Emoji Generation using GANs

A Deep Learning Project

Introduction

Figure 1: A depiction of Memoji

Memojis (or emojis) have become a significant part of our daily communication. They are a fun way to express our emotions and feelings. They also are a significant product of companies like Apple, Samsung and Google.

The challenge with creating Memojis is that they are unique to each person. This makes it difficult to create a general model that can generate Memojis for everyone. However, we can create a model that can generate Memojis that look like the actual person. They can resemble the person in terms of their hair color, skin tone etc.

For this deep learning project, we will use a Generative Adversarial Network (GAN) to generate Memojis. We will use a dataset of images of people's faces to train the GAN. The GAN will learn the features of the faces and generate Memojis that look like the faces in the dataset.

What is a GAN?

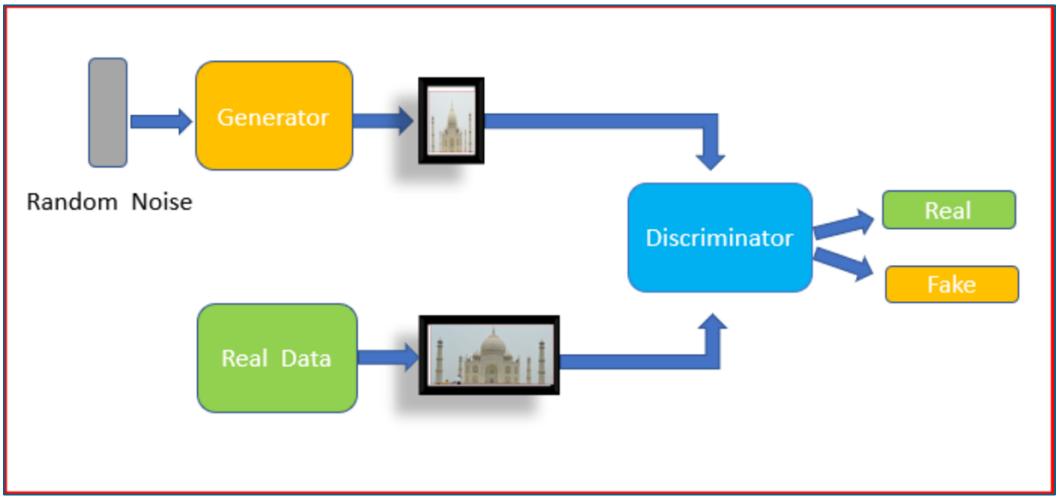

Figure 2: A general architecture of GAN

A Generative Adversarial Network (GAN) is a type of deep learning model that is used to generate new data samples. It consists of two neural networks - a generator and a discriminator. The generator generates new data samples, while the discriminator tries to distinguish between real data samples and generated data samples. The two networks are trained together in a competitive setting, where the generator tries to fool the discriminator, and the discriminator tries to detect the generated data samples.

The concept was initially developed by Ian Goodfellow and his colleagues in June 2014. The core idea of a GAN is based on the "indirect" training through the discriminator, another neural network that can tell how "realistic" the input seems, which itself is also being updated dynamically. This means that the generator is not trained to minimize the distance to a specific image, but rather to fool the discriminator. This enables the model to learn in an unsupervised manner.

Model Architecture

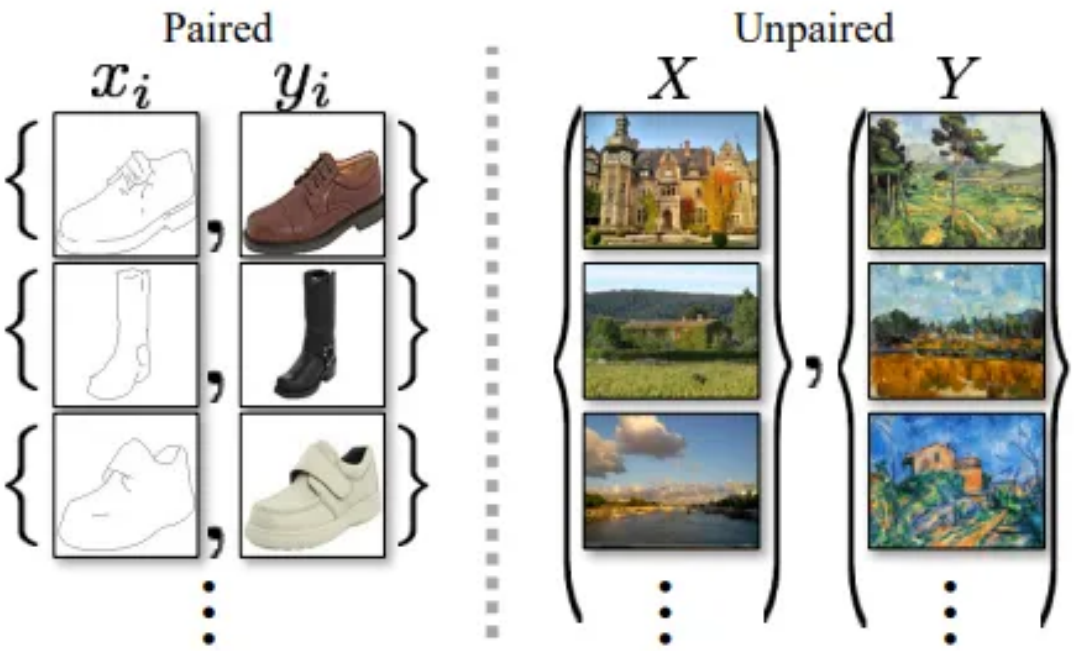

Figure 3: Inspiration for CycleGAN

The GAN model consists of two neural networks - a generator and a discriminator. The generator takes random noise as input and generates new data samples, while the discriminator takes data samples and tries to distinguish between real data samples and generated data samples.

In this project, we will try to use the paper CycleGAN to generate Memojis. CycleGAN is a type of GAN that is used for image-to-image translation. It can learn to translate images from one domain to another without paired examples. This means that we can train the GAN on images of human faces and generate Memojis that look like the faces.

There are 2 Generators and 2 Discriminators in CycleGAN. The generators are responsible for translating images from one domain to another, while the discriminators are responsible for distinguishing between real and generated images.

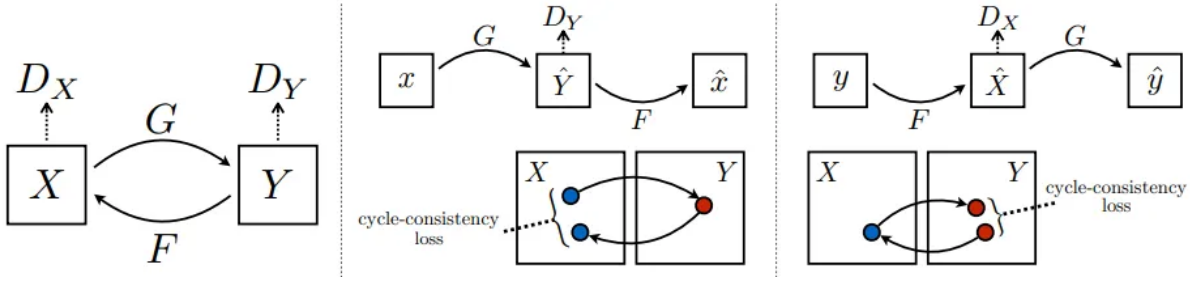

Figure 4: Working of CycleGAN

Although for our use case, we only need one generator and one discriminator, we will still use the architecture of CycleGAN to generate Memojis.

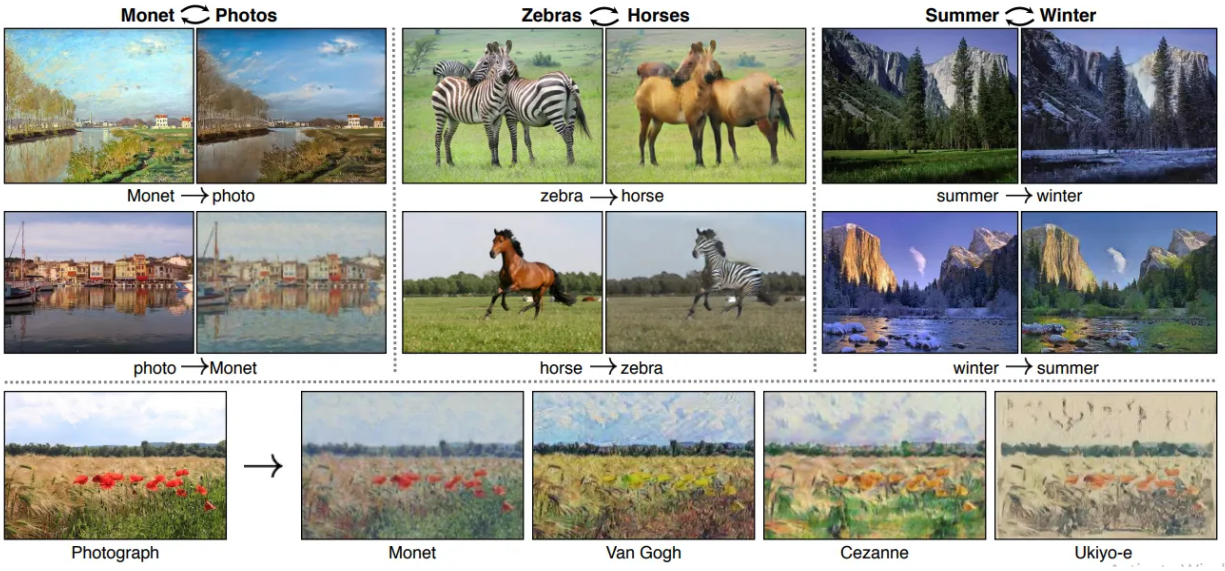

Figure 5: A working example of GANs

Dataset

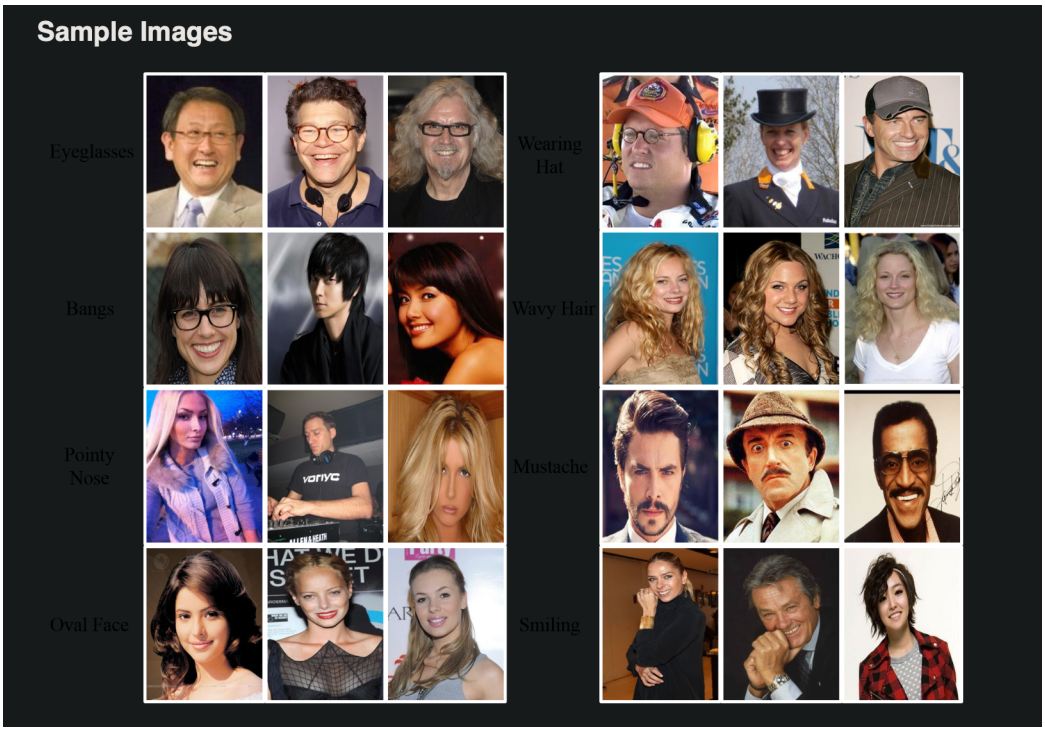

Figure 6: Sample Images of CelebA Dataset

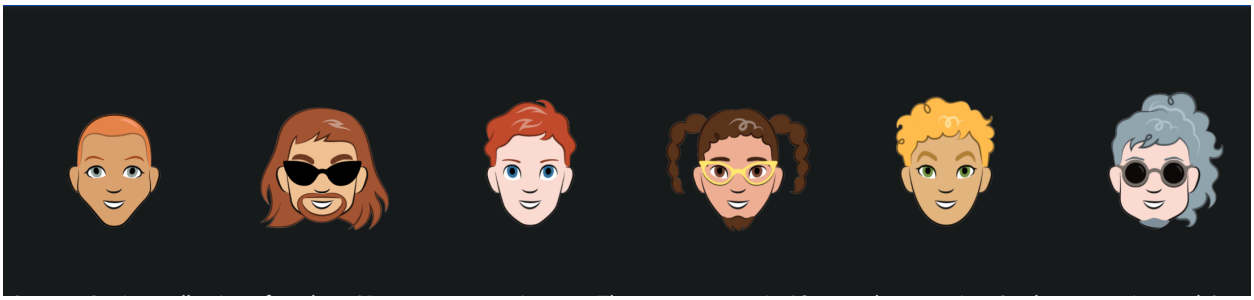

Figure 7: Sample Images of Cartoonset

Two datasets are being used for this project. They are:

- **CelebA Dataset**: This dataset contains over 2,000,000 celebrity images with annotations. We will use this dataset to train the GAN to learn the features of human faces.
- **Cartoonset**: This dataset by Google contains over 10^{13} images of emojis that each represent various features of a person. This dataset was indeed used a build a ML model for **Google Allo**.

Data Preparation

- The model is implemented as is from the paper and let learn the features of the faces and generate Memojis that look like the faces. CelebA dataset is used to train the GAN and generate Memojis.
- For preprocessing this dataset, the images are resized to 475x475 and added some padding.
- For the Cartoonset dataset, a Center Crop of the images to size of 475x475 is applied.

Loss Function and Training

The CycleGAN model uses two loss functions - the adversarial loss and the cycle-consistency loss:

- Adversarial Loss: The adversarial loss is used to train the generator to generate realistic images.

$$\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))]$$

Figure 8: Adversarial Loss

- Cycle Loss: The cyclic loss is used to ensure that the generated images are consistent with the inputs.

$$\mathcal{L}_{\text{cyc}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].$$

Figure 9: Cyclic Loss

Learnings and Conclusion

1. Most basic cloud offerings don't work. Be ready to spend money. When working on scale, it is better to optimize for money or use cloud-managed services.
2. Don't write the code for multi-device logic. Use tools like Lightning AI to modularize your code. Lightning AI takes care of moving between devices and other logic, allowing you to focus on the core logic.
3. Keep monitoring your metrics. This helps to save money and preempt training thus reducing your experiment time. Neptune is a good option to monitor your experiments.
4. Keep your image size and batch size in mind. If your image size is too large (e.g. 1024 x 1024), then your batch size will be limited to 1, which is very inefficient use of compute and storage.

To conclude, a complete model couldn't be trained for the limitations of resources and other constraints. But the process of building this project proved to be invaluable and taught a lot of things that go into building a machine learning model in a professional and academic setting.

Sample Outputs

Human Image

Generated Cartoon Image