MRIDiff: Image Synthesis of Brain MRI Scans with Denoising Diffusion Probabilistic Model

Zesheng Jia Faculty of Science

Abstract

We are proposing a new model that applies denoising steps by UNet for improving the diffusion process to improve the result of medical image synthesis on Brain MRI scan Dataset.

1. INTRODUCTION

Image synthesis is widely applied by GANs for a few years, the result of GANs on medical images are relatively satisfied but lack of flexibility and the precision in image details [1]. The raising of Diffusion models are showing the supreme outperformance than GANs in Text-to-image generation tasks, like DALL-E [2]. The images that generated by Diffusion models are increasingly improving and have a trending that is even better than human being's creations.[3] However, the original Diffusion Models are suffering from slow reverse inference time cost and the lack of stability of the result in each Markov Chain. This is not ideal in medical domain image generation. Hence, here we applied a new Denoising Diffusion models that can be adjusted and lower the noises in every few steps in the reverse inference process. [4] are showing delightful results by increasing step size T with this idea. And hereby, we wish to use admitting a progressive lossy decompression scheme that can be interpreted as a generalization of autoregressive decoding [5] to further improve the performance of [4]'s model.

2. LITERATURE SURVEY

In recent years, people are devoted to using Deep learning models to solve the computer vision tasks in Medical domain's realistic problems. Image synthesis is a crucial part of those problems. MRI, radiology, and CT scans' synthesis are showing good results by GANS [6]. In that paper, they used two state-of-art methods in 2018 for MRI image synthesis. First, they used Replica method that estimates a nonlinear mapping between image patches from the source contrast onto the individual voxels from the target contrast [16]. Then they used Multimodal method that uses an end-to-end neural network to estimate the target image given the source image as input. By applying this two-step GANs model, they successfully achieved $0.952 \pm .018$ SSIM score.

3. PROBLEM STATEMENT

Currently, rare diseases' medical scans are very hard to obtain in world-wide range. Most of time, those images are belong to patients themselves or some of the best hospitals, who would like not to share. Hence, by using image synthesis from limited amount of rare medical scans that are already publicly published, we can create more scans that are related to certain diseases, which are unrestricted by the confidential agreement between two parties. And using those model generated images, we can help doctors or medical students to learn those diseases without too many difficulties.

4. DATASET

We use kaggle dataset Brian Tumor MRI to set the baseline images for our model to inference. This dataset contains 4 types of images, Glioma Tumor MRI scans, Meningioma Tumor Scans, Healthy Brain Scans, and Pituitary Tumor Scans (As in Figure 1). In total, we have 3264 different Brian MRI scan images.

5. PROPOSED TECHNIQUE

5.1. Summary

From Figure 2, we are showing the process of how originally diffusion models work in image syn-

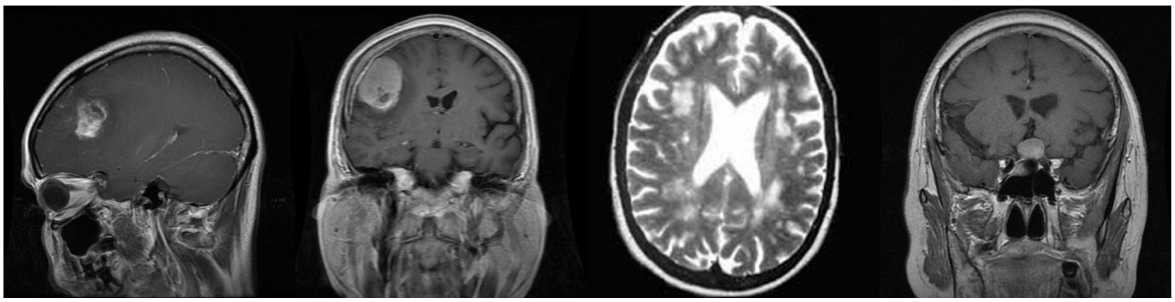

Figure 1. MRI Scan images

Four different types of Brain MRI Scan from Left to Right: Glioma Tumor Scan, Meningioma Tumor Scan, Healthy Brain Scan, and Pituitary Tumor Scan.

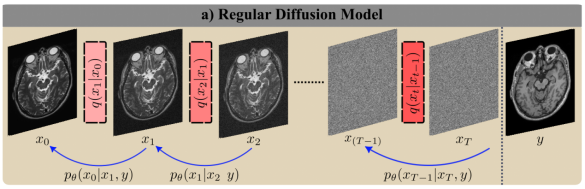

Figure 2. Regular Diffusion models inference process from [4]

Adding noises in each step of Markov Chain process until the data distribution converges to a given prior, i.e., standard Gaussian distribution [10].

thesis problems. We use our target image and add random Gaussian noise in each step to make our MRI scan into a purely noisy image that has the same distribution with a Gaussian noise distribution.

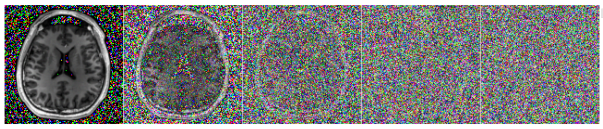

Figure 3. Demonstration of adding Gaussian Noises into our MRI image

Forward pass procedure on one Brain MRI scan image.

However, the original method are suffering from arbitrary results from a given Gaussian distribution prior by the reverse path. So [5] proposed a novel idea to use UNet model that combines with ResNet, attention, positional embeddings, and residual connection to downsampling the input images into a much smaller dimension matrices. Then the model upsamples the matrices into a segmentation feature map that captures the most import features of our input, as the autoregressive decoding method [5]. In our model, we will use this technique with huber loss function to regenerate those four types of images in our MRI dataset.

5.2. Detailed Model Structure

First we define the same function of position embeddings like the one in the Transformers paper [17]. The position of each step in the forward pass and backward pass is important for the model to know when and where to stop. This makes the neural network "understand" at which particular time step (noise level) it is operating, for every image in a batch. And in the next step, we use CNN for down sampling the input images from pure RGB values to the segmentation maps. But for recognizing the images values, we use two fundamental CNN block as ResNet [18] and ConvNext block [19]. And By the popular and sucess of Transformers [17] bring us, we also add attention module in our model. For measuring and retrieving the most important features from all images feature maps. And finally, we define our conditional U-Net model structure. As shown in the Figure 4.

1. Add one convolutional layer on the batch of images that are added noise and positional embeddings on each step during the markov chain.

2. And we use 2 ResNet/ConvNext blocks with group norm + attention layer + residual connection + downsample operation for downsampling the original image RGB inputs.

3. Then we use ResNet or ConvNext blocks with attention again for further downsampling.

4. After we get into the lowest dimension of our downsampling stage, we appling upsampling by the same combination of blocks, such that CNN blocks + group norm + attention and residual connection. All those steps are for upsampling the features we got before.

5. At the end, we apply one ResNet + convolutional layer for getting the output segmentation map.

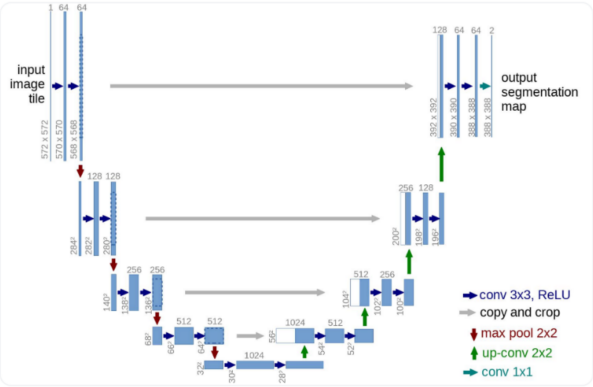

Figure 4. Demonstration of UNet structure

We downsample the input values from one image to a 1D array with 1024 dimension then upsampling the linear projection into a 388 by 388 output segmentation for further inference.

5.3. Inference and Sampling

Here, we introduce the timesteps. From the original paper, it is the number of denosing or add nosing steps of one simple forward or backward diffusion procedure. The more timesteps it has in one step, the more clearly the image could be. In the original paper, they use 2000 timesteps for one pass. Here, in our project, due to the limiation of my GPU resources. We set the timesteps as 300. Then We start from time T = 300, and we sample pure Gaussian noise values, then we use our UNet model to gradually denoising it, until the timesteps equals to 1 at the beginning.

In the best case, we will get a new and original MRI scan from the random Gaussian noise as the model generate a brain scan for us.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \mathbf{z}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 5. Sampling algorithms from [5]

We use for loop and the p-sample algorithm

$$x = \frac{1}{\sqrt{(\alpha_t)}}(X_t - \frac{1-\alpha_t}{\sqrt{1-\hat{\alpha}_t}} Z_\theta(X_t, t)) + \sigma_t z$$

to denoising the Gaussian random noise distribution at T = 300 to our target MRI scan image at T = 0.

6. EVALUATION METRICS

In this project, we used Huber loss as the loss function and evaluation metrics. The Huber loss function describes the penalty incurred by an estimation procedure f. Huber (1964) defines the loss function piecewise by

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \le \delta \\ \delta \cdot \left(|a| - \frac{1}{2}\delta\right), & \text{otherwise} \end{cases}$$

There is a reason why we didn't use SSIM for my evaluation metrics. Due to the very limited GPU power on training our model. The image was limited by a very small resolution as we discussed in the next section. That made our model only can capture the overall shape. If we can use more power GPU with more GPU memory, we can get more satisfied result. Hence, only at that point, the SSIM evaluation metrics is meaningful.

7. PERFORMANCE EVALUATION AND RESULTS

7.1. Training Result

In this project, we tried dozens of different hyperparameter combinations until finally we successfully got the model started to train. We used one Nvidia Tesla T4 with 16G GPU memory and continued to train different model settings for more than 80 hours before we called it the end.

For the best model by Figure 6, we set the image size as 44 by 44 pixels, channels = 1 as only

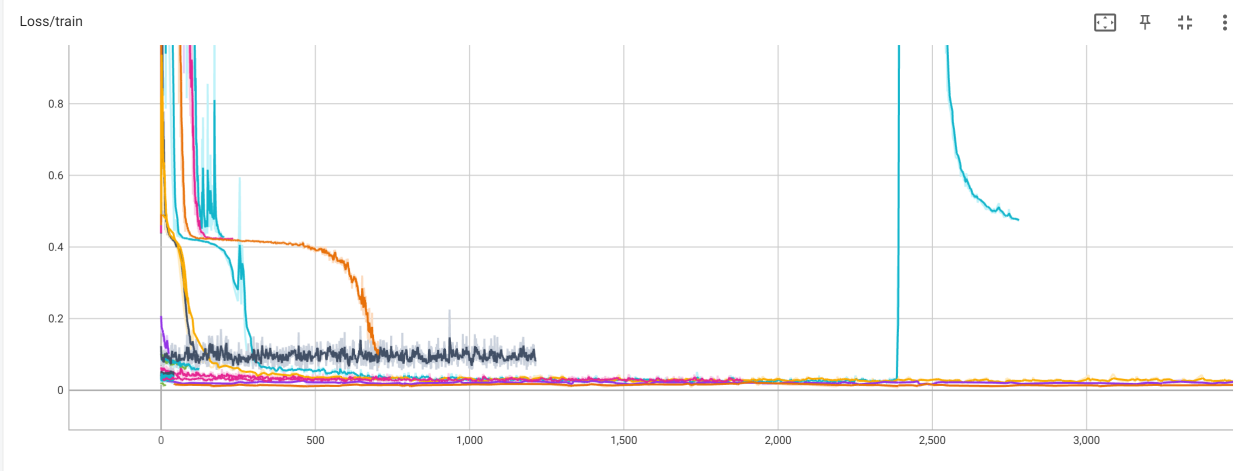

Figure 6. Validation Loss graph of all models' training steps
Different hyperparameter makes big difference on the Huber loss.

black and white image, and batch size for parameter updating as 80. We used Adam optimizer with learning rate 1e-3 and maximum 8000 epochs during the whole process. The minium huber loss we got from the best model is 0.0126018. And if any loss values less than 0.015, the model has no meaningful result. The shape of generated image is nothing like the target image.

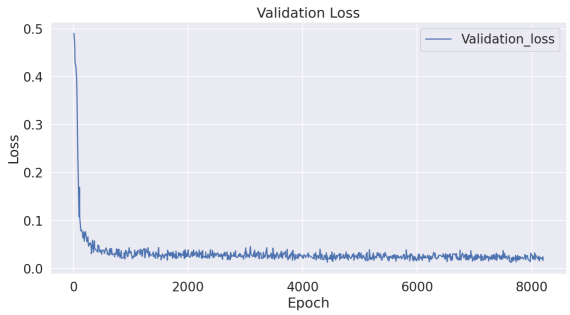

Figure 7. Validation Loss during the training of image size 42, batch size 80

The model improving very fast at the beginning, and turn to slow after a few hundreds epochs. But it still keep improving very slowly.

And for all the models we tried by Figure 6, the huber loss were so different on different hyperparameter. We found that if we increase the image size, then we must also increase the batch size. The ratio of (Image size / batch size) must be sustained. If the ratio is decreasing in the model design, the model won't improve after no matter how many epochs we gave to it. This took me more than 4 hours training during the night on a higher resolution target image hyperparameter setting.

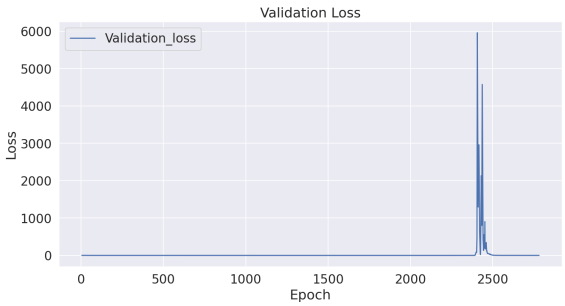

Figure 8. Validation Loss during the training of image size 72, batch size 40

The model improving very fast at the beginning, and turn to slow after a few hundreds epochs. But it still keep improving very slowly.

For one certain training with image size = 72 and batch size = 40, which is the best hyperparameter we can get from the Google Colab Free resources. In this model by Figure 8, we trained more than 2200 epochs, then we got a surge of Loss at 2450 epoch up to nearly 6000 loss value. However, during the first 2000 epochs, the loss values are around 0.025 by Figure 9.

Hence, for the training result by comparing those two model settings by Table 1, we finalized our hyperparameter on Nvidia Tesla T4 with image size 42, batch size 80, channels 1, and Adam optimizer with learning rate 1e-3. Base on this setting, now we will show the inference results.

Table 1. Models Validation Loss

Model setting	count	mean	std	min	25%	50%	75%	max
Image size 44, batch size 80	8000	0.0312	0.0381	0.0126	0.0224	0.0262	0.0306	0.4902
Image size 72, batch size 40	2600	23.5068	272.1552	0.0164	0.0266	0.0327	0.068	5959.6943

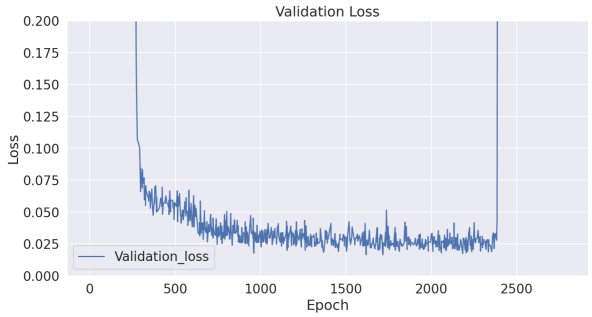

Figure 9. Validation Loss of y axis from 0 to 0.2 of image size 72, batch size 40

We can see that the validation loss is decreasing but suddenly had a surge around 2040 epochs.

7.2. Inference Result

In this section, we take 3 different sampling results with the model generated images form epoch 0, 120, and 585. The rest of epochs have similar human evaluation performance, the only difference is the Huber loss was still decreasing.

For the initial model without any training epochs, we have the inference sampling result at Figure 10. We can see that there is only random Gaussian noises as it suppose to be. At epoch 120, the shape of human head was becoming more clearly. We can see that the contour of human skull and some shape of human mouth or jaw. At epoch 585, we can clearly see the four different types of human Brain tumor or healthy brain MRI scans. The human eyes, skulls or even the Amygdala is clear to observe. I didn't end up here, but the rest of the epochs are lower the Huber loss, but there is no big difference than the result at 585 epochs.

8. CONCLUSION AND FUTURE SCOPE

From this project, we can see that the potential of generating rare diseases with small number of images are valid. The Diffusion model gave us a satisfied result even on a small resolution image input. And by testing and trying all the combinations of hyperparameter, we can find one setting that can

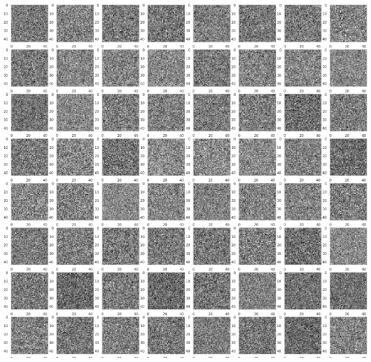

Figure 10. Inference result on Epoch 0
We can see that there is only exist pure Gaussian random noises.

Figure 11. Inference result on Epoch 120
Model start to learn the shape of the input images.

Figure 12. Inference result on Epoch 585
The model already captured the essence of Human Brain MRI Scan.

be trained and improved by thousands of epochs. In the meantime, improving the image resolution need more computation resources is one obstacle that I currently have, but not for the whole world or further investigation with funding. And we also noticed that the certain ratio of (image size / batch size) is a critical training signal that will indicate whether the model will successfully train or not.

Therefore, in the future if we still have chance to continue this project, I would like to use more powerful GPU with different hyperparameter settings for getting a better result.

References

[1] H. Huang, P. S. Yu, C. Wang, 'An Introduction to Image Synthesis with Generative Adversarial Nets'. arXiv, 2018.

[2] A. Ramesh . ., 'Zero-Shot Text-to-Image Generation'. arXiv, 2021.

[3] J. Yu . ., 'Vector-quantized Image Modeling with Improved VQGAN'. arXiv, 2021.

[4] M. Özbey . ., 'Unsupervised Medical Image Translation with Adversarial Diffusion Models'. arXiv, 2022.

[5] J. Ho, A. Jain, and P. Abbeel, 'Denoising Diffusion Probabilistic Models'. arXiv, 2020.

[6] S. U. Dar, M. Yurt, L. Karacan, A. Erdem, E. Erdem, and T. C¸ ukur, "Image synthesis in multi-contrast MRI with conditional generative adversarial networks," IEEE Trans. Med. Imag., vol. 38, no. 10, pp. 2375–2388, 2019.

[7] B. Yu, L. Zhou, L. Wang, Y. Shi, J. Fripp, and P. Bourgeat, "Ea-GANs: Edge-aware generative adversarial networks for cross-modality MR image synthesis," IEEE Trans. Med. Imag., vol. 38, no. 7, pp. 1750–1762, 2019.

[8] A. Sharma and G. Hamarneh, "Missing MRI pulse sequence synthesis using multi-modal generative adversarial network," IEEE Trans. Med. Imag., vol. 39, pp. 1170–1183, 2020.

[9] G. Wang et al., "Synthesize high-quality multi-contrast magnetic resonance imaging from multi-echo acquisition using multi-task deep generative model," IEEE Trans. Med. Imag., vol. 39, no. 10, pp. 3089–3099, 2020.

[10] L. Yang . ., 'Diffusion Models: A Comprehensive Survey of Methods and Applications'. arXiv, 2022.

[11] F. A. Fardo, V. H. Conforto, F. C. de Oliveira, P. S. Rodrigues, 'A Formal Evaluation of PSNR as Quality Measurement Parameter for Image Segmentation Algorithms'. arXiv, 2016.

[12] J. Nilsson T. Akenine-Möller, 'Understanding SSIM'. arXiv, 2020.

[13] M. Mirza S. Osindero, 'Conditional Generative Adversarial Nets'. arXiv, 2014.

[14] M.-Y. Liu, T. Breuel, J. Kautz, 'Unsupervised Image-to-Image Translation Networks'. arXiv, 2017.

[15] J. Ho, A. Jain, P. Abbeel, 'Denoising Diffusion Probabilistic Models'. arXiv, 2020.

[16] A. Jog, A. Carass, S. Roy, D. L. Pham, and J. L. Prince, "Random forest regression for magnetic resonance image synthesis," Med. Image Anal., vol. 35, pp. 475–488, 2017.

[17] J.A. Vaswani et al., 'Attention Is All You Need'. arXiv, 2017.

[18] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition'. arXiv, 2015.

[19] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, 'A ConvNet for the 2020s'. arXiv, 2022.

A. Appendix A: Team Contribution

This project is 100% individually done by author Zesheng Jia. There is no other party involved with.

Component	Team member
All/100% components	Zesheng Jia

B. Appendix B: Contribution Percentage

Team member	Contribution
Zesheng Jia	100%