Generative Adversarial Networks

Aaron Mishkin

UBC MLRG 2018W2

Generative Adversial Networks

"Two imaginary celebrities that were dreamed up by a random number generator."

https://research.nvidia.com/publication/2017-10_Progressive-Growing-of

Why care about GANs?

Why to spend your limited time learning about GANs:

- GANs are achieving state-of-the-art results in a large variety of image generation tasks.
- There's been a veritable explosion in GAN publications over the last few years – many people are very excited!
- GANs are stimulating new theoretical interest in min-max optimization problems and "smooth games".

Why care about GANs: Hyper-realistic Image Generation

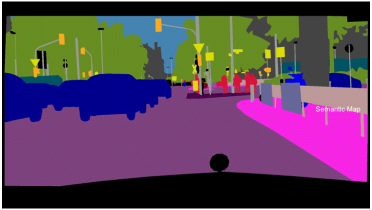
StyleGAN: image generatation with hierarchical style transfer [3].

https://arxiv.org/abs/1812.04948

Why care about GANs: Conditionally Generative Models

Conditional GANs: high-resolution image synthesis via semantic labeling [8].

Input: Segmentation **Output: Synthesized Image**



https://research.nvidia.com/publication/2017-12_High-Resolution-Image-Synthesis

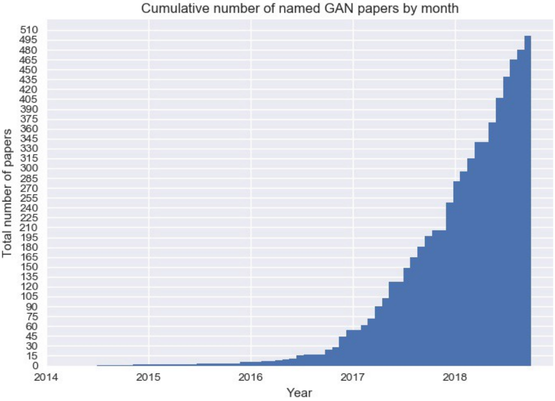
Why care about GANs: Image Super Resolution

SRGAN: Photo-realistic super-resolution [4].

Bicubic Interp. **SRGAN** **Original Image**

https://arxiv.org/abs/1609.04802

Why care about GANs: Publications



Approximately 500 papers GAN papers as of September 2018!

See https://github.com/hindupuravinash/the-gan-zoo for the exhaustive list of papers.

Generative Models

Generative Modeling

Generative Models estimate the probabilistic process that generated a set of observations $\mathcal{D}$.

- $\mathcal{D} = \left\{ \left(\mathbf{x}^i, \mathbf{y}^i \right) \right\}_{i=1}^{n}$: supervised generative models learn the joint distribution $p(\mathbf{x}^i, \mathbf{y}^i)$, often to compute $p(\mathbf{y}^i \mid \mathbf{x}^i)$.
- $\mathcal{D} = \left\{ \mathbf{x}^i \right\}_{i=1}^{n}$: unsupervised generative models learn the distribution of $\mathcal{D}$ for clustering, sampling, etc. We can:
  - directly estimate $p(\mathbf{x}^i)$,
  - introducing latents $\mathbf{y}^i$ and estimate $p(\mathbf{x}^i, \mathbf{y}^i)$.

Generative Modeling: Unsupervised Parametric Approaches

- **Direct Estimation:** Choose a parameterized family $p(\mathbf{x} \mid \theta)$ and learn θ by maximizing the log-likelihood

$$\theta^* = \arg\max \theta \sum_{i=1}^{n} \log p(\mathbf{x}^i \mid \theta).$$

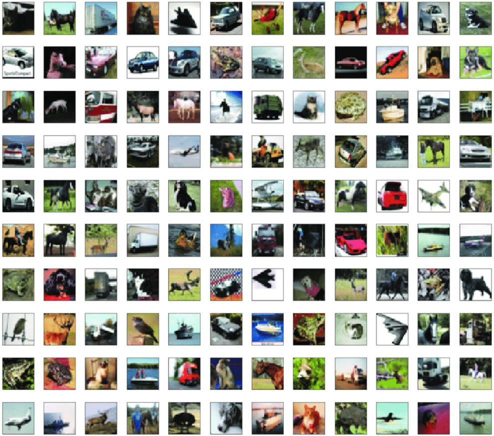
- **Latent Variable Models:** Define a joint distribution $p(\mathbf{x}, \mathbf{y} \mid \theta)$ and learn θ by maximizing the log-marginal likelihood

$$\theta^* = \arg\max \theta \sum_{i=1}^{n} \log \int_{\mathbf{z}^i} p(\mathbf{x}^i, \mathbf{z}^i \mid \theta) d\mathbf{z}.$$

Both approaches require that $p(\mathbf{x} \mid \theta)$ is easy to evaluate.

Generative Modeling: Models for (Very) Complex Data

How can we learn such models for very complex data?



https://www.researchgate.net/figure/Heterogeneousness-and-diversity-of-the-CIFAR-10-entries-in-their-10-

Generative Modeling: Normalizing Flows and VAEs

Design parameterized densities with huge capacity!

- **Normalizing flows:** sequence of non-linear transformations to a simple distribution $p_{\mathbf{z}}(\mathbf{z})$

$$p(\mathbf{x} \mid \theta_{0:k}) = p_{\mathbf{z}}(\mathbf{z}) \text{ where } \mathbf{z} = f_{\theta_k}^{-1} \circ \cdots \circ f_{\theta_1}^{-1} \circ f_{\theta_0}^{-1}(\mathbf{x})\,.$$

  $f_{\theta_j}^{-1}$ must be invertible with tractable log-det. Jacobians.

- **VAEs:** latent-variable models where inference networks specify parameters

$$p(\mathbf{x}, \mathbf{y} \mid \theta) = p(\mathbf{x} \mid f_\theta(\mathbf{y}))p_{\mathbf{y}}(\mathbf{y}).$$

  The marginal likelihood is maximized via the ELBO.

GANs

GANs: Density-Free Models

Generative Adversial Networks (GANs) instead use an unrestricted generator $G_{\theta_g}(\mathbf{z})$ such that

$$p(\mathbf{x} \mid \theta_g) = p_{\mathbf{z}}(\{\mathbf{z}\}) \text{ where } \{\mathbf{z}\} = G_{\theta_g}^{-1}(\mathbf{x}).$$

- **Problem:** the inverse image of $G_{\theta_g}(\mathbf{z})$ may be huge!
- **Problem:** it's likely intractable to preserve volume through $G(\mathbf{z}; \theta_g)$.

So, we can't evaluate $p(\mathbf{x} \mid \theta_g)$ and we can't learn θ_g by maximum likelihood.

GANs: Discriminators

GANs learn by comparing model samples with examples from $\mathcal{D}$.

- Sampling from the generator is easy:

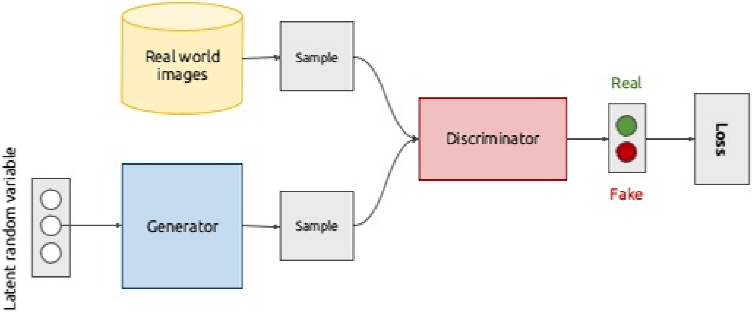
$$\hat{\mathbf{x}} = G_{\theta_g}(\hat{\mathbf{z}}), \text{ where } \hat{\mathbf{z}} \sim p_{\mathbf{z}}(\mathbf{z}).$$

- Given a sample $\hat{\mathbf{x}}$, a discriminator tries to distinguish it from true examples:

$$D(\mathbf{x}) = \Pr\left(\mathbf{x} \sim p_{\text{data}}\right).$$

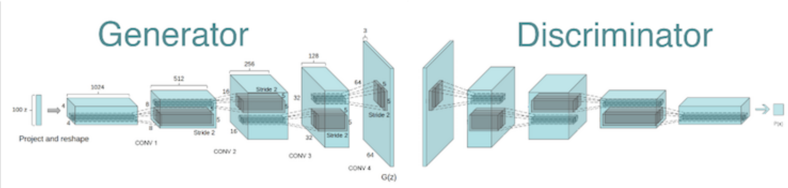
- The discriminator "supervises" the generator network.

GANs: Generator + Descriminator



https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

- Let $\mathbf{z} \in \mathbb{R}^m$ and $p_{\mathbf{z}}(\mathbf{z})$ be a simple base distribution.
- The generator $G_{\theta_g}(\mathbf{z}) : \mathbb{R}^m \to \tilde{\mathcal{D}}$ is a deep neural network.
    - $\tilde{\mathcal{D}}$ is the manifold of generated examples.
- The discriminator $D_{\theta_d}(\mathbf{x}) : \mathcal{D} \cup \tilde{\mathcal{D}} \to (0, 1)$ is also a deep neural network.

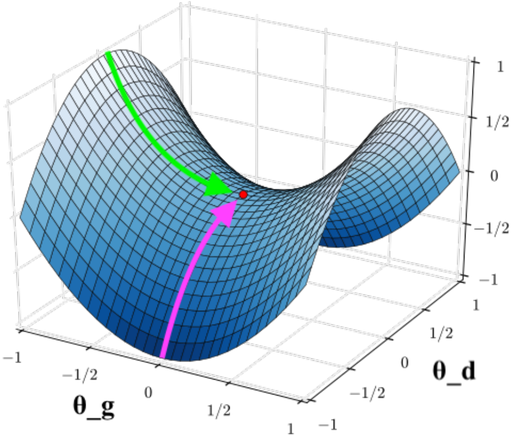


https://arxiv.org/abs/1511.06434

GANs: Saddle-Point Optimization

Saddle-Point Optimization: learn $G_{\theta_g}(\mathbf{z})$ and $D_{\theta_d}(\mathbf{x})$ jointly via the objective $V(\theta_d, \theta_g)$:

$$\min_{\theta_g} \max_{\theta_d} \underbrace{\mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})]}_{\text{likelihood of true data}} + \underbrace{\mathbb{E}_{p_{\mathbf{z}}(\mathbf{z})} \left[\log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right]}_{\text{likelihood of generated data}}$$



GANs: Optimal Discriminators

Claim: Given G_{θ_g} defining an implicit distribution $p_g = p(\mathbf{x} \mid \theta_g)$, the optimal descriminator is

$$D^*(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{g}}(\mathbf{x})}.$$

Proof Sketch:

$$\begin{aligned} V(\theta_d, \theta_g) &= \int_{\mathcal{D}} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) d\mathbf{x} + \int_{\tilde{\mathcal{D}}} p(\mathbf{z}) \log(1 - D(G_{\theta_g}(\mathbf{z}))) d\mathbf{z} \\ &= \int_{\mathcal{D} \cup \tilde{\mathcal{D}}} p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x})) d\mathbf{x} \end{aligned}$$

Maximizing the integrand for all $\mathbf{x}$ is sufficient and gives the result (see bonus slides).

Previous Slide: https://commons.wikimedia.org/wiki/File:Saddle_point.svg

GANs: Jensen-Shannon Divergence and Optimal Generators

Given an optimal discriminator $D^*(\mathbf{x})$, the generator objective is

$$C(\theta_g) = \mathbb{E}_{p_{\text{data}}}\left[\log D^*_{\theta_d}(\mathbf{x})\right] + \mathbb{E}_{p_g(\mathbf{x})}\left[\log\left(1 - D^*_{\theta_d}(\mathbf{x})\right)\right]$$

$$= \mathbb{E}_{p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{g}}(\mathbf{x})}\right] + \mathbb{E}_{p_g(\mathbf{x})}\left[\log \frac{p_g(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{g}}(\mathbf{x})}\right]$$

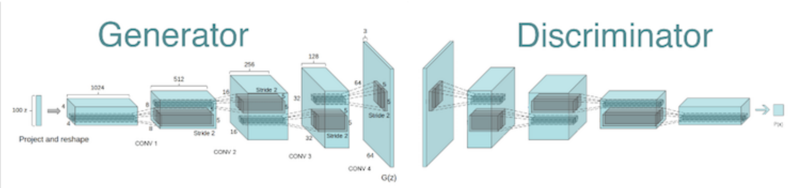
$$\propto \underbrace{\frac{1}{2} KL\left(p_{\text{data}} \middle\| \frac{(p_{\text{data}} + p_g)}{2}\right) + \frac{1}{2} KL\left(p_g \middle\| \frac{(p_{\text{data}} + p_g)}{2}\right)}_{\text{Jensen-Shannon Divergence}}$$

$C(\theta_g)$ achives its global minimum at $p_g = p_{\text{data}}$ given an optimal discriminator!

GANs: Learning Generators and Discriminators

Putting these results to use in practice:

- High-capacity discriminators D_{θ_d} approximate the Jensen-Shannon divergence when close to global maximum.
- D_{θ_d} is a "differentiable program".
- We can use D_{θ_d} to learn G_{θ_g} with our favourite gradient descent method.



https://arxiv.org/abs/1511.06434

GANs: Training Procedure

for $i = 1 \dots N$ **do**

for $k = 1 \dots K$ **do**

- Sample noise samples $\{\mathbf{z}^1, \dots, \mathbf{z}^m\} \sim p_{\mathbf{z}}(\mathbf{z})$
- Sample examples $\{\mathbf{x}^1, \dots, \mathbf{x}^m\}$ from $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator D_{θ_d}:

$$\theta_d = \theta_d - \alpha_d \nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\mathbf{x}^i\right) + \log\left(1 - D\left(G\left(\mathbf{z}^i\right)\right)\right)\right].$$

end for

- Sample noise samples $\{\mathbf{z}^1, \dots, \mathbf{z}^m\} \sim p_{\mathbf{z}}(\mathbf{z})$.
- Update the generator G_{θ_g}:

$$\theta_g = \theta_g - \alpha_g \nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\mathbf{z}^i\right)\right)\right).$$

end for

Problems (c. 2016)

Problems with GANs

- **Vanishing gradients:** the discriminator becomes "too good" and the generator gradient vanishes.
- **Non-Convergence:** the generator and discriminator oscillate without reaching an equilibrium.
- **Mode Collapse:** the generator distribution collapses to a small set of examples.
- **Mode Dropping:** the generator distribution doesn't fully cover the data distribution.

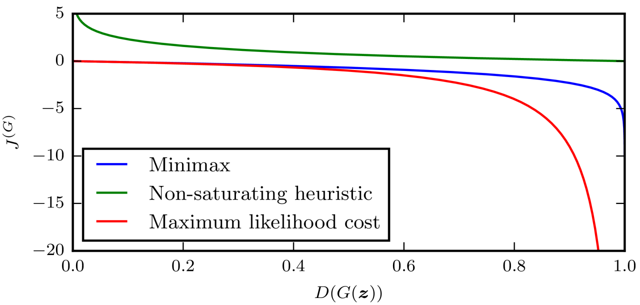
Problems: Vanishing Gradients

- The **minimax** objective saturates when D_{θ_d} is close to perfect:

$$V(\theta_d, \theta_g) = \mathbb{E}_{p_{\text{data}}}[\log D_{\theta_d}(\mathbf{x})] + \mathbb{E}_{p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right].$$

- A **non-saturating heuristic** objective for the generator is

$$J(G_{\theta_g}) = -\mathbb{E}_{p_{\mathbf{z}}(\mathbf{z})}\left[\log\left(D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right].$$



https://arxiv.org/abs/1701.00160

Problems: Addressing Vanishing Gradients

Solutions:

- **Change Objectives:** use the non-saturating heuristic objective, maximum-likelihood cost, etc.
- **Limit Discriminator:** restrict the capacity of the discriminator.
- **Schedule Learning:** try to balance training D_{θ_d} and G_{θ_g}.

Problems: Non-Convergence

Simultaneous gradient descent is not guaranteed to converge for minimax objectives.

- Goodfellow et al. only showed convergence when updates are made in the function space [2].
- The parameterization of D_{θ_d} and G_{θ_g} results in highly non-convex objective.
- In practice, training tends to oscillate – updates "undo" each other.

Problems: Addressing Non-Convergence

Solutions: Lots and lots of hacks!

6: Use Soft and Noisy Labels

- Label Smoothing, i.e. if you have two target labels: Real=1 and Fake=0, then for each incoming sample, if it is real, then replace the label with a random number between 0.7 and 1.2, and if it is a fake sample, replace it with 0.0 and 0.3 (for example).
  - Salimans et. al. 2016
- make the labels the noisy for the discriminator: occasionally flip the labels when training the discriminator

7: DCGAN / Hybrid Models

- Use DCGAN when you can. It works!
- if you cant use DCGANs and no model is stable, use a hybrid model : KL + GAN or VAE + GAN

8: Use stability tricks from RL

- Experience Replay
  - Keep a replay buffer of past generations and occassionally show them
  - Keep checkpoints from the past of G and D and occassionaly swap them out for a few iterations
- All stability tricks that work for deep deterministic policy gradients
- See Pfau & Vinyals (2016)

9: Use the ADAM Optimizer

- optim.Adam rules!
  - See Radford et. al. 2015
- Use SGD for discriminator and ADAM for generator

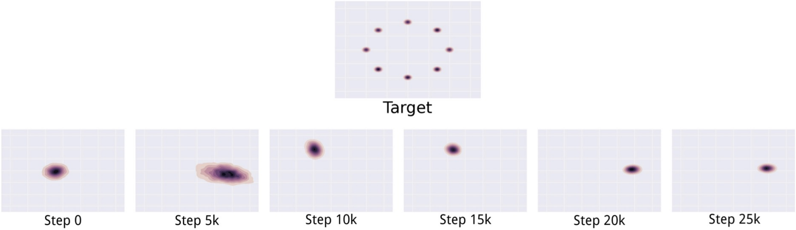
https://github.com/soumith/ganhacks

Problems: Mode Collapse and Mode Dropping

One Explanation: SGD may optimize the max-min objective

$$\max_{\theta_d} \min_{\theta_g} \mathbb{E}_{p_{\text{data}}} [\log D_{\theta_d}(\mathbf{x})] + \mathbb{E}_{p_{\mathbf{z}}(\mathbf{z})} \left[\log \left(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}))\right)\right]$$

Intuition: the generator maps all $\mathbf{z}$ values to the $\hat{\mathbf{x}}$ that is mostly likely to fool the discriminator.



https://arxiv.org/abs/1701.00160

A Possible Solution

A Possible Solution: Alternative Divergences

There are a large variety of divergence measures for distributions:

- **f-Divergences:** (e.g. Jensen-Shannon, Kullback-Leibler)

$$D_f \ (P \,||Q) = \int_{\chi} q(\mathbf{x}) f(\frac{p(\mathbf{x})}{q(\mathbf{x})}) d\mathbf{x}$$

  - GANs [2], f-GANs [7], and more.
- **Integral Probability Metrics:** (e.g. Earth Movers Distance, Maximum Mean Discrepancy)

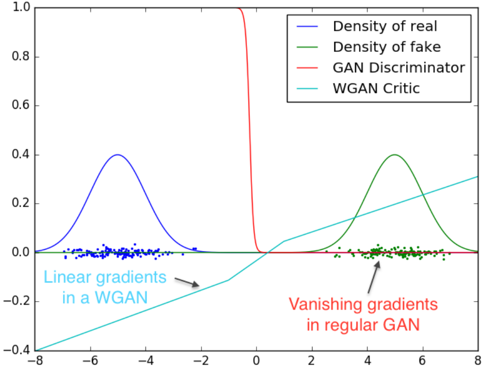
$$\gamma_F \ (P \,||Q) = \sup_{f \in F} \left| \int f dP - \int f dQ \ \right|$$

  - Wasserstein GANs [1], Fisher GANs [6], Sobolev GANs [5] and more.

A Possible Solution: Wasserstein GANs

Wasserstein GANs: Strong theory and excellent empirical results.

- "In no experiment did we see evidence of mode collapse for the WGAN algorithm." [1]



https://arxiv.org/abs/1701.07875

Summary

Summary

Recap:

- GANs are a class of density-free generative models with (mostly) unrestricted generator functions.
- Introducing adversial discriminator networks allows GANs to learn by minimizing the Jensen-Shannon divergence.
- Concurrently learning the generator and discriminator is challenging due to
  - Vanishing Gradients,
  - Non-convergence due to oscilliation
  - Mode collapse and mode dropping.
- A variety of alternative objective functions are being proposed.

Agknowledgements and References

There are lots of excellent references on GANs:

- Sebastian Nowozin's presentation at MLSS 2018.
- NIPS 2016 tutorial on GANs by Ian Goodfellow.
- A nice explanation of Wasserstein GANs by Alex Irpan.

Bonus: Optimal Discriminators Cont.

The integrand

$$h(D(\mathbf{x})) = p_{\text{data}}(\mathbf{x}) \log D(\mathbf{x}) + p_g(\mathbf{x}) \log(1 - D(\mathbf{x}))$$

is concave for $D(\mathbf{x}) \in (0, 1)$. We take the derivative and compute a stationary point in the domain:

$$\frac{\partial h(D(\mathbf{x}))}{\partial D(\mathbf{x})} = \frac{p_{\text{data}}(\mathbf{x})}{D(\mathbf{x})} - \frac{p_g(\mathbf{x})}{1 - D(\mathbf{x})} = 0$$
$$\Rightarrow D(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_{\text{g}}(\mathbf{x})}.$$

This minimizes the integrand over the domain of the discriminator, completing the proof.

References i

Martin Arjovsky, Soumith Chintala, and Léon Bottou.
Wasserstein gan.
arXiv preprint arXiv:1701.07875, 2017.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio.
Generative adversarial networks. arxiv e-prints.
arXiv preprint arXiv:1406.2661, 2014.

Tero Karras, Samuli Laine, and Timo Aila.
A style-based generator architecture for generative adversarial networks.
arXiv preprint arXiv:1812.04948, 2018.

Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al.
Photo-realistic single image super-resolution using a generative adversarial network.
In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4681–4690, 2017.

Youssef Mroueh, Chun-Liang Li, Tom Sercu, Anant Raj, and Yu Cheng.
Sobolev gan.
arXiv preprint arXiv:1711.04894, 2017.

Youssef Mroueh and Tom Sercu.
Fisher gan.
In *Advances in Neural Information Processing Systems*, pages 2513–2523, 2017.

References iii

Sebastian Nowozin, Botond Cseke, and Ryota Tomioka.
f-gan: Training generative neural samplers using variational divergence minimization.
In *Advances in neural information processing systems*, pages 271–279, 2016.

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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8798–8807, 2018.