Synthetic Data Generation for MRI and Tabular Data

Mohamed Radwan (SLATE930, UiB)

Abstract— **In this study, synthetic data generation methods are used in the domain of brain disorders data. Several methods are utilized on tabular and MRI data. The tabular data is used for dementia diagnosis while the MRI data is used for the study of brain tumors. In this study, the fidelity of the generated data are explored in details in order to evaluate the used models in generating realistic images and tabular data. Using the MRI data, Patch/Pixel based method called Pix2Pix model achieved the best results in terms of synthetic data fidelity. Pix2Pix model generated images with almost identical structural features to the real data. This is supported through studying the structural similarity index and Peak Signal to Noise Ratio of the synthetic and real images. For the Tabular data domain, GAN and VAE based methods are used on the tabular data which achieved the best synthetic data fidelity. There are certain limitations in this study regarding the size of the data, resolution of the images and clinical validation. A future possible direction could be through studying the utility and privacy of the synthetic data. The code, methods and results of the experiments in this report is shared on GitHub [1]**

Index Terms— **Synthetic, MRI, Dementia, Tabular**

I. INTRODUCTION AND BACKGROUND

Brain Disorders such as Dementia, Alzheimer Disease (AD) affect many people worldwide. Alzheimer is a neurodegenerative disorder that causes cognitive decline, memory loss and changes in behavior of patients. Alzheimer and Dementia disorders remain poorly understood in terms of its pathogenic mechanism [1]. Dementia and Alzheimer Disease can be studie using electroencephalogram (EEG), Magnetic Resonance Imaging (MRI) or Tabular data. In this study, the main focus will be on MRI and tabular data. Magnetic Resonance Imaging (MRI) is a useful diagnostic tool for Dementia and Alzheimer which should be used intensively by researchers to understand more about this disease pathology and develop machine learning models to help clinicians in the diagnostics.

The medical data are in general highly sensitive and private data. This introduces major limitations in developing machine learning models as these models highly depend on the availability of big data repository for training. One prominant solution for this challenge is the use of synthetic data. This way the original data can be protected and not shared while the synthetic data is used instead. Synthetic data generation has the capability to augment limited training datasets to improve model performance and provide privacy for the original data. Several approaches and variants are used in synthetic generations are mainly based on Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs) and diffusion models.

GANs can produce visually realistic MRI images but can be prone to hallucinations and mode collapse. The study [2] reports several methods for MRI and CT scans. The study [3] used Conditional-GAN by adding labels in medical image synthesis to add constraints to the models. Another study [4] used U-Net [5] based architecture combined with GANs to generate CT and MRI scans. U-Net [6] was used to generate synthetic CT images from MRI. CycleGAN [7] was applied on MRI and CT scans, however, anatomical inconsistencies were reported. Diffusion models such as DDPMs have recently outperfomed GANs in fidelity [8], [9]. Diffusion models make the data generation as the reversal of a noising process which produce robust results. These models show great promise in the medical imaging. However, This is beyond the scope of this study.

Several challenges remain open. One of the challenges is Preventing hallucinations and ensuring structural fidelity. Furthermore, rigorous Clinical evaluation is needed to evaluate the data. Furthermore, handling data leakage risks in synthetic data remain an open question in research.

Our Objectives in this study are:

- Building different Synthesizers for Dementia tabular data and comparing between the fidelity of the generated data for each of used models.
- Building different Synthesizers for MRI data on both the fine-grained patches/pixels and on large feature maps levels and comparing between the fidelity of the generated images.

The study will report results on generic MRI from brain tumor diagnosis. However, the same methods should be extended to MRI for dementia diagnosis. The tabular data is from dementia disease patients. The format of this study is as in the following: In the section II, the details about the chosen data is reported aligned with the motivation of study. Section III explains the used models in this study. Here, Six different models are used in generating synthetic data. Section IV reports the main results in this study summarized and compared through figures. Section V compare and summarize the findings in this study and discuss the fidelity of generated data from all the models. Section VI concludes the study main findings with limitations and future development.

II. USED DATA

In this study, two types of datasets were used. The first dataset is structured tabular dataset in which three tabular different synthetic generation models were used. The second data is unstructured medical MRI images in which three different models were used to generate synthetic data. Both the used data are publicly available for research purposes.

1) Dementia Tabular data: This data is from the the Open Access Series of Imaging Studies (OASIS) [2] from Washington University Alzheimer's Disease Research Center. OASIS project aims at making MRI data sets of the brain available to the scientific applications for future discovery and clinical uses. OASIS provides rich content with several versions of MRI and tabular data. These data is provided by [10] from OASIS and can be used for Alzheimer and Dementia diagnosis and detection. In this study, the focus on using the metadata content from MRI. This data has nondemented and demented subjects that covers a cross-sectional collection of 416 subjects aged 18 to 96. The dataset consists of both men and women.

2) MRI data: The MRI data in this study is shared by [11]. The dataset is hosted publicly on the Kaggle platform[3]. It consists of collection of brain MRI scans. The dataset is mainly for brain tumors detections. It includes labeled images with and without tumors. This dataset has 253 subjects in the two classes of subjects. 155 subjects have tumors and 98 subjects are healthy.

---

[2] https://sites.wustl.edu/oasisbrains/
[3] https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/data

```
============================================================
Layer (type:depth-idx)          Output Shape         Param #
============================================================
Generator                       [1, 5]               --
├─Sequential: 1-1               [1, 5]               --
│    └─Linear: 2-1              [1, 64]              704
│    └─ReLU: 2-2               [1, 64]              --
│    └─Linear: 2-3              [1, 128]             8,320
│    └─ReLU: 2-4               [1, 128]             --
│    └─Linear: 2-5              [1, 5]               645
============================================================
Total params: 9,669
Trainable params: 9,669
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 0.01
============================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.04
Estimated Total Size (MB): 0.04
============================================================
```

(a) Generator

```
============================================================
Layer (type:depth-idx)          Output Shape         Param #
============================================================
Discriminator                   [1, 1]               --
├─Sequential: 1-1               [1, 1]               --
│    └─Linear: 2-1              [1, 128]             768
│    └─LeakyReLU: 2-2          [1, 128]             --
│    └─Linear: 2-3              [1, 64]              8,256
│    └─LeakyReLU: 2-4          [1, 64]              --
│    └─Linear: 2-5              [1, 1]               65
│    └─Sigmoid: 2-6            [1, 1]               --
============================================================
Total params: 9,089
Trainable params: 9,089
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 0.01
============================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.04
Estimated Total Size (MB): 0.04
============================================================
```

(b) Discriminator

Fig. 1: General GAN architecture layers

```
============================================================
Layer (type:depth-idx)          Output Shape         Param #
============================================================
VAE                             [1, 5]               --
├─Sequential: 1-1               [1, 64]              --
│    └─Linear: 2-1              [1, 64]              384
│    └─ReLU: 2-2               [1, 64]              --
├─Linear: 1-2                   [1, 5]               325
├─Linear: 1-3                   [1, 5]               325
├─Sequential: 1-4               [1, 5]               --
│    └─Linear: 2-3              [1, 64]              384
│    └─ReLU: 2-4               [1, 64]              --
│    └─Linear: 2-5              [1, 5]               325
│    └─Sigmoid: 2-6            [1, 5]               --
============================================================
Total params: 1,743
Trainable params: 1,743
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 0.00
============================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.01
Estimated Total Size (MB): 0.01
============================================================
```

Fig. 2: General VAE architecture layers. The two sequential layers are the encoder and decoder, respectively.

III. METHODOLOGY

One of the main method used for generating synthetic data is the Generative Adversarial Network (GAN) [12]. GAN consists of two elements: Generator (G) and Discriminator (D) which are trained in an adversarial manner. The general architecture of GAN is as shown in Figure 1.

The generator aims to produce realistic samples from random noise z, while the discriminator aims to distinguish real data x from generated data $G(z)$. The general objective function consists of the discriminator loss which is:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{\text{data}}}\big[\log D(x)\big] - \mathbb{E}_{z \sim p_z}\big[\log(1 - D(G(z)))\big]$$

In the first term, discriminator wants to classify real data as 1. The second term, discriminator wants to classify fake data as 0. Generator wants to maximize $D(G(z))$ to fool the discriminator with the following loss:

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}\big[\log D(G(z))\big]$$

The other main method used for generating synthetic data is the Variational Autoencoder (VAE) [13]. VAE consists of encoder and decoder. The encoder maps input data to the latent representation and the decoder maps Maps latent representation back to reconstructed data. This is done by optimizing reconstruction Loss such as MSE between the original and reconstructed and KL Divergence loss to regularize latent space. KLD loss penalizes the encoder if it makes distributions that drift too far from the original.

A. Tabular Synthesizers

1) Conditional Tabular GAN (CTGAN): CTGAN was introduced by [14] as a GAN method designed for tabular data. This model can handle mixture of categorical and numerical features. CTGAN is simply a GAN [12] variant with several modifications. The generator outputs synthetic tabular rows where both continuous and categorical variables are generated. The Discriminator attempts to distinguish real from synthetic rows, while being conditioned on the same categorical variable used by the generator. Here, Gaussian Mixture Model (GMM) is used to transform Continuous columns to model skewed and multimodal distributions. CTGAN uses categorical variable for sampling and feed as a condition during training which outputs a synthetic row conditioned on that category. Here, CTGAN outputs valid combinations of categorical and continuous values In other words, CTGAN extends the standard GAN objective with conditional information from categorical variables.

2) Tabular Variational Autoencoder (TVAE): TVAE was introduced by [14] as VAE method for synthesizing tabular data with a mixture of categorical and continuous features. TVAE is a variant of the VAE model [13] with extra preprocessing steps for tabular datasets. So, TVAE has encoder and decoder. Here, continuous variables are used directly while categorical variables are represented with one-hot encoding and used as a softmax layer in the decoder. The loss function is a combination of MSE for Continuous variables and Cross-entropy loss for categorical variables. This means that training objective are applied separately per feature type.

3) Deep Regret Analytic GAN (DRAGAN): DRAGAN [15] is another variant of GAN [12] with gradient penalty to help to stabilize training and reduce mode collapse. This works by training GANs and penalizing the discriminator's gradient around real data points. Gradient penalty forces the discriminator to behave smoothly around real data. In practice, This works by the adding small perturbation noise around real data points before calculating the gradient of discriminator with respect to the real data point. This penalty serves as a term in the loss function of the discriminator during training.

B. Image Synthesizers

1) Deep Convolutional Generative Adversarial Networks (DC-GAN): DCGAN [3] is a class of GANs [12] that uses deep convolutional neural networks (CNN) in the generator and discriminator to be used mainly for images. Instead of full connected layers, the DCGAN generator uses a series of transposed convolutional layers with batch normalization and ReLU function. The final layer uses Tanh activations. The discriminator uses convolutional layers that classifies an input image as either real or fake. It uses convolutions, LeakyReLU activations followed by final sigmoid activation. The architecture is shown in Figure 3.

2) Pix2Pix Model: Pix2Pix [16] is an image to image translation model based on GANs. This model learns a translation mapping from an input image to an output image using the adversarial loss. In other words, Pix2Pix generates output image conditioned on the input image. The generator in Pix2Pix is based on U-Net [5] which is encoder-decoder structure with skip connections that allow the

```
=================================================================
Layer (type:depth-idx)              Output Shape        Param #
=================================================================
Generator                           [1, 3, 64, 64]      --
├─Sequential: 1-1                   [1, 3, 64, 64]      --
│    └─ConvTranspose2d: 2-1         [1, 512, 4, 4]      819,200
│    └─BatchNorm2d: 2-2             [1, 512, 4, 4]      1,024
│    └─ReLU: 2-3                    [1, 512, 4, 4]      --
│    └─ConvTranspose2d: 2-4         [1, 256, 8, 8]      2,097,152
│    └─BatchNorm2d: 2-5             [1, 256, 8, 8]      512
│    └─ReLU: 2-6                    [1, 256, 8, 8]      --
│    └─ConvTranspose2d: 2-7         [1, 128, 16, 16]    524,288
│    └─BatchNorm2d: 2-8             [1, 128, 16, 16]    256
│    └─ReLU: 2-9                    [1, 128, 16, 16]    --
│    └─ConvTranspose2d: 2-10        [1, 64, 32, 32]     131,072
│    └─BatchNorm2d: 2-11            [1, 64, 32, 32]     128
│    └─ReLU: 2-12                   [1, 64, 32, 32]     --
│    └─ConvTranspose2d: 2-13        [1, 3, 64, 64]      3,072
│    └─Tanh: 2-14                   [1, 3, 64, 64]      --
=================================================================
Total params: 3,576,704
Trainable params: 3,576,704
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 428.35
=================================================================
Input size (MB): 0.00
Forward/backward pass size (MB): 2.06
Params size (MB): 14.31
Estimated Total Size (MB): 16.37
=================================================================
```

(a) Generator

```
=================================================================
Layer (type:depth-idx)              Output Shape        Param #
=================================================================
Discriminator                       [1, 1, 1, 1]        --
├─Sequential: 1-1                   [1, 1, 1, 1]        --
│    └─Conv2d: 2-1                  [1, 64, 32, 32]     3,072
│    └─LeakyReLU: 2-2               [1, 64, 32, 32]     --
│    └─Conv2d: 2-3                  [1, 128, 16, 16]    131,072
│    └─BatchNorm2d: 2-4             [1, 128, 16, 16]    256
│    └─LeakyReLU: 2-5               [1, 128, 16, 16]    --
│    └─Conv2d: 2-6                  [1, 256, 8, 8]      524,288
│    └─BatchNorm2d: 2-7             [1, 256, 8, 8]      512
│    └─LeakyReLU: 2-8               [1, 256, 8, 8]      --
│    └─Conv2d: 2-9                  [1, 512, 4, 4]      2,097,152
│    └─BatchNorm2d: 2-10            [1, 512, 4, 4]      1,024
│    └─LeakyReLU: 2-11              [1, 512, 4, 4]      --
│    └─Conv2d: 2-12                 [1, 1, 1, 1]        8,192
│    └─Sigmoid: 2-13               [1, 1, 1, 1]        --
=================================================================
Total params: 2,765,568
Trainable params: 2,765,568
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 103.82
=================================================================
Input size (MB): 0.05
Forward/backward pass size (MB): 1.44
Params size (MB): 11.06
Estimated Total Size (MB): 12.55
=================================================================
```

(b) Discriminator

Fig. 3: DCGAN architecture layers

```
=================================================================
Layer (type:depth-idx)              Output Shape        Param #
=================================================================
UNetGenerator                       [1, 3, 64, 64]      --
├─Sequential: 1-1                   [1, 256, 8, 8]      --
│    └─Conv2d: 2-1                  [1, 64, 32, 32]     3,136
│    └─LeakyReLU: 2-2               [1, 64, 32, 32]     --
│    └─Conv2d: 2-3                  [1, 128, 16, 16]    131,200
│    └─BatchNorm2d: 2-4             [1, 128, 16, 16]    256
│    └─LeakyReLU: 2-5               [1, 128, 16, 16]    --
│    └─Conv2d: 2-6                  [1, 256, 8, 8]      524,544
│    └─BatchNorm2d: 2-7             [1, 256, 8, 8]      512
│    └─LeakyReLU: 2-8               [1, 256, 8, 8]      --
├─Sequential: 1-2                   [1, 3, 64, 64]      --
│    └─ConvTranspose2d: 2-9         [1, 128, 16, 16]    524,416
│    └─BatchNorm2d: 2-10            [1, 128, 16, 16]    256
│    └─ReLU: 2-11                   [1, 128, 16, 16]    --
│    └─ConvTranspose2d: 2-12        [1, 64, 32, 32]     131,136
│    └─BatchNorm2d: 2-13            [1, 64, 32, 32]     128
│    └─ReLU: 2-14                   [1, 64, 32, 32]     --
│    └─ConvTranspose2d: 2-15        [1, 3, 64, 64]      3,075
│    └─Tanh: 2-16                   [1, 3, 64, 64]      --
=================================================================
Total params: 1,318,659
Trainable params: 1,318,659
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 351.50
=================================================================
Input size (MB): 0.05
Forward/backward pass size (MB): 2.98
Params size (MB): 5.27
Estimated Total Size (MB): 8.31
=================================================================
```

(a) Generator

```
=================================================================
Layer (type:depth-idx)              Output Shape        Param #
=================================================================
Discriminator                       [1, 1, 7, 7]        --
├─Sequential: 1-1                   [1, 1, 7, 7]        --
│    └─Conv2d: 2-1                  [1, 64, 32, 32]     6,208
│    └─LeakyReLU: 2-2               [1, 64, 32, 32]     --
│    └─Conv2d: 2-3                  [1, 128, 16, 16]    131,200
│    └─BatchNorm2d: 2-4             [1, 128, 16, 16]    256
│    └─LeakyReLU: 2-5               [1, 128, 16, 16]    --
│    └─Conv2d: 2-6                  [1, 256, 8, 8]      524,544
│    └─BatchNorm2d: 2-7             [1, 256, 8, 8]      512
│    └─LeakyReLU: 2-8               [1, 256, 8, 8]      --
│    └─Conv2d: 2-9                  [1, 1, 7, 7]        4,097
│    └─Sigmoid: 2-10               [1, 1, 7, 7]        --
=================================================================
Total params: 666,817
Trainable params: 666,817
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 73.72
=================================================================
Input size (MB): 0.10
Forward/backward pass size (MB): 1.31
Params size (MB): 2.67
Estimated Total Size (MB): 4.08
=================================================================
```

(b) Discriminator

Fig. 4: Pix2Pix architecture layers

transfer of fine-grained spatial features from the encoder to the decoder. The discriminator does not classify the entire image as real or fake. Instead, the discriminator (PatchGAN) looks at small patches of the image (pixels wise) and outputs a prediction for each patch as real or fake. If all patches look real, the whole image is considered real. In this case, the real image is fed into the model as input and output. The architecture is shown in Figure 4.

3) Convolutional Variational Autoencoder: The architecture is based on CNN with B-VAE. B-VAE was introduced by [17] is an extension of VAE with addition of Beta weight in the KL divergence term in the loss function to help further with regularization. Convolutional Variational Autoencoder consists of encoder and decoder. The Encoder has four layers of convolution followed by ReLU layers while the Decoder has four transposed convolutional followed by ReLU layers to reconstruct the image. The architecture is shown in Figure 5.

C. Evaluation Metrics

In this study, several synthetic data evaluation metrics are used to measure the similarity between real and generated data. These evaluation metrics are used to give details about the fidelity of the synthetic data. The following metrics are used:

1) Mean Square Error (MSE): This metric is the Mean Squared Error between the real R and the synthetic image S.

$$\text{MSE} = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \big(R(i,j) - S(i,j) \big)^2$$

```
=================================================================
Layer (type:depth-idx)              Output Shape        Param #
=================================================================
BetaVAE                             [1, 3, 64, 64]      --
├─Encoder: 1-1                      [1, 128]            --
│    └─Conv2d: 2-1                  [1, 32, 32, 32]     1,568
│    └─ReLU: 2-2                    [1, 32, 32, 32]     --
│    └─Conv2d: 2-3                  [1, 64, 16, 16]     32,832
│    └─ReLU: 2-4                    [1, 64, 16, 16]     --
│    └─Conv2d: 2-5                  [1, 128, 8, 8]      131,200
│    └─ReLU: 2-6                    [1, 128, 8, 8]      --
│    └─Conv2d: 2-7                  [1, 256, 4, 4]      524,544
│    └─ReLU: 2-8                    [1, 256, 4, 4]      --
│    └─Linear: 2-9                  [1, 128]            524,416
│    └─Linear: 2-10                 [1, 128]            524,416
├─Decoder: 1-2                      [1, 3, 64, 64]      --
│    └─Linear: 2-11                 [1, 4096]           528,384
│    └─ConvTranspose2d: 2-12        [1, 128, 8, 8]      524,416
│    └─ReLU: 2-13                   [1, 128, 8, 8]      --
│    └─ConvTranspose2d: 2-14        [1, 64, 16, 16]     131,136
│    └─ReLU: 2-15                   [1, 64, 16, 16]     --
│    └─ConvTranspose2d: 2-16        [1, 32, 32, 32]     32,800
│    └─ReLU: 2-17                   [1, 32, 32, 32]     --
│    └─ConvTranspose2d: 2-18        [1, 3, 64, 64]      1,539
│    └─Tanh: 2-19                   [1, 3, 64, 64]      --
=================================================================
Total params: 2,957,251
Trainable params: 2,957,251
Non-trainable params: 0
Total mult-adds (Units.MEGABYTES): 135.40
=================================================================
Input size (MB): 0.05
Forward/backward pass size (MB): 1.08
Params size (MB): 11.83
Estimated Total Size (MB): 12.96
=================================================================
```

Fig. 5: Convolutional Beta Variational Autoencoder architecture layers

2) Peak Signal-to-Noise Ratio (PSNR): PSNR is used to measure the quality of a synthetic compared to the original image. It indicates how much noise in the synthetic image. Higher PSNR means the better quality of synthetic images. The PSNR is given by the following equation:

$$\text{PSNR} = 10 \cdot \log_{10}\left(\frac{\text{MAX}^2}{\text{MSE}}\right)$$

where MAX is the maximum pixel value of the image. It is usually 255 for RGB images and MSE is the Mean Squared Error between the real and the synthetic images. In practice, PSNR values that are higher than 30 dB are good quality while values less than 20 dB indicates heavy distortion in the synthetic image

3) Structural Similarity Index (SSIM): The Structural Similarity Index (SSIM) is a metric that quantifies the similarity between two images brought from classical image analysis. Given real R and synthetic S images, the SSIM is defined as:

$$\text{SSIM}(x,y) = \frac{(2\mu_R\mu_S + C_1)(2\sigma_{RS} + C_2)}{(\mu_R^2 + \mu_S^2 + C_1)(\sigma_R^2 + \sigma_S^2 + C_2)}$$

where μ_R, μ_S are the mean intensities of R and S, σ_R^2, σ_S^2 are the variances of R and S, σ_{RS} is the covariance between R and S, C_1, C_2 are small constants to stabilize the division. The values of SSIM lies between 0 and 1 where 0 indicates limited structural similarity and 1 indicates perfect structural similarity.

4) Principal Component Analysis (PCA): Principal Component Analysis (PCA) is a method for dimensionality reduction. It is used to reduce the high dimension data into lower dimensions space while preserving data features variance. PCA theoretical details are beyond the scope of this article. However, PCA is used here to study the similarity between synthetic and real data in low dimension space assuming that similar data points will share local neighborhood in the 2D space.

D. Technical Specifications

The used MRI images are already preprocessed for artifacts and cleaning. Before training, the data was resize to (64x64) pixel size. This small size is used for training the model as it is more efficient. However, bigger size can be used but this will essentially lead to long training time. The data was standardized to the scale (-1, 1). The Image synthesizers are trained for 200 epochs. Here the epoch means that the model is trained on the whole dataset 200 times. During the training, learning rates, number of layers are optimized for each model.

For the tabular data, NaN features are removed from the data in addition to the "Hand" feature which has only one value (R). The remaining features that are used for training are five. Four of those are numerical while only one is categorical. More details of experiments are shown in the GitHub repository [4].

IV. RESULTS

In this section, the generated synthetic data for both the used datasets are shown and studied for fidelity. The results for tabular data are shown in section IV-A and the results for image data are shown in section IV-B.

A. Tabular Synthetics

The generated synthetic in this section is generated using three different tabular models: CTGAN, TVAE and DRAGAN. In the following subsection IV-A.1, IV-A.2, IV-A.3, results are shown in terms of statistical and fidelity properties.

[4] https://github.com/mhmdrdwn/synthetic_MRI/

(a) Sex

(b) Age

(c) ASF

(d) eTIV

(e) nWBV
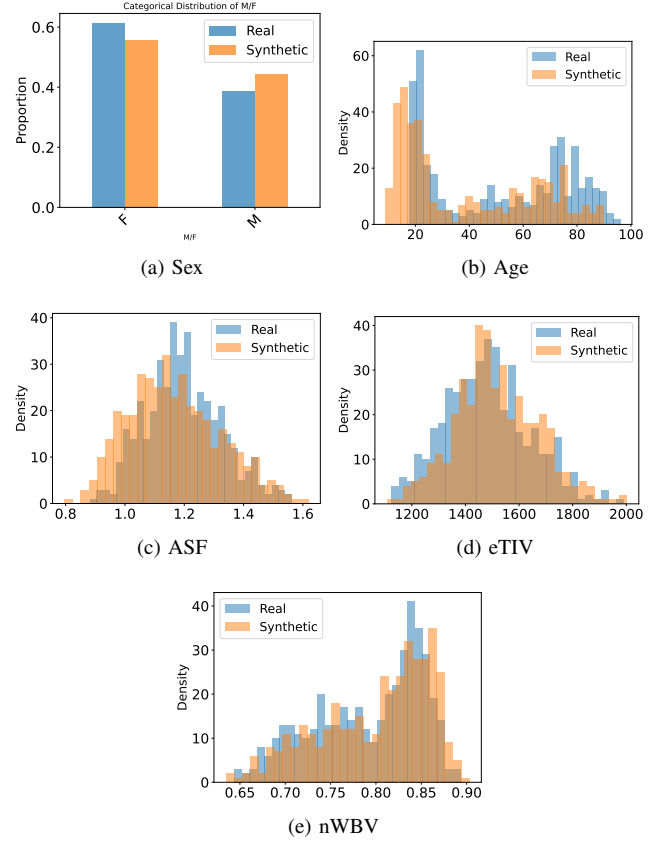
Fig. 6: Features Distribution of original vs synthetic data using CTGAN model

1) CTGAN generated tabular: After generating synthetic data using CTGAN tabular model, Figure 6 shows the distribution of the synthesized features versus the real features. In this figure, five features are shown. One categorical and four numerical. The distributions of the real and synthetic features show high overlap in general.

For the sex variable, the synthetic data is well represented for both the sexes as shown in Figure 6. Furthermore, Age, ASF, eTIV and nWBV synthetic distributions show high overlaps with the real data.

2) TVAE generated tabular: In a similar fashion to the previous figure of CTGAN, the synthetic and real features distributions are shown in Figure 8. The distributions of the real and synthetic features show high overlap in general for all the features. TVAE shows better overlap than CTGAN generated features.

3) DRAGAN generated tabular: Figure 8 shows the distributions of the real and synthetic features using the DRAGAN model. The distributions have limited overlaps. For the sex variable, the synthetic data has comparable distribution of both sex. However, for the eTIV and nWBV features, the synthetic data generate data with ranges that are not covered in the original data.

4) Comparison: Table I reports and compares mean, standard deviation, skewness and kurtosis of the synthetic data features and the real data. The comparison is shown for the three used models on the tabular data. From the statistical analysis, the features (eTIV, mWBV, ASF), CTGAN and TVAE generate the synthetic data has comparable mean, std, Kurtosis and Skewness values. DRAGAN has the worst generated data in general except for Age feature. DRAGAN model generated data have the closest mean to the real data for the Age feature.

(a) Sex

(b) Age

(c) ASF

(d) eTIV

(e) nWBV
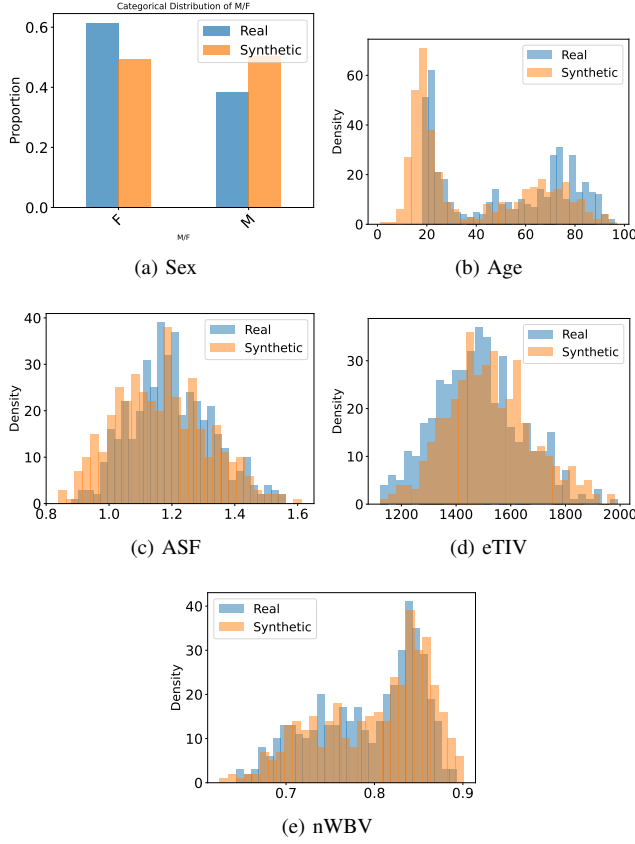
Fig. 7: Distribution of original vs synthetic data using TVAE model

(a) Sex

(b) Age

(c) ASF

(d) eTIV

(e) nWBV
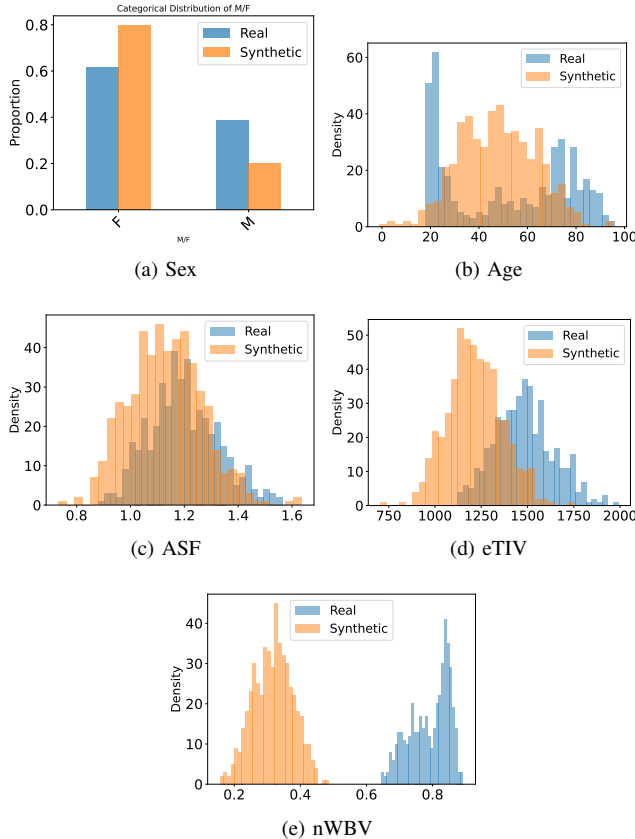
Fig. 8: Distribution of original vs synthetic data using DRAGAN model

Row	Real Data				CTGAN Synthetic			
	mean	std	skewness	kurtosis	mean	std	skewness	kurtosis
Age	51.35	25.26	0.00	-1.59	38.3	24.6	0.53	-1.25
eTIV	1481.9	158.74	0.26	-0.15	1517.45	156.0	0.27	0.022
nWBV	0.79	0.059	-0.52	-0.88	0.8	0.06	-0.65	-0.58
ASF	1.19	0.128	0.28	-0.17	1.16	0.15	0.36	-0.382
Row	TVAE Synthetic				DRAGAN Synthetic			
	mean	std	skewness	kurtosis	mean	std	skewness	kurtosis
Age	38.12	25.36	0.592	-1.230	47.48	15.70	-0.01	-0.271
eTIV	1522.3	156.32	0.292	-0.08	1216.3	147.89	0.16	0.089
nWBV	0.80	0.062	-0.57	-0.72	0.317	0.06	-0.039	-0.39
ASF	1.16	0.14	0.26	-0.385	1.131	0.13	0.27	0.24

TABLE I: Statistical properties of the generated synthetic data for the used three models in comparison with the real data.
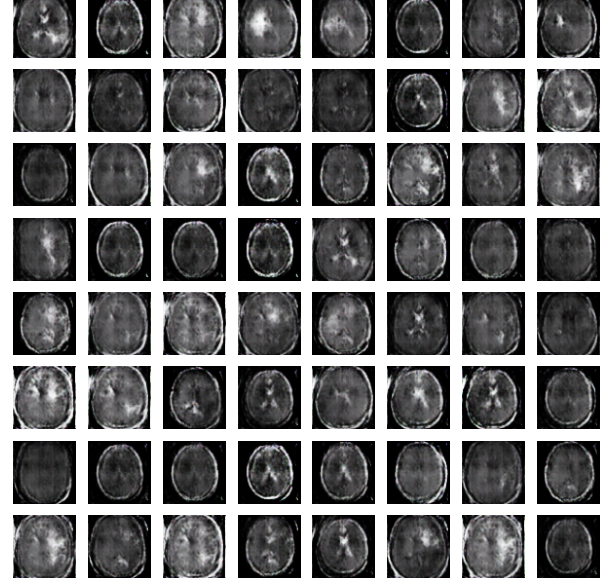
Fig. 9: Samples of normalized Generated synthetic images of DC-GAN model. This batch of images has 64 MRI scans to be used for visual inspections.

B. MRI Image Synthetics

Here, the generated synthetic images are shown for visual inspection and further fidelity analysis of those generated images are shown in details.

1) DCGAN generated images: Figure 9 shows randomly selected images from the synthetic data for visual inspection. Those samples are normalized between [0, 1]. However, it is hard to tell whether the images are realistic based only visual inspection specially in GAN generated images as we can not compare synthetic to real images directly. However, we argue if the GAN is working to generate realistic images, that some of generated synthetic images maybe are similar to the real images.

For better analysis, Figure shows the distribution of raw images samples when reduced on 2D space using PCA. Here the distributions of synthetic and real images have smaller overlaps. This indicates poor synthetic data generations.

2) Pix2Pix generated images: To study the fidelity of the generated images using Pix2Pix model, randomly selected images from the synthetic and real images are shown in Figure 11 for visual inspection. Here, unlike to GAN models, generated images can be compared directly to real images. It is noticed from Figure 11 that the generated images are almost identical to real images even from non expert experience.

By looking further on one of images, Figure 12 shows the top
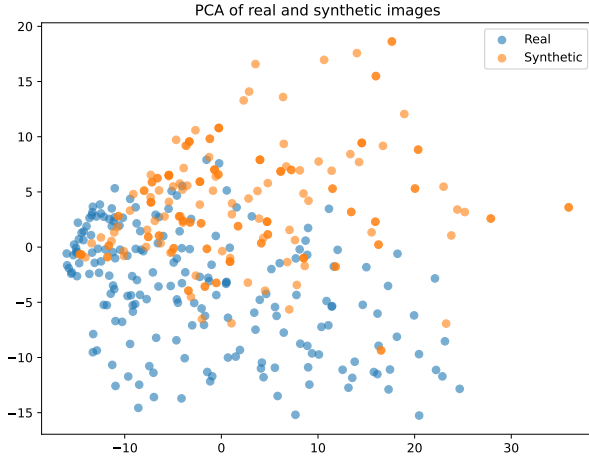
Fig. 10: PCA Distribution of real and synthetic images using DC-GAN model. In this figure, raw pixels are used.

left real and synthetic image. It is challenging to tell the difference between the two images from visual inspection. However, there remains slight amount of noise in the synthetic image. It is worth noting here that the used images for training the model is (64x64) which is very low resolution in this case. This used resolution is aimed for efficient model development and training. However, the development of this model for higher resolution can be done using the same method but this will lead to slow and less efficient training process.

Figure 13 shows the distributions of data points from real and synthetic images after applying dimensionality reduction on the images using PCA. The synthetic and real images showing local similarity in which similar samples end up near each other. This is a sign of robust results.

For further analysis, Figure 14 shows the MSE, PSNR and SSIM between each of the generated and real images. Those metrics are counted over three batches of the data. This means there are comparisons between 192 synthetic and real images. The MSE values are between 0.0005 and 0.0025 with mode around 0.0007. The PSNR has a distribution of mode around 30 and the minimum value is 26. The SSIM has a distribution of mode around 0.96 and the minimum value is 0.9. This is another sign of robust results.

3) Convolutional VAE generated images: Generated images using VAE model are randomly selected and compared with real images and are shown in Figure 15. Here, the real data is the same ones used in Figure 11. It is noticed from Figure 11 that the generated images are highly similar to the real images.

Figure 16 shows one single MRI image that is the top left real and synthetic. There are differences between the real and synthetic images. These differences are higher compared to the Figure 12 which is generated by Pix2Pix model.

Figure 17 shows the PCA reduced features visualizations of the data points from real and synthetic images. The synthetic and real images showing similarity in which similar samples end up near each other.

Figure 18 shows the MSE, PSNR and SSIM between each of the generated and real images. The MSE values are between 0.007 and 0.0002 with mode around 0.003. The PSNR has a distribution of mode around 24 and the minimum value is 19. The SSIM has a distribution of mode around 0.7 and the minimum value is 0.5.

V. DISCUSSION
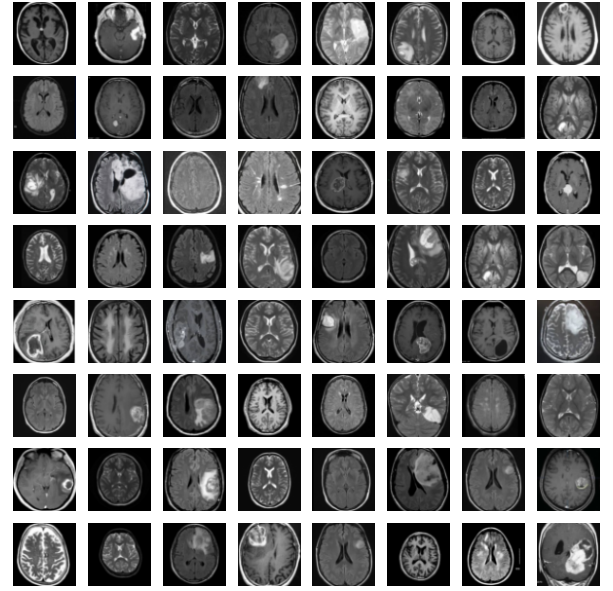

(a) Real Data
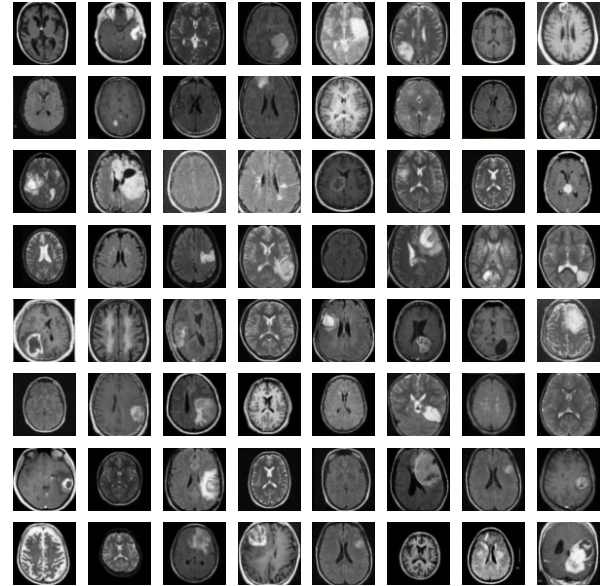
(b) Synthetic Data

Fig. 11: Normalized real and generated synthetic images using Pix2Pix model

### A. Tabular synthetic Data

CTGAN and TVAE generate the synthetic data has more similar to the real data. This is supported from studying the overlapping distribution and studying of statistical analysis such as mean, std, Kurtosis and Skewness. This suggests that fidelity of the synthetic data generated by CTGAN and TVAE and relatively higher than DRAGAN. DRAGAN generates the worst synthetic data in terms of fidelity. In the case mWBV feature, DRAGAN generate feature values that are entirely out of distribution. It is also worth noting that the results from DRAGAN are also not numerically stable. This maybe is related to the small size of the data.
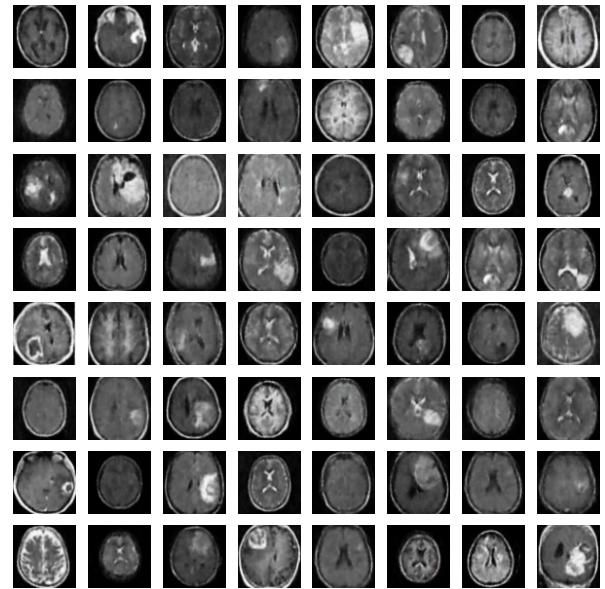
(a) Real Image   (b) Synthetic Image
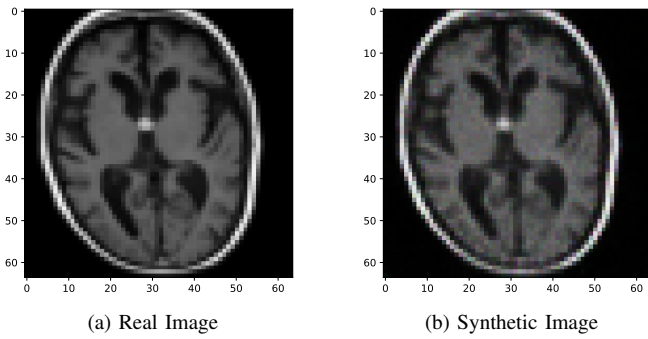
Fig. 12: Sample Normalized real and synthetic image 64x64 for comparison using Pix2Pix model. The model can be extended for higher resolution images.
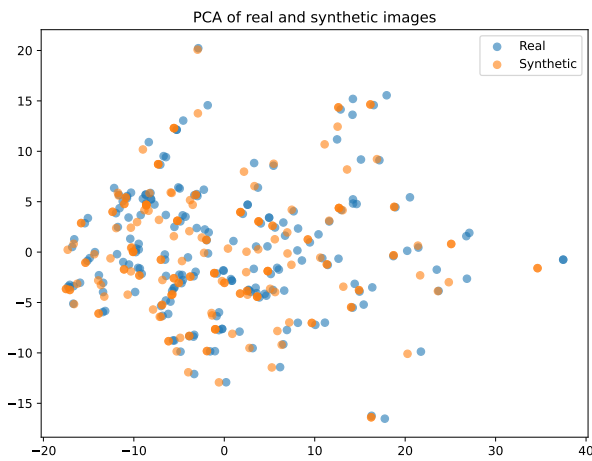
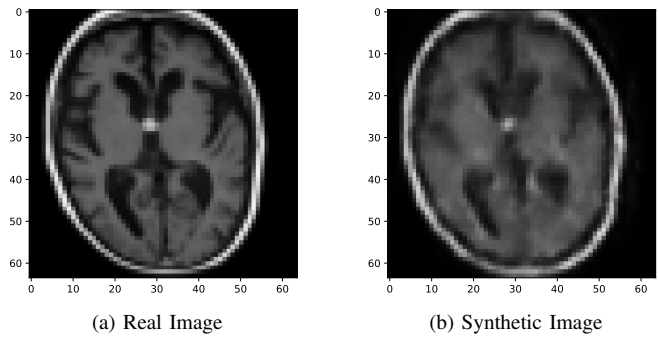Fig. 13: PCA Distribution of real and synthetic images using Pix2Pix model. In this figure, raw pixels are used.
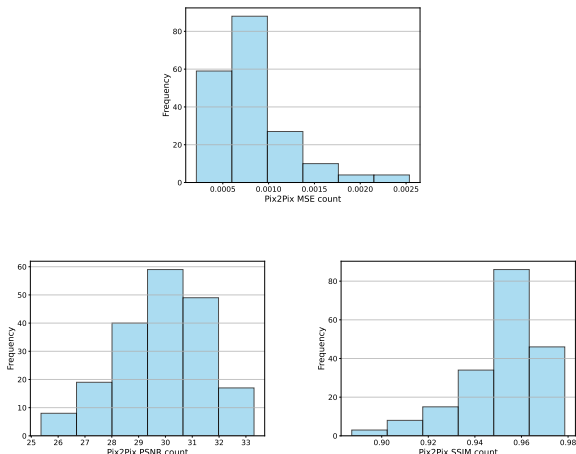
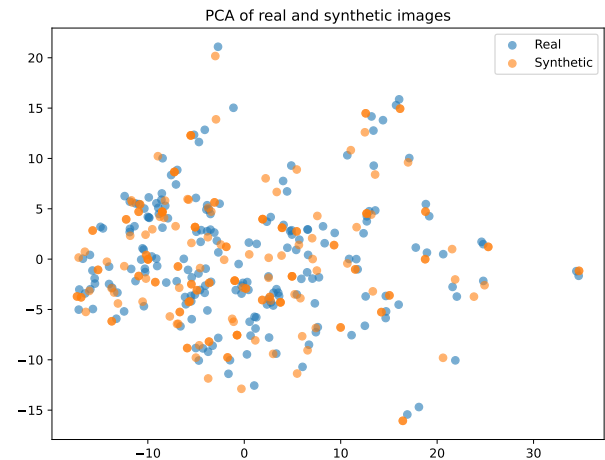Fig. 14: Normalized Generated synthetic images of Pix2Pix model

(a) Synthetic Data

Fig. 15: Normalized Generated synthetic images of DCGAN model. Here, only synthetic data is shown while the real data is the same subset used in Fig 11

(a) Real Image   (b) Synthetic Image

Fig. 16: Sample Normalized real and synthetic image 64x64 for comparison using VAE model.

Fig. 17: PCA Distribution of real and synthetic images using VAE model. In this figure, raw pixels are used.
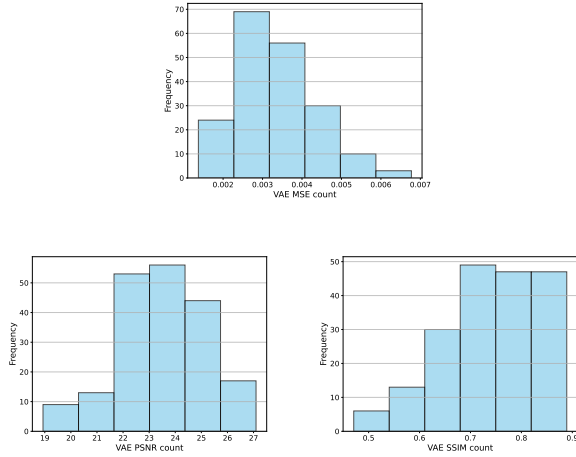
Fig. 18: Normalized Generated synthetic images of Pix2Pix model

B. MRI synthetic Data

The PCA plots from Pix2Pix and VAE models suggests that the synthetic data points preserve local neighborhood structure with real data. As overlapping clusters usually mean the synthetic data has learned many of the local relationships present in the real data. The synthetic data may have similar modes and variance as the real data. These are good sign of high fidelity for Pix2Pix and VAE.

On the other hand, the PCA plot of the real versus generated images using DCGAN indicates difference in modes with limited overlap. This indicates limited fidelity and the synthetic data maybe are not realistic.

By comparing the three models results, It is evident that Pix2Pix model achieved the best performances in terms of generating synthetic MRI images. This is supported by the visual inspection and evaluations metrics (SSIM, MSE, PSNR). The MSE values between real and generated are lower in Pix2Pix compared to VAE. Similarly, The PSNR values between real and generated are higher in Pix2Pix compared to VAE. In some cases, Pix2Pix generate images that almost identical to the real data with values between 0.96 and 0.98 SSIM value. Pix2Pix generated images show PSNR values between 26 and 33. It is noted that PSNR values between 30 and 40 dB are good quality. Values less than 20 dB indicates heavy distortion in the synthetic image. VAE generated images show PSNR values between 19 and 27 which indicates that large portions of the images are of good quality. The PSNR values are in general higher for the Pix2Pix model than Convolutional VAE. This suggests that Pix2Pix generated the highest fidelity data in this study.

VI. CONCLUSION AND FUTURE WORK

In this study, the objective is building high fidelity synthetic MRI and tabular data for the goal of diagnosis of brain disorders is approached. Several models are used in the study including tabular and image based synthetic models. DCGAN and TVAE tabular models generate relatively high fidelity tabular features. For the synthetic image generation, it is noted that Pix2Pix model achieves the best performances in generating highly realistic images. This is supported by measuring differnet evaluation metrics such as SSIM, MSE and PSNR. The convolutional VAE achieves medium performances compared to Pix2Pix in synthetic image generation.

There remain room for developments in terms of the used methods and building other methods such as Diffusion models. Diffusion models such as DDPMs have recently outperfoms GANs in data fidelity. Several recent medical imaging studies report that diffusion models generate higher quality MRI scans.

The image synthesizers are trained on low resolution images (64x64 images) for faster and efficient training and development. It would be beneficial to train on the models on higher resolution images in order to validate the robustness of the models.

In this study, MRI dataset with brain tumors has been used. A possible future direction is to use on actual dementia datasets. The same workflow still applies on several problems such as dementia and Alzheimer. However, applying the workflow will require further extensive work such as MRI data preprocessing and cleaning such as removing artifacts and noise before training.

This study is done with limited resources. A future direction could be use medical validation, auditing and rigorous expert review which is essential in this case. A clinical or neuroscientific insights are helpful to validate and test the generated data in order to avoid hallucinations and mistakes given the sensitivity of the tasks.

Synthetic MRI is often proposed as a solution to sensitive data privacy. However, risks of leakage of potential patient data such as membership inference still exist. Differential privacy mechanisms are highly recommended in this case.

VII. ACKNOWLEDGMENT

REFERENCES

[1] O. Sheppard and M. Coleman, "Alzheimer's disease: etiology, neuropathology and pathogenesis," *Exon Publications*, pp. 1–21, 2020.

[2] S. Dayarathna, K. T. Islam, S. Uribe, G. Yang, M. Hayat, and Z. Chen, "Deep learning based synthesis of mri, ct and pet: Review and analysis," *Medical image analysis*, vol. 92, p. 103046, 2024.

[3] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[4] B. Sun, S. Jia, X. Jiang, and F. Jia, "Double u-net cyclegan for 3d mr to ct image synthesis," *International Journal of Computer Assisted Radiology and Surgery*, vol. 18, no. 1, pp. 149–156, 2023.

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[6] L. Zimmermann, B. Knäusl, M. Stock, C. Lütgendorf-Caucig, D. Georg, and P. Kuess, "An mri sequence independent convolutional neural network for synthetic head ct generation in proton therapy," *Zeitschrift für Medizinische Physik*, vol. 32, no. 2, pp. 218–227, 2022.

[7] A. Chartsias, T. Joyce, R. Dharmakumar, and S. A. Tsaftaris, "Adversarial image synthesis for unpaired multi-modal cardiac data," in *International workshop on simulation and synthesis in medical imaging*. Springer, 2017, pp. 3–13.

[8] F. Khader, G. Müller-Franzes, S. Tayebi Arasteh, T. Han, C. Haarburger, M. Schulze-Hagen, P. Schad, S. Engelhardt, B. Baeßler, S. Foersch *et al.*, "Denoising diffusion probabilistic models for 3d medical image generation," *Scientific Reports*, vol. 13, no. 1, p. 7303, 2023.

[9] A. Kazerouni, E. K. Aghdam, M. Heidari, R. Azad, M. Fayyaz, I. Hacihaliloglu, and D. Merhof, "Diffusion models for medical image analysis: A comprehensive survey," *arXiv preprint arXiv:2211.07804*, 2022.

[10] D. S. Marcus, T. H. Wang, J. Parker, J. G. Csernansky, J. C. Morris, and R. L. Buckner, "Open access series of imaging studies (oasis): cross-sectional mri data in young, middle aged, nondemented, and demented older adults," *Journal of cognitive neuroscience*, vol. 19, no. 9, pp. 1498–1507, 2007.

[11] N. Chakrabarty, "Brain mri images for brain tumor detection," https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/data.

[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[13] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[14] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," *Advances in neural information processing systems*, vol. 32, 2019.

[15] N. Kodali, J. Abernethy, J. Hays, and Z. Kira, "On convergence and stability of gans," *arXiv preprint arXiv:1705.07215*, 2017.

[16] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1125–1134.

[17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *International conference on learning representations*, 2017.