Semantic Image Inpainting with Deep Generative Models

CS 736 Course Project Report

Karan Taneja (15D070022)
Pranav Kulkarni (15D070017)

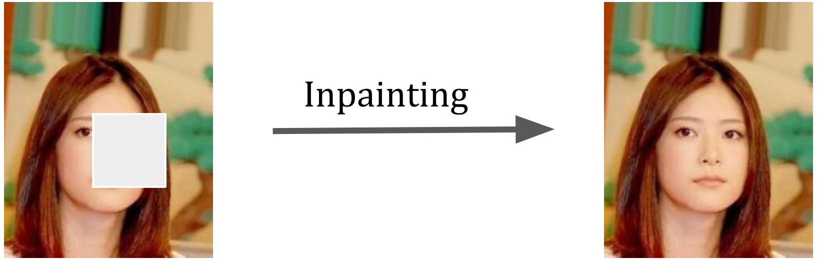
Contents

Based on the paper: *Semantic Image Inpainting with Deep Generative Models* in CVPR 2017 by R.A. Yeh et al.

Problem Statement

Motivation: For medical images - useful for processing (segmentation/ registration etc) in presence of lesions (suffered part)

Semantic image inpainting: large missing regions have to be filled based on the available visual data



Extracting information from single image loses out on high level context leading to poor results. So we use a deep generative model!

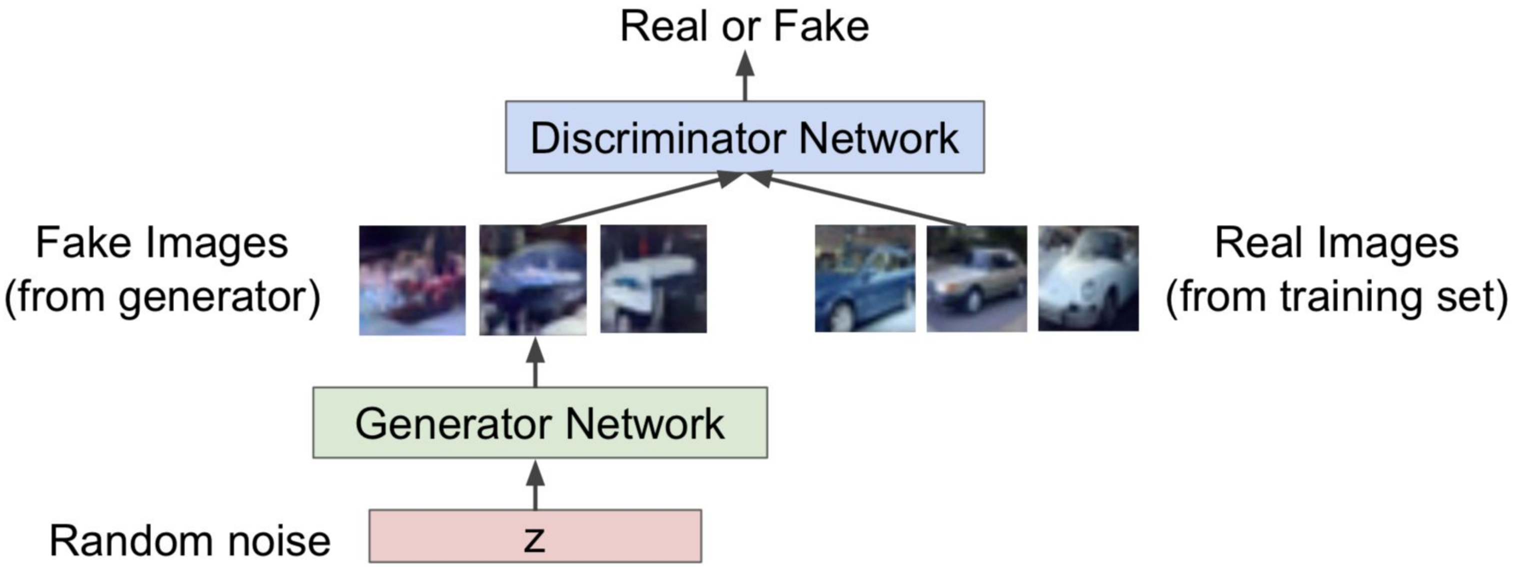
Overview of the approach

- Generate the missing content by conditioning on the available data.
- Use generative models (like GANs) with a generator which act as a mapping from latent space to images.
- For inpainting, find closest encoding of the corrupted image in latent space using context loss and prior loss.
- Pass the encoding through the generative model to infer missing content.
- Blend the predicted patch intensities to have coherence with surrounding known pixel intensities using blending.

Advantages of the approach

- Inference is possible independent of the structure of missing content.
- Requires no knowledge about shape and size of corrupted patches while training the model.
- Have provided realistic state of the art results on face images.

Generative Adversarial Network (GAN)



Training a GAN

Generator network: try to fool the discriminator by generating real-looking images
Discriminator network: try to distinguish between real and fake images

Train jointly in **minimax game**

Discriminator outputs likelihood in (0,1) of real image

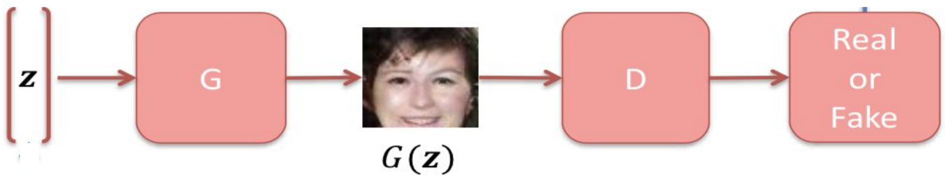
Minimax objective function:

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

- Discriminator (θ_d) wants to **maximize objective** such that D(x) is close to 1 (real) and D(G(z)) is close to 0 (fake)
- Generator (θ_g) wants to **minimize objective** such that D(G(z)) is close to 1 (discriminator is fooled into thinking generated G(z) is real)

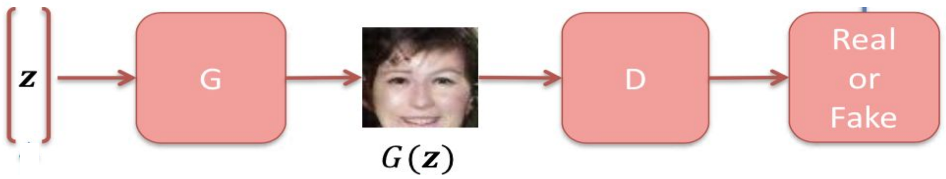
Importing GAN setup for inpainting

- Generator G and discriminator D are trained with uncorrupted data.
- After training, the generator G is able to map a point z drawn from p_Z and generate an image mimicking samples from pdata.



Importing GAN setup for inpainting

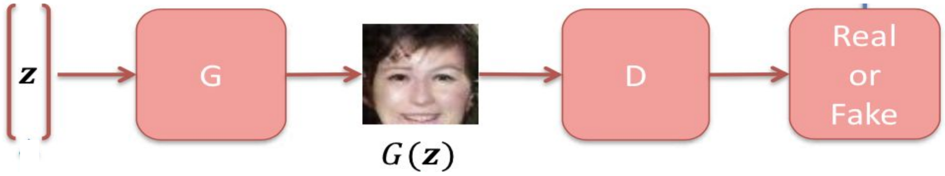
- Assumption: G is efficient in its representation then an image that is not from p_{data} (e.g., corrupted data) should not lie on the learned encoding manifold z.
- Aim to recover the encoding $\hat{z}$ "closest" to the corrupted image while being constrained to the manifold



Optimization Problem and Loss Terms

Optimization problem: y is the corrupted image, M is the binary mask.

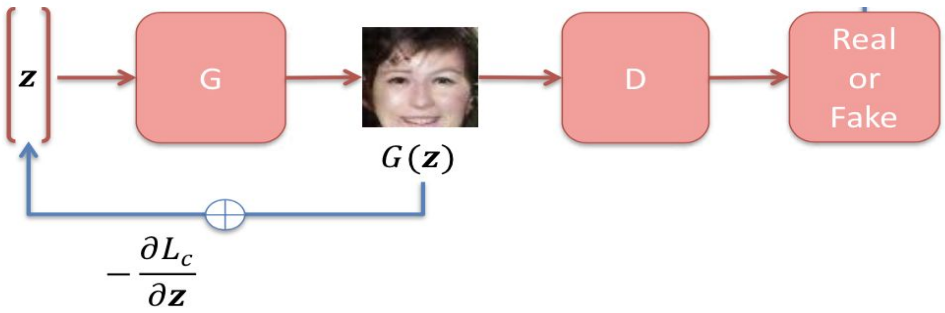
$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\}$$



Optimization Problem and Loss Terms

Optimization problem: y is the corrupted image, M is the binary mask.

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\}$$



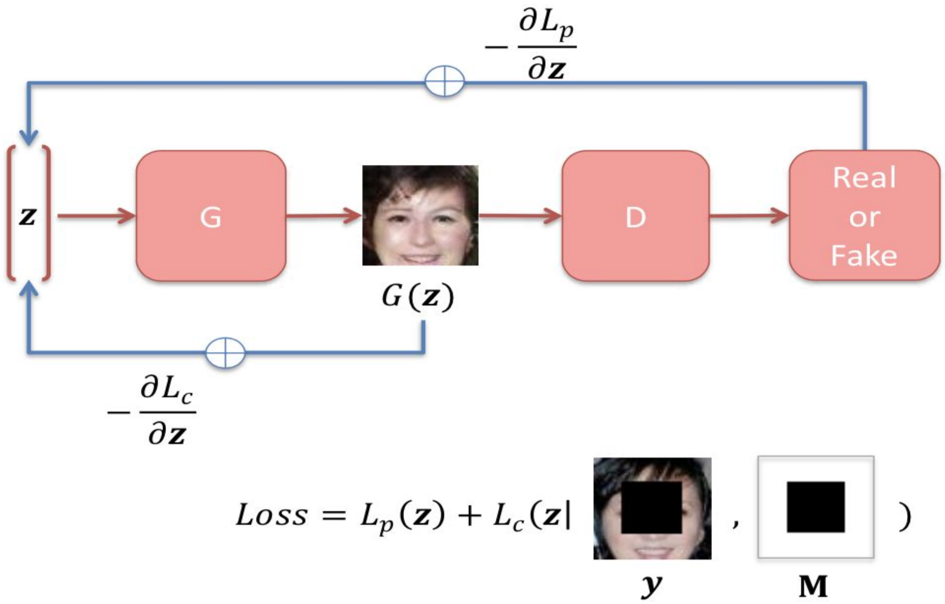
$\mathcal{L}_c$ is the context loss: constrains the generated image given the input corrupted image y and the hole mask M

Optimization Problem and Loss Terms

Optimization problem: y is the corrupted image, M is the binary mask.

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}} \{\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) + \mathcal{L}_p(\mathbf{z})\}$$

$\mathcal{L}_p$ is the prior loss: penalizes unrealistic images



Weighted Context Loss

- L_2 loss over uncorrupted part: equal importance to all pixels.
- Importance of an uncorrupted pixel should depend on the number of corrupted pixels surrounding it.
- A pixel that is very far away from any hole should play very little role in the inpainting process.

Weighted Context Loss

- *W(i)* importance of pixel location *i.*
- *|N(i)|* cardinality of set of neighbors of pixel *i* in a local window.

$$\mathbf{W}_i = \begin{cases} \sum_{j \in N(i)} \frac{(1-\mathbf{M}_j)}{|N(i)|} & \text{if } \mathbf{M}_i \neq 0 \\ 0 & \text{if } \mathbf{M}_i = 0 \end{cases}$$

According to the paper, empirically L_1 loss is slightly better!

$$\mathcal{L}_c(\mathbf{z}|\mathbf{y}, \mathbf{M}) = \|\mathbf{W} \odot (G(\mathbf{z}) - \mathbf{y})\|_1$$

Prior Loss

Penalties based on high-level image feature representations instead of pixel-wise differences.

Recovered image should be similar to the samples drawn from the training set.

Since D is trained to differentiate generated images from real images...

Hence the prior loss is taken identical to the GAN loss for training the discriminator D

$$\mathcal{L}_p(\mathbf{z}) = \lambda \log(1 - D(G(\mathbf{z})))$$

Here, λ is the balancing parameter between the two losses.

Inpainting

- Let $\hat{z}$ be closest z in latent space based on the prior and context loss.
- We can overlay uncorrupted pixels on $G(\hat{z})$.
- But, predicted pixels may not exactly preserve the same intensities of the surrounding pixels, although the content is correct and well aligned.
- Solution: Poisson Blending

Poisson Blending

Instead of keeping the intensity from the generated image, use the gradients of *G(ẑ)* to preserve image details!

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \|\nabla \mathbf{x} - \nabla G(\hat{\mathbf{z}})\|_2^2,$$

$$\text{s.t. } \mathbf{x}_i = \mathbf{y}_i \;\text{ for }\; \mathbf{M}_i = 1$$

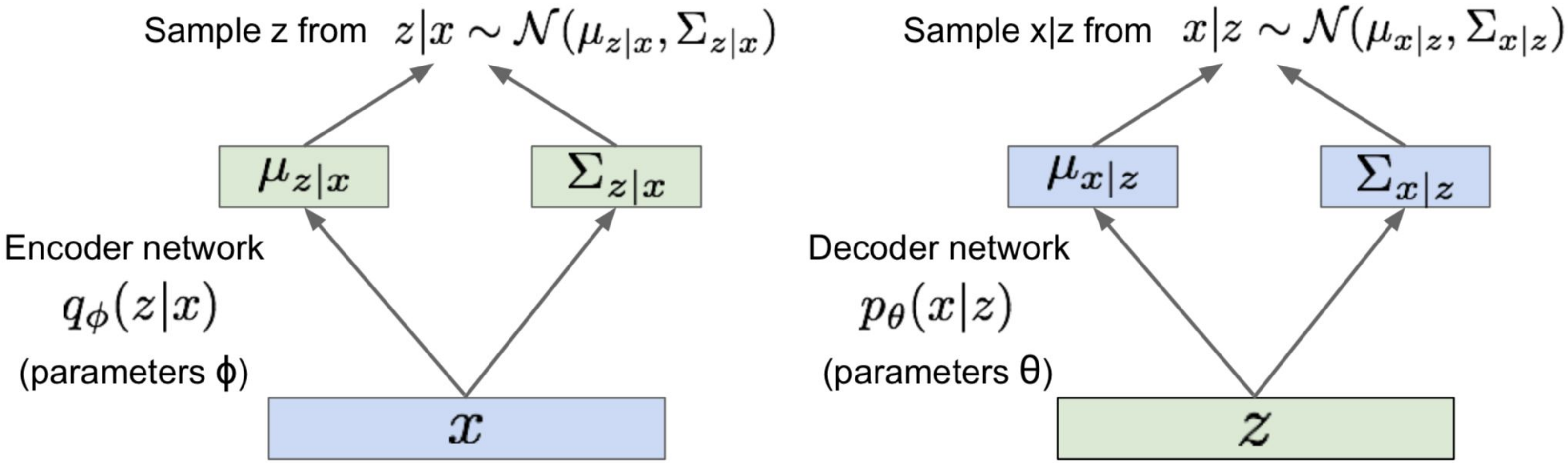
0	1	0
1	-4	1
0	1	0

the Laplace filter

Equivalent to minimizing the norm of difference of Laplacians of *x* and *G(ẑ)*!

And it has a unique solution!

Variational Autoencoders

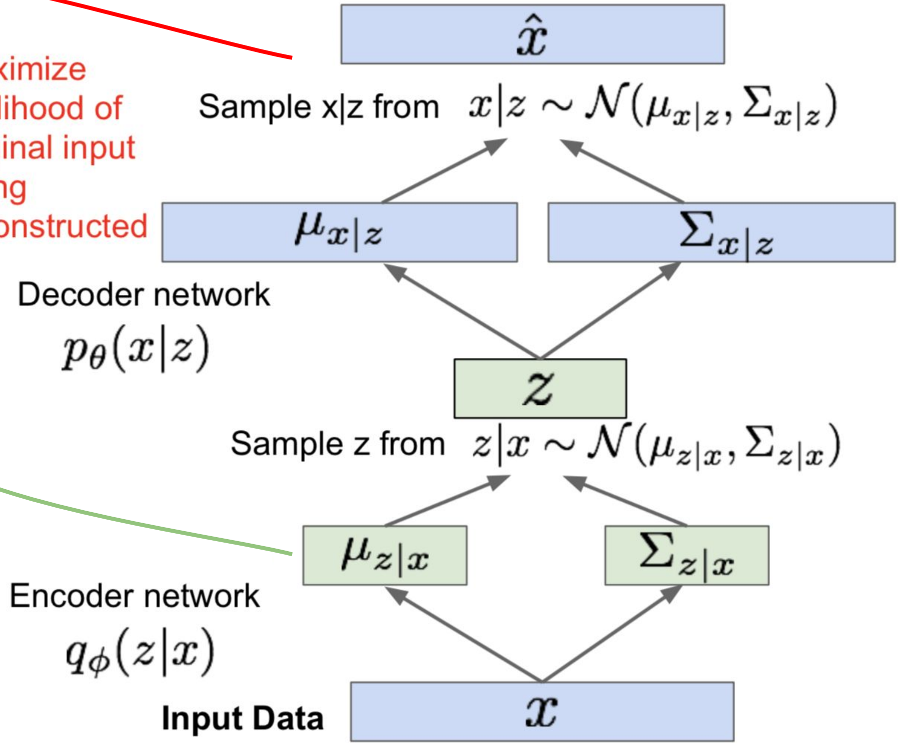


Variational Autoencoders

$$\underbrace{\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Maximize likelihood of original input being reconstructed

Make approximate posterior distribution close to prior



Importing VAE setup for inpainting

$\mathcal{L}_p$ Prior loss: $||z||^2$

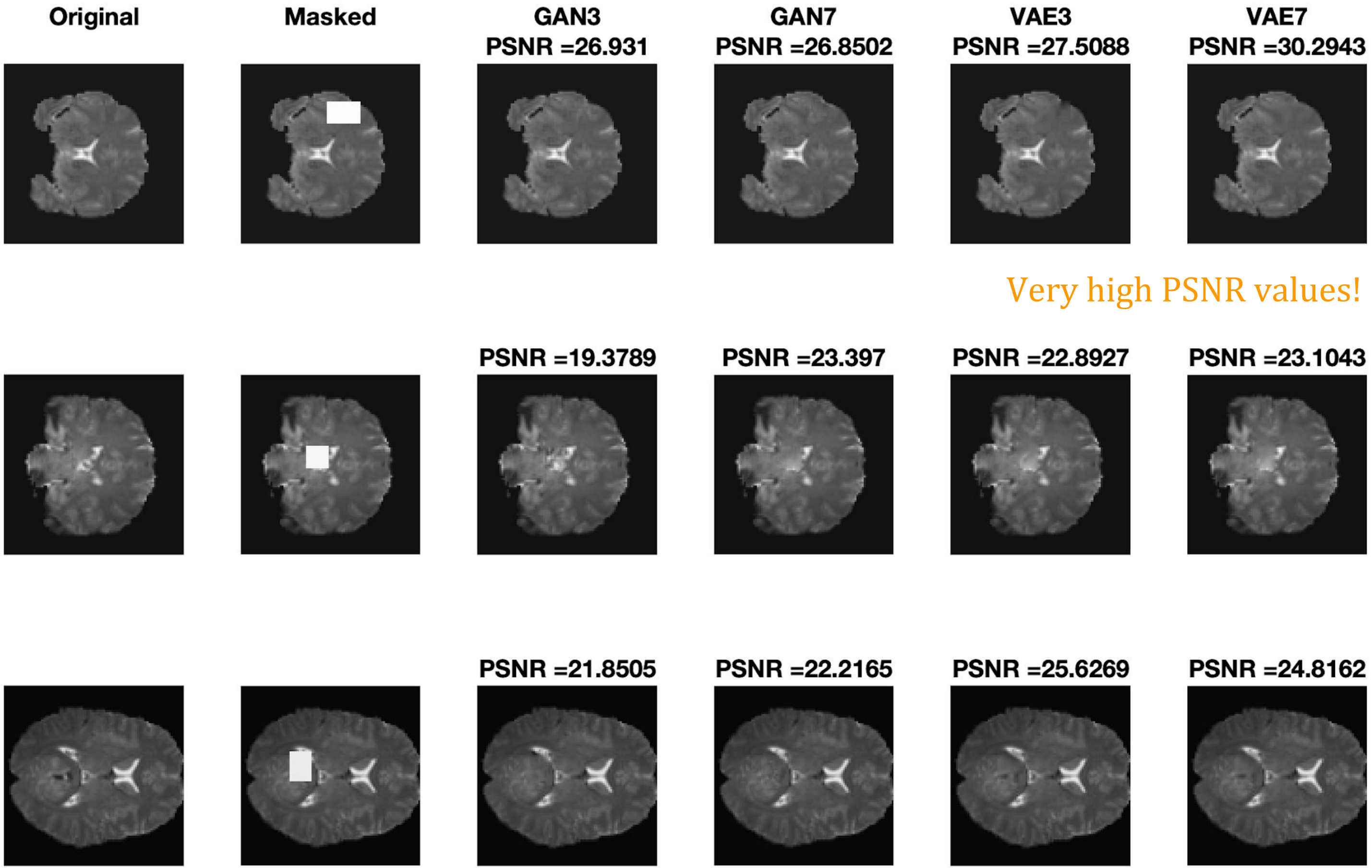
penalty on hidden representation vector being away from assumed prior distribution (standard normal distribution)

$\mathcal{L}_c$ Context loss: Same as before

L_1 norm of weighted perpixel difference

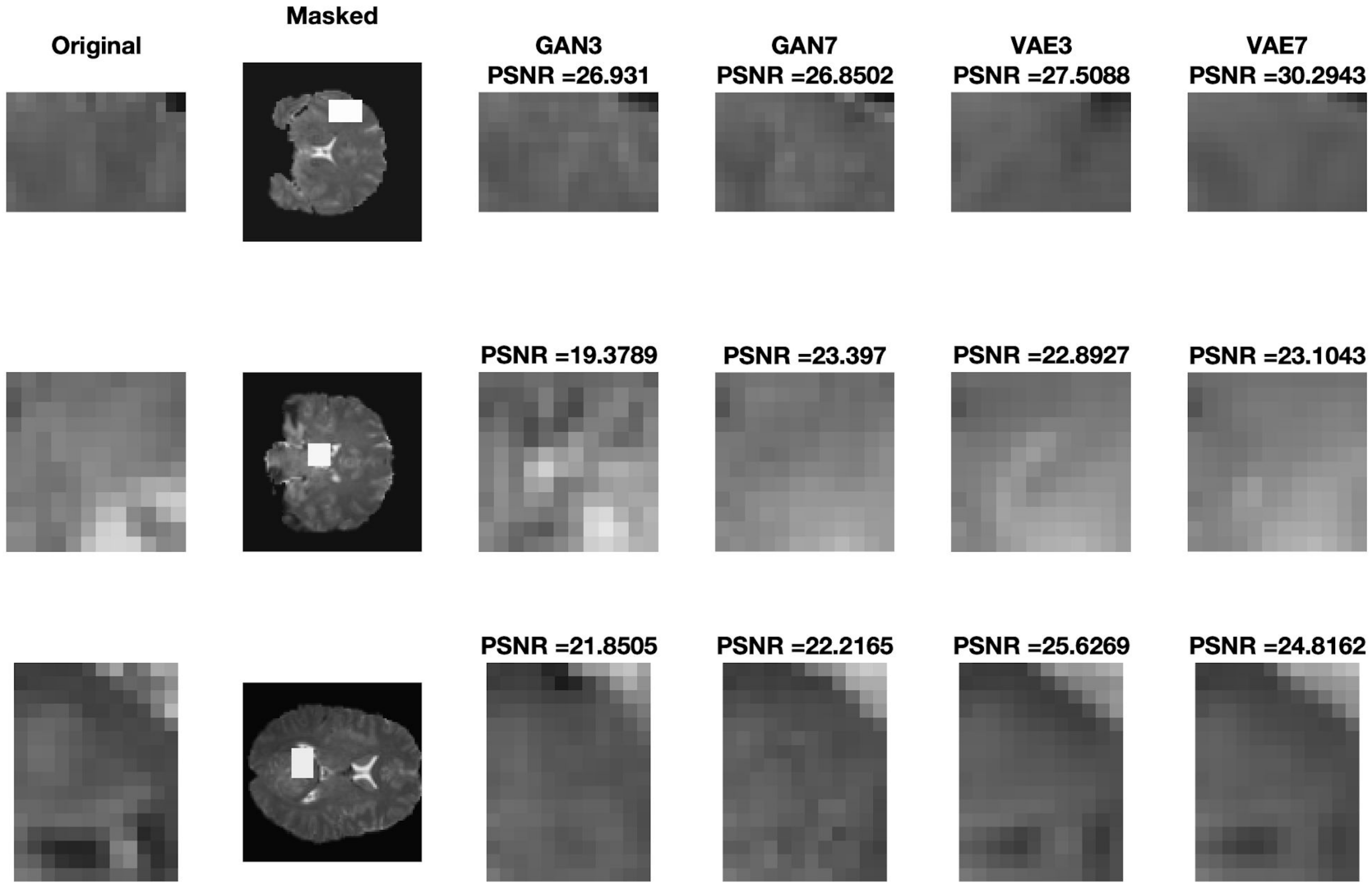
Experiments

Comparison of GANs and VAEs with convolution kernels of size 3 and 7

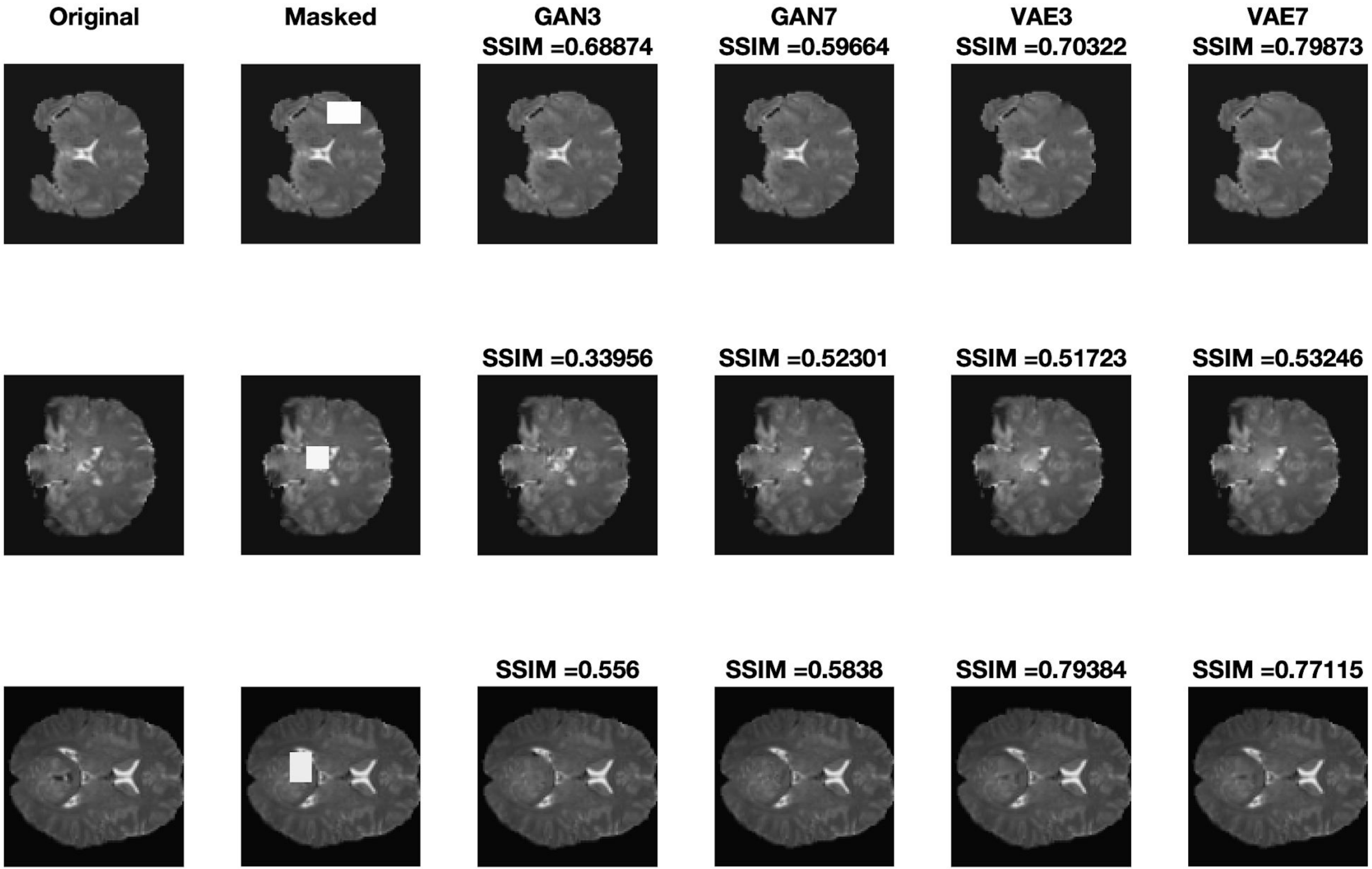


Very high PSNR values!

Inpainted images are visually almost indifferentiable...

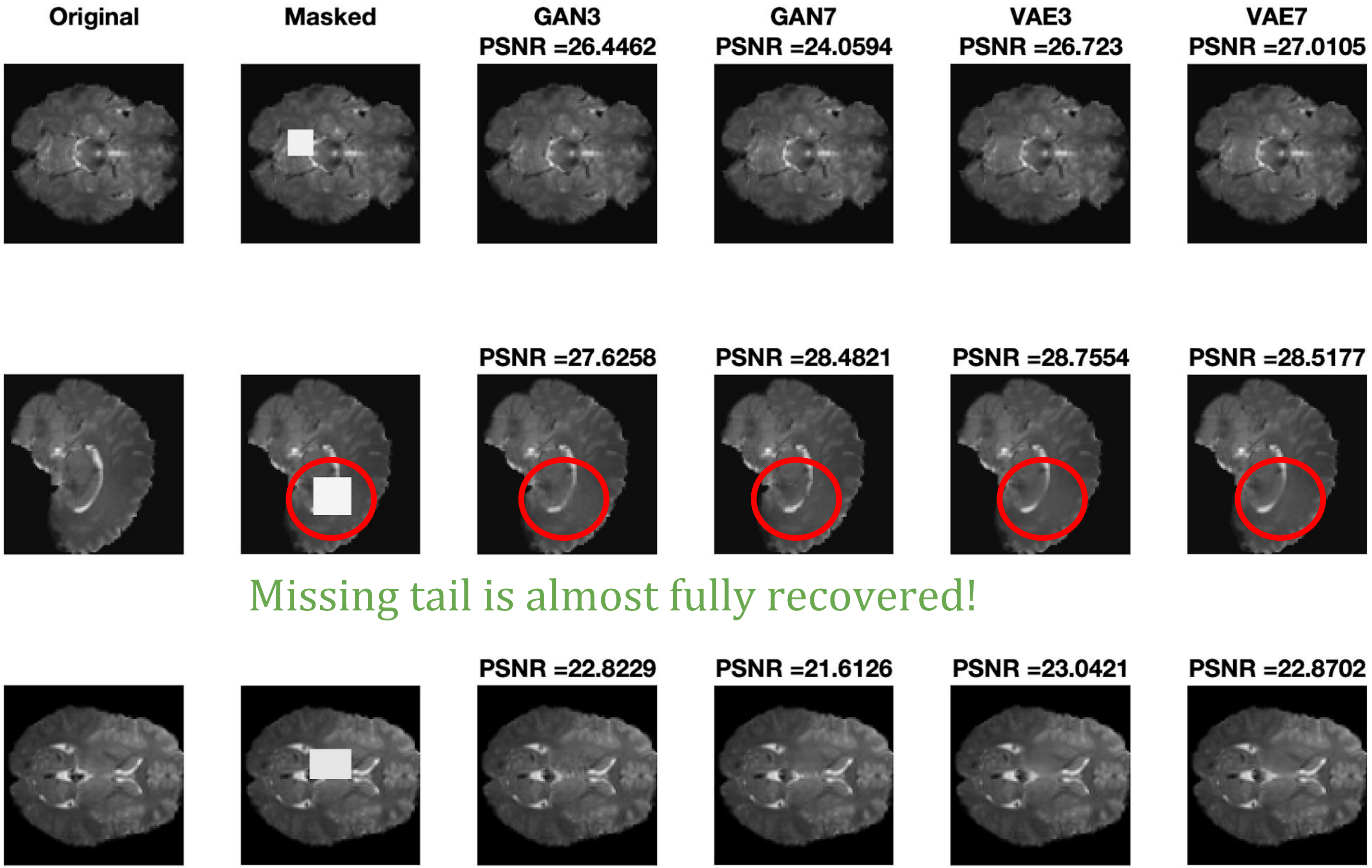


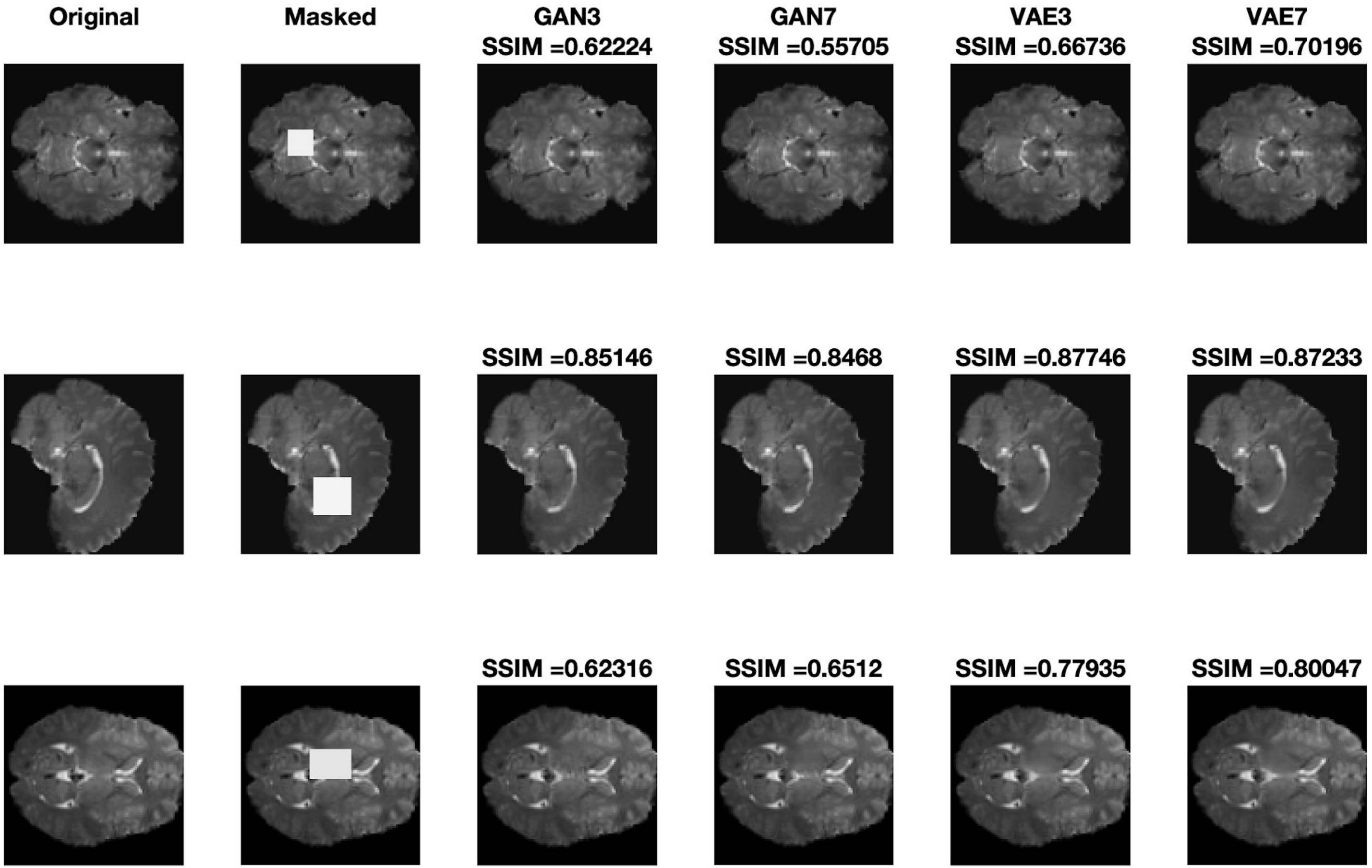
Original
Masked
GAN3
PSNR =26.931
GAN7
PSNR =26.8502
VAE3
PSNR =27.5088
VAE7
PSNR =30.2943
PSNR =19.3789
PSNR =23.397
PSNR =22.8927
PSNR =23.1043
PSNR =21.8505
PSNR =22.2165
PSNR =25.6269
PSNR =24.8162



Significantly better results for VAEs than GANs!

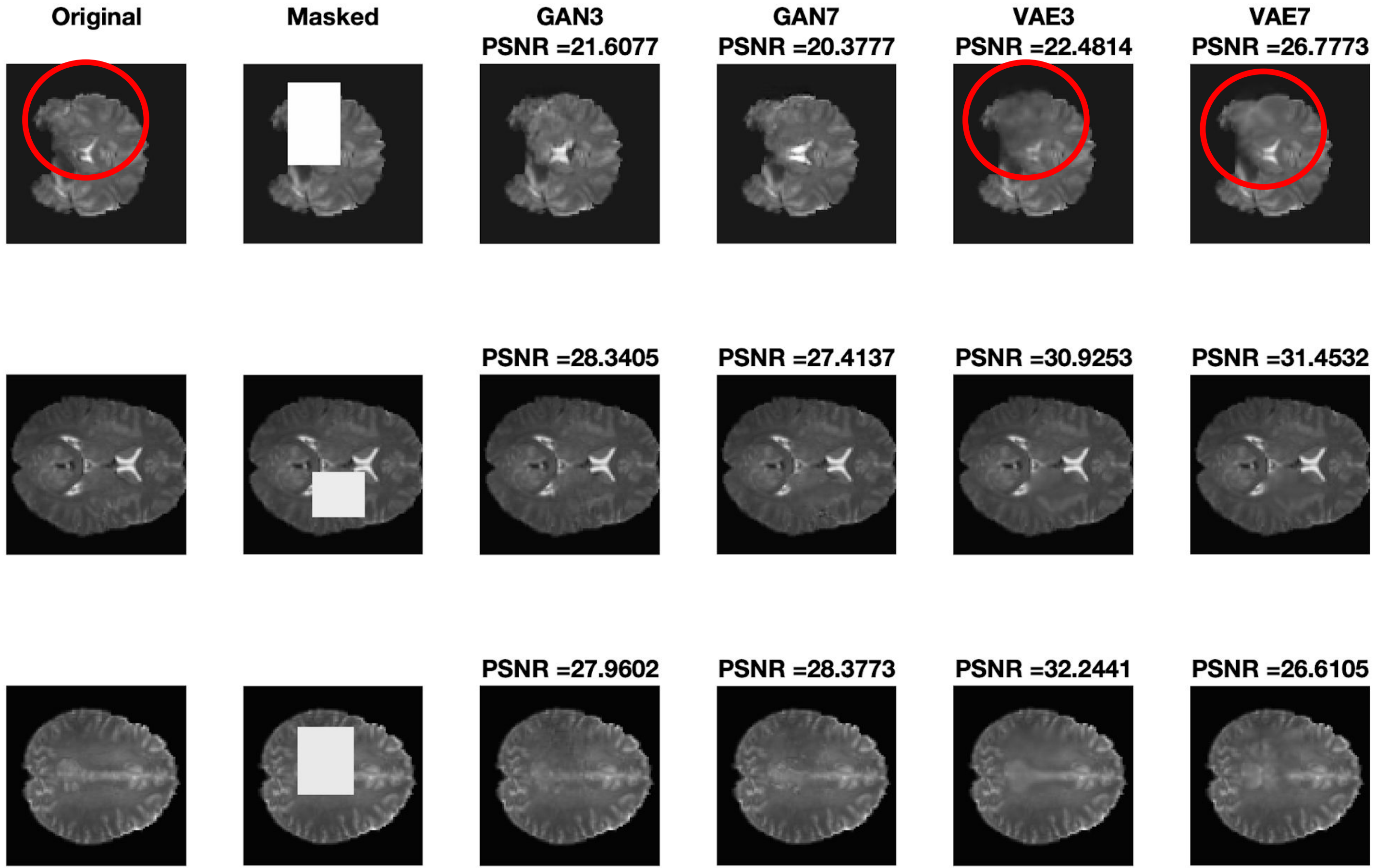
Similar trend with PSNR as well as SSIM measure.


Original
Masked
GAN3
PSNR =26.4462
GAN7
PSNR =24.0594
VAE3
PSNR =26.723
VAE7
PSNR =27.0105
PSNR =27.6258
PSNR =28.4821
PSNR =28.7554
PSNR =28.5177
Missing tail is almost fully recovered!
PSNR =22.8229
PSNR =21.6126
PSNR =23.0421
PSNR =22.8702

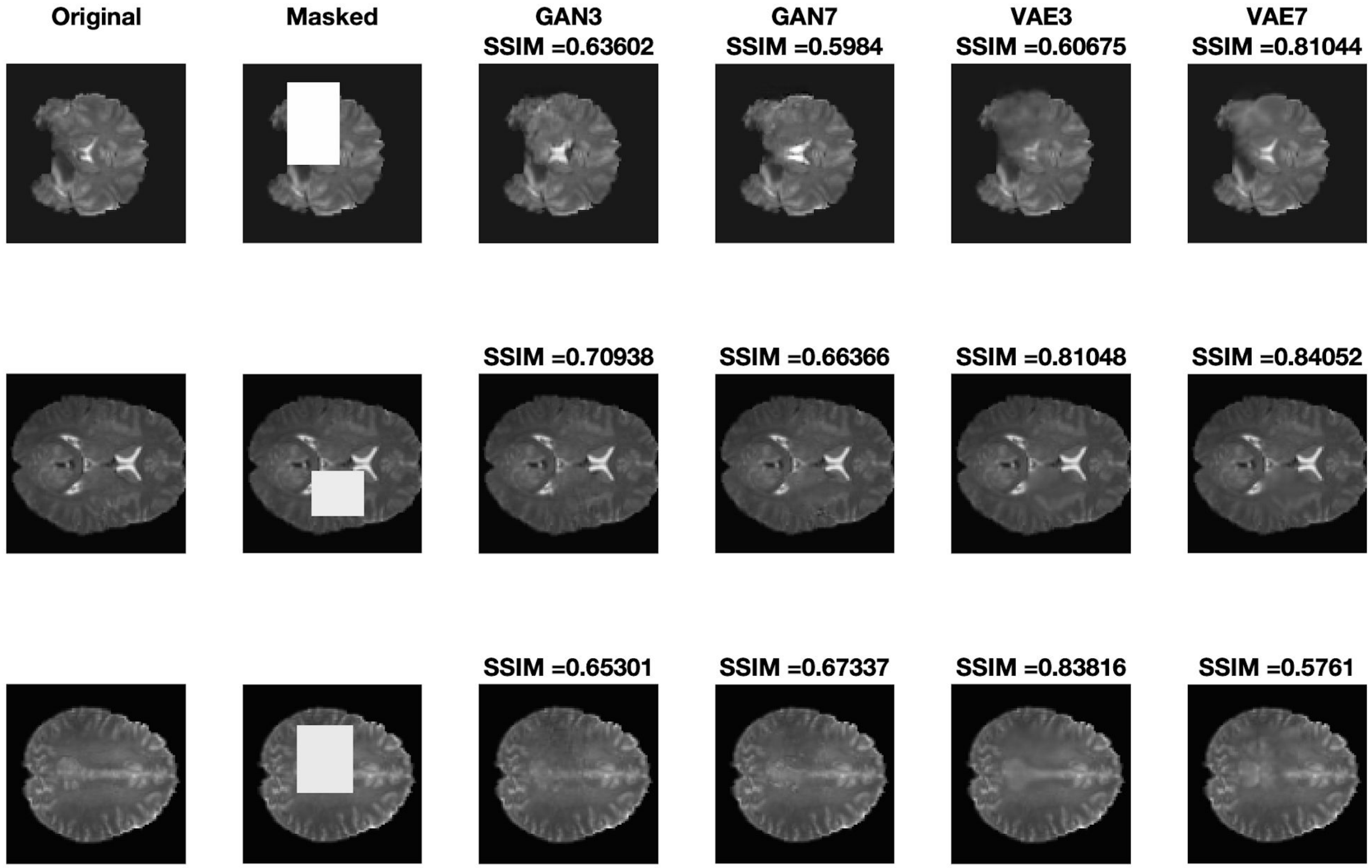


Able to inpaint any part of any slice of the brain irrespective of the patch size!

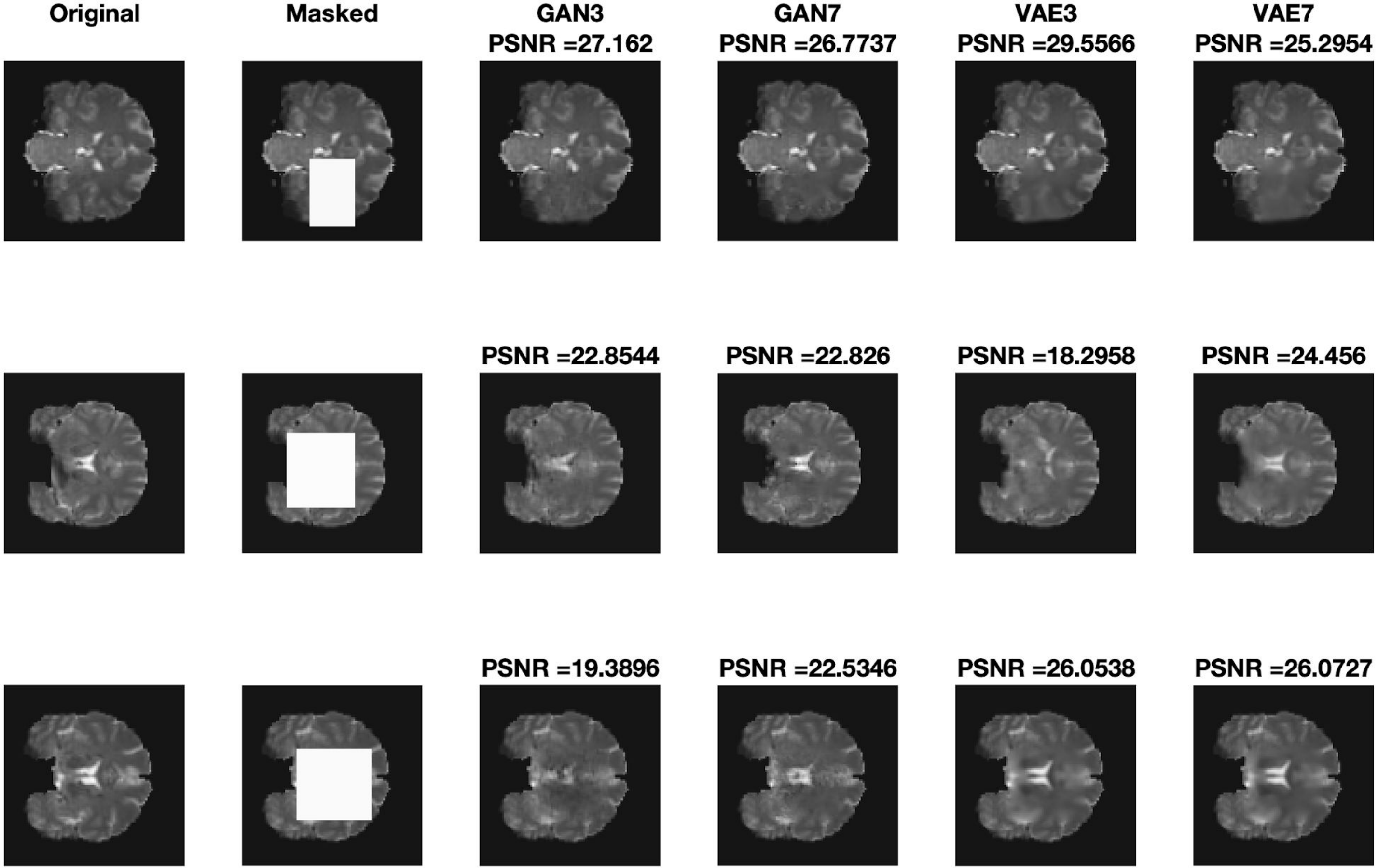
Comparison of GANs and VAEs with large masks and kernel sizes 3 and 7



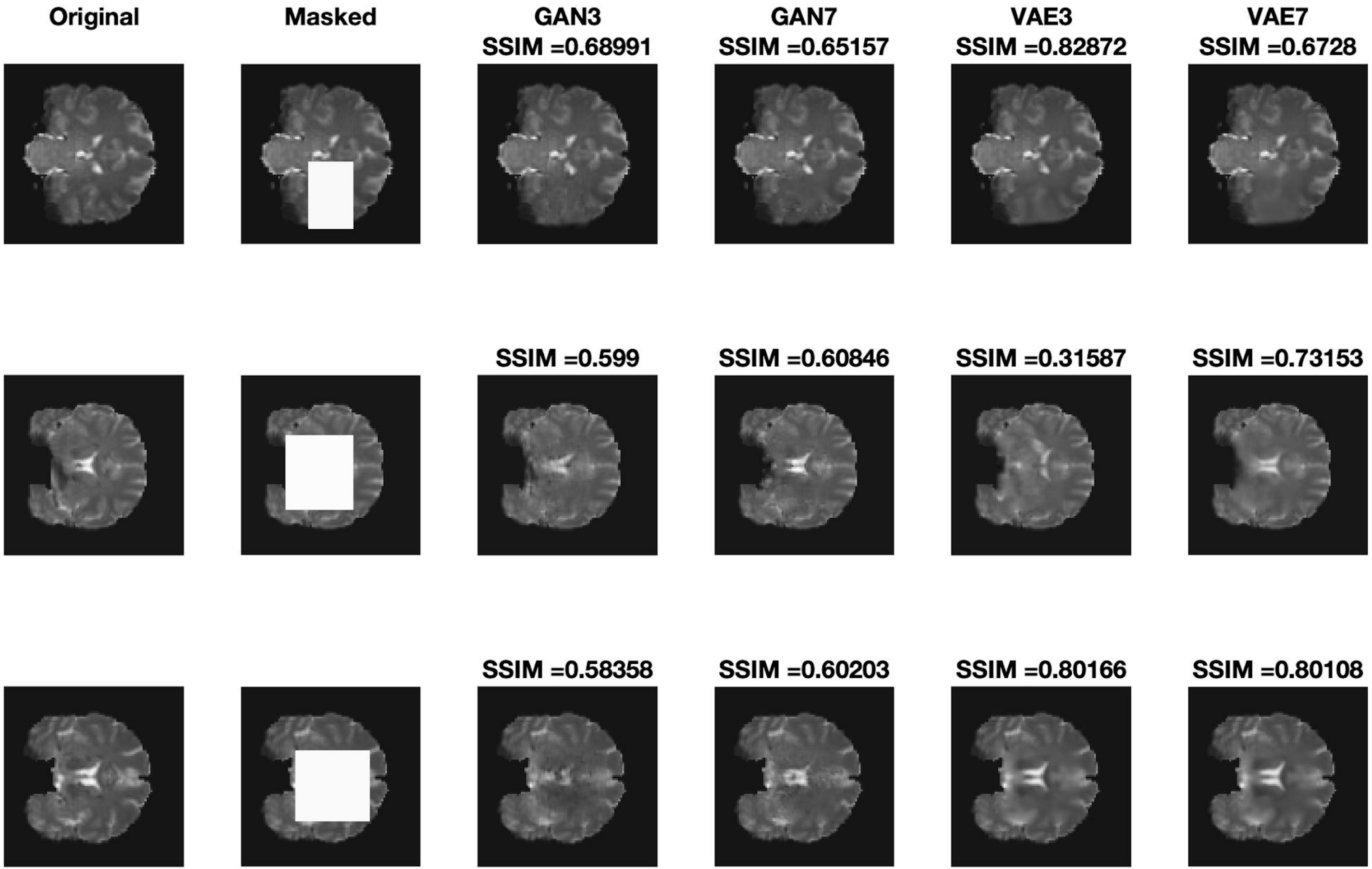
With larger patches, some fold structure is observed to be missing!



Original
Masked
GAN3
SSIM =0.63602
GAN7
SSIM =0.5984
VAE3
SSIM =0.60675
VAE7
SSIM =0.81044
SSIM =0.70938
SSIM =0.66366
SSIM =0.81048
SSIM =0.84052
SSIM =0.65301
SSIM =0.67337
SSIM =0.83816
SSIM =0.5761

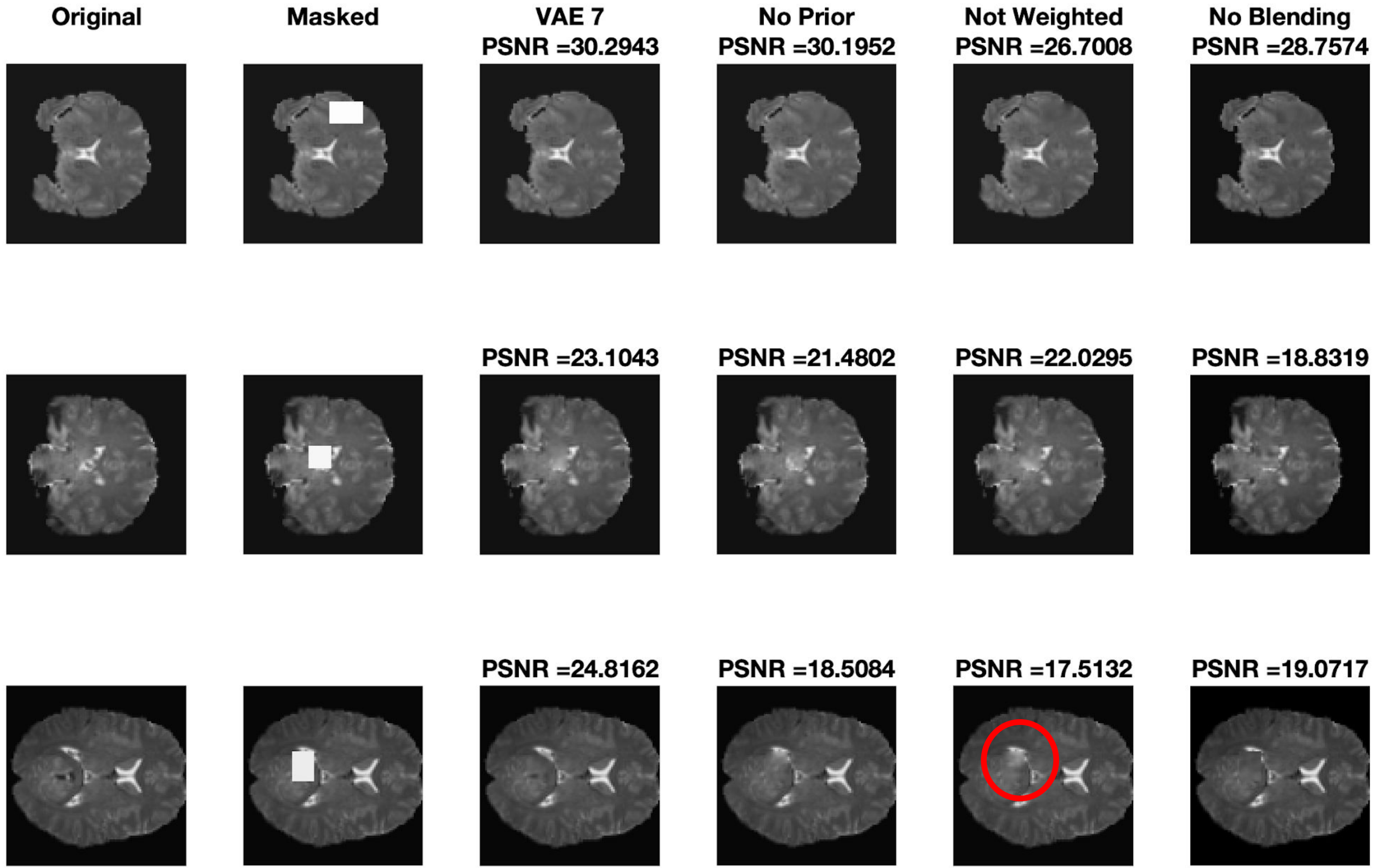


Almost completely occluded images recovered reasonably well!

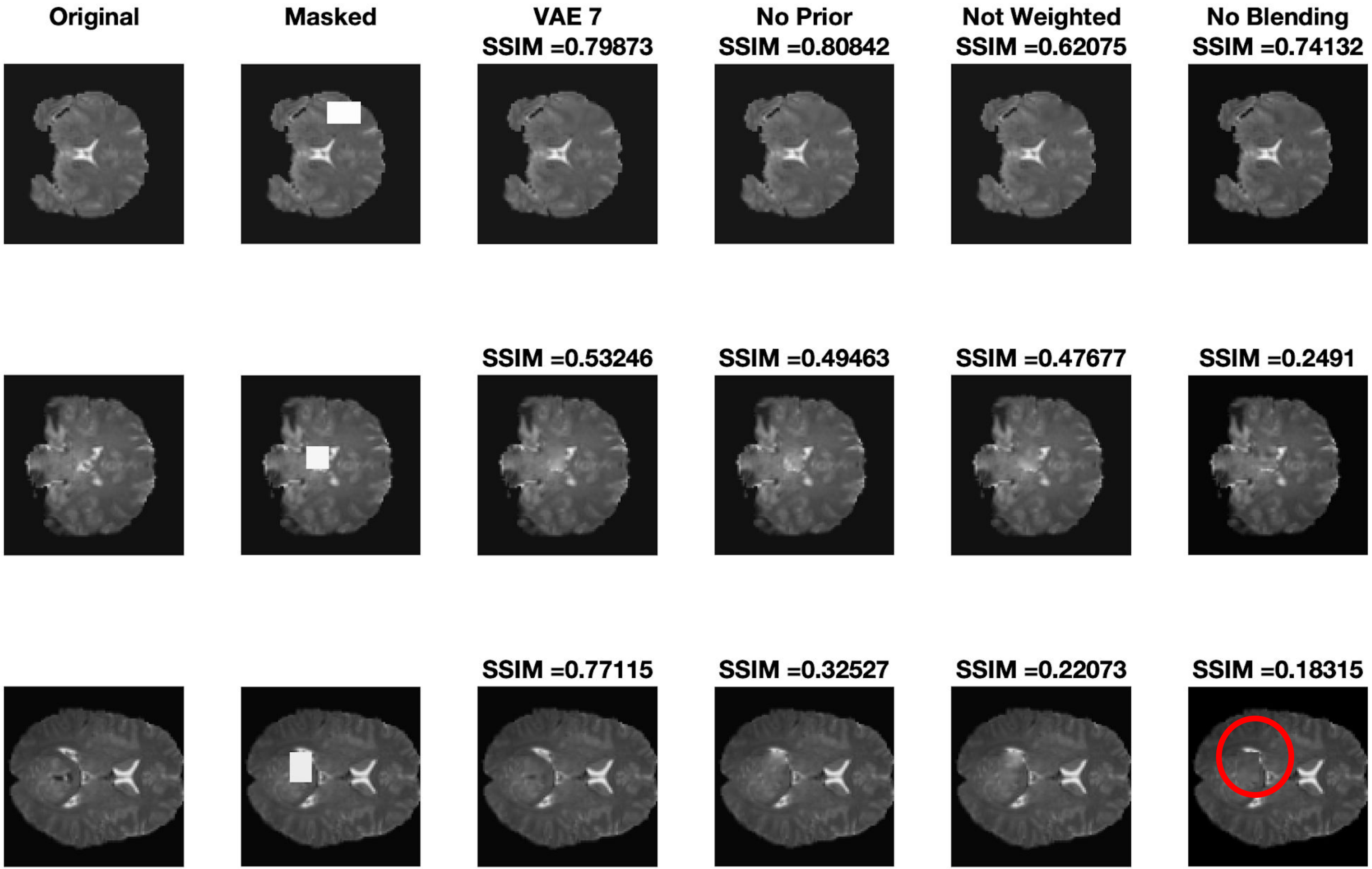


VAEs continue to perform better, even with larger patches.

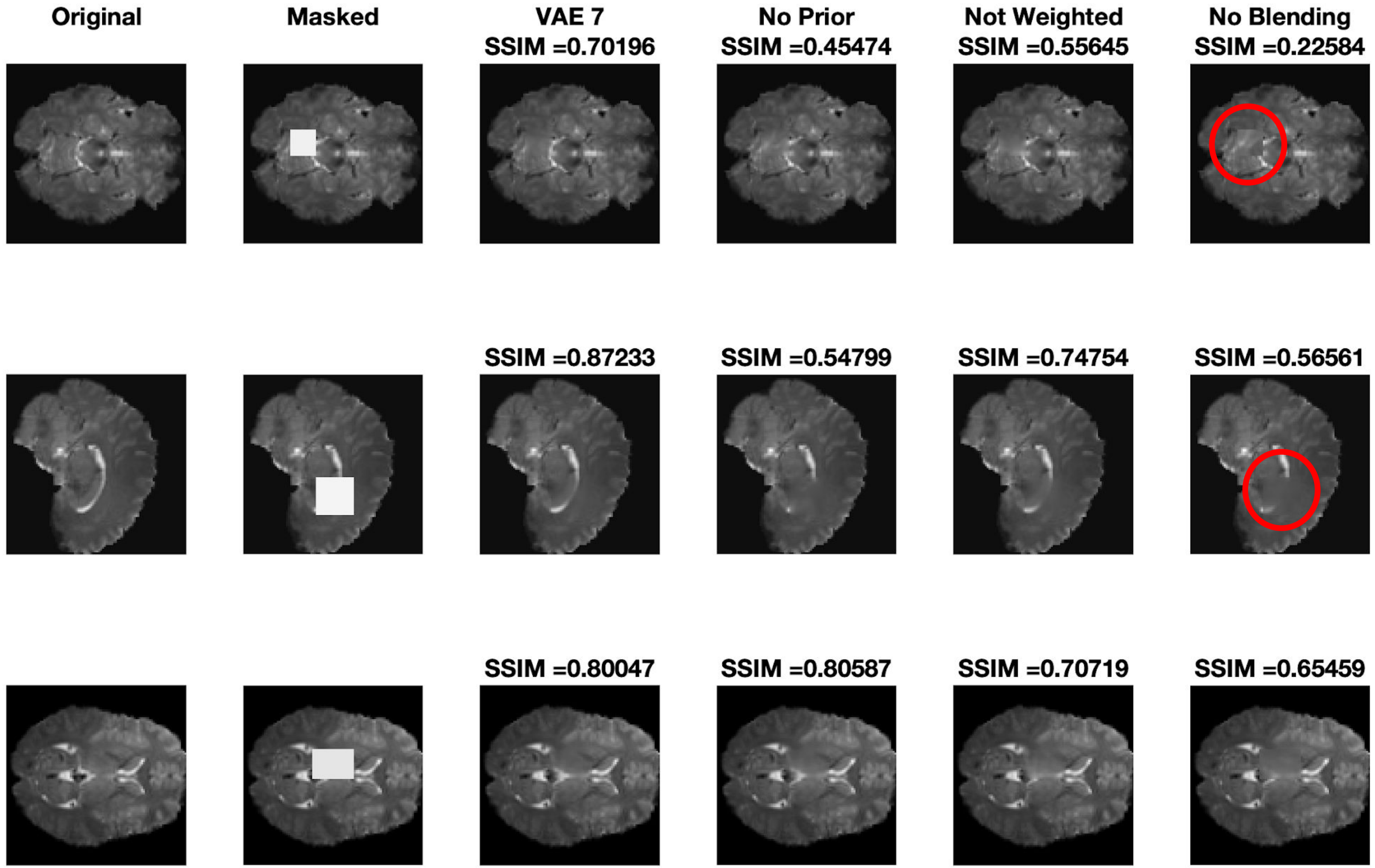
VAE 7 : Demonstrating effect of prior loss, weighted context loss and blending

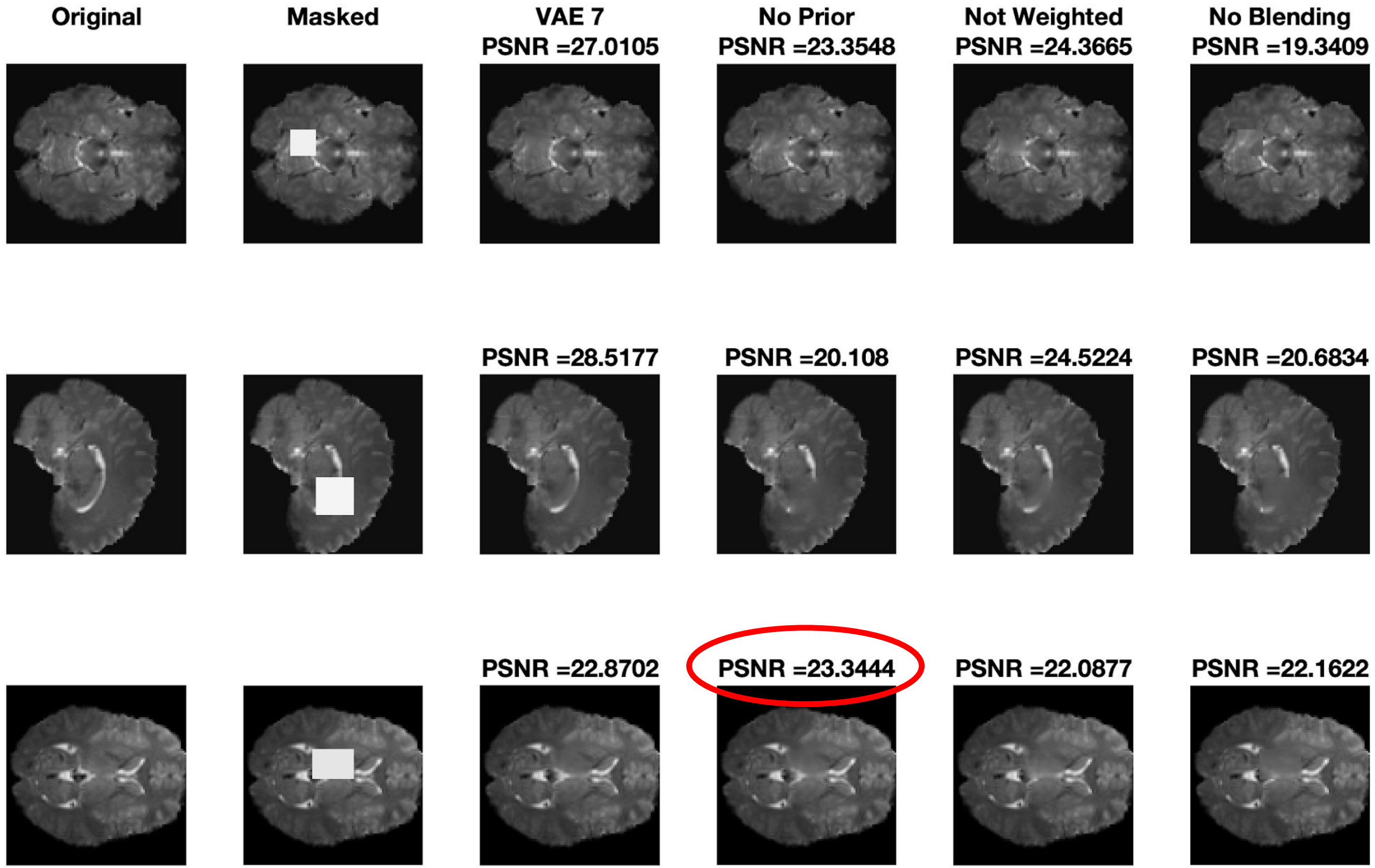


Prior, Weighted loss and blending all improve the result quality!

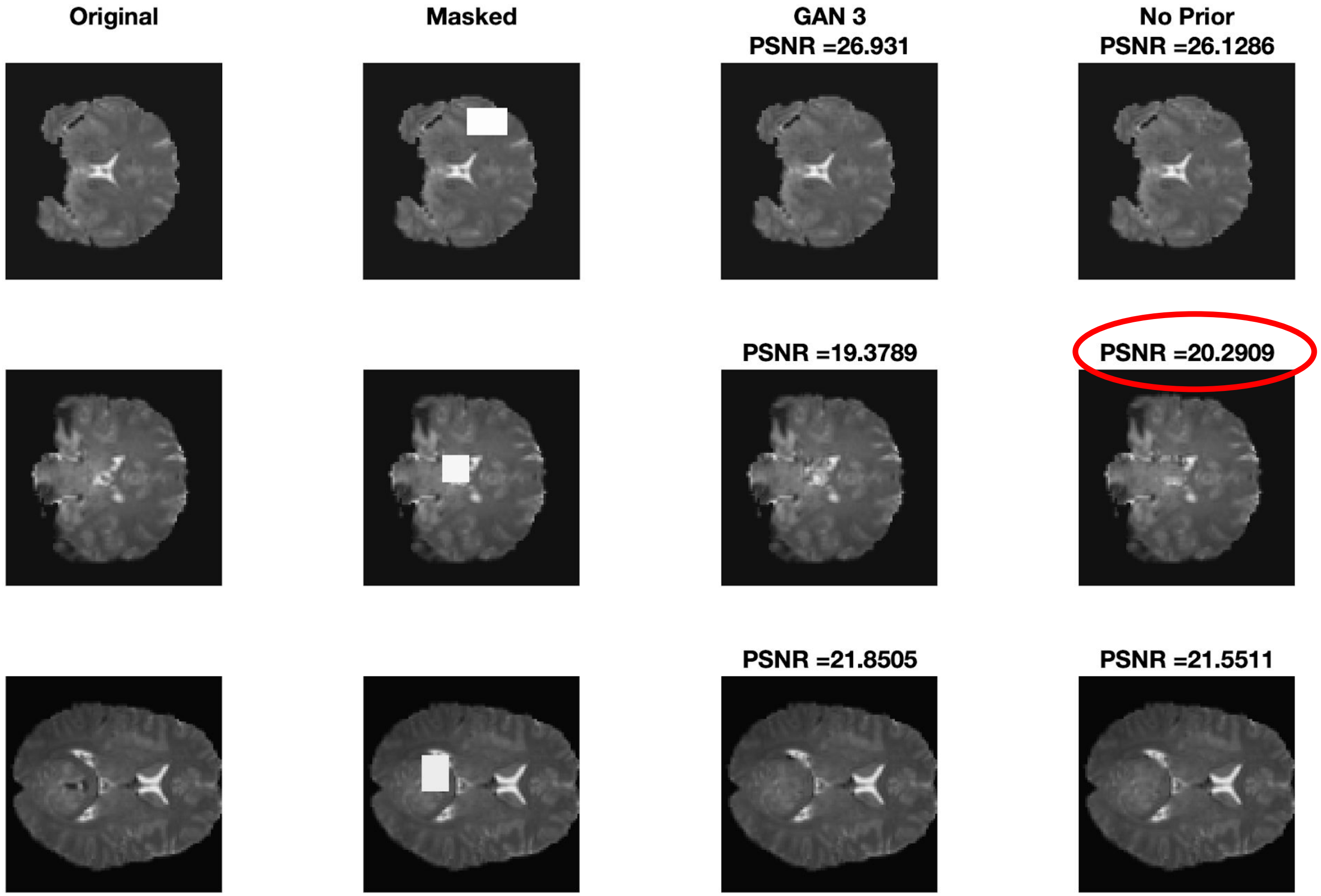


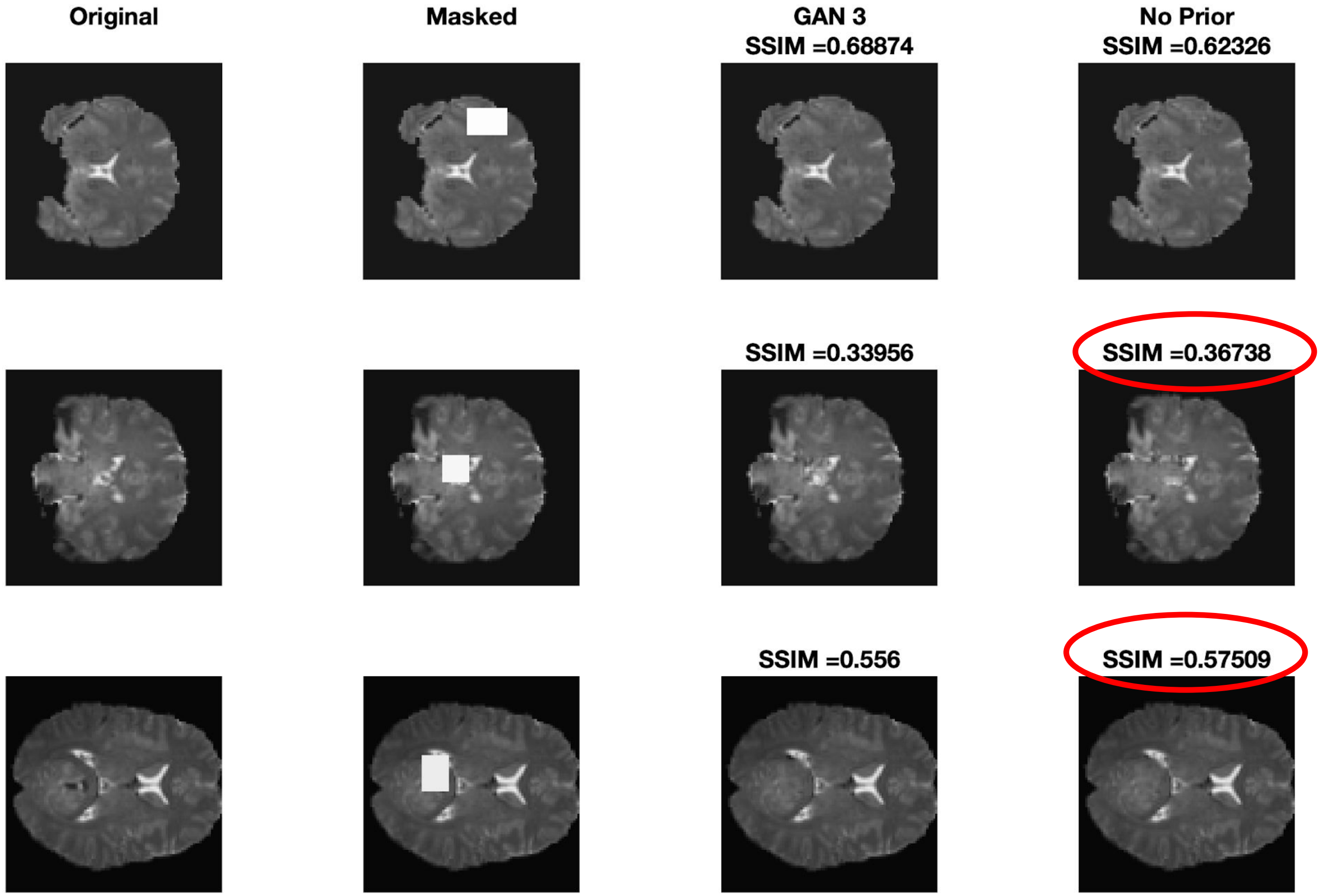
Patch structure visible when blending is not used - discontinuity along patch boundary

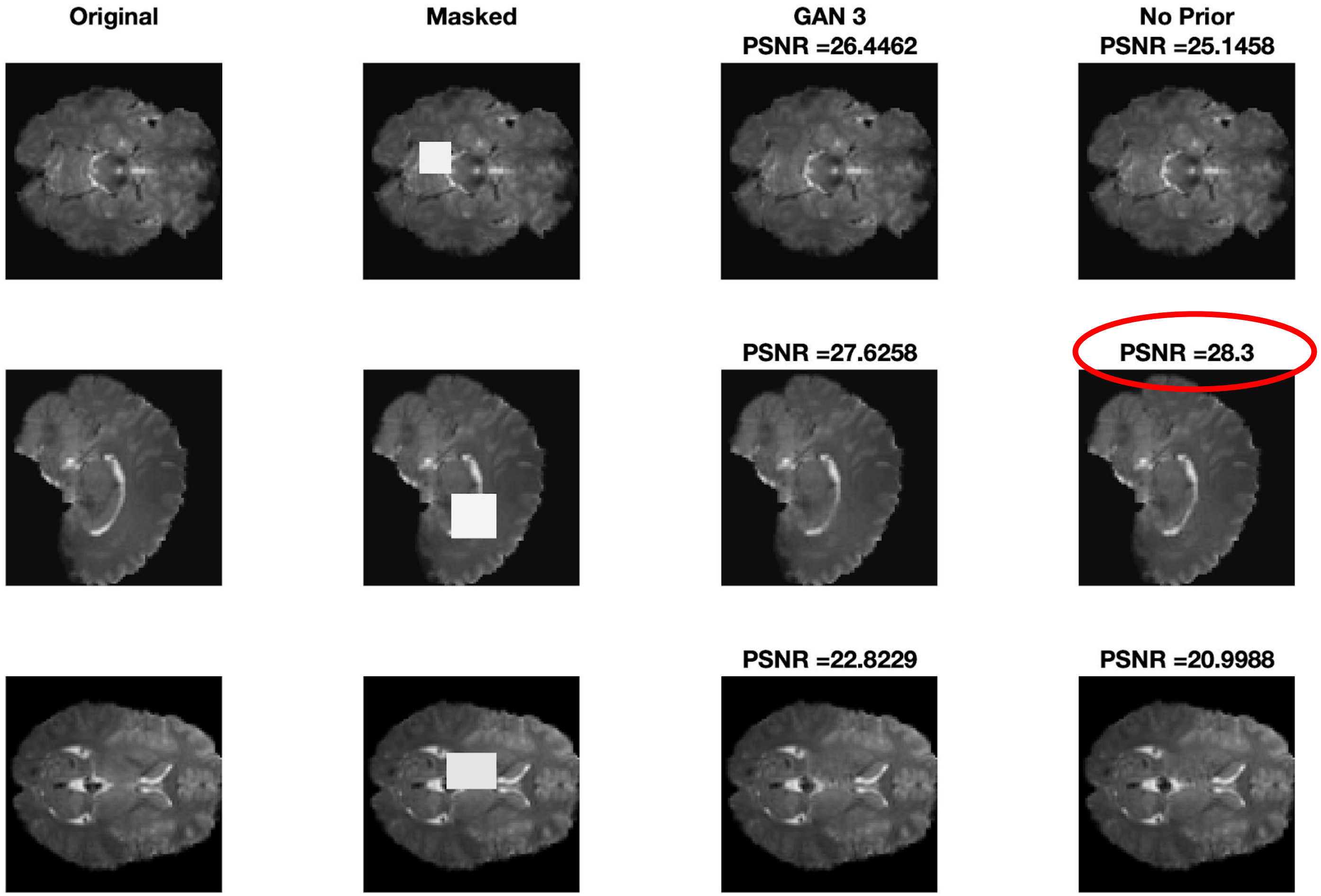

Original
Masked
VAE 7
SSIM =0.70196
No Prior
SSIM =0.45474
Not Weighted
SSIM =0.55645
No Blending
SSIM =0.22584
SSIM =0.87233
SSIM =0.54799
SSIM =0.74754
SSIM =0.56561
SSIM =0.80047
SSIM =0.80587
SSIM =0.70719
SSIM =0.65459

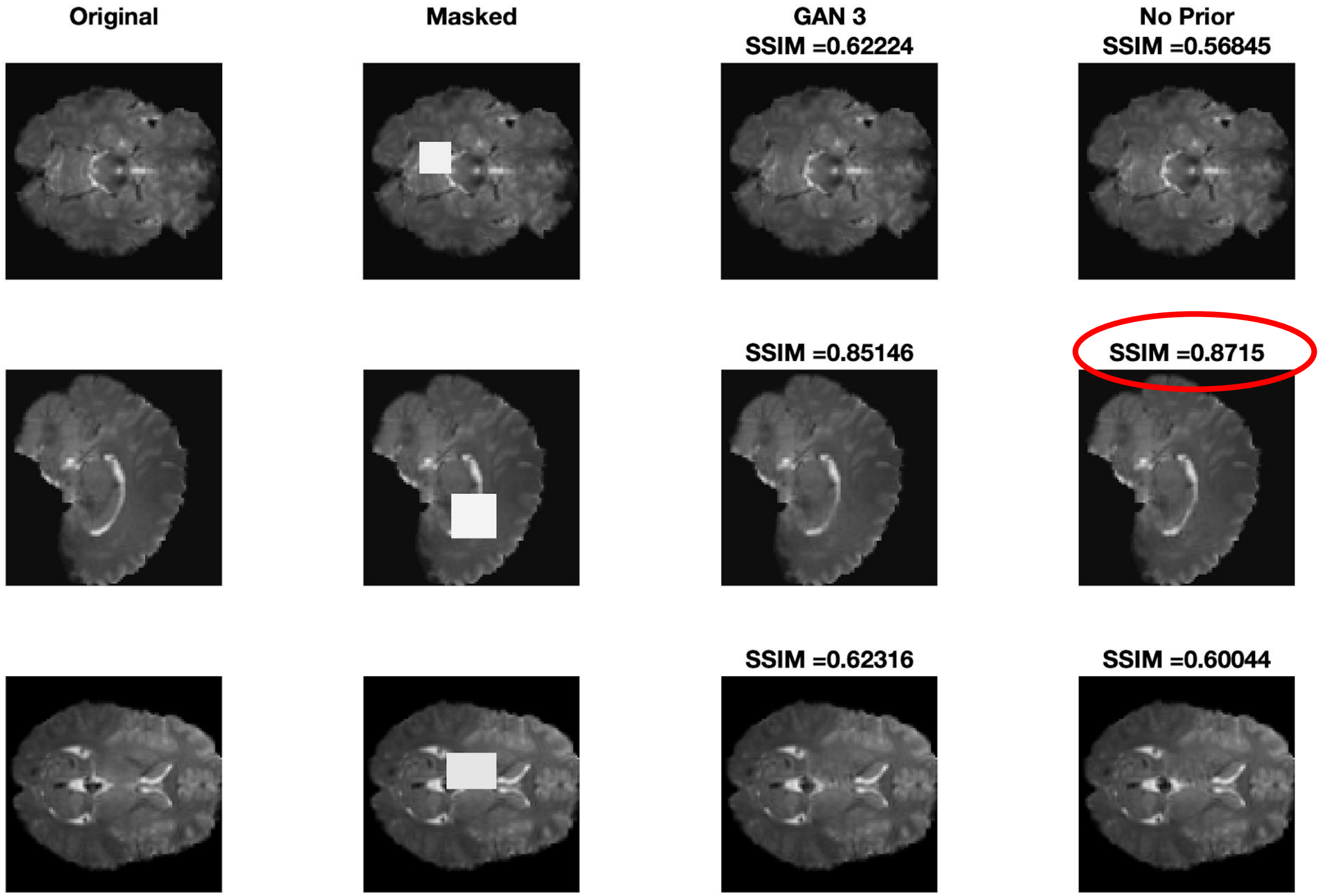

Original
Masked
VAE 7
PSNR =27.0105
No Prior
PSNR =23.3548
Not Weighted
PSNR =24.3665
No Blending
PSNR =19.3409
PSNR =28.5177
PSNR =20.108
PSNR =24.5224
PSNR =20.6834
PSNR =22.8702
PSNR =23.3444
PSNR =22.0877
PSNR =22.1622

GAN 3: Effect of prior loss


Original
Masked
GAN 3
PSNR =26.931
No Prior
PSNR =26.1286
PSNR =19.3789
PSNR =20.2909
PSNR =21.8505
PSNR =21.5511


Original
Masked
GAN 3
SSIM =0.68874
No Prior
SSIM =0.62326
SSIM =0.33956
SSIM =0.36738
SSIM =0.556
SSIM =0.57509


Original
Masked
GAN 3
PSNR =26.4462
No Prior
PSNR =25.1458
PSNR =27.6258
PSNR =28.3
PSNR =22.8229
PSNR =20.9988


Original
Masked
GAN 3
SSIM =0.62224
No Prior
SSIM =0.56845
SSIM =0.85146
SSIM =0.8715
SSIM =0.62316
SSIM =0.60044

Conclusions

- VAE worked better than GAN in most cases. Why?
  - VAE is directly trained on real images.
  - VAE realizes three clusters faster!
  - Trained in 25% less epochs, each consumed 25% less time. VAEs are 78% faster to train! Improvement over method used in the paper.
  - Maybe, GANs are better than VAEs on face data though.
- We also confirmed the importance of
  - Prior loss
  - Weighted context loss
  - Blending
- Advantage due to prior loss more clearly observed in VAEs than GANs.