Generate Faster: Accelerate Generative Models Using VAE

Group 7

Ziyue Lin, Liangyue Zhou,

Jiayi Yao, Dixin Deng

April 23, 2023

Overview

Motivation

Our Journey to Generative Models

- Generative models (e.g., DALLE2, stable-diffusion) usually demand a substantial amount of time and resources to train and test.
- We constantly feel depleted after a long period of failed attempts.
- Can we sacrifice a little amount of accuracy to achieve a significant acceleration of generative models?
- Engineers/Corporations $\rightarrow$ trail and error (e.g., finding SOTA model architectures or weights)
- Researchers/Students $\rightarrow$ more intricate and general directions (e.g., explaining the network, more efficient training diagrams)

Motivation

Generalizable Training Framework of Generative Image Models

- **High-level idea:** Leverage high-quality VAEs to aid both the efficiency and quality of generative image models.
- **Current implementation:** Deploy well-trained VAE to vanilla GAN.

Some Potential Benefits

- In some applications, such as in gaming and virtual environments, where user interact with the generative model in real-time, a faster generative model can **enhance the user experience**.
- Accelerating generative models can lead to significant **energy savings**, as training and inference of deep learning models can require a considerable amount of computational resources.
- Boost researcher/engineers' **work iteration efficiency**!

Background

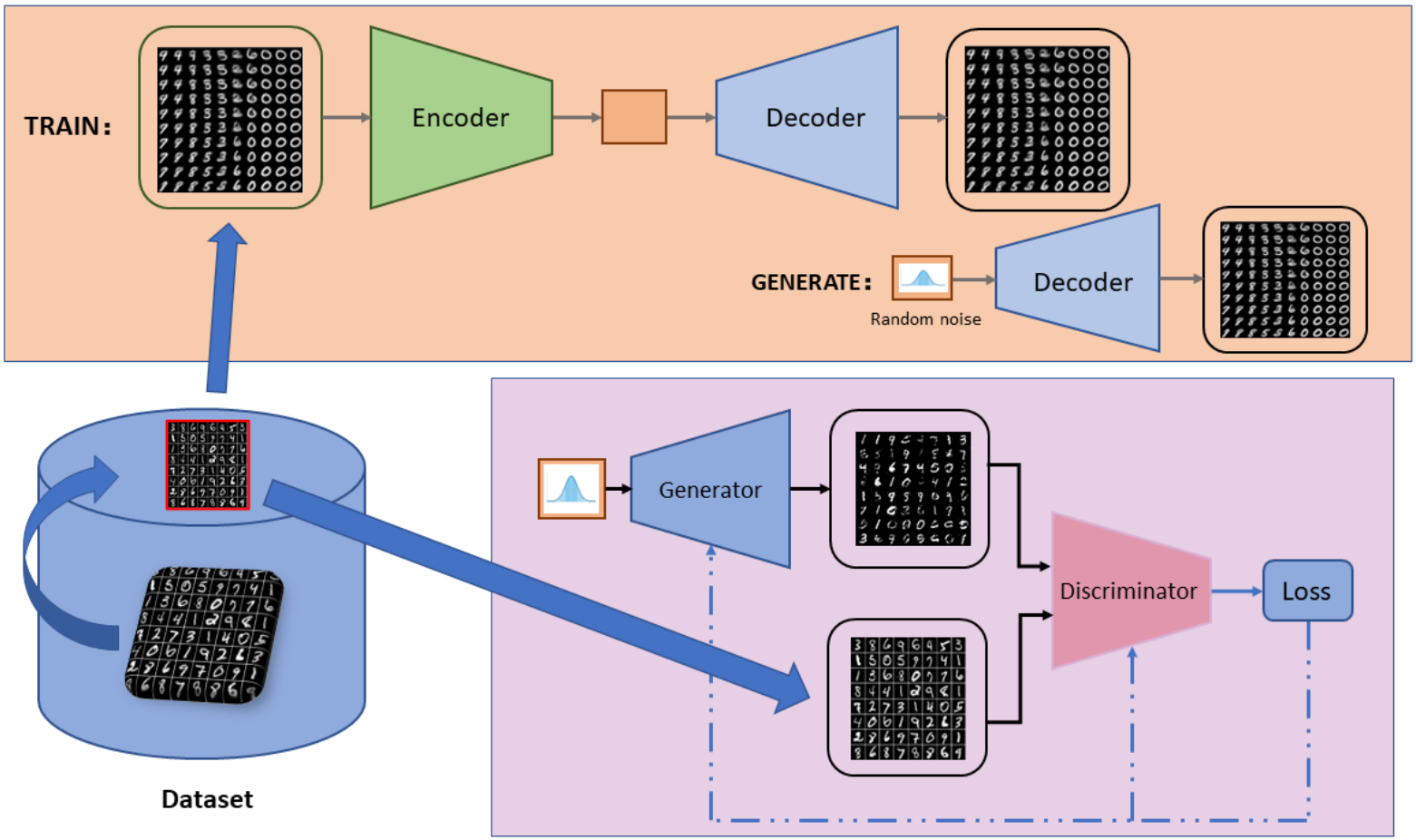

Figure: Traditional GAN and VAE models

Methodology

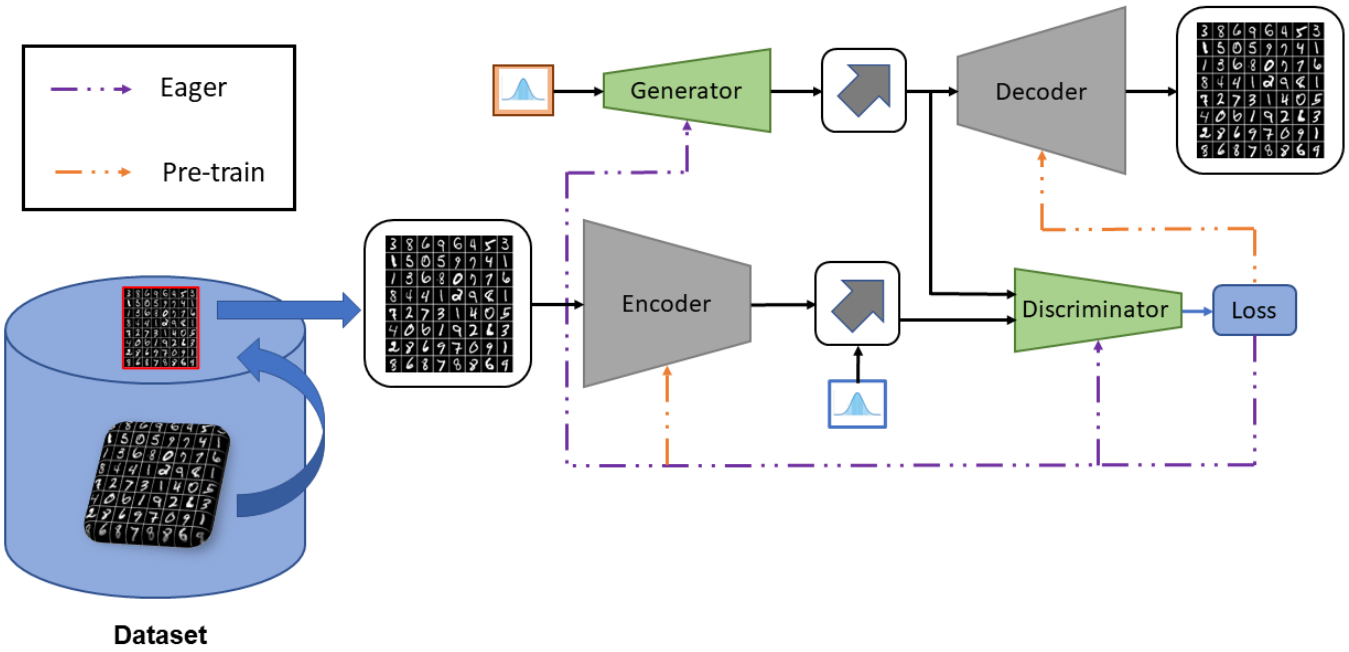

Figure: GAN model accelerated by pretrained\eager VAE

- By only generating and discriminating latent features, the computational work in the training process shrinks notably, which will expedite the algorithm.

Methodology

Algorithm Minibatch version of the Auto-Encoding VB (AEVB) algorithm.

$\boldsymbol{\theta}$, $\phi \leftarrow$Initialize parameters

for *number of learning iterations* **do**

- $\mathrm{X}^M \leftarrow$Random minibatch of M datapoints (drawn from full dataset)
- $\epsilon \leftarrow$Random samples from noise distribution z($\boldsymbol{\epsilon}$)
- g $\leftarrow \nabla_{\boldsymbol{\theta},\phi} \widetilde{\mathcal{L}}^M \left(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon}\right)$
- $\boldsymbol{\theta}$, $\phi \leftarrow$Update parameters using gradients g (e.g. SGD)

end

return $p(\theta, \phi)$

Methodology

Algorithm Minibatch stochastic gradient descent training of generative adversarial nets.

for *number of learning iterations* **do**

  for *k steps* **do**

    • Sample minibatch of m noise samples $\{p^{(1)}, \ldots, p^{(m)}\}$ from noise prior $p_g(z)$.

    • Sample minibatch of m examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from Algorithm 1 result $p(\theta, \phi)$.

    • Update the discriminator by ascending its stochastic gradient:

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

  end

end

Methodology

Algorithm Generating image process

for *k steps* **do**

- Sample minibatch of m noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
- Update the generator by descending its stochastic gradient:

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

end

Data Collection

We test our model using the following three datasets:

- MNIST Dataset: A dataset of **handwritten digits** consists of 60,000 training images and 10,000 testing images
- Fashion MNIST Dataset: A dataset of **fashion images** consists of 60,000 training images and 10,000 testing images

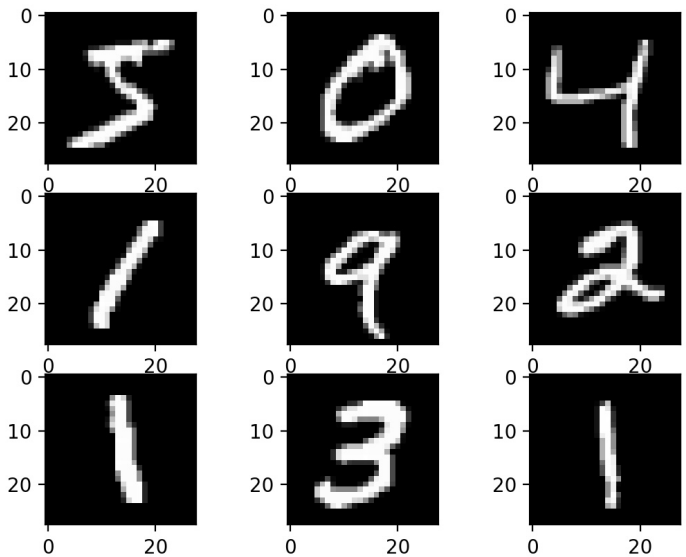

Figure: Sample Images from MNIST Dataset

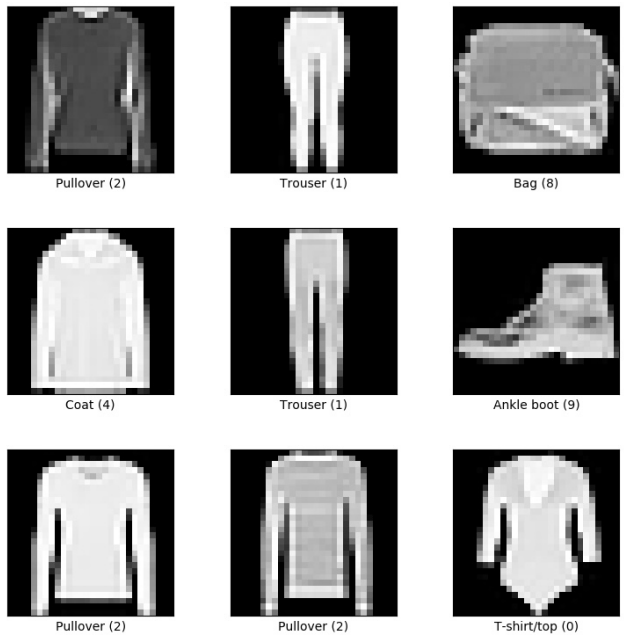

Figure: Sample Images from Fashion MNIST Dataset

Data Collection

We test our model using the following three datasets (continued):

- CelebA Dataset: 202,599 number of **face images** of various celebrities

Figure: Sample Images from CelebA Dataset

Experimental Results

Running Time Statistics

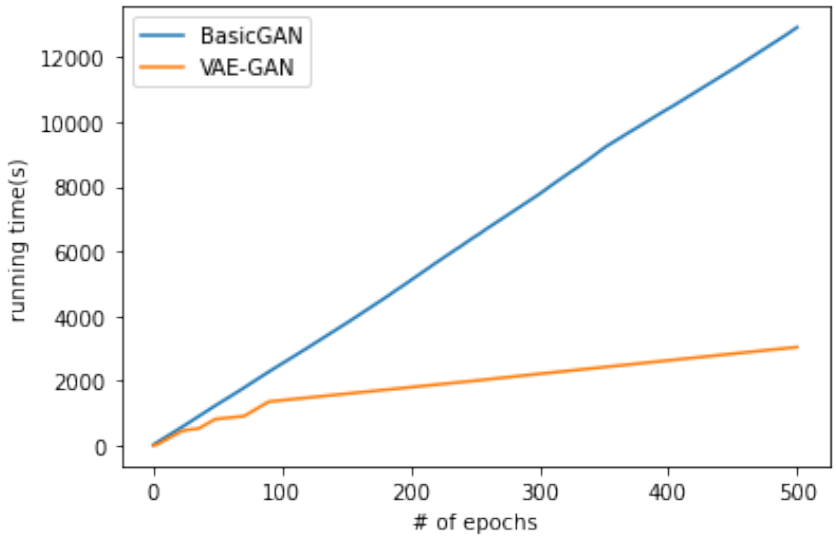

Figure: Trained on MNIST

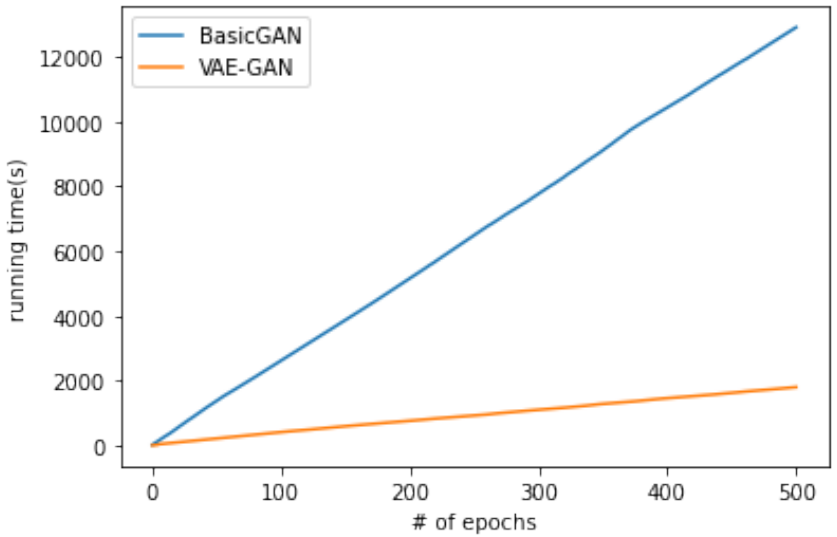

Figure: Trained on FashionMNIST

Experiment Results

Generator Loss and Discriminator Loss

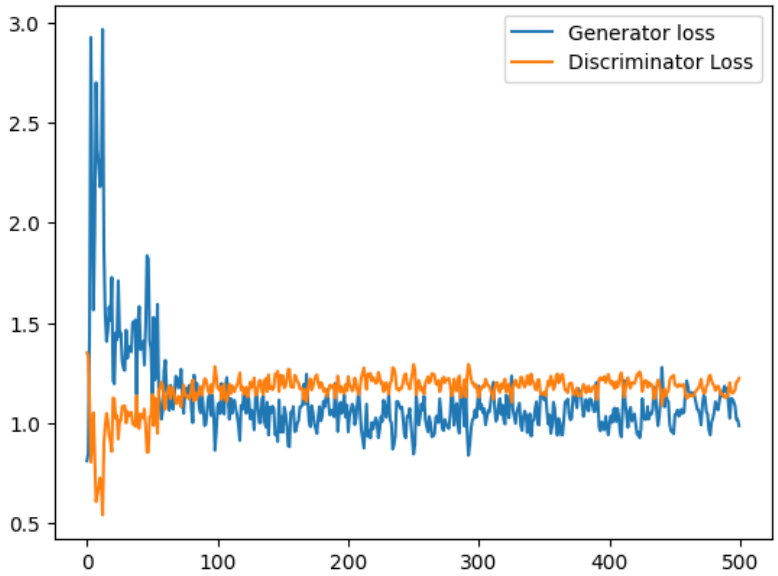

Figure: VAE-GAN trained on MNIST

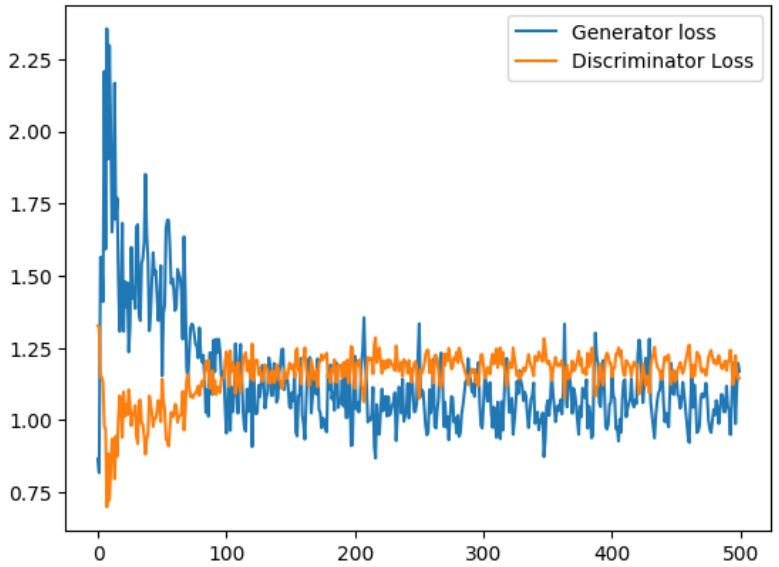

Figure: VAE-GAN trained on FashionMNIST

The undulating similarity of generator loss and discriminator loss implies the fierceness of competition and the soundness of network complexity selection.

Experimental Results

VAE-GAN(Upper row) V.S. Basic GAN(Lower row)

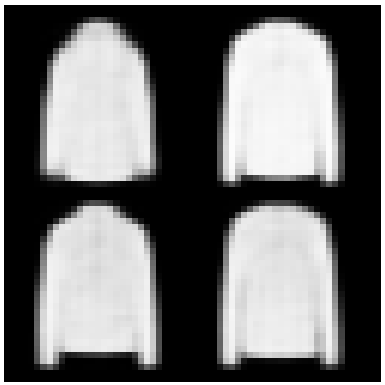

Figure: epoch1

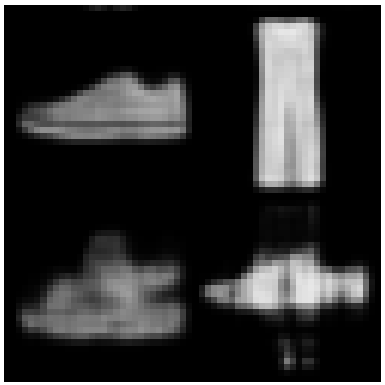

Figure: epoch100

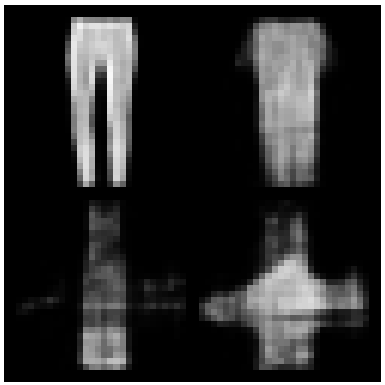

Figure: epoch300

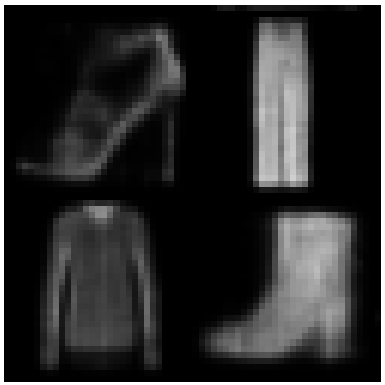

Figure: epoch500

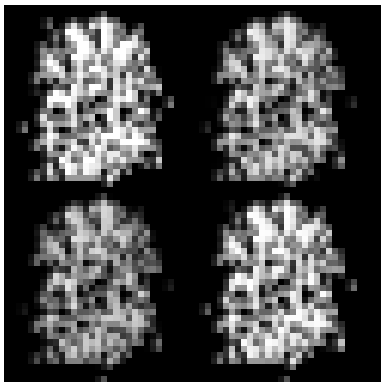

Figure: epoch1

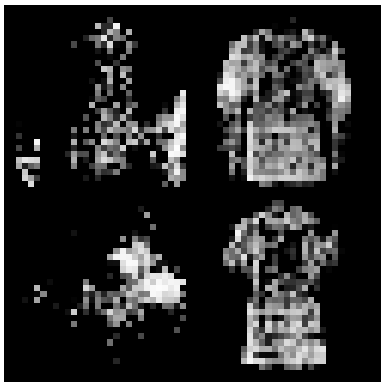

Figure: epoch100

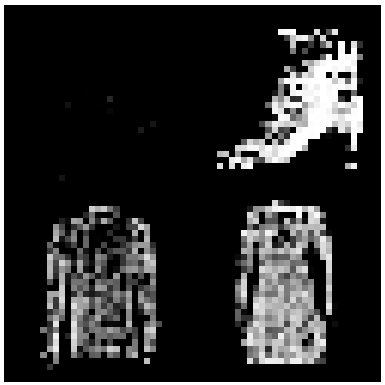

Figure: epoch300

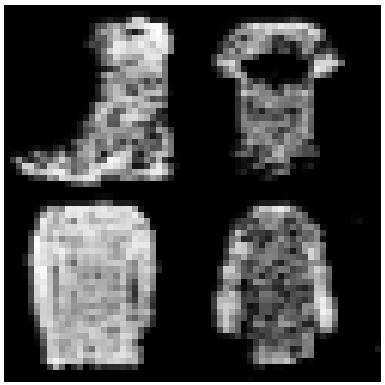

Figure: epoch500

Experimental Results

Image Quality Exhibition

Figure: Basic GAN on MNIST

Figure: VAE-GAN on MNIST

Experimental Results

Image Quality Exhibition

Figure: Basic GAN on FashionMNIST

Figure: VAE-GAN on FashionMNIST

Surprisingly, VAE-GAN incorporates the advantages of VAE model and produces clearer images.

Conclusion

Summary

We successfully materialized our idea of **accelerating basic GAN model with a pre-trained VAE model**, which could be applied and generalized to other generative models.

A view of the future

Our architectural novelty permits pre-trained VAE model to be grafted onto any other generative models, which enables researchers to combine the advantages of VAE models and other models

- Profitability of GAN: Governability (Conditioned Outputs), Excellent performance while generating new images
- Desirability of VAE: Learning hidden representations, Proficiency while training, Reusable module
- Eligibility of other generative models: .....

Conclusion

Ongoing Project: Synthesis of StarGAN and VAE

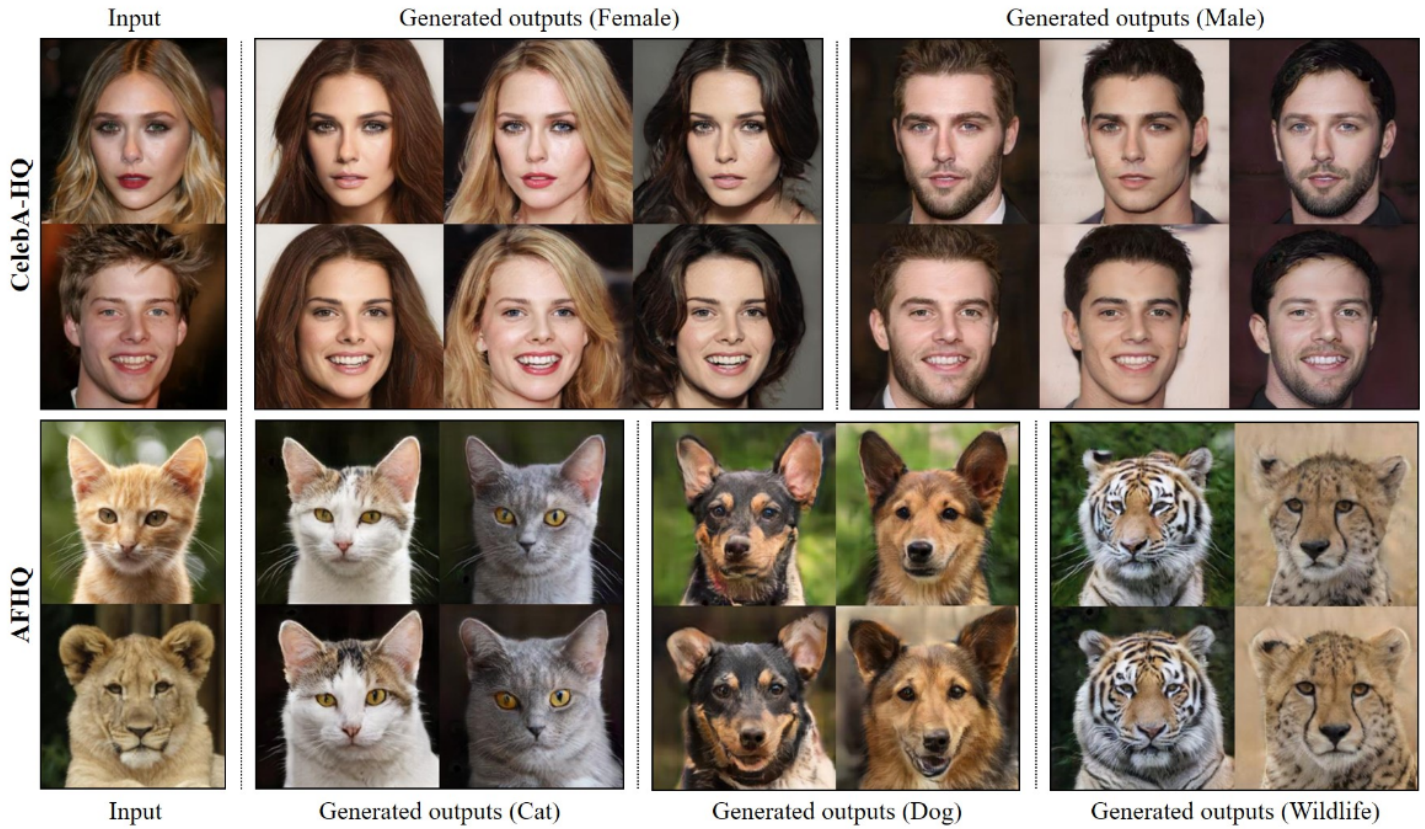

Figure: Conditioned outputs of StarGAN

Conclusion

Ongoing Project: Synthesis of StarGAN and VAE

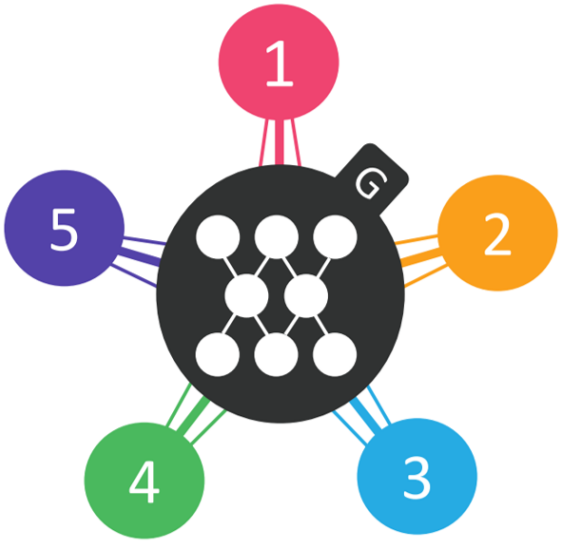

Figure: Original starGAN

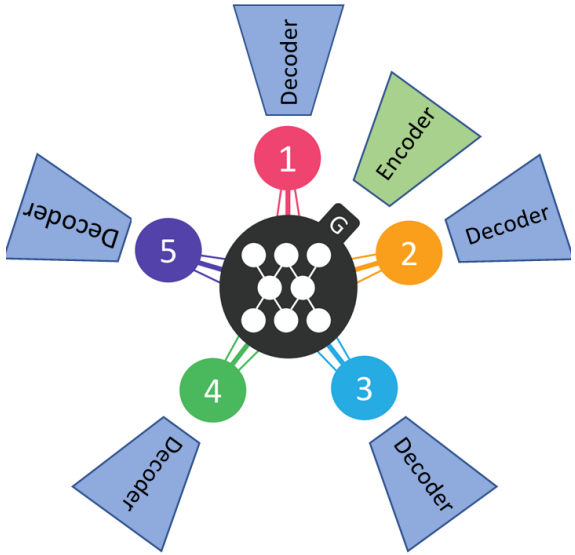

Figure: Combine starGAN and VAE

References

- https://github.com/lyeoni/pytorch-mnist-GAN/tree/master
- https://github.com/lyeoni/pytorch-mnist-VAE
- [Kingma and Welling, 2013] Auto-Encoding Variational Bayes
- [Choi et al., 2018] StarGAN: Unified Generative Adversarial Networks for Multi-Domain Image-to-Image Translation
  https://github.com/clovaai/stargan-v2

Thank You!

Q & A