John Wang 015938845 john.wang@sjsu.edu MSAI, Team Coordinator
Jack Zhu 012218128 jiali.zhu@sjsu.edu MSAI
Amy Phan 012194572 amylan-anh.phan@sjsu.edu MSSE
David Archer 013900770 guosheng.zhang@sjsu.edu MSAI

Generative Adversarial Networks: Face Generators

Works contributed by each team member

John Wang	Team coordinator, Primary coder for Homebrew GANs, helped other group member run implementations	25% in coding, discussion, organizing, testing, document and PPT
Jack Zhu	Fine tuning and testing the Vanilla GAN, DCGans prototype, providing ideas	25% in coding, discussion, organizing, testing, document and PPT
Amy Phan	Experiments on Vanilla GAN and tested Keras' StyleGAN example code for comparison	25% in coding, discussion, organizing, testing, document and PPT
David Archer	Experiments on Vanilla GAN, creating Cartoonizer app and Restorer app	25% in coding, discussion, organizing, testing, document and PPT

Abstract

Machine learning has revolutionized human vision related tasks as it reduced the human cost of such tasks and led to the improvement of society. Recently, variational autoencoders, generative adversarial networks (2014) and diffusion models (2015) have become very influential in the task of generating data using deep neural networks.

In our project, we will focus on the generative adversarial network (GANs) approach for face generation in order to explore the topic. We will develop a GAN model to generate faces from the celebA dataset and other resources. We hope to get an understanding of the mechanics, limitations and the right situations to apply the GANs technique. Our stretch goal is to see if we can cartoonize these generated faces into thumbnails.

In our project, we will first use tensorflow, google Colab Pro+, and celebA faces datasets to generate human faces. We will be using python, tensorflow keras for CNNs, numpy and pandas for data processing, PIL for image processing and matplotlib for graphing data.

We were able to accomplish generating human faces, but we faced challenges when it came to runtime and the system crashing. In order to address these problems, we moved to google colaband upgrade to the PRO+ for performance. Additionally, we faced tuning problems as the occasional either the generator or discriminator models would overpower the other causing the model to collapse into noise. We spent significant time tuning the network. In general, we have learned that GANs are a good algorithm to apply for generating images as long as you have a good dataset and a lot of time to tune the GANs. The difficulty in creating a good GANs requires a good dataset, lots of processing power and time to tune the model. In our project, we limited our image to 64 by 64 pixels and fine tuned our model to be suitable for the project. Finally, with the required latent space of 100 will achieve a better result to run. This process still took ~6 hours to run and achieved ok results. In situations where processing power is limited GANs is not an appropriate algorithm.

In the beginning, we prototyped our system using DCGANs and Mnist dataset to generate simple digits from 0-9. Using this as a basis for our project, we moved onto other techniques. From here we attempted to build our own homebrewed GANs system which incorporated various GANs architectures such as DCGANs and StyleGANs. We then compared it to pre-made state of the art implementations of StyleGANs. The premade keras implementation of Style GANs outperformed our model due to its incorporation of methodologies from progressive GANs (starting with small images and growing them larger) and WGANs (Wasserstein loss).

After creating and comparing different GANs, we created an application to cartoonize photos and images. We utilized bilateralFilter and drawContours functions in OpenCV and clusterize the original colors. The result is adorable.

Finally, we experimented with GFPGANs which is based on another app BasicSR and dedicated to restoring old or damaged photos. The app can even activate the faces in arts and make them alive.

All in all, the GAN family is huge and we could not traverse all popular applications. The famous AlphaGo Zero is also another version of GAN. In the future we would like to explore other styles of GAN as well as techniques to improve GANs such as: noise injection, cycling multiple models, growing from smaller images, and Wasserstein loss.

References

1. Ian J. Goodfellow, Generative Adversarial Networks, https://arxiv.org/abs/1406.2661
2. Tensorflow Team, DCGANs tutorial, https://www.tensorflow.org/tutorials/generative/dcgan
3. Nagesh Singh Chauhan, Noise GANs example, https://www.kaggle.com/code/nageshsingh/generate-realistic-human-face-using-gan
4. Sayak, DCGANs example, https://www.kaggle.com/code/sayakdasgupta/fake-faces-with-dcgans/
5. Himanshu Nayal, DCGANs example,https://www.kaggle.com/code/himanshunayal/human-face-generation-dcgans/notebook
6. Alec Radford, Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, https://arxiv.org/abs/1511.06434
7. Tero Karras, A Style-Based Generator Architecture for Generative Adversarial Networks, https://arxiv.org/abs/1812.04948
8. Soon-Yau Cheong, StyleGANs tutorial, https://keras.io/examples/generative/stylegan/
9. Xintao Wang, Towards Real-World Blind Face Restoration with Generative Facial Prior, https://arxiv.org/pdf/2101.04061v2.pdf
10. Vladimir Bok, GANs in Action, https://www.manning.com/books/gans-in-action