PHOTO RECONSTRUCTION CHALLENGE

Final Report

Group-33

GROUP MEMBERS

Aman Sharma	-	20CS30063
Dasari Giridhar Ram Chand	-	20CS10022
Lathiya Ronak Rameshbhai	-	22BM6JP24
Siddhant Shamrao Shambharkar	-	22CS60R78

1. Project Overview

Image reconstruction is an important task in computer vision, where the goal is to fill in missing or damaged parts of an image. This will help in reconstructing old images, and help with images having low resolution. The challenge is to build models that can reconstruct animal images that have missing sections caused by degradation or network issues. The task involves predicting the missing pixels in the images, given information about the masks' location.

The project offers different methodologies to approach the problem, including pixel reconstruction with Bidirectional LSTM, Generative Adversarial Networks, and Stable Diffusion/CLIP Embeddings. Pix2Pix, CycleGAN, and DCGAN are some of the generative models that can be used in this project. The Pix 2 Pix model is well-suited for this use case.

The dataset for this deep-learning assignment consists of images of four different animals: cats, dogs, elephants, and tigers. Each image in the dataset has dimensions of 256x256 pixels thus no resizing was necessary as such during preprocessing. Additionally, each image contains two masked areas, which are represented as 75x75 boxes. The goal of this assignment is to predict the output for these two masked areas in each image.

The dataset is organized into several sub-folders, each corresponding to one of the four animal categories. Within each sub-folder, there are multiple image files in the JPEG/JPG/PNG format, each with a unique filename. The details of the bounding boxes for each image are given in a csv file. To improve the robustness of the deep learning model on this dataset, several data augmentation techniques were applied. These techniques include random cropping, random jitter, and random flipping of the images.

2. Challenges

There are some challenges in this project, including:

1. **Compute availability:**

   Since the image size was relatively large, getting compute resources to train models on it was a challenging task. Colab and kaggle have limits on the free users in the amount of usage and thus in the end we had to resort to training the model on local machines' GPUs for hours on end. Setting up CUDA for training locally was also a challenge.

2. **Non-convergence of Generative Models**:

   Generative models are notorious for not converging and being difficult to train. Thus setting the right loss parameters and testing out for different batch sizes was crucial in getting semi-decent-looking results. A lot of our training runs resulted in node collapse and did not produce good results.

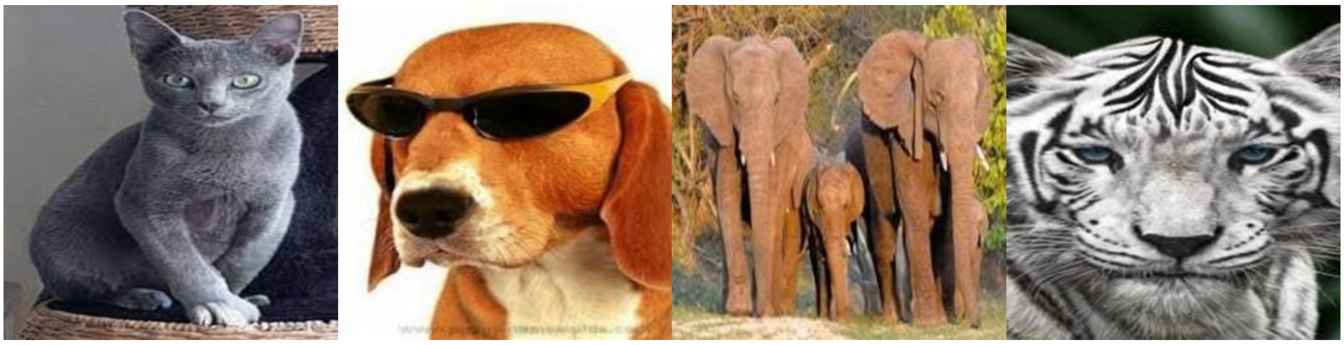
3. Models used/tried

Stable Diffusion:

Stable Diffusion is a generative model that uses a sequence of diffusion steps to synthesize images. It is a variant of the well-known diffusion process that models image generation as the iterative process of adding noise to an image and gradually reducing the level of noise. By performing a sequence of such steps, the model can generate highly realistic images with a range of different styles and features.

We used a PyTorch implementation of stable diffusion and used a pre-trained model to synthesize outputs. We generated image masks using the information provided and the image masks and image themselves were both resized to 512x512 as that is the data the stable diffusion model was trained on. The output was then resized back to 256x256 and then stored.

The model produced excellent results, with very real-looking images however the Kaggle score was worse than the pix2pix model we trained later. We used different prompts like "Real image of an <animal_name>, no text" and "Natural looking seamless image, no text" among others to get results, the latter of which turned out to be better. No text was required as sometimes it added text into the masked regions and adding no text reduced such instances and improved the quality of outputs. Selective images generated are attached below

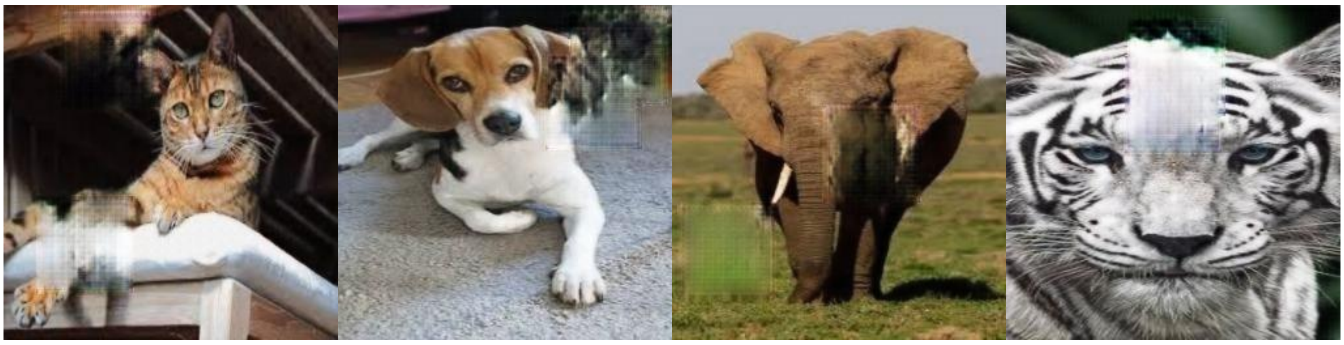


Pix2pix:

We ended up using pix2pix model, which is a state-of-the-art conditional generative adversarial network (GAN) model that is capable of learning a mapping between input and output images. In the case of this particular dataset, the input images contain two masked areas, and the corresponding output images contain those same masked areas but with the predicted animal type.

The Pix2pix model is comprised of two main components: a generator network and a discriminator network. The generator network is an encoder-decoder architecture with skip connections. The encoder network compresses the input image into a lower-dimensional representation, while the decoder network generates the output image by upscaling this representation. The discriminator network is a CNN that takes in pairs of images, either a real input/output pair or a generated input/output pair, and determines whether the pair is real or fake. The discriminator is trained to correctly classify the input/output pairs, while the generator is trained to produce output images that fool the discriminator into thinking they are real.

Upon training for 6.5 hours for 50 epochs with a batch size of 1 on a 4GB Nvidia GeForce GTX 1650Ti GPU and using 5600 images to train on, the model ended up giving a score of 0.2033 on Kaggle, which we submitted on the 16th but is the best score we achieved. The output images from the pix2pix model are shown below.



4. Conclusion

Upon manual inspection, it can be easily observed that the outputs from stable diffusion are more consistent and look more real, with the masks being perfectly filled in whereas the pix2pix model usually generates distorted or different coloured pixels in the masked areas. Thus stable diffusion would be the go-to choice for real-world implementations of image inpainting. However as far as the score on Kaggle goes, since we are taking RMSE of pixel values, pix2pix generates better mappings in those terms and thus provides the “better” model for this project.

The code is provided in the same zip file as this report, and contains notebook used for stable diffusion as well as code for pix2pix. The instructions to run the code are in a README.md file, along with the link of the drive where the model is stored.