

---

---

# Project Presentation

## Context Encoders: Feature Learning by Inpainting

Abhisek Mohapatra : 2020201020

Parth Arora : 2020201054

Paras Joshi : 2020201028

---

# Overview

**Date of Submission**

April 30, 2021

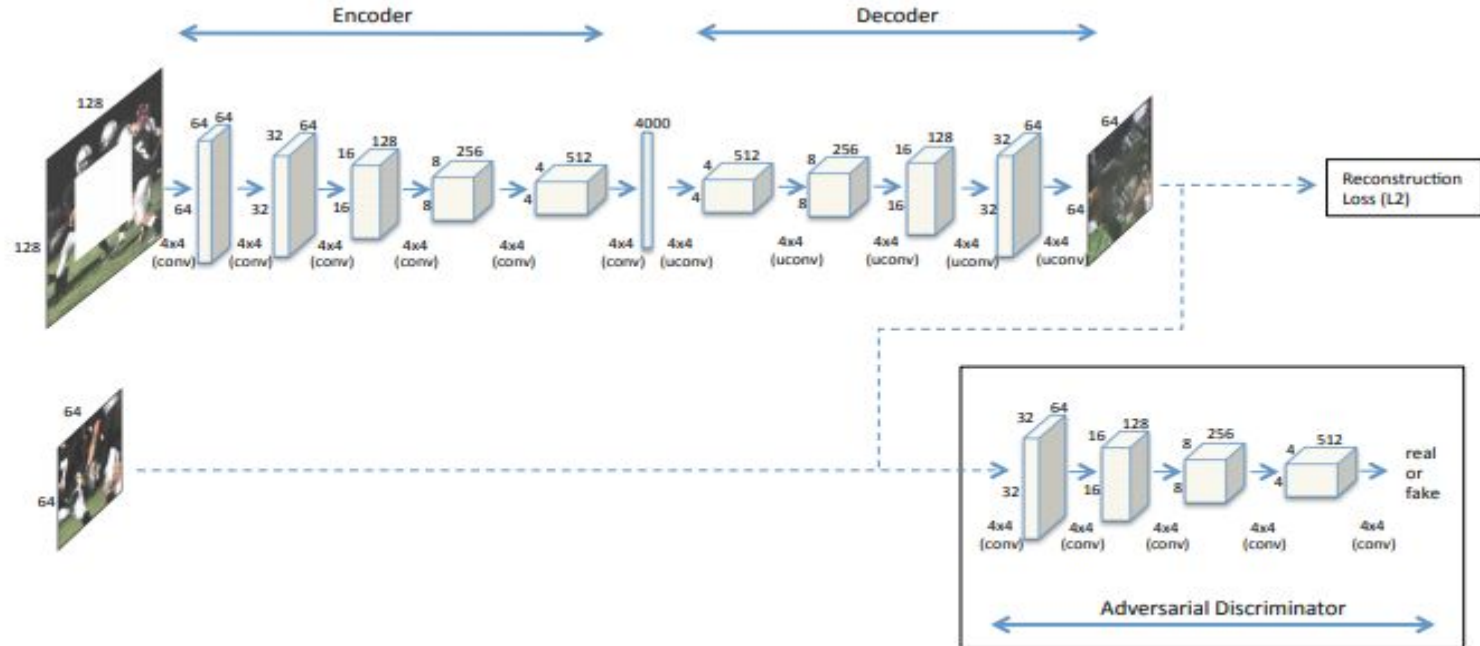
---

---

# Understanding the Problem Statement :

- We are given a set of complete coloured images from which we mask the centre region from the original image.
  - The task is to build a model which takes the masked image as the input and reconstruct a complete coloured image via Context Encoder model.
  - Context Encoder makes use of Semantic Inpainting Concept to reconstruct the original input image and the idea is to explore the semantic embeddings in the image.
  - We make use of 2 Loss Functions : Reconstructive Loss and Adversarial Loss and Generator and Discriminator networks to train the model.
-

# Basic Architecture of Context Encoder



(a) Context encoder trained with joint reconstruction and adversarial loss for semantic inpainting. This illustration is shown for *center region dropout*. Similar architecture holds for arbitrary region dropout as well. See Section 3.2.

---

# Function of the architecture

- The Generator model emulates the encoder-decoder pipeline concept similar to Autoencoder architecture.
  - The context image is passed through the encoder to obtain features which are connected to the decoder.
  - The encoder first takes an input image with missing regions and produces a latent feature representation of that image.
  - Then the decoder takes this feature representation and regenerates image pixels in the missing regions of the image.
-

---

# Encoder and Decoder

**Encoder** -- Encoder architecture is derived from AlexNet architecture. In the present architectures, this information propagation is handled by fully connected or inner product layers, where all the activations are directly connected to each other. This is so because convolutional layers connect all the feature maps together, but never directly connect all locations within a specific feature map.

**Decoder**-- The channel-wise fully-connected layer is followed by a series of five up-convolutional layers with learned filters. The intuition behind this is straightforward – the series of up-convolutions and non-linearities comprises a non-linear weighted upsampling of the feature produced by the encoder until we roughly reach the original target size.

---

---

# Loss Functions

**Loss Function** -- We train our context encoders by regressing to the ground truth content of the missing (dropped out) region. We model this behavior by having a **joint loss function** to handle both continuity within the context and multiple modes in the output.

- **The Reconstruction loss** is responsible for capturing the overall structure of the missing region and coherence with regards to its context, but tends to average together the multiple modes in predictions. The AutoEncoder model is trained over this loss to regenerate the pixels in dropped out regions.

Reconstruction loss or Mean Square loss calculates the L2 distance between the original masked portion and portion generated by the model. It is used to penalize the model if generated image not seems to be close the original one.

- **The Adversarial loss** is actually binary\_crossentropy loss that tries to make prediction look real, and has the effect of picking a particular mode from the distribution. The Discriminator Model is trained over this loss so that the final output image reconstructed gets as close and real and sharp to the complete coloured image.

$$L = \lambda_{rec} L_{rec} + \lambda_{adv} L_{adv}$$

- We define the overall loss function as :
-

---

# Attention areas

## Task 1

- Our context encoder focuses on solving a particular task: to fill in large missing areas of the image, where it can't get "hints" from nearby pixels. This requires a much deeper semantic understanding of the scene, and the ability to synthesize high-level features over large spatial extents.

## Task 2

- Like autoencoders, context encoders are trained in a completely unsupervised manner. Our results demonstrate that in order to succeed at this task, a model needs to both understand the content of an image, as well as produce a plausible hypothesis for the missing parts.
-



---

# Dataset:

- The dataset that we have used is the **Scene classification** dataset from Kaggle which contains ~25K images from a wide range of natural scenes.
- <https://www.kaggle.com/nitishabharathi/scene-classification>



---

# Experiments:

## Data Pre-processing :

- Dataset is split into train and test datasets in ratio 80:20 .
  - Input Images are scaled to 128\*128 dimensions.
  - For every image input there exists three different types --
    - Real Image
    - Cropped Image
    - Reconstructed Image
  - Normalized data using mean=0.5 and standard deviation=0.5 over 3 input channels (rgb components) across all dimensions.
-

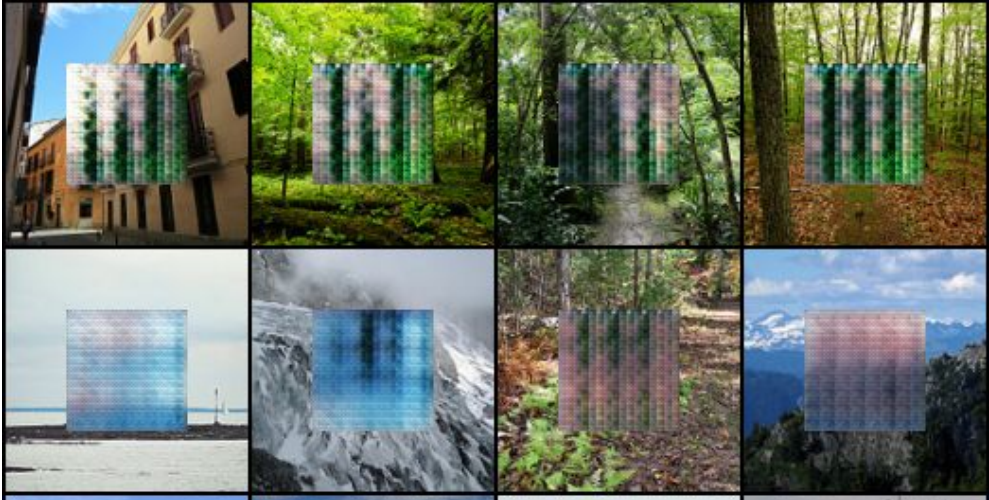
---

# Experiment cont..

- Vary learning rate to get the better output and less reconstruction loss.
  - Experimented with different optimizers such as Adam and SGD.
  - Tuned different hyperparameters of the respective optimizer.
  - We are creating the plot for epoch vs total loss of the model to get an idea about the reconstruction of the image formed.
-

---


## Case 1: Weight initialization: Random

l_rate	wt_recon	optimizer	result
0.001	0.9	Adam	

---

---

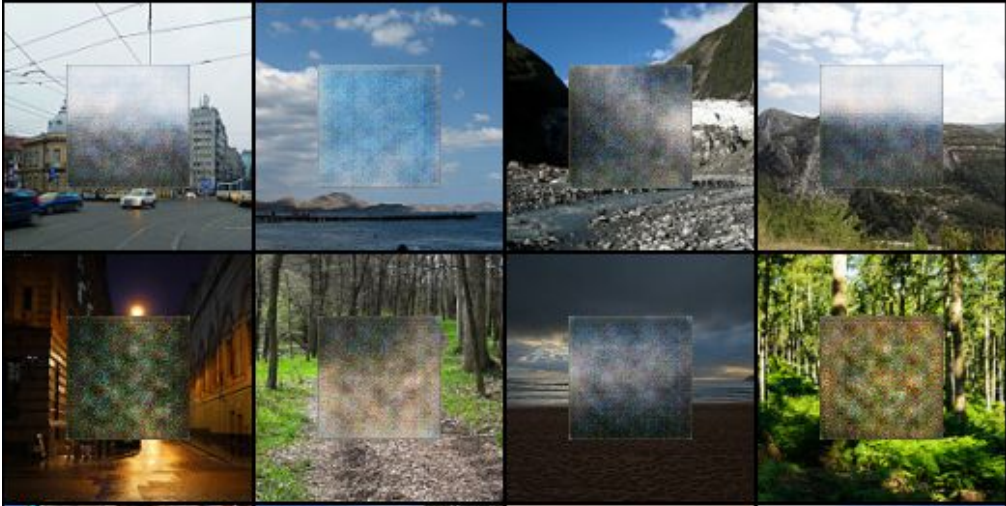
## Case 2: Weight initialization: Random

I_rate	wt_recon	optimizer	result
0.001	0.999	Adam	

---

---


## Case 3: Weight initialization: Normal

L_Dis	L_Gen	wt_r.	optim.	
0.0001	0.0005	0.999	<b>SGD</b>	

---

---

## Case 4: Weight initialization: Normal

L_Dis	L_Gen	wt_r.	optim.	result
0.0001	0.0005	0.999	Adam	

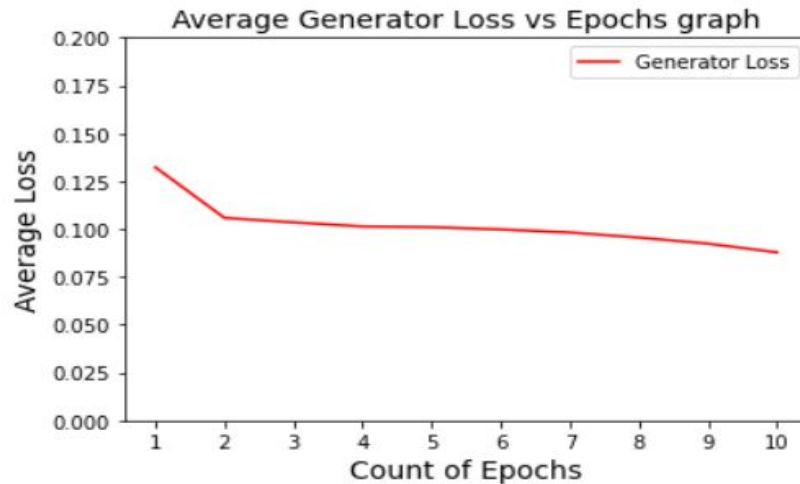
---



---

# Conclusion

- The model used in **Case 4** which uses **Adam** as the optimizer and **0.0001** and **0.0005** as the learning rates for Discriminator and Generator respectively.
- The filter weights in this model are initialised using the **Normal** method.
- The graph depicting the average loss of generator in the subsequent epochs is shown below.

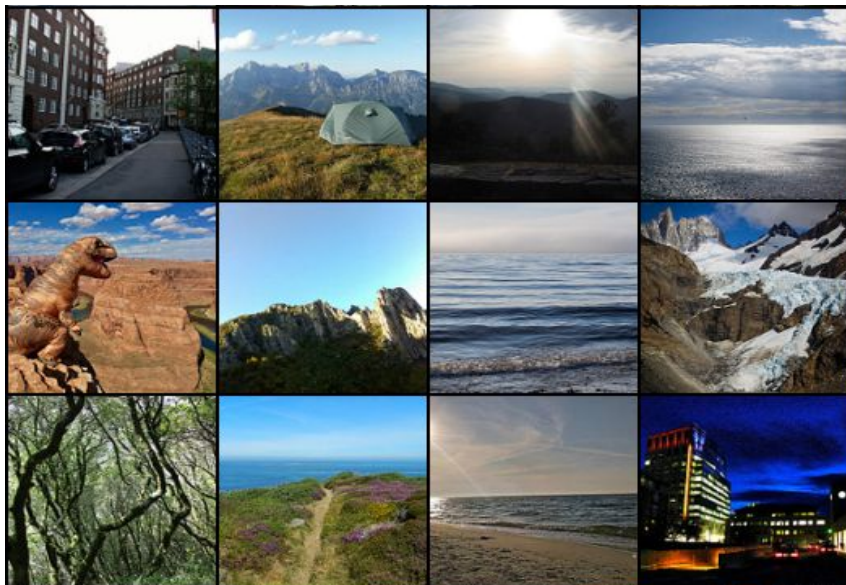




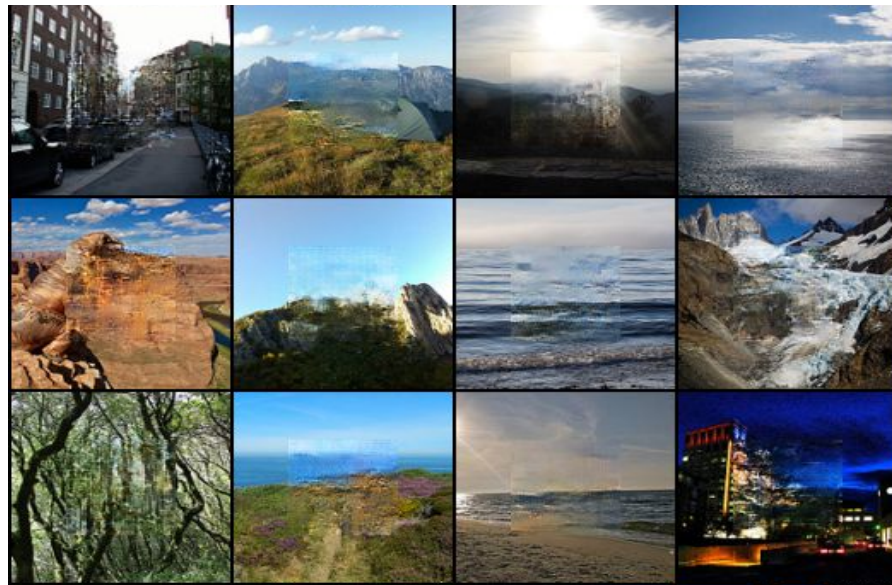
---

- Some more results for Case 4

Real Images



Reconstructed Images



---

# Individual Contribution

- **Paras Joshi :**

- Data preprocessing
- Training the Model
- Model Development

- **Parth Arora :**

- Model Development
- Training the Model
- Testing of the model

- **Abhisek Mohapatra :**

- Implementing Loss functions
- Training the model
- Hyper-parameters tuning & plotting Performance graph

**Thank You**

---