ABSTRACT:

This report documents the development and deployment of Generative Adversarial Network (GAN) to generate unseen Pokemon faces using a Kaggle dataset. The model was trained for 450 epochs, and the final results were deployed via Streamlit . The project demonstrates the application of deep learning techniques for synthetic image generation and practical deployment using a lightweight web framework. Key outcomes include successfully generating novel Pokémon faces and creating an interactive interface for users to explore the results.

INTRODUCTION:

The goal of this project was to explore the capabilities of GANs in generating synthetic images of Pokémon faces that do not exist in the original dataset. GANs, composed of a generator and a discriminator, are ideal for this task due to their ability to learn data distributions and produce realistic outputs. The project involved:

- Curating and preprocessing a Pokémon image dataset.
- Designing and training a GAN architecture.
- Evaluating the model's performance over 400 epochs.
- Deploying the trained model via Streamlit for user interaction.

DATASET AND PREPROCESSING:

Dataset Source:

We will be using 3 datasets totalling approximately 29000 images

1. **Pokemon_V2 Dataset (kaggle)**
   9002 images from kaggle hub dataset (pokemon v2 , containing augmented images of original 813 images form v1 dataset)
   Link: https://www.kaggle.com/datasets/rajatvisitme/pokemon-image-dataset-v2
   the original 813 images are acquired from https://veekun.com/dex/downloads

2. **Pokemon Generation One - 20,100 Gen 1 Pokémon (kaggle)**
   20000 images being a combination of multiple existing datasets - All 151 Pokemon sorted by name

   Link: https://www.kaggle.com/datasets/bhawks/pokemon-generation-one-22k/data

Preprocessing Steps:

To process the dataset links, we have creating a pokemon subset class that load only Pokémon subfolders from a root directory (for github dataset) since 7k and 20k datasets are from kaggle, we simply used the kaggle api to call them. In the case for the github dataset, it was extracted into our cloud folder where it will be extracted.

- Load the dataset: We will be combining all of them
- Dataloader: we will be using the dataloader to feed pokemon images into the dcgan model

METHODOLOGY:

GAN Architecture:

The model leverages a deep convolutional GAN (DCGAN) architecture with 5 convolutional layers in the discriminator and transposed convolutional layers in the generator. Below is the detailed breakdown:

Generator

Input: 100-dimensional noise vector (latent space).
Layers:
Dense layer: Expands noise vector to 8x8x512.
Reshape to 8x8x512.
Transposed Convolutional Blocks (4 layers):
Each block: Conv2DTranspose → BatchNorm → ReLU.
Upsampling: Doubles spatial dimensions at each layer (8x8 → 16x16 → 32x32 → 64x64).
Final layer: Conv2DTranspose with 3 filters, tanh activation (output: 64x64x3 image).

Discriminator

- **Input:** 64x64x3 RGB image.
- **Layers:**5 Convolutional Blocks:
- **Each block:**Conv2D (kernel: 5x5, stride: 2) → BatchNorm → LeakyReLU (α=0.2).
- **Filter progression:** 64 → 128 → 256 → 512 → 512.
- **Downsampling:** Halves spatial dimensions at each layer (64x64 → 32x32 → 16x16 → 8x8 → 4x4).
- Flatten layer.
- Dense layer with sigmoid activation (real/fake classification).

pGAN Architecture:

The progressive GAN (pGAN) was used as well.
We gave random noise (fixed dimension through pytorch) as input , it basically has a function that will add layers during training time, and take weighted sum of the weights to pass onto for the next epoch.

Training Process:

Epochs: 450

Hardware:
Software: Google Collab

RESULTS AND ANALYSIS:

DEPLOYMENT USING STREAMLIT:

User Interface: A minimalistic, user-friendly interface with a button to generate new Pokémon faces.
Real-Time Generation: Instant display of outputs using Streamlit's st.image and caching (@st.cache) for efficient model loading.
Hosting: Locally deployed with potential for public sharing via Streamlit Community Cloud.

TOOLS:

Frontend/Backend: Streamlit (unified framework for UI and logic).
Integration: Generator model embedded directly into the Streamlit script.

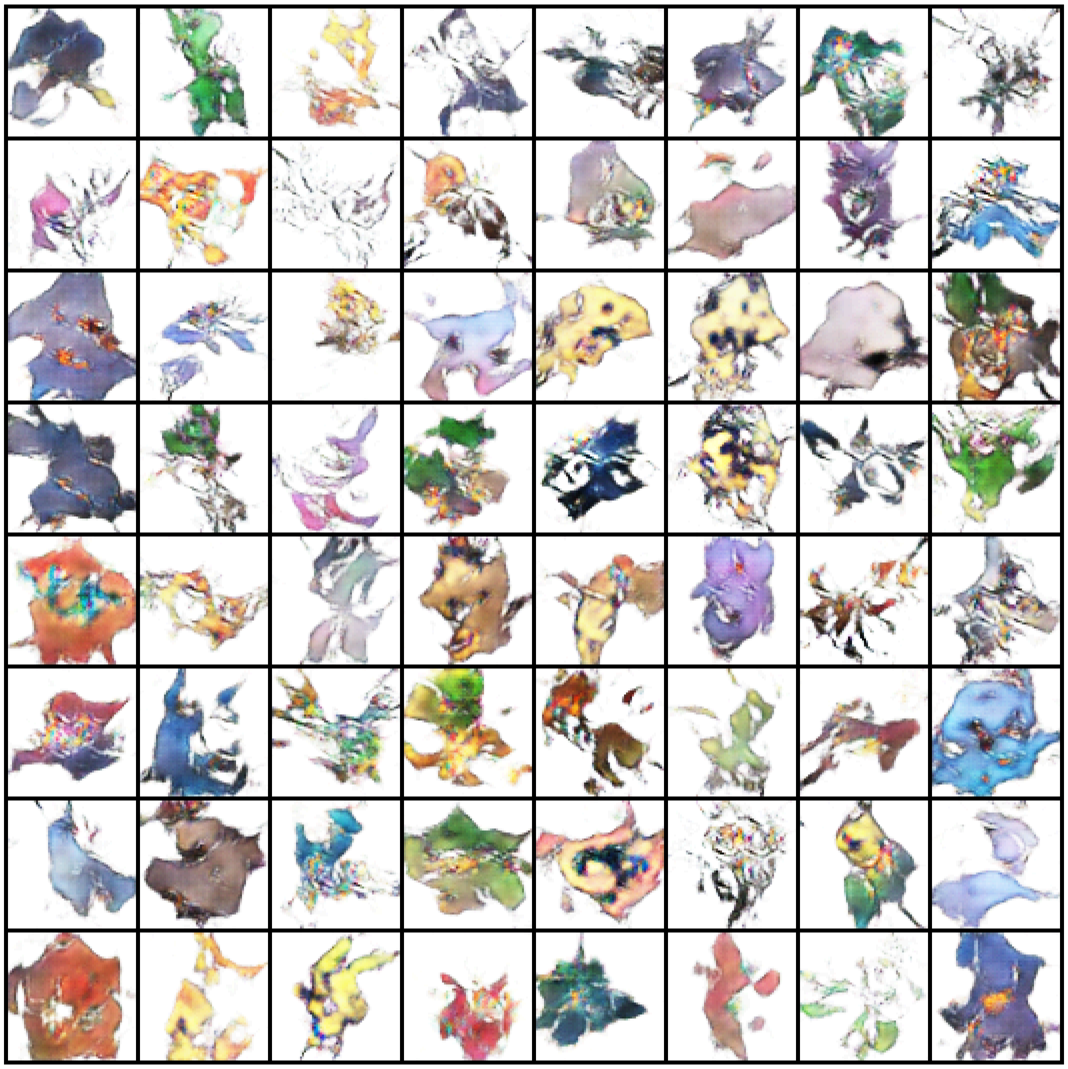
RESULTS:

Training logs:

These are the original images from the dataset we used.We used 3 datasets which included 36k images in total.

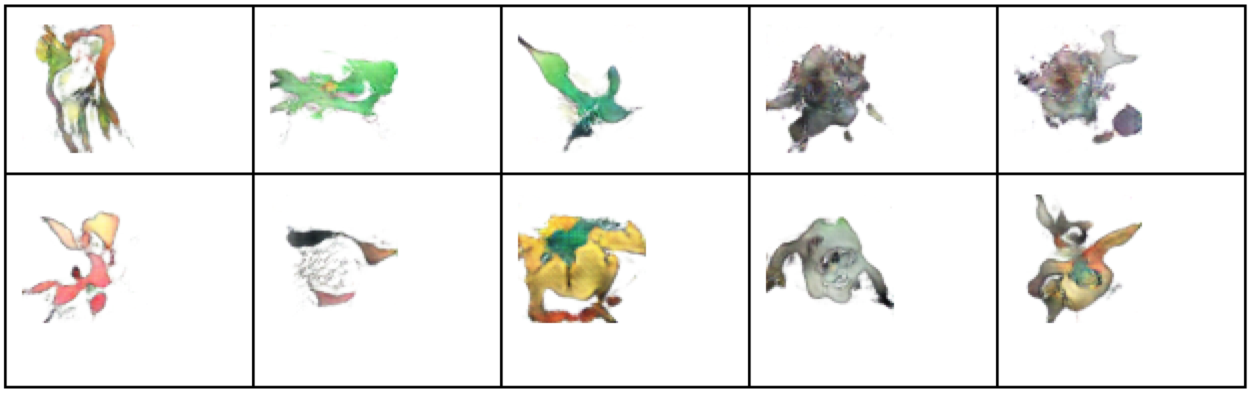
Intermediate results:

These are the intermediate results we got after 50 epochs.



Generated images:

These are the final results.We achieved these results(images) after 450 epochs.



CONCLUSION:

This project successfully demonstrated the use of GANs for unseen Pokemon face generation.We can further enhance these features given larger dataset and more computational power and resources.