Final project for the Deep Learning course

Da Vinchie Lisa

SIMPLE IMAGE INPAINTING

01

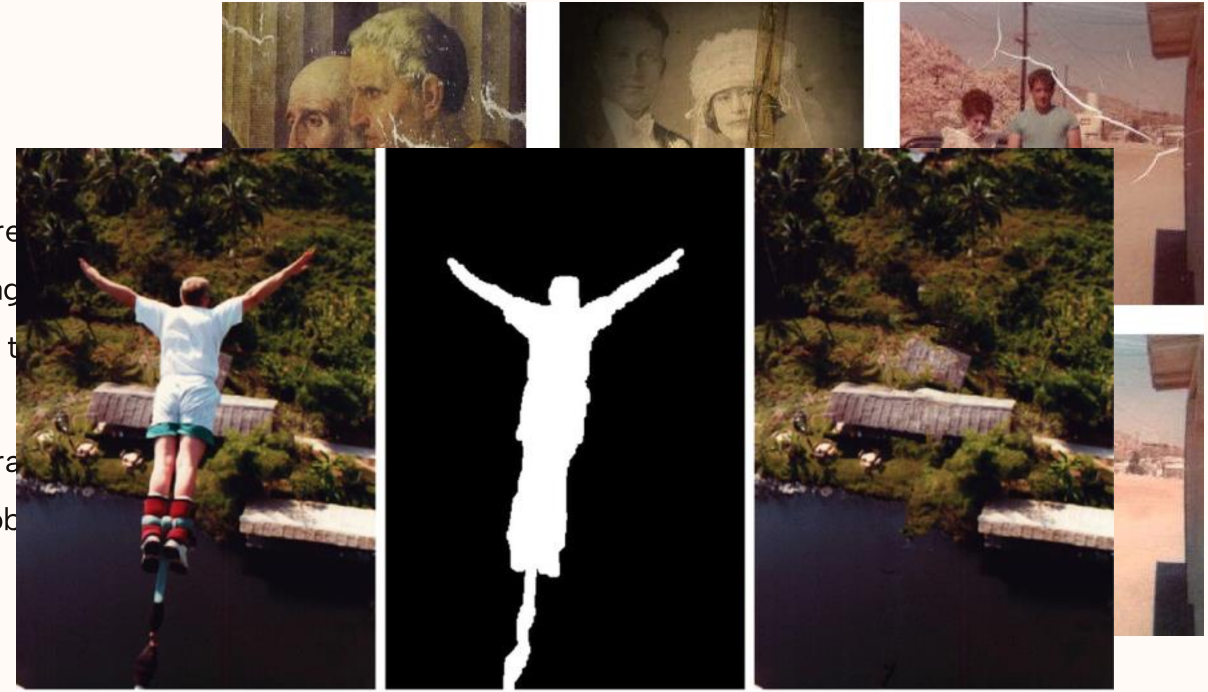
Introduction

What is image inpainting?

- It is a process used to restore missing parts of an image by using information from the surrounding pixels
- Examples: restoring scratched old photos, removing objects from a picture

What is image inpainting?

- It is a process used to re missing parts of an imag using information from t surrounding pixels
- Examples: restoring scra old photos, removing ob from a picture



What is image inpainting?

Challenges:

- There isn't a single correct way to fill in a missing part of an image

Main Techniques

- Partial differential equations
- Patch based
- Deep-learning based

Deep-Learning based techniques

- Autoencoders
- GANs
- Transformers
- Diffusion Models

Why the autoencoder?

Training stability

- Deterministic loss
- No adversarial training

Why the autoencoder?

Computational efficiency

- Single forward pass
- No discriminator => lower memory usage
- Faster training than GANs and diffusion models

Disadvantages

- Less realistic results than GANs
- Deterministic result
- Less accurate textures

The dataset

Cifar10

- 50000 training images
- 10000 validation images
- 3 x 32 x 32 images
- 10 classes

The dataset

Cifar10

- 50000 training images
- 10000 validation images
- 3 x 32 x 32 images
- 10 classes

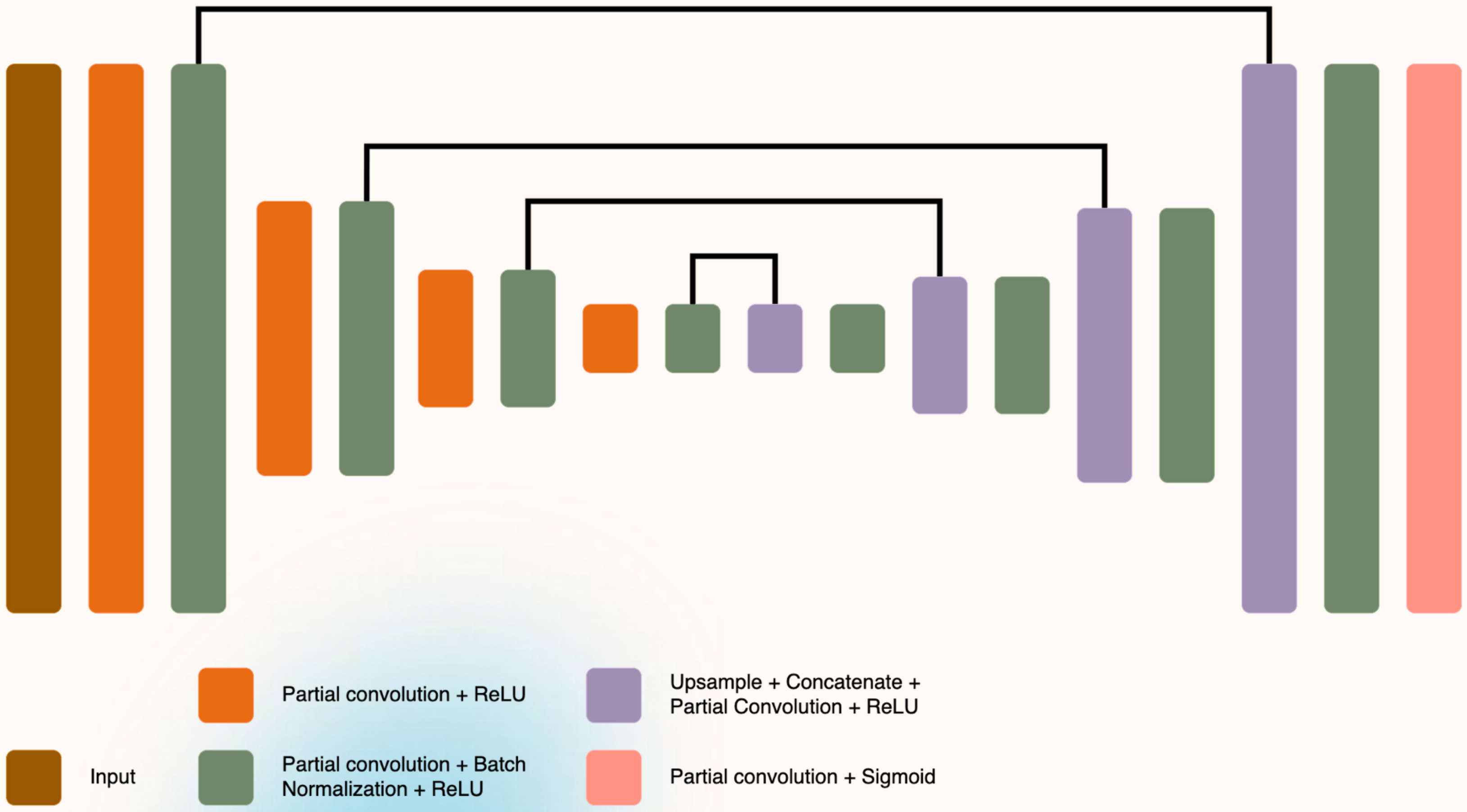
The dataset

Cifar10

- 50000 training images
- 10000 validation images
- 3 x 32 x 32 images
- 10 classes

02

Architecture



Partial convolution + ReLU
Upsample + Concatenate + Partial Convolution + ReLU
Input
Partial convolution + Batch Normalization + ReLU
Partial convolution + Sigmoid

Encoder

#	Layer	Output Size	#	Layer	Output Size
1	Input	3 x 32 x 32	8	Partial Conv + ReLU	128 x 8 x 8
2	Partial Conv + ReLU	32 x 32 x 32	9	Partial Convolution	128 x 4 x 4
3	Partial Convolution	32 x 16 x 16	10	Batch Normalization + ReLU	128 x 4 x 4
4	Batch Normalization + ReLU	32 x 16 x 16	11	Partial Conv + ReLU	256 x 4 x 4
5	Partial Conv + ReLU	64 x 16 x 16	12	Partial Convolution	256 x 2 x 2
6	Partial Convolution	64 x 8 x 8	13	Batch Normalization + ReLU	256 x 2 x 2
7	Batch Normalization + ReLU	128 x 8 x 8			

Decoder

#	Layer	Output Size	#	Layer	Output Size
14	Upsample		24	Upsample	
15	Concatenate		25	Concatenate	
16	Partial Convolution + ReLu	256 x 4 x 4	26	Partial Convolution + ReLu	64 x 16 x 16
17	Partial Convolution	128 x 4 x 4	27	Partial Convolution	32 x 16 x 16
18	Batch Normalization + ReLU	128 x 4 x 4	28	Batch Normalization + ReLU	32 x 16 x 16
19	Upsample		29	Upsample	
20	Concatenate		30	Concatenate	
21	Partial Convolution + ReLu	128 x 8 x 8	31	Partial Convolution + ReLu	32 x 32 x 32
22	Partial Convolution	64 x 8 x 8	32	Partial Convolution	3 x 32 x 32
23	Batch Normalization + ReLU	64 x 8 x 8	33	Batch Normalization + ReLU	3 x 32 x 32
			34	Partial Convolution + Sigmoid	3 x 32 x 32

Why this architecture?

- Based on previous works, adapted for smaller images
- Convolutions are very efficient in learning features and are translation invariant
- Batch Normalization helps with convergence
- Skip connections mitigate information loss during
- The sigmoid maps the values of the pixels to values between 0 and 1
- Hierarchical layers learn different features at each layer

Training an autoencoder

Training

- Take a fully know image
- Simulate a mask by replacing values under a specific area with a placeholder
- During training, reconstruct the whole image but calculate the loss only on the masked pixels

Convolutional layer

$$x' = \sum \left(W^T \circ X \right)$$

Convolutional layer

$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 5 & 3 & 8 & 4 \\ 3 & 3 & 2 & 8 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 6 & -9 & -8 \\ -3 & -2 & -3 \\ -3 & 0 & -2 \end{bmatrix}$$

Convolutional layer

$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 0 & 0 & 8 & 4 \\ 3 & 3 & 0 & 0 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 11 & -6 & -13 \\ 2 & 10 & -8 \\ -1 & 8 & -4 \end{bmatrix}$$

$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 10 & 10 & 8 & 4 \\ 3 & 3 & 10 & 10 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} -9 & -6 & 7 \\ -18 & 1 & 12 \\ -11 & -2 & 6 \end{bmatrix}$$

Partial Convolutions

- Only the valid pixels contribute to the convolution

$$x' = \begin{cases} \mathbf{W}^T(\mathbf{X} \odot \mathbf{M}) \frac{\mathrm{sum}(\mathbf{1})}{\mathrm{sum}(\mathbf{M})} + b, & \text{if } \mathrm{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Partial Convolutions

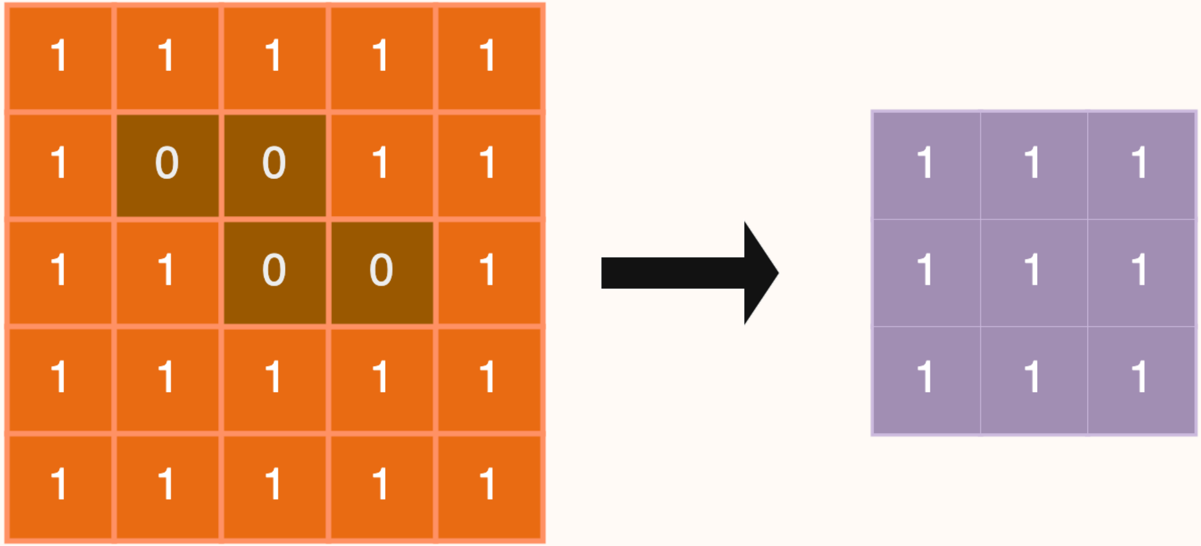
$$\begin{bmatrix} 7 & 2 & 3 & 3 & 8 \\ 4 & 0 & 0 & 8 & 4 \\ 3 & 3 & 0 & 0 & 4 \\ 2 & 8 & 7 & 2 & 7 \\ 5 & 4 & 4 & 5 & 4 \end{bmatrix} \otimes \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 16.5 & -10.8 & -19.5 \\ 3 & 18 & -12 \\ -1.1 & 10.3 & -5.1 \end{bmatrix}$$

Partial Convolutions

- After each operation, the mask is updated to indicate which regions have been filled

$$m' = \begin{cases} 1, & \text{if } \mathrm{sum}(\mathbf{M}) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Partial Convolutions



Metrics

- MSE loss: measures the pixel-per-pixel similarity

$$MSE = \frac{1}{N}\sum_{i,j}(A_{i,j} - B_{i,j})^2$$

Metrics

- Dice Coefficient: measures the overlap between two sets

$$Dice = \frac{2 * |A \cap B|}{|A| + |B|}$$

Metrics

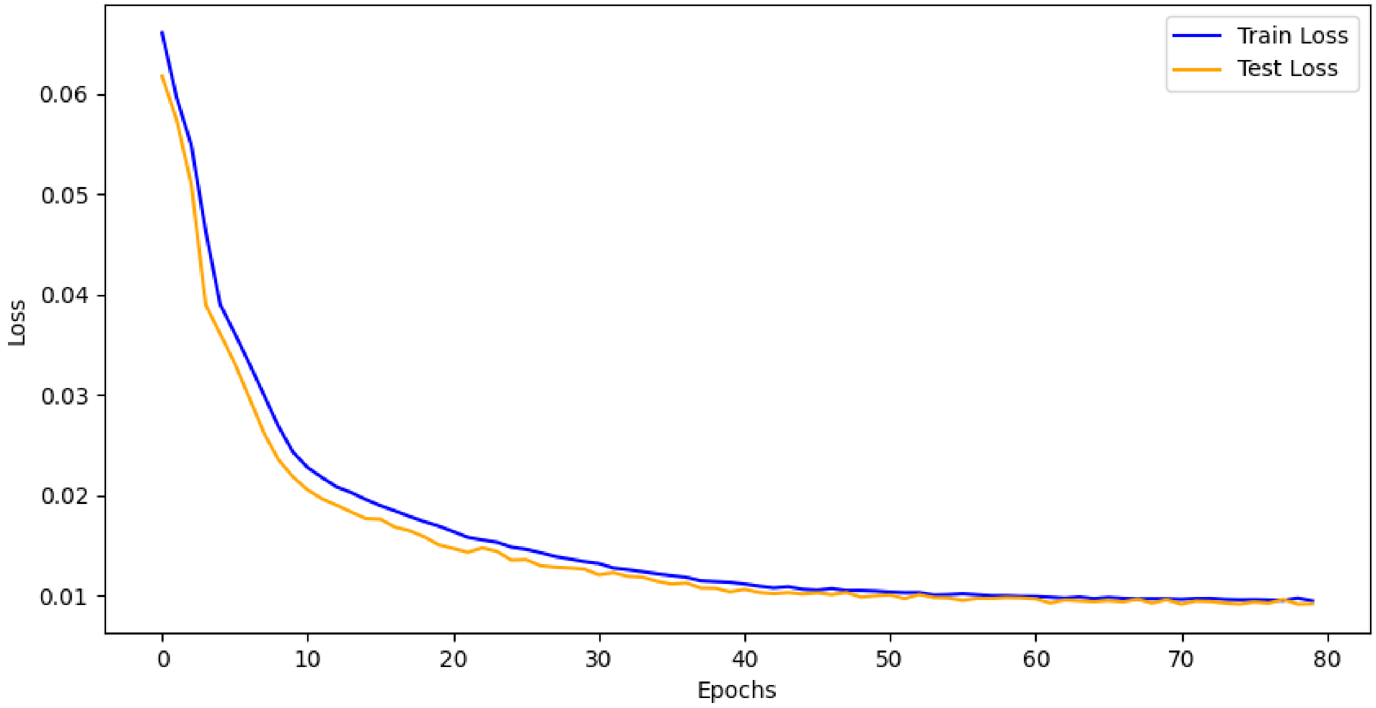
- Dice Coefficient: measures the overlap between two sets

$$Dice = \frac{2 * \sum_{i,j} A_{i,j} B_{i,j}}{\sum_{i,j} A_{i,j} + \sum_{i,j} B_{i,j}}$$

03

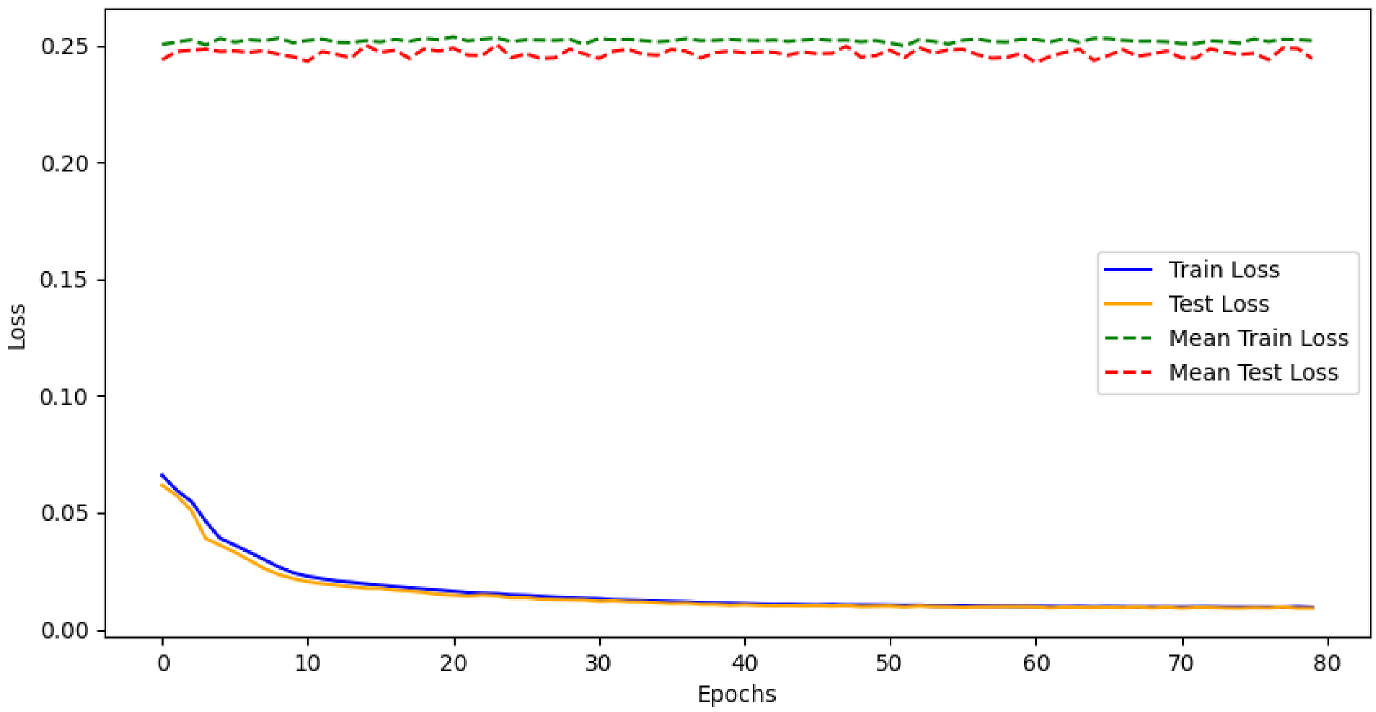
Results

Loss

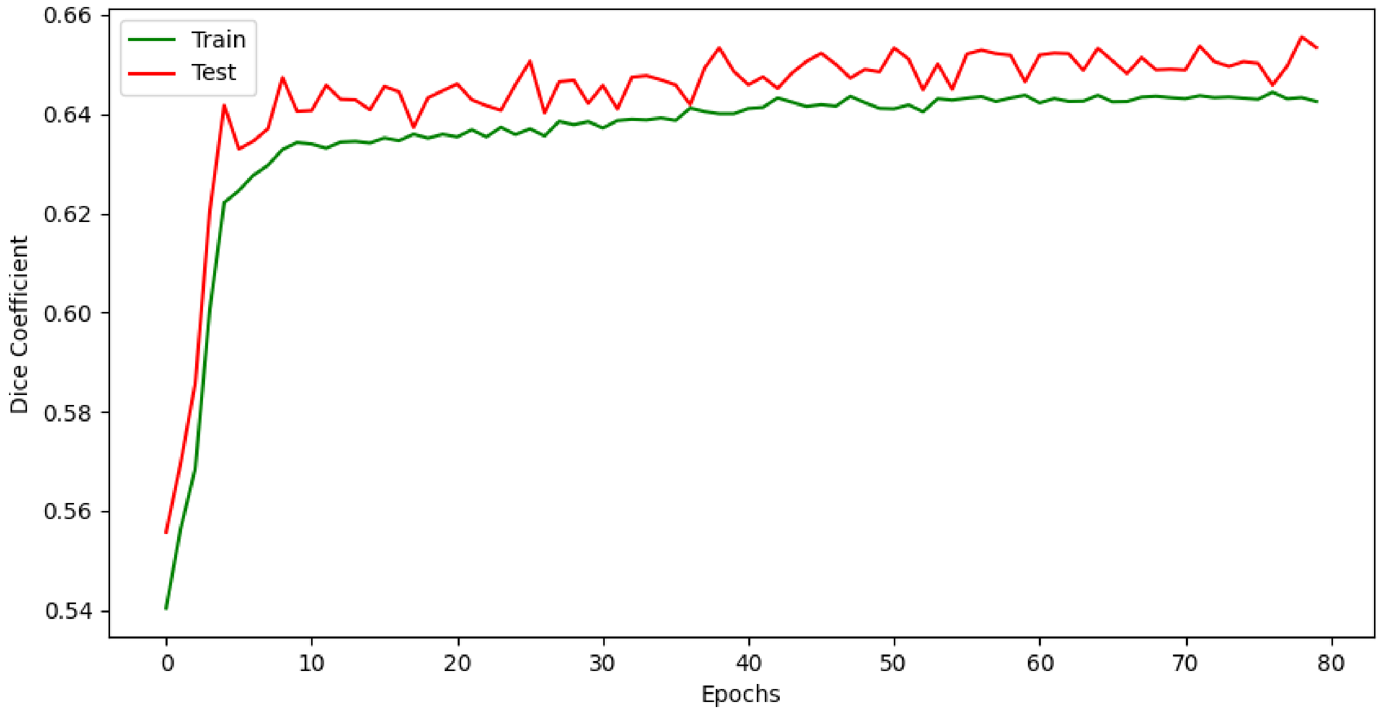


Train Loss
Test Loss
0.06
0.05
0.04
0.03
0.02
0.01
Loss
0
10
20
30
40
50
60
70
80
Epochs

Comparison with the mean

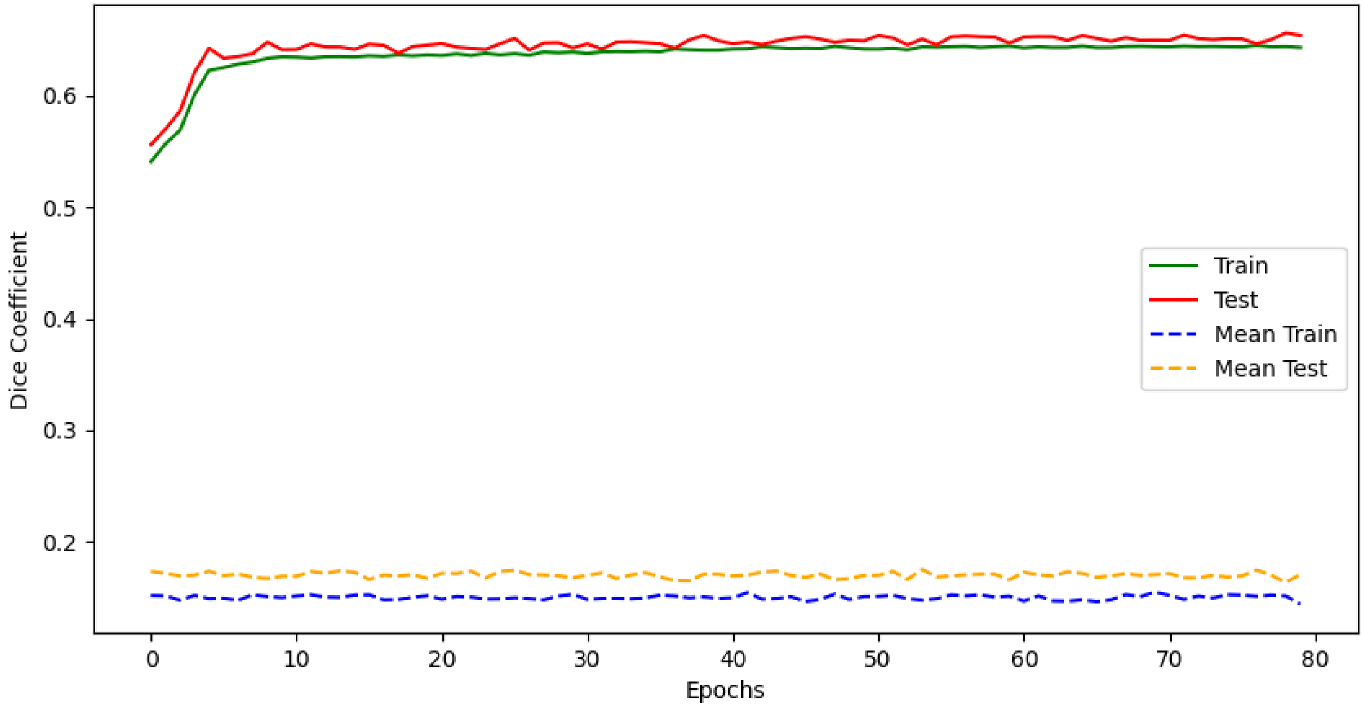


Dice

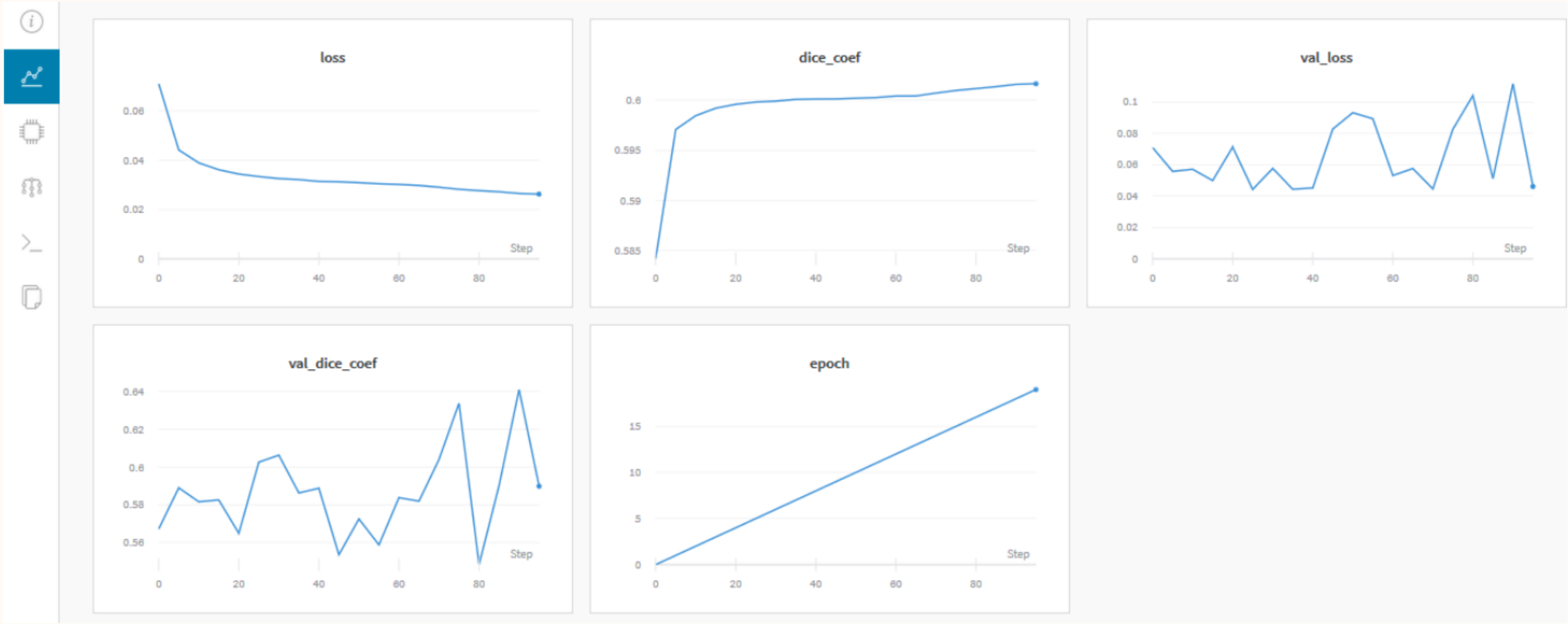


Train
Test
Dice Coefficient
0.66
0.64
0.62
0.60
0.58
0.56
0.54
0
10
20
30
40
50
60
70
80
Epochs

Comparison with the mean



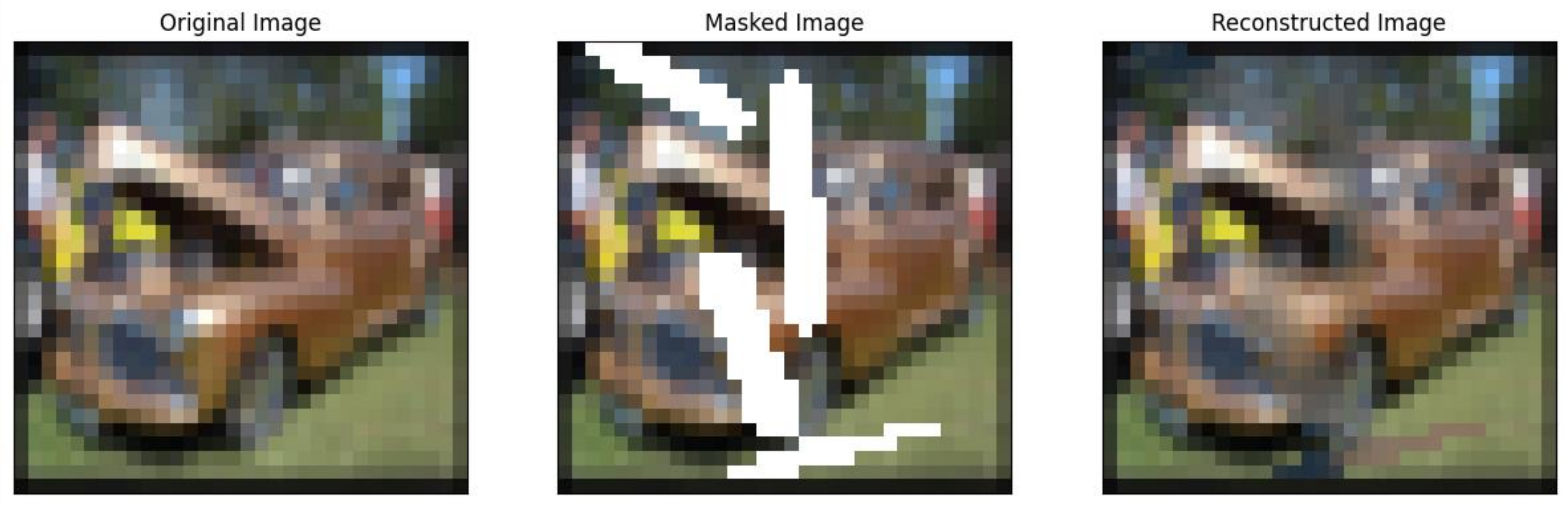
Comparison



Hyperparameters

- Adam Optimizer
- MSE loss
- Initial learning rate 0.0002
- Learning rate multiplied by 0.1 every 10 epochs
- 50000 training images, 10000 test images
- Images normalized in the range [−1, 1]

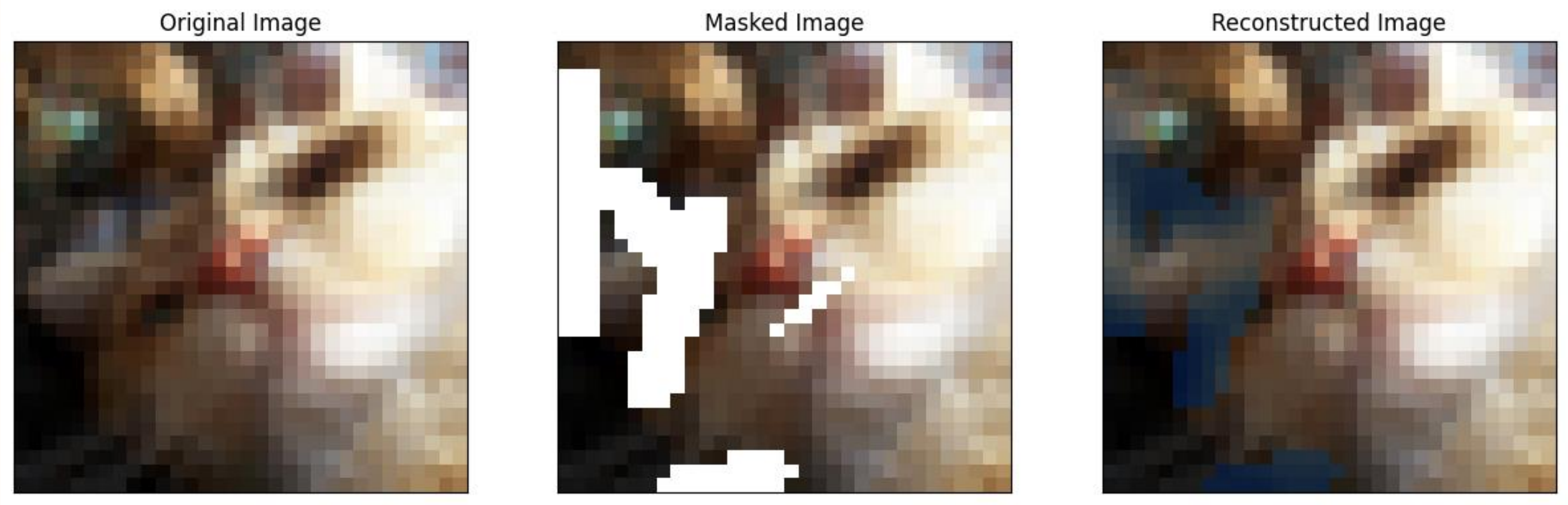
Reconstructed Images



Reconstructed Images

Reconstructed Images

Reconstructed Images



04

Improvements

More complex loss

From “Image Inpainting for Irregular Holes Using Partial Convolutions“, Liu et al.

$$\mathcal{L}_{total} = \mathcal{L}_{valid} + 6\mathcal{L}_{hole} + 0.05\mathcal{L}_{perceptual} + 120(\mathcal{L}_{style_{out}} + \mathcal{L}_{style_{comp}}) + 0.1\mathcal{L}_{tv}$$

Coarse-to-fine network

From “Generative Image Inpainting with Contextual Attention“, Yu et al.

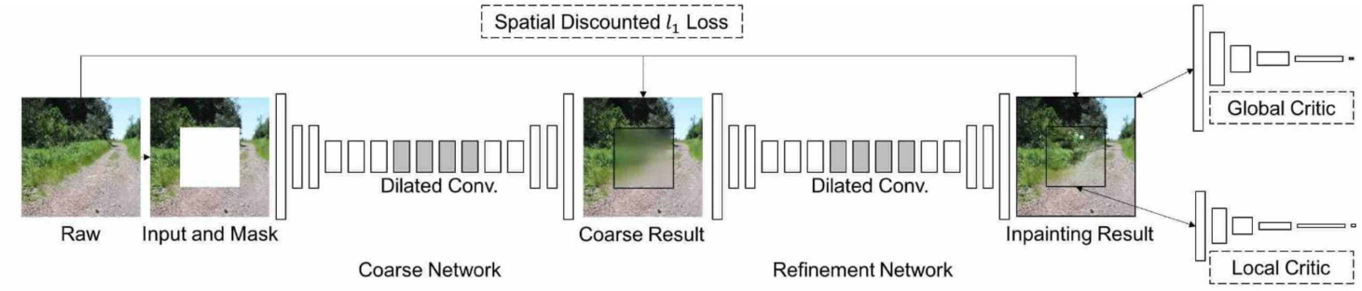


Figure 2: Overview of our improved generative inpainting framework. The coarse network is trained with reconstruction loss explicitly, while the refinement network is trained with reconstruction loss, global and local WGAN-GP adversarial loss.

Contextual Attention

From “Generative Image Inpainting with Contextual Attention“, Yu et al.

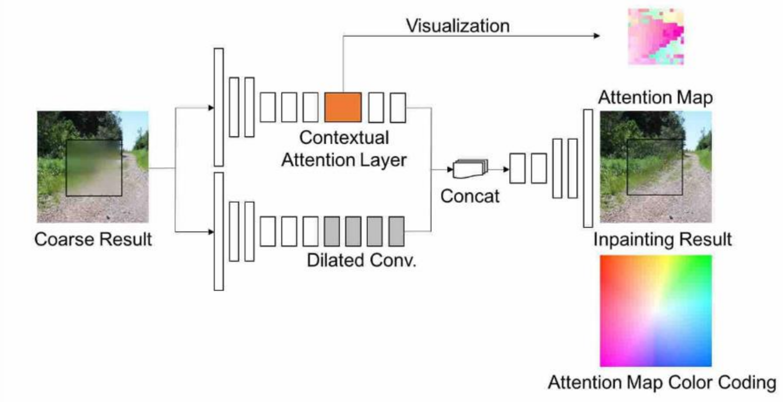


Figure 4: Based on coarse result from the first encoder-decoder network, two parallel encoders are introduced and then merged to single decoder to get inpainting result. For visualization of attention map, color indicates relative location of the most interested background patch for each pixel in foreground. For examples, white (center of color coding map) means the pixel attends on itself, pink on bottom-left, green means on top-right.

Thanks!

Sources:

- Introduction to image inpainting with deep learning
- Generative Image Inpainting with Contextual Attention
- Image Inpainting for Irregular Holes Using Partial Convolutions
- deepimageinpainting by ayulockin
- PConv-Keras by MathiasGruber

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon** and infographics & images by **Freepik**

Thanks!