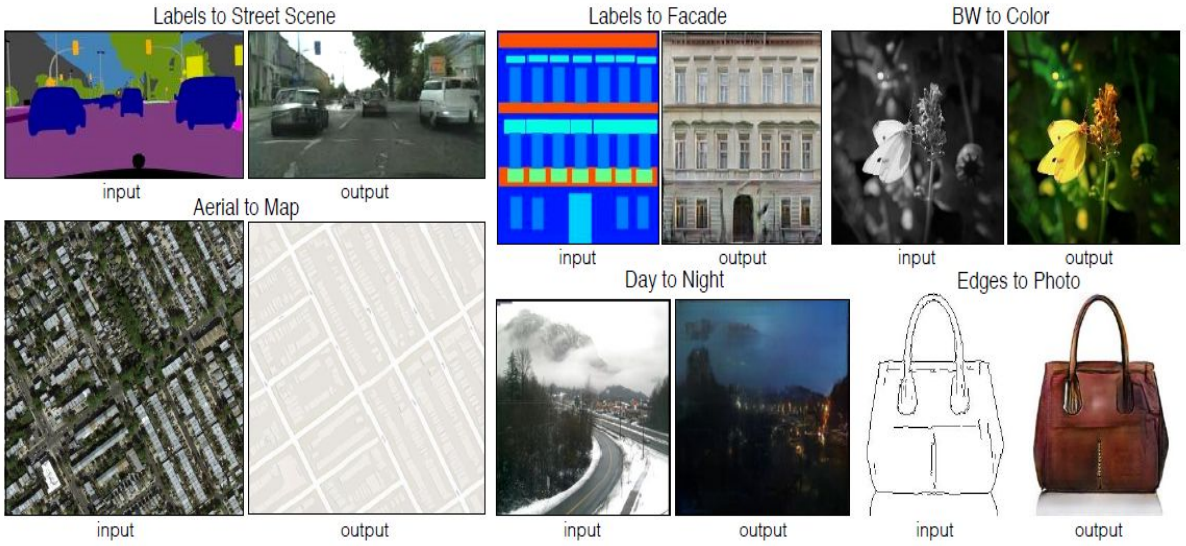
Project: Update Report II

Image to Image translation using cGANs

Mentor: Dr. Pawan Kumar
Aman Krishna - 2018201070
Gyanshu Azad Singh - 2018201073



Introduction:

We have implemented a Deep Convolutional Generative Adversarial Network using Tensorflow for creating new Pokemons using a Pokemon dataset. The report includes:

- Working of a Generative Adversarial Network
- Generator function
- Discriminator Function
- Result obtained

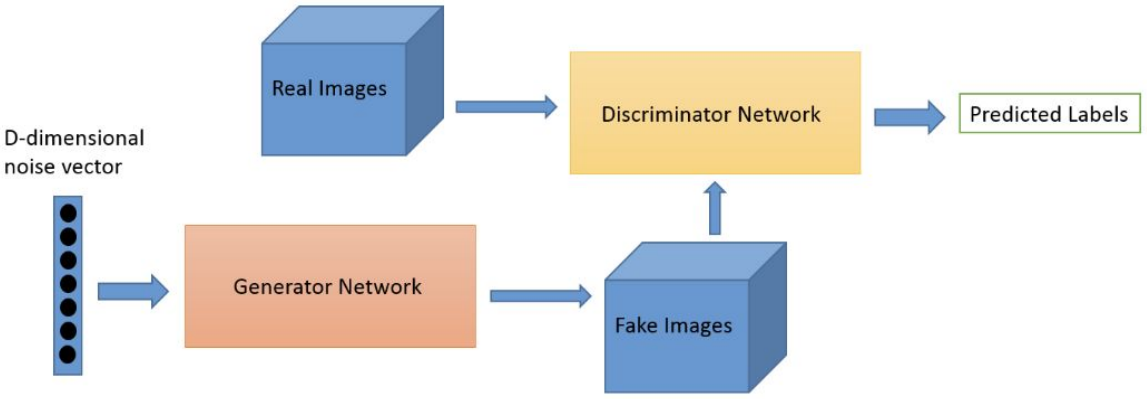
GANs (Generative Adversarial Networks)

A GAN has 4 components:

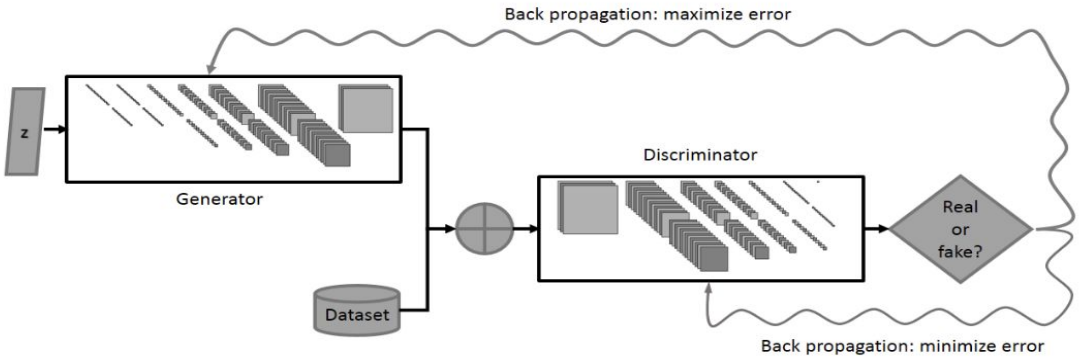
- Generative Network: In traditional GANs the generative model learn a mapping from random noise vector ***z*** to output image ***y***

$$G: z \rightarrow y$$

- Discriminator Network: The goal of the discriminator network is to distinguish real images from the fake images (provided by the generator)



- Dataset: This contains the real image dataset used to train the Discriminator network
- Random Noise: The random noise that goes into the generator as a source of entropy



Objective Function:

- The following is the objective function for the network

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Objective Function

- Since a game-theoretic approach is taken, our objective function is represented as a minimax function
- The discriminator tries to maximize the objective function, therefore we can perform gradient ascent on the objective function

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

Gradient Ascent on Discriminator

- The generator tries to minimize the objective function, therefore we can perform gradient descent on the objective function

$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Gradient Descent on Generator

- By alternating between gradient ascent and descent, the network can be trained

Pokemon Generation using GAN:

- We used Tensorflow to create a GAN to create new pokemons
- The real pokemon dataset contains around 100 pokemons

- The dataset was preprocessed to resize the images to 128x128x3 size
- Number of channels used for each convolutional layer of the Generator network is

  c4, c8, c16, c32, c64 = 512, 256, 128, 64, 32

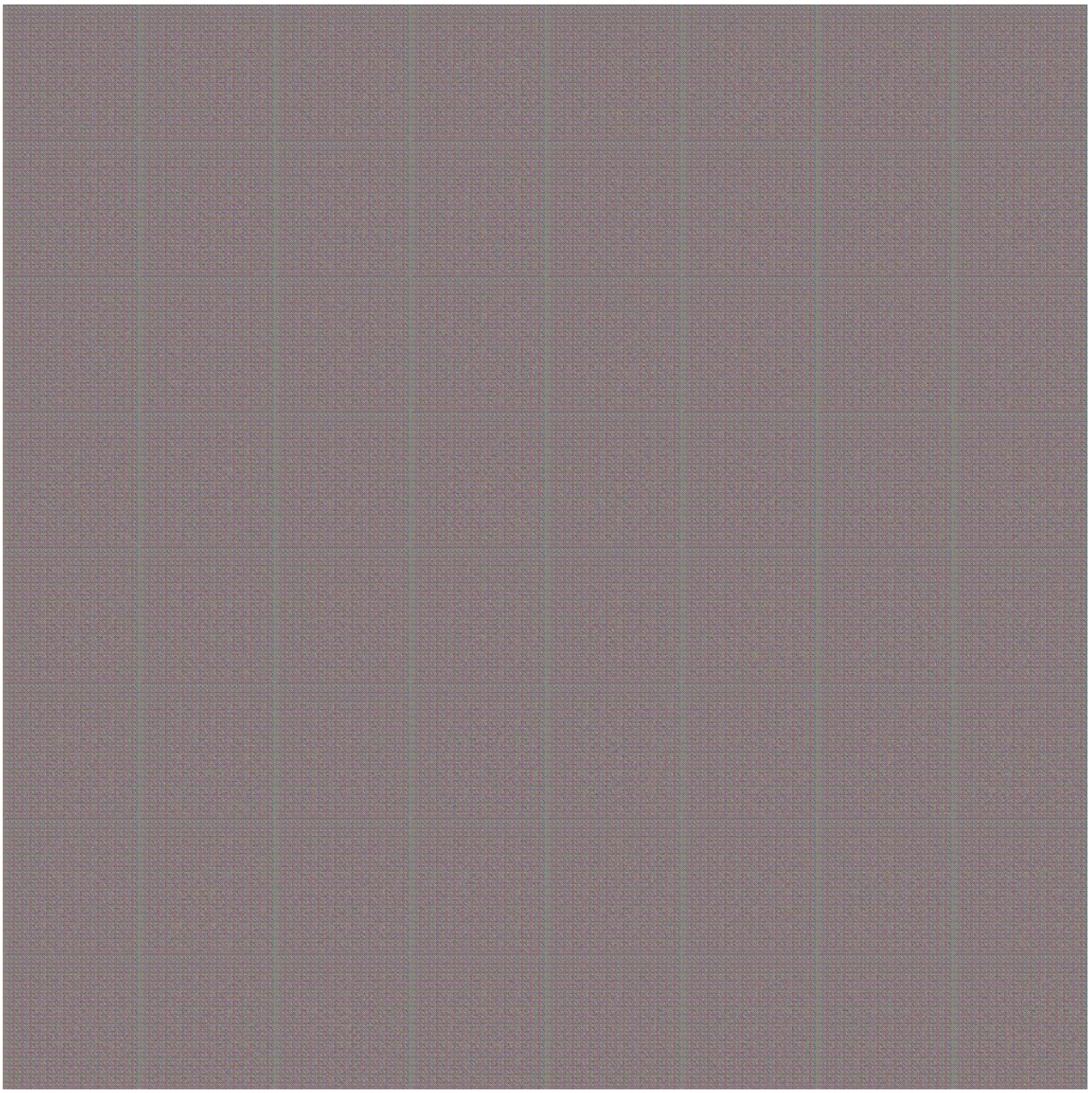
- The same for the discriminator is

  c2, c4, c8, c16 = 64, 128, 256, 512

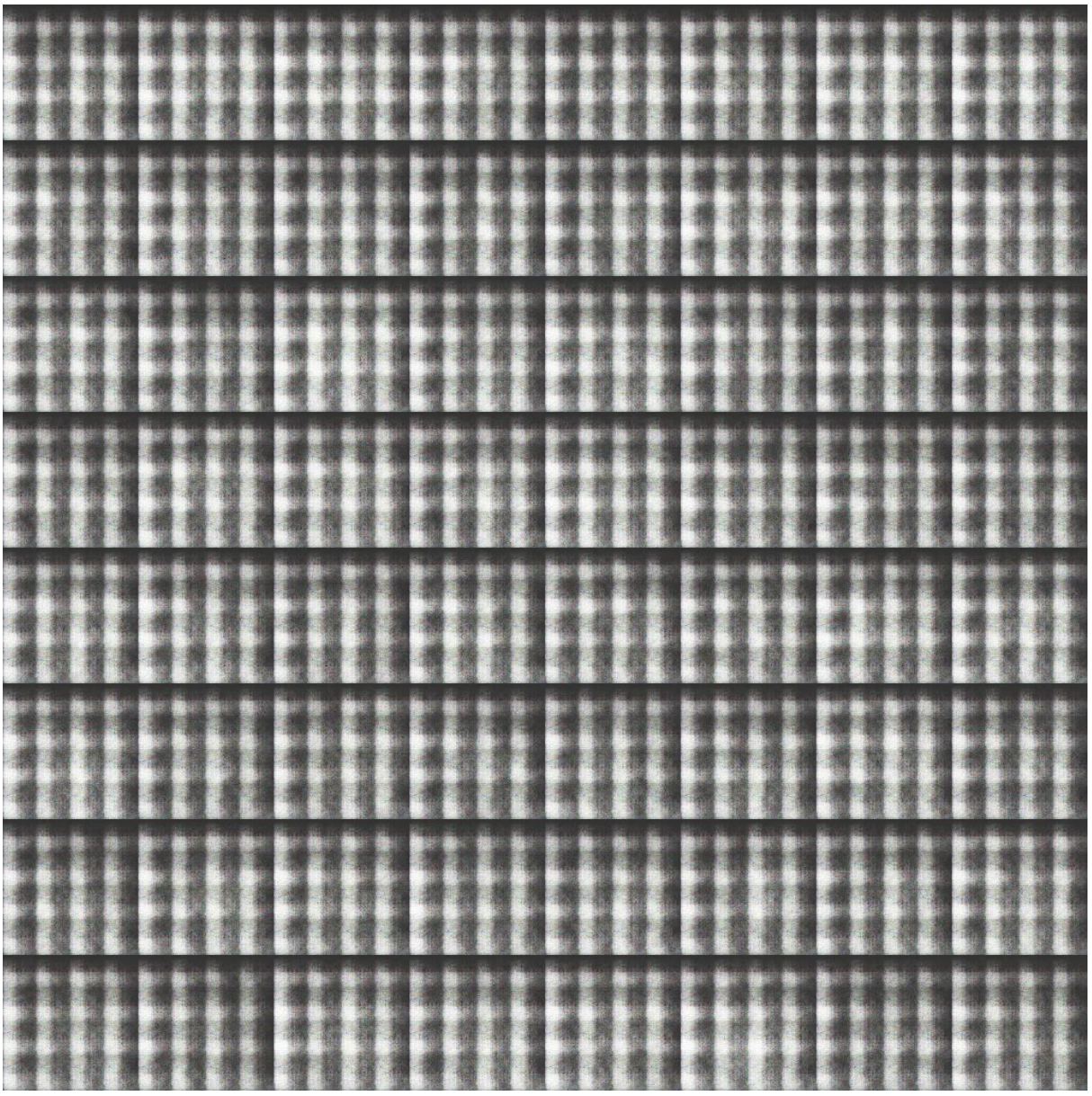
- The Discriminator is trained more than the Generator to obtain better results

Results:

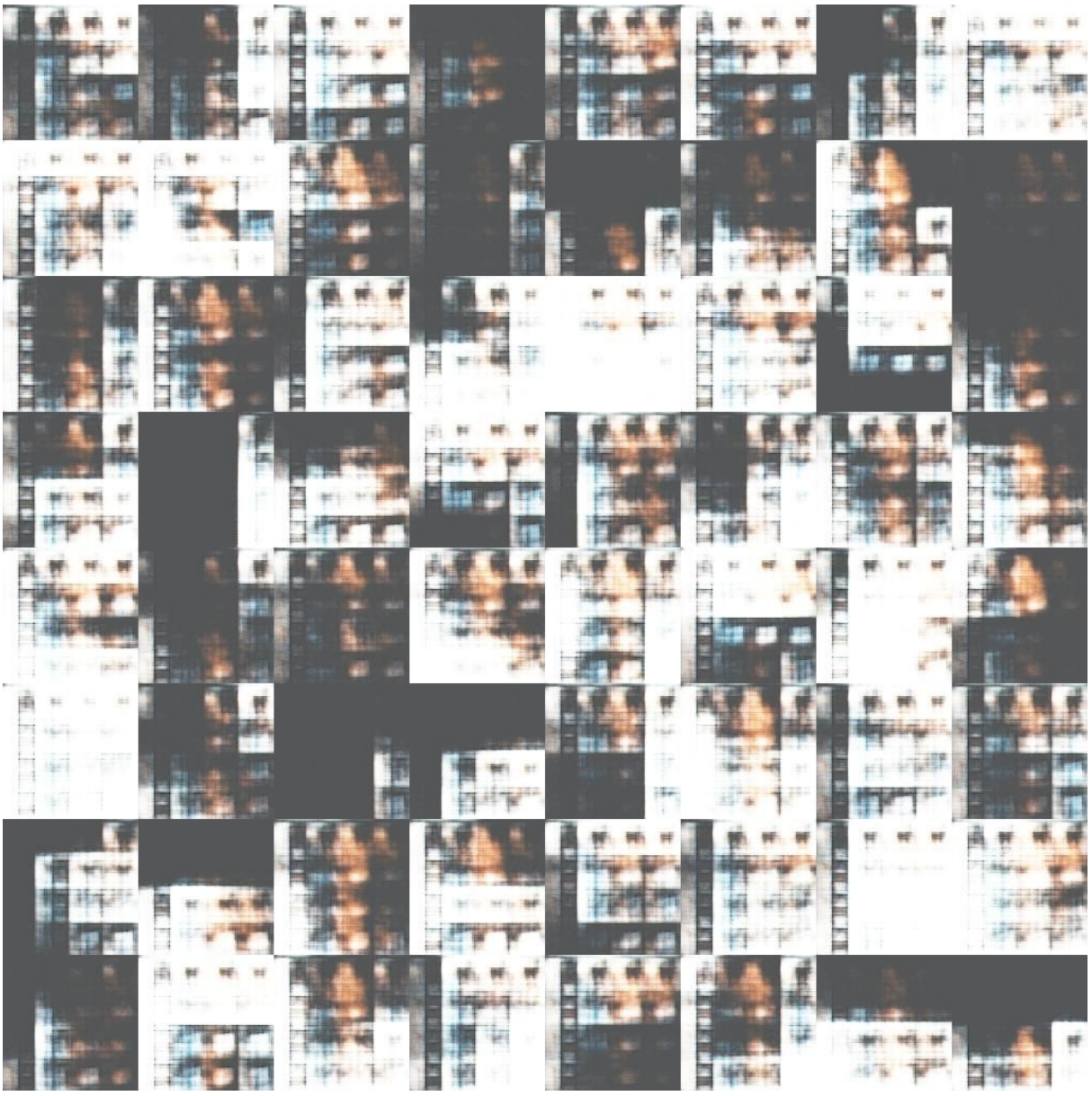
- **After 1 epoc**



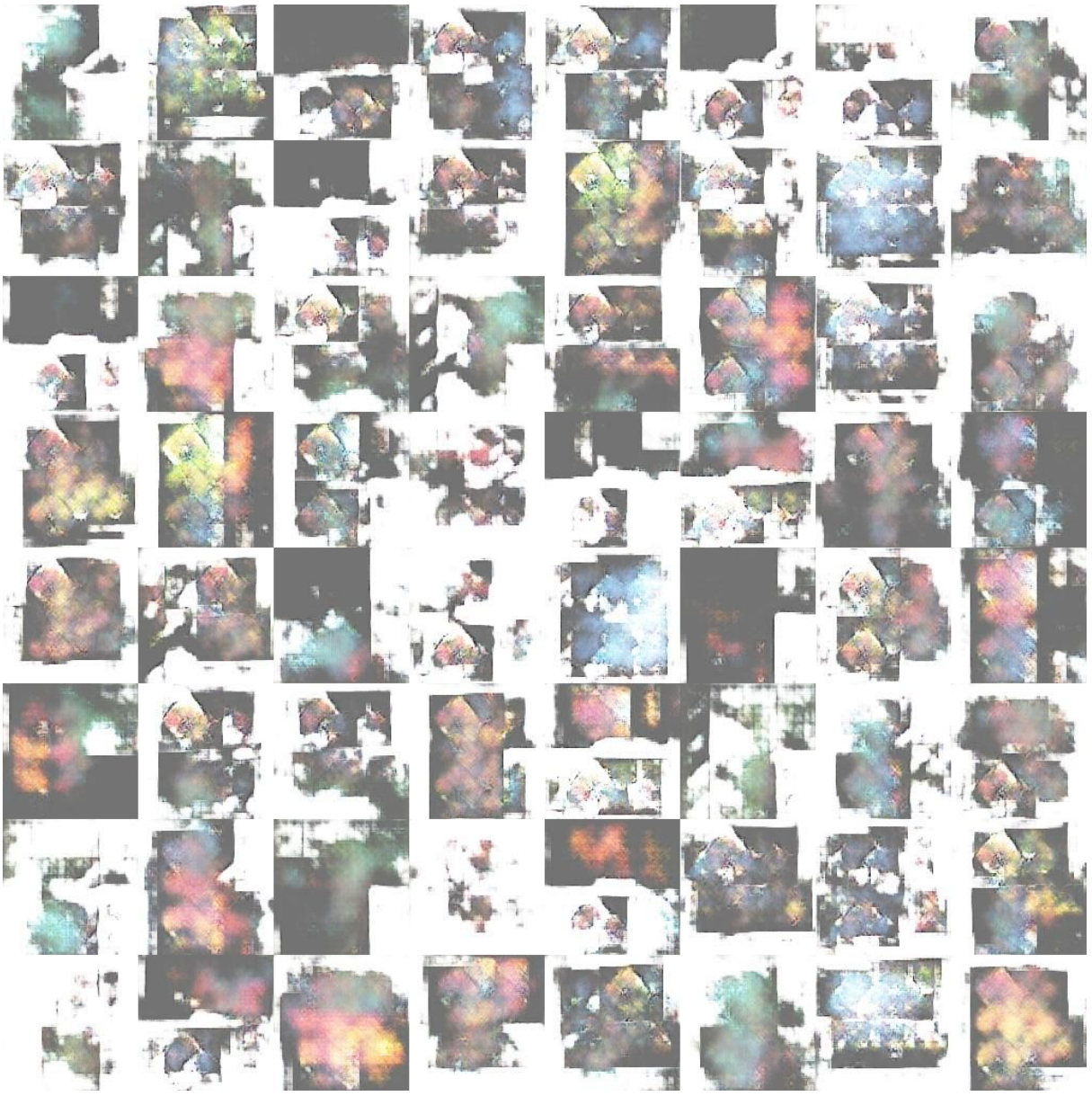
- **After 100**



- **After 500**



- **After 2500**



- **Final results**

Next Steps:

- The next step of the project will be to build a Conditional Generative adversarial network mentioned in the paper and obtaining similar results