Image Inpainting Using Deep learning Techniques

CS 736 Course Project under Prof. Suyash Awate

By:- Dhananjay Singh (180050029)
Dhruva Dhingra (190070020)
Liza Dahiya (190050063)

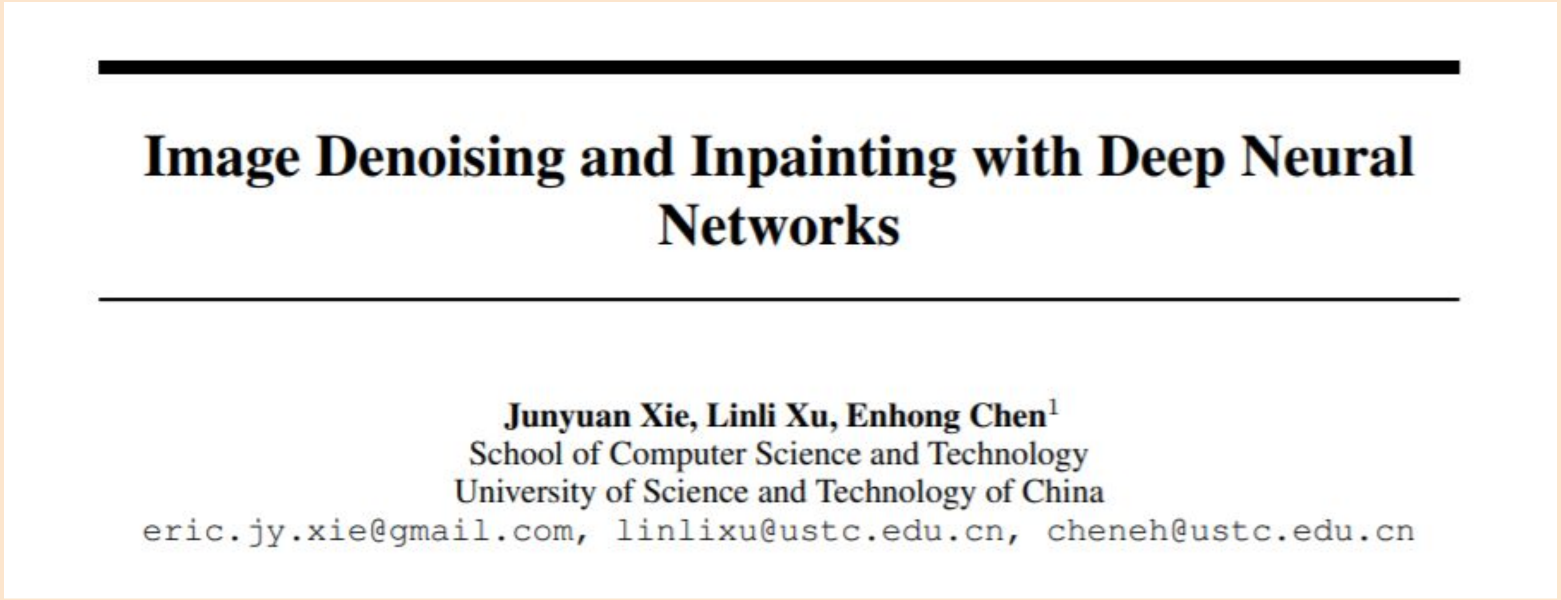


Image Denoising and Inpainting with Deep Neural Networks

Junyuan Xie, Linli Xu, Enhong Chen[1]
School of Computer Science and Technology
University of Science and Technology of China
eric.jy.xie@gmail.com, linlixu@ustc.edu.cn, cheneh@ustc.edu.cn

What does this paper propose?

1. The paper first describes a Denoising Auto-encoder which takes a corrupted version (xi) of original image(yi) and apply two sigmoid activation layers to get a form y(xi) and then optimize it weights and biases to achieve :-

$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{i=1}^{N} \|\mathbf{y_i} - \hat{\mathbf{y}}(\mathbf{x_i})\|.$$

1. This Denoising autoencoder is used as the basic unit of “Stacked Sparse Denoising Auto-encoders” where K DAs are stacked one after another, where (i-1)th DA’s first layer activation becomes the input of the ith DA
2. These K DA’s are trained such that it weights optimize the following reconstruction loss :-

$$L_1(\mathbf{X},\mathbf{Y};\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\|\mathbf{y_i} - \hat{\mathbf{y}}(\mathbf{x_i})\|_2^2 + \beta\,\mathrm{KL}(\hat{\rho}\|\rho) + \frac{\lambda}{2}(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2)$$

MSE + Sparsity inducing term + Regularization term

These pre-trained weights obtained from K stacked DAs then initialize our final deep network having 2*K-1 hidden activation layers and one input and output each. Then this deep network is trained to minimize the following objective :-

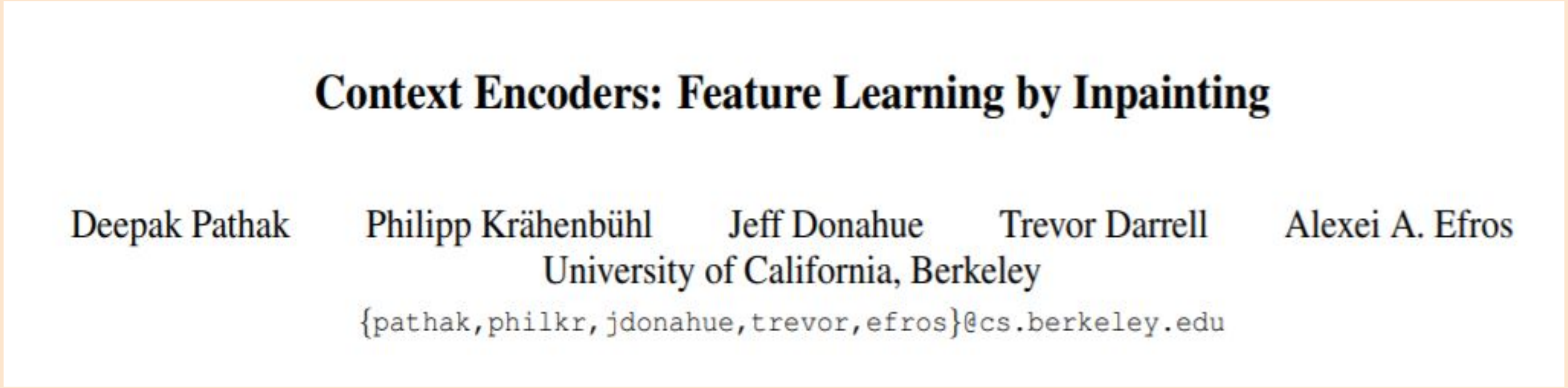
MSE + Regularization term

$$L_2(\mathbf{X}, \mathbf{Y}; \theta) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2} \|\mathbf{y_i} - \mathbf{y}(\mathbf{x_i})\|_2^2 + \frac{\lambda}{2} \sum_{j=1}^{2K} (\|\mathbf{W_j}\|_F^2).$$

Note here no sparsity regularization term is present as pre-trained weights will serve as the regularization.

Implementation

1. We tried implementing the Denoising Autoencoder and Stacked model , but unfortunately we did not get any desired results.
2. We could not comprehend if something not described in the paper is also required in the model
3. However we will try training it and tuning it more to check if some desired result could be achieved

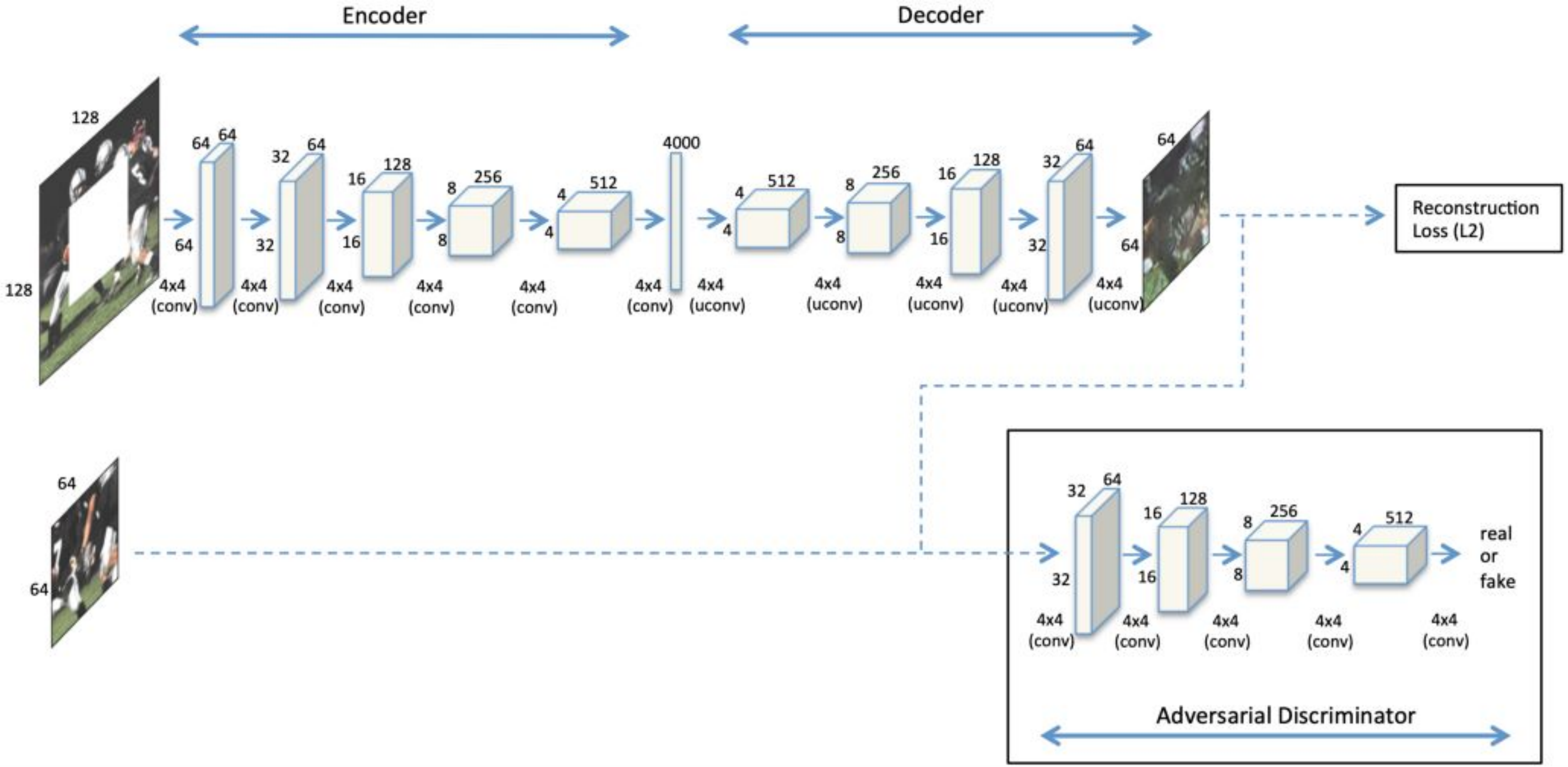


Context Encoders: Feature Learning by Inpainting

Deepak Pathak   Philipp Krähenbühl   Jeff Donahue   Trevor Darrell   Alexei A. Efros
University of California, Berkeley
{pathak,philkr,jdonahue,trevor,efros}@cs.berkeley.edu

What does this paper propose?

1. This paper proposes to employ Generative Adversarial Network (GANs) to perform image inpainting
2. The model proposed in the paper consists of an encoder capturing the context of an image into a compact latent feature representation and a decoder which uses that representation to produce the missing image content and thus it is known as **context encoder**


Encoder
Decoder
128
128
64
64
64
32
32
64
16
16
128
8
8
256
4
4
512
4000
4
4
512
8
8
256
16
16
128
32
32
64
64
64
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (uconv)
4x4 (uconv)
4x4 (uconv)
4x4 (uconv)
4x4 (uconv)
Reconstruction Loss (L2)
64
64
32
32
64
16
16
128
8
8
256
4
4
512
real or fake
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (conv)
4x4 (conv)
Adversarial Discriminator

Loss Functions

Reconstruction Loss

$$\mathcal{L}_{rec}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2, \qquad (1)$$

Adversarial Loss

$$\mathcal{L}_{adv} = \max_{D} \quad \mathbb{E}_{x \in \mathcal{X}} [\log(D(x)) \\ + \log(1 - D(F((1 - \hat{M}) \odot x)))], \qquad (2)$$

$$\mathcal{L} = \lambda_{rec}\mathcal{L}_{rec} + \lambda_{adv}\mathcal{L}_{adv}.$$

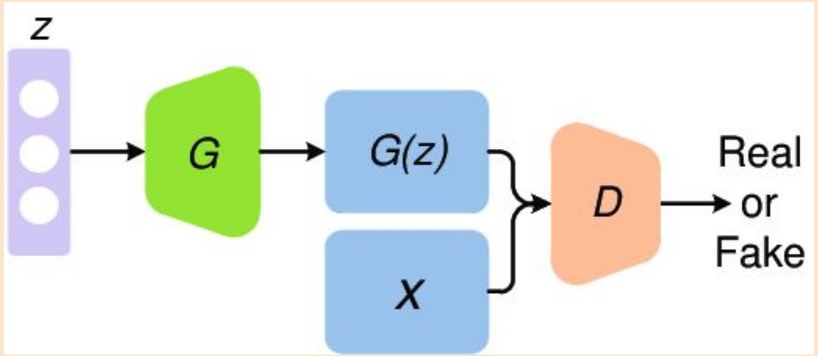
Combined Loss

How is this model trained?

1. Both the generator and discriminator are trained alternatively using Adam optimizer
2. We first create a random mask on a known image and then we make a prediction by our generator for the masked region
3. Then we train our discriminator with both the original masked region intensities labeled as real and predicted generator masked region labeled as fake
4. Then we train our generator using a combined model because our loss takes into account the prediction by discriminator (adversarial loss) as well as reconstruction loss

What is a Generative Adversarial Network?

- Typical GANs consist of one generator and one discriminator.
- The generator is responsible for filling the missing parts in an image
- And the discriminator is responsible for distinguishing filled images for real images

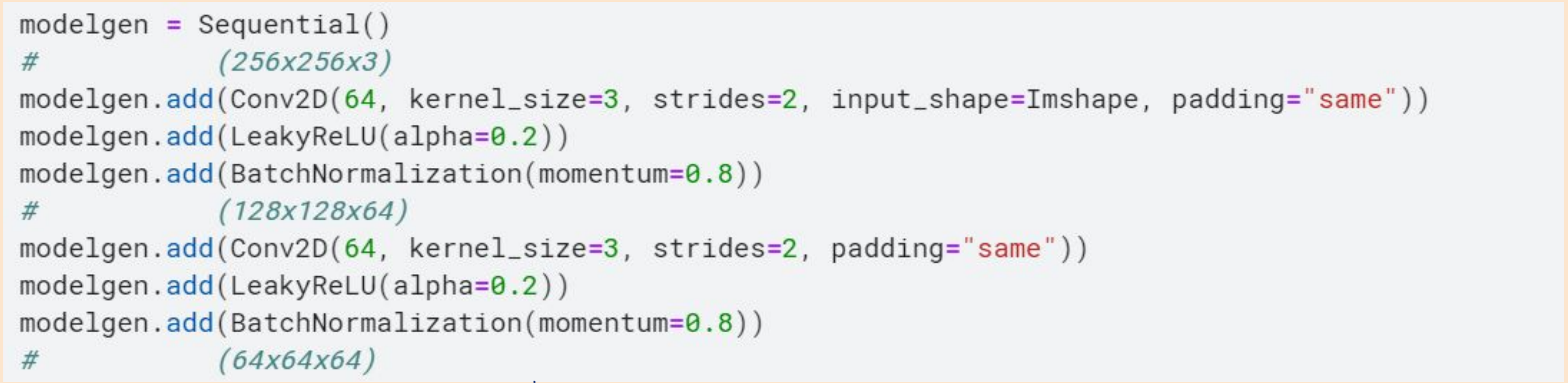


Main intuition behind GANs

We will randomly feed filled images and real images to our discriminator to fool it

Eventually when our discriminator cannot figure out that the given filled image is fake , then we will be certain that our generator created a good visual quality image

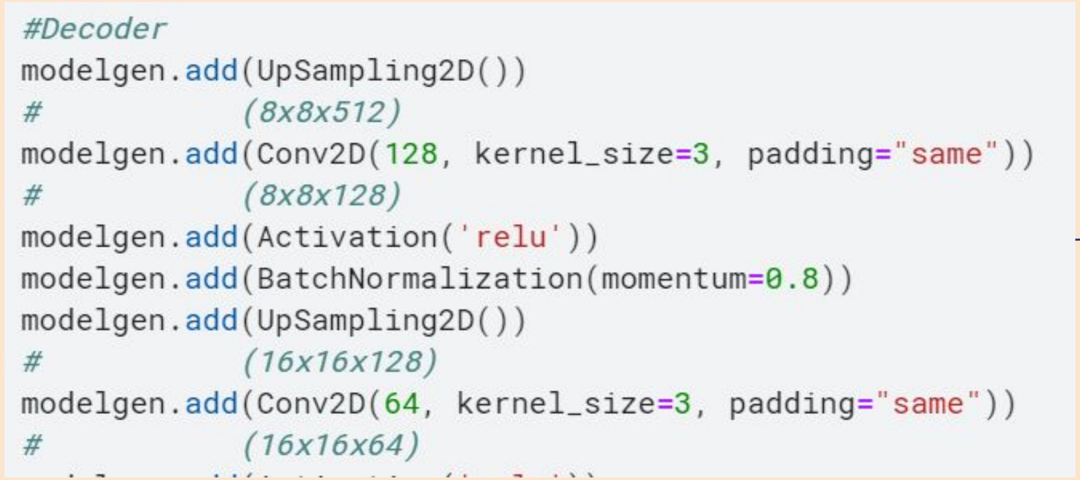
Generator Model - Encoder



```
modelgen = Sequential()
#            (256x256x3)
modelgen.add(Conv2D(64, kernel_size=3, strides=2, input_shape=Imshape, padding="same"))
modelgen.add(LeakyReLU(alpha=0.2))
modelgen.add(BatchNormalization(momentum=0.8))
#            (128x128x64)
modelgen.add(Conv2D(64, kernel_size=3, strides=2, padding="same"))
modelgen.add(LeakyReLU(alpha=0.2))
modelgen.add(BatchNormalization(momentum=0.8))
#            (64x64x64)
```

Input image size is 256x256x3
On adding Conv2D layer having 64 filters and stride =2 makes the output shape as 128x128x64

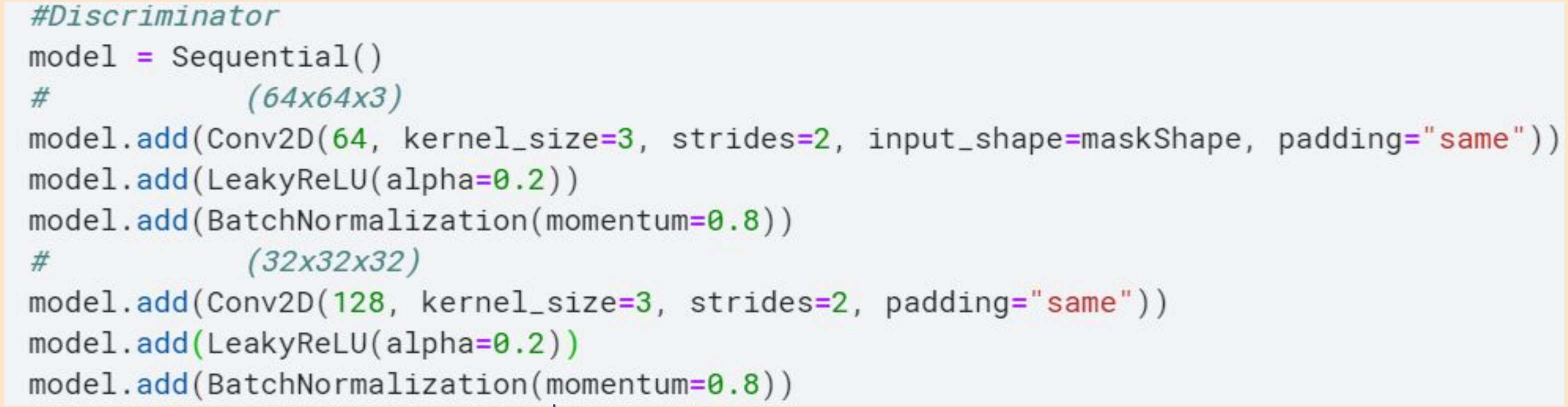
Generator Model - Decoder



```
#Decoder
modelgen.add(UpSampling2D())
#             (8x8x512)
modelgen.add(Conv2D(128, kernel_size=3, padding="same"))
#             (8x8x128)
modelgen.add(Activation('relu'))
modelgen.add(BatchNormalization(momentum=0.8))
modelgen.add(UpSampling2D())
#             (16x16x128)
modelgen.add(Conv2D(64, kernel_size=3, padding="same"))
#             (16x16x64)
```

On adding Upsampling2D() layer the channels remain same whereas the other 2 dimensions gets doubled and Conv2D with stride=1 just changes the number of channels and keeps the other 2 dimensions same

Discriminator Model



```python
#Discriminator
model = Sequential()
#           (64x64x3)
model.add(Conv2D(64, kernel_size=3, strides=2, input_shape=maskShape, padding="same"))
model.add(LeakyReLU(alpha=0.2))
model.add(BatchNormalization(momentum=0.8))
#           (32x32x32)
model.add(Conv2D(128, kernel_size=3, strides=2, padding="same"))
model.add(LeakyReLU(alpha=0.2))
model.add(BatchNormalization(momentum=0.8))
```

Input image size is 64x64x3
On adding Conv2D layer having 64 filters and stride =2 makes the output shape as 32x32x64

Alternate Training

```
inputImages, missingParts, _ = maskImages(imgs)

generatorPredicted = generator.predict(inputImages)

d_loss_real = discriminator.train_on_batch(missingParts,valid)
d_loss_fake = discriminator.train_on_batch(generatorPredicted,fake)
d_loss = 0.5*np.add(d_loss_fake,d_loss_real)

g_loss = combined.train_on_batch(inputImages, [missingParts,valid])
```

As described in slide 11

Results of Context Encoder on ImageNet Dataset

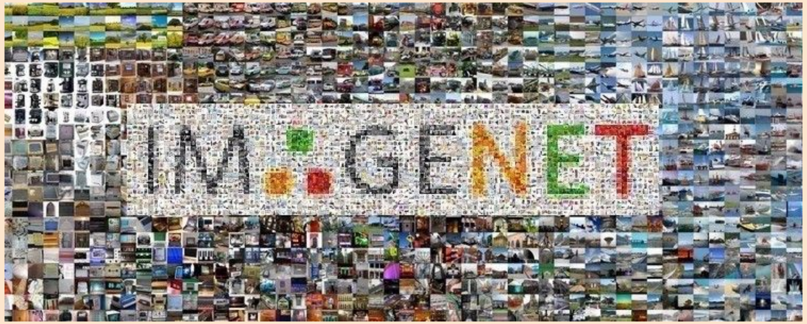
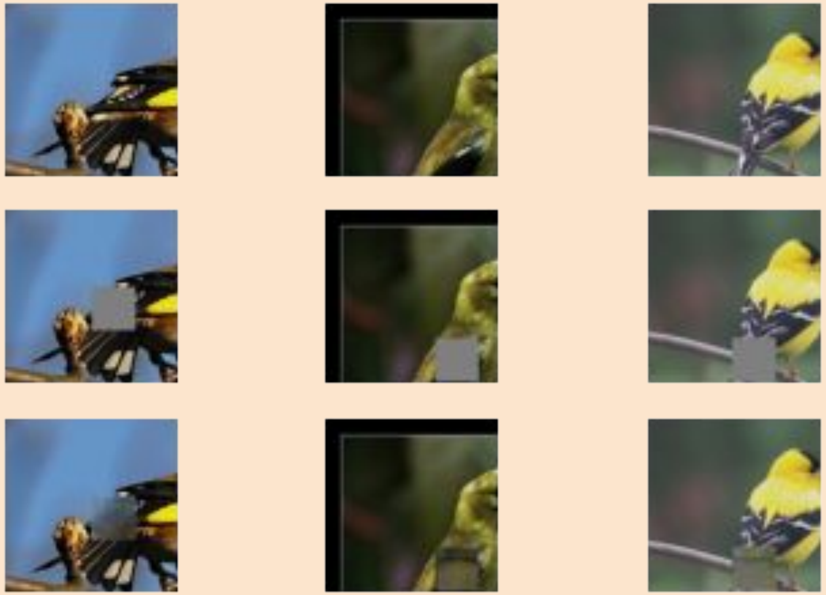
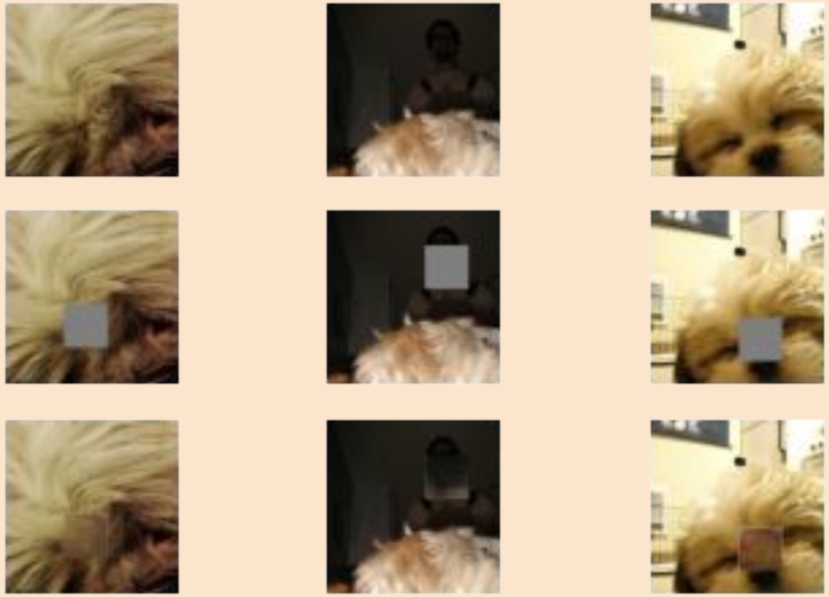


Image Inpainting - I



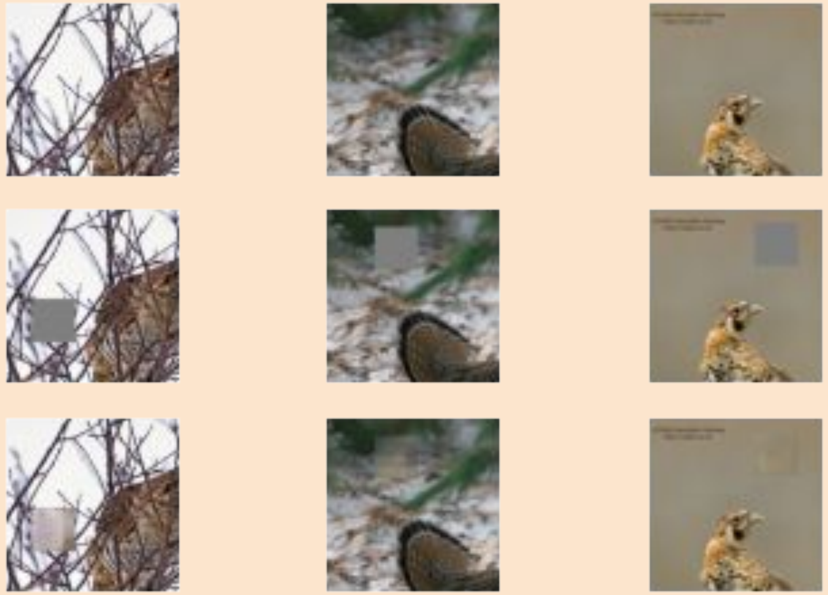
- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - II



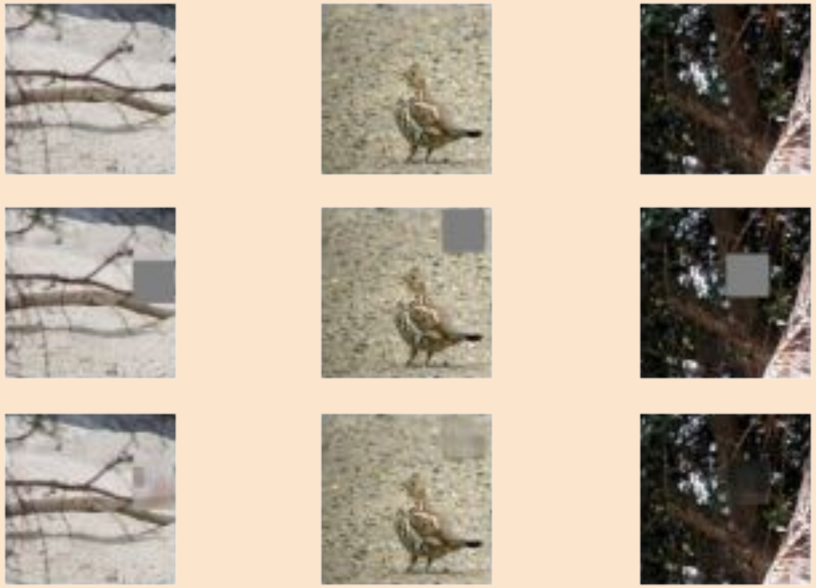
- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - III



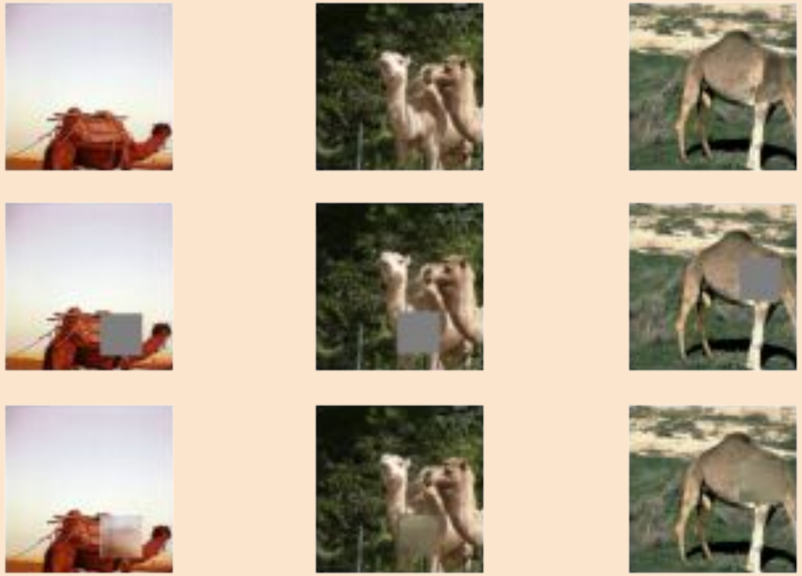
- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - IV



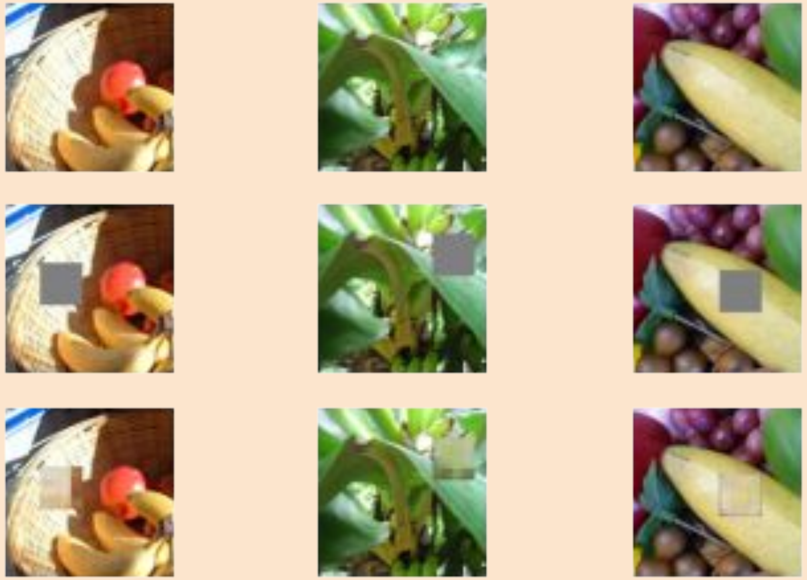
- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - V



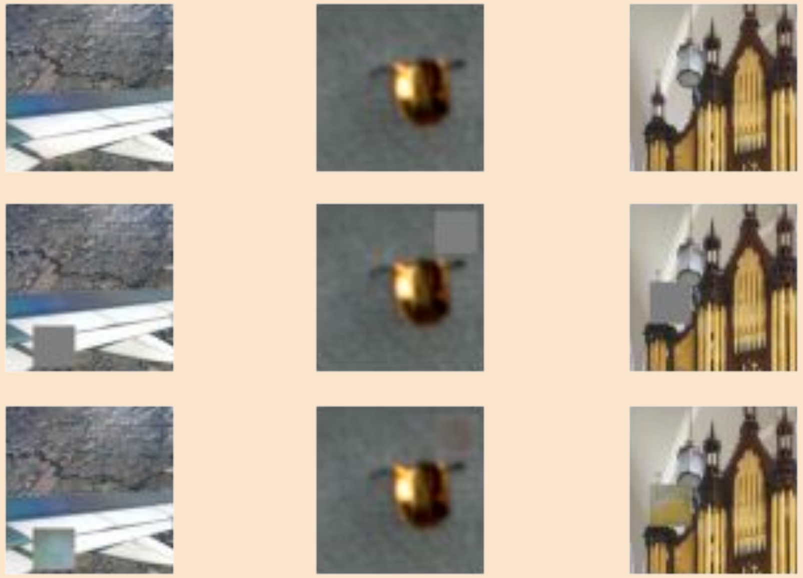
- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - VI



- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Image Inpainting - VII



- First Row: Original Image
- Second Row: Masked Image
- Third Row: Impainted Image

Further Advancements in Deep Image Inpainting

1. After the context encoder model, some models have been developed to transfer the style of the most similar valid pixels to the generated pixels to enhance local texture details
2. Some papers have tried to take into account both the local and global information of an image to enforce locally and globally consistency (by having both global and local discriminators)

Thank You