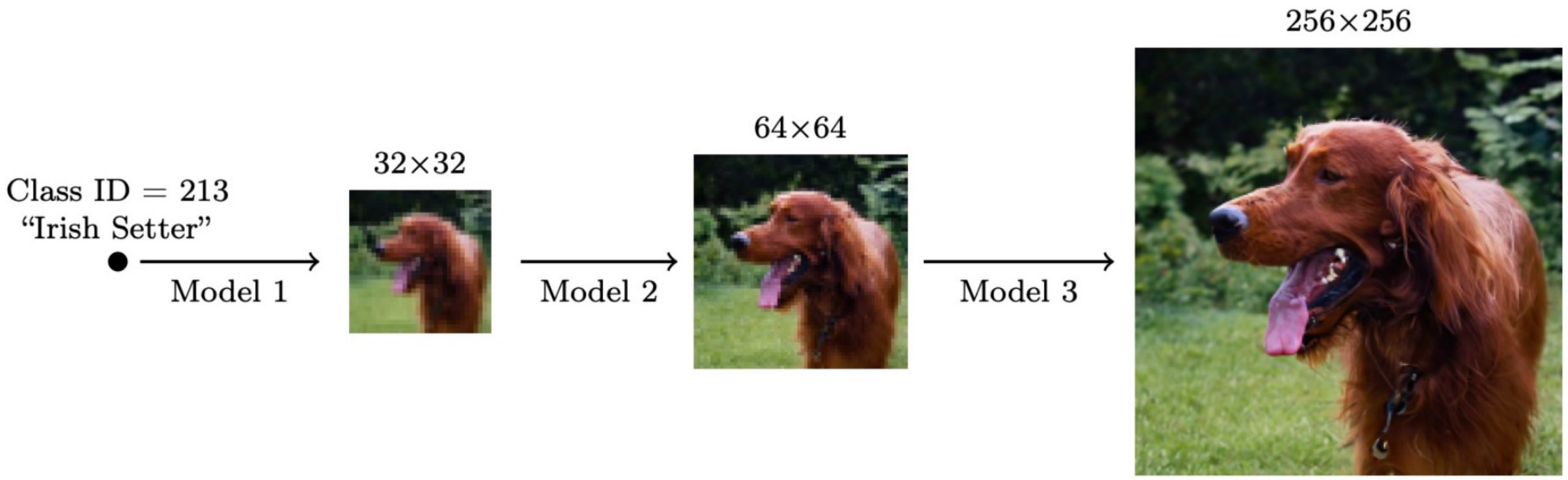
Conditioning a Diffusion Model

- Conditioned Diffusion: include the label / caption as an input
- Classifier Guidance: add the gradient of a (pre-trained) classifier wrt to target class as signal to the denoising model
  (push diffusion process towards target class at each step)
- CLIP Guidance (similar to classifier guidance but with CLIP model)
- Classifier-Free Guidance: loss based on difference between conditional vs unconditional likelihood

DALL-E 2 Architecture

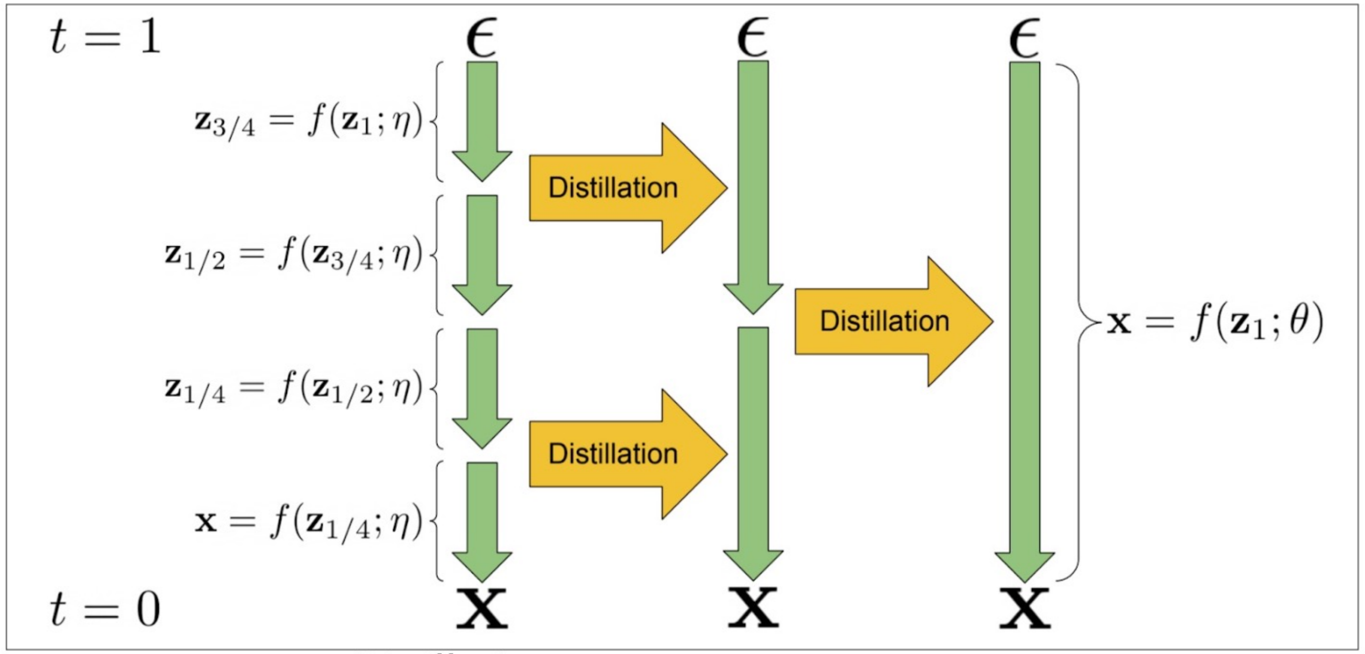


Cascaded Diffusion Models



Progressive distillation

- Distill a deterministic DDIM sampler to the same model architecture.
- At each stage, a "student" model is learned to distill two adjacent sampling steps of the "teacher" model to one sampling step.
- At next stage, the "student" model from previous stage will serve as the new "teacher" model.


$t = 1$
ϵ
ϵ
ϵ
$\mathbf{z}_{3/4} = f(\mathbf{z}_1; \eta)$
$\mathbf{z}_{1/2} = f(\mathbf{z}_{3/4}; \eta)$
$\mathbf{z}_{1/4} = f(\mathbf{z}_{1/2}; \eta)$
$\mathbf{x} = f(\mathbf{z}_{1/4}; \eta)$
Distillation
Distillation
Distillation
$\mathbf{x} = f(\mathbf{z}_1; \theta)$
$t = 0$
$\mathbf{x}$
$\mathbf{x}$
$\mathbf{x}$

Distillation stage $\longrightarrow$

Salimans & Ho, "Progressive distillation for fast sampling of diffusion models", ICLR 2022.

slide credit: Kreis, Gao, Vahdat tutorial