Synthesis of brain tumor MRI images using GAN aggregation with style transfer

Sarah Hindawi, Suman Bagri, Vignesh Edithal

Abstract—Tumor classification and detection are critical for a quick and effective cure. This motivates the idea of developing an automated deep learning (DL) method for classification of brain tumor images. However, DL has raised concerns regarding invading patients' privacy. Also, it is expensive and time-intensive to collect large amounts of MRI images. In addition, many medical imaging datasets are imbalanced which makes it harder for the model to detect outliers. Hence, data augmentation is a crucial necessity in medical image analysis. Traditional data augmentation methods such as rotation, scale, crop, etc. create highly correlated images that lack variance which may prevent DL models from learning the underlying features of an image. Meanwhile, Generative Adversarial Networks (GAN) have shown promising results in generating synthetic data with good generalization ability to large image datasets. GANs also serve as an anonymization tool which reduces data handling costs. In this work, we use the Aggregation GAN (AGGrGAN) [1] model to capture both the unique features and localized information of a source image using style transfer and also the shared information among the different latent representations of multiple images using multiple GANs. We apply style transfer after aggregation to increase resemblance to the original images. Then, we perform an ablation study of aggregation and style transfer to evaluate their impact on performance. Finally, we train a classification network to study the impact of injecting fake images into the training dataset on the performance of the classifier, this also allows us to evaluate the images qualitatively. We make our code available at https://github.com/edithal-14/csc2529-2022-project.

Index Terms—GAN, MRI, Brain, Tumor, Style Transfer, Aggregation, DCGAN, WGAN, UNet, Data Augmentation

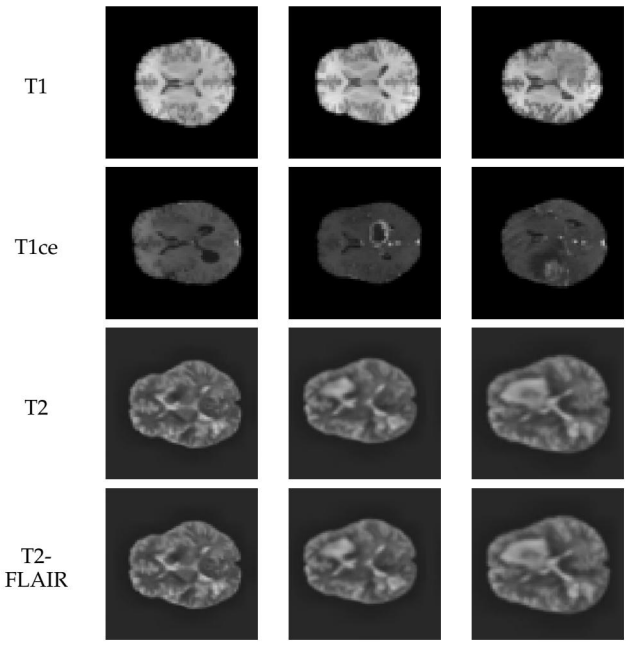
1 Introduction

Brain tumor identification is crucial to prevent long term disabilities. Severe cases such as High Grade Glioma may be fatal. Magnetic Resonance Imaging (MRI) is a powerful non-invasive tool for obtaining these brain scans. MRI scans can provide key information such as the location, shape, size and the growth stage of the brain tumor. To perform any medical image analysis using deep learning techniques, a sufficient volume of data with variability is required. However, traditional image augmentation methods such as scale, rotation, crop etc. create highly correlated images which are unable to capture the underlying features of the source images. In addition, they might change the pattern useful for diagnosis. Class imbalance is another reason to apply augmentation. Generative Adversarial Network (GAN) models have shown promising results in generating synthetic data with good generalization ability to large datasets. In this work, we use the Aggregation GAN (AGGrGAN) model to capture both the unique features and localized information of a source image using style transfer and also the shared information among the different latent representation of multiple images. We then perform an ablation study to quantitatively evaluate (using PSNR and SSIM scores) the generated images and also to study the impact of aggregation followed by style transfer. For a qualitative analysis, we train a classification network using both real images and a mixture of real and fake images to study the effectiveness of the images generated by our models. All our experiments have been performed on the BraTS 2020 dataset [2] [3] [4]. We start off with a literature study of related works in the next section. Then, we describe the BraTS 2020 dataset and explain the data pre-processing steps in detail. In the methodology section, we explain in brief the internal working of a GAN model as described in [5], then we move on to describing various GAN architectures such as DCGAN [6], WGAN [7], UNet GAN [8]. We also explain the aggregation logic used in AGGrGAN and provide details of the style transfer technique. The last section contains detailed quantitative/qualitative analysis followed by some concluding remarks.

2 Related Work

Han et al. [9] have applied DCGAN and WGAN separately on the BraTS 2016 dataset to generate artificial MRI scans. To validate their results, they conducted the Visual Turing test with 53% accuracy for WGAN. Nie et al. [10] have used Fully Convolution Network (FCN) as generator and a basic CNN as the discriminator, they have proposed 3D FCN to estimate target image from the corresponding source image, they have used ADNI dataset and have obtained a mean PSNR of 34.1. Emami et al. [11] have proposed a GAN based model where ResNet is used as the generator and discriminator is a CNN with five convolutional layers which classify the image as real or fake, they achieved a mean PSNR of 26.6 ± 1.2 for an IRB approved dataset. Shin et al. [12] segmented the overall scans of Alzheimer's Disease Neuroimaging Initiative (ADNI) [13] dataset and BraTS dataset into brain anatomy and tumors using pix2pix GAN [14]. They have obtained the augmented scans by applying different combinations of the segmented brain anatomy and tumor labels by introducing some alternations. Sarkar et al. [15] created a CNN model to detect the type of brain tumor using MRI scans to classify meningioma, glioma and pituitary tumors.

TABLE 1
Sample images from BraTS 2020 dataset where each row represents the MRI imaging modality



3 METHODOLOGY

3.1 Dataset description

We use BraTS 2020 dataset for all our experiments. It consists of four MRI modality classes namely, T1 weighted images (T1), Post contrast T1 weighted images (T1ce), T2 weighted images (T2) and T2 Fluid Attenuated Inversion Recovery (T2-FLAIR). These samples have been acquired with different clinical protocols and various scanners from multiple institutions. Training data contains a total of 369 NIfti files for each class and testing data contains 125 Nifti files for each class. A Nifti file corresponds to a 3D image acquired by a MRI scan [1]. The middle layer of the 3D image is chosen for all experiments since that is the biggest in size and contains more details as compared to the other slices which are smaller in size. Each image has a resolution of 240 x 240 pixels. However, due to limited resources at our disposal we chose to resize the image to 64 x 64 pixels resolution for all our experiments. This also helps with training the GAN model as explained in the later sections. Sample images from the dataset are shown in Table 1.

3.2 A brief introduction to Generative Adversarial Network (GAN)

GAN is a deep learning (DL) framework to capture the distribution of training data such that we can generate new data from that distribution. They were first described in [5]. They consist of two competing models namely Generator and Discriminator. The job of the Generator is to spawn "fake images" that look like the training images. The job of the Discriminator is to classify whether a given image is real or fake. During training, both of the models compete in a game with each other to minimize their losses. The equilibrium of the game is when the Generator produces perfect "fake images" and the Discriminator is left to always guess at 50% probability.

Before we formally define the losses of both the models in a GAN, we would like to mention some notation. Let, x be the data representing an image, ρ_{data} represents the training distribution, ρ_{latent} represents the latent distribution. The output of Discriminator $D(x)$ is the probability of the given image being real. The output of the Generator $G(z)$ is a fake image where z is the latent distribution of the training distribution. So, $D(G(z))$ is the probability of a generated image ("fake image") being classified as real. As described in [5], D and G play a **minmax** game where D tries to **maximize** the log-probability that it classifies correctly ($\log(D(x))$). G tries to **minimize** the log-probability that D will predict its output to be fake ($\log(1 - D(G(z)))$). From the paper, the GAN loss function is described as follows.

$$min_G max_D V(D, G) =$$
$$\mathbb{E}_{x \sim \rho_{data}(x)}[\log D(x)] + \mathbb{E}_{z \sim \rho_{latent}(z)}[\log(1 - D(G(z)))]$$

In theory the solution to this game is where $\rho_{real} = \rho_{fake}$, however, in practice this is rarely achieved, this makes training a GAN an inherently difficult task. The convergence of GANs is still an active area of research.

3.3 AGGrGAN model

The aggregation GAN (AGGrGAN) model as described in [1] consists of two different Deep Convolutional GANs (DCGANs) and one Wasserstein GAN (WGAN). The two DCGANs differ in the way the latent vector is up-sampled to generate the fake image, one uses transposed convolution layers and the other one an explicit up-sampling operation. The images from each of the GANs are merged together using a weighting scheme which is based on edge detection and PSNR/SSIM metrics. To further increase the quality of the images, style transfer technique is applied to the output of each GAN and the aggregated image. The architectural diagram of this model is presented in Figure 1. In the upcoming sections we talk about each of these components in detail.

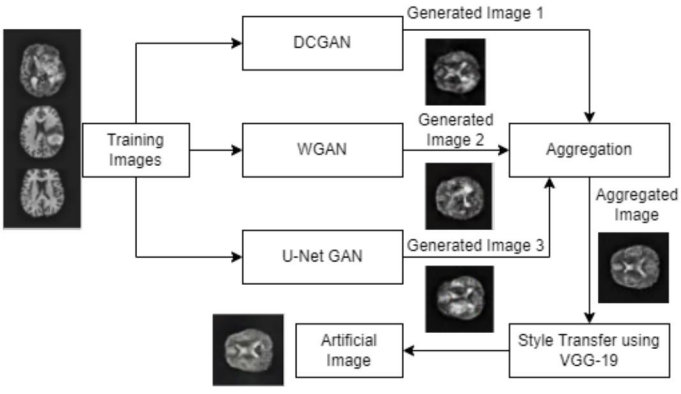


Fig. 1. Architecture diagram of Aggregate GAN (AGGrGAN)

3.4 DCGAN model

DCGAN was introduced in [6], it is an extension of the GAN model described above where Convolutional and Transposed Convolutional layers are used repeatedly in the Discriminator and Generator model respectively. The Discriminator uses strided convolutional layers to repeatedly downsample the image to a scalar value, it is suggested to use strided convolution instead of max/mean pooling layer since the former allows the model to learn the downsampling operation suitable for the given data. The Generator uses strided transposed convolutional layers to repeatedly upscale the latent vector (noise sampled from a multivariate normal distribution) to generate a "fake image". Both the models use Batch Normalization (BatchNorm) layers to regulate gradients. The Generator and Discriminator use ReLU and Leaky ReLU activation functions respectively, LeakyReLU helps in healthy flow of gradients across the network and prevents issues such as vanishing gradient problem.

The Generator uses 5 layers of transposed convolutional layers interspersed with BatchNorm and ReLU layers to upsample a latent vector of size 128 to a single channel 64 x 64 image, it uses the Tanh activation function as the last layer for output. The Discriminator uses 5 layers of convolutional layers interspersed with BatchNorm and Leaky ReLU layers to downsample the 64 x 64 image into a scalar value, it uses the Sigmoid activation function as the final layer to output a scalar probability value.

DCGAN is trained using a Binary Cross Entropy (BCE) loss where the ground truth y value for real image is 1 and fake image is 0, this allows us to choose between two different log values in the BCE loss.

$$l_n = -((y_n * \log x_n) + (1 - y_n) * \log(1 - x_n)) \quad (1)$$

The Discriminator loss is calculated in two steps. First, we feed forward the Discriminator with real images and use $y = 1$ to calculate BCE loss. Then we create a fake image using feed forward of Generator which is fed to the Discriminator and BCE loss is calculated using $y = 0$. The sum of these losses is the Discriminator loss. The Generator loss is calculated by creating a fake image using feed forward of Generator, then passing it through Discriminator where the BCE loss is calculated using $y = 1$. At the end of every mini batch (batch size = 128) the optimizer step is performed (Adam with beta1 = 0.5 and lr = 2e-4). We perform a total of 500 epochs on the training data (369 images) for each modality. The Discriminator loss, Generator loss, mean PSNR and SSIM scores of each mini batch is stored for plotting in order to visualize the training process.

3.5 WGAN model

Wasserstein GAN proposed by [7] is an extension of GAN which improves training stability and uses a loss function which is more indicative of the quality of generated images. It uses a Discriminator (called Critic in WGAN) which outputs different score for real and fake images instead of a probability value. This change is motivated by the fact that Generator should seek a minimization of Earth Movers (EM) distance between the observed distribution and generating distribution. EM distance is chosen since it produces large gradients even for large difference in the observed and generated distributions. The Discriminator loss in this case is the negation of difference between the critic scores of real and fake images. The Generator loss is the negation of the critic score of fake images. It is to be noted that when training WGAN the generator model is updated only one time per n_{critic} (5 in our case) updates of the critic model. Following is the equation of critic loss in WGAN.

$$W(P_r, P_\theta) = sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim P_r}[f(x)] - \mathbb{E}_{x \sim P_\theta}[f(x)] \quad (2)$$

Here sup is the least upper bound and f is a 1-Lipschitz function following the constraint.

$$\|f(x_1) - f(x_2)\| \leq \|x_1 - x_2\| \quad (3)$$

We use two different WGANs based on how the 1-Lipschitz constrained function is implemented. In the WGAN model, we clip the weights of the critic between -0.01 and 0.01. However, the WGAN paper suggests that this is not the best way to enforce the 1-Lipschitz constraint. In the WGAN-GP (WGAN Gradient Penalty) model, we add a penalty term to the critic loss based on how much the second norm of the gradient moves away from 1. The gradient is calculated at an interpolated point which lies between observed and generated distributions. It is suggested not to used BatchNorm layers in the Critic model when using gradient penalty.

$$W(P_r, P_\theta) += \lambda * \mathbb{E}_{\hat{x} \sim P_{\hat{x}}}[(\nabla_{\hat{x}} D(\hat{x}) - 1)^2] \quad (4)$$

Here λ is the regularization parameter which is set to 10 in our case and $\hat{x}$ is the interpolated point.

For the training of WGAN and WGAN-GP we use image size of 64 x 64 pixels resolution and latent vector size of 128. We run 500 epochs with a batch size of 32 with lr = 1e-3 and beta1 parameter of Adam optimizer set to 0.5

3.6 U-Net GAN

One of the major challenges for GANs is the ability to produce locally and globally coherent images [8]. In order to tackle this problem, the authors of [8] proposed a modification to the "vanilla" GAN where the standard classification network used as a discriminator (D) was replaced with an encoder-decoder framework of a U-Net. In an encoder-decoder network, the encoder acts as a classification network that downsamples the input capturing global image context which is then fed into a bottleneck layer. The decoder, subsequently, upsamples the output of the bottleneck by accommodating encoder output at each level using skip-connections thus learning locally-relevant details. A U-Net classifier also specializes in providing state-of-the-art results while requiring only a few training images to learn. It does so by using each training image efficiently to learn a more precise segmentation map.

Our implementation of the U-Net GAN differs from [8] in two main areas:

Architecture:

Method	T1_PSNR	T1_SSIM	T1CE_PSNR	T1CE_SSIM
DCGAN	21.3	0.69	25.57	0.7
DCGAN+style transfer	**29.64**	**0.87**	**32.46**	**0.86**
WGAN	19.43	0.63	21.4	0.36
WGAN+style transfer	28.49	0.85	30.74	0.76
WGAN-GP	21.26	0.73	25.14	0.71
WGAN-GP+style transfer	28.88	0.85	32.27	0.85
UNetGAN (normal weight init)	15.79	0.63	12.77	0.29
UNetGAN (normal weight init)+style transfer	25.21	0.69	19.9	0.31
UNetGAN (ortho weight init)	17.88	0.71	20.4	0.66
UNetGAN (ortho weight init)+style transfer	27.63	0.79	26.91	0.79
AGGrGAN (total agg)	19.42	0.67	20.01	0.38
AGGrGAN (total agg)+style transfer	28.08	0.81	28.45	0.6
AGGrGAN (top 3 PSNR)	19.12	0.69	22.85	0.58
AGGrGAN (top 3 PSNR)+style transfer	28.62	0.84	31.1	0.8

Fig. 2. Ablation study showing the effectiveness of aggregation and style transfer. Without the style transfer the order of performance that we get is DCGAN $>$ WGAN-GP $>$ WGAN $>$ AGGrGAN $>$ UNet GAN (orthonormal weights) $>$ UNet GAN (normal weights). The impact of style transfer seems to be significant. In the case of DCGAN it led to a PSNR increase of 8 points and SSIM increase of nearly 0.2 points.

We use a standard U-Net architecture (as described in [16]) as base for the Discriminator(D^U) with a few modifications. First, we use Leaky ReLU (with negative slope $= 0.2$) as activation functions between convolutional layers. Second, for down-sampling (in the encoder), we use average pooling with a 2×2 kernel size. Third, for up-sampling (in the decoder), we use transposed convolutional layers with a 2×2 kernel size and a stride of 2. Finally, we use a convolutional layer with a 24×24 kernel size which produces a singular value for 64×64 input image size. This is, subsequently passed through a sigmoid activation function to give a probabilistic value indicating real/fake image. As part of our exploratory study, we also used 2 different weight initialization methods for the Discriminator: Normal(sampled from $\mathcal{N}(0, 0.02)$) and Orthogonal (as described in [17]). The Generator(G^U) architecture is exactly the same as that for DCGAN.

Training:

We adopt a training methodology similar to that for DCGAN. The BCE loss is used to train both the G^U and the D^U as described in 3.4. Training is done for 1000 epochs with a batch size of 128. Adam optimizer with beta1=0.5, beta2=0.999 and lr=2e-4 is used for both G^U and D^U.

The main reason behind using a different U-Net GAN architecture than [8] was to keep the training process same for all the GANs. However, it is worthwhile to note that this architecture may not result in the best possible synthetic image generation in terms of evaluation metrics as this architecture does not make use of the per-pixel outputs from the Discriminator to train the Generator but rather combines them to produce a singular value (fake/real input image) to compute the Generator loss.

3.7 Aggregation

From the set of 5 (1 DCGAN + 2 WGAN + 2 UNet GAN) fake images generated, we choose the top 3 images based on SSIM/PSNR as suggested by [1], aggregation algorithm is then run on these images. Firstly, we apply a Sobel filter to produce edge mapped images, then a Gaussian filter is applied to smooth the edges (we found sigma = 2 to be the best for this purpose). Then, the images are weighted based on the value of this smoothened edge mapped image. Finally a weighted addition of the images is carried out to generate the aggregated image. We also experimented with a PSNR/SSIM metric based weighting scheme but found the results to be similar to that of the above. In the aggregation algorithm proposed in [1], they only chose top 2 out of 3 fake images based on PSNR/SSIM metrics. However, we experimented with an ablation study of removing this top n selection component by performing aggregation across all the images. As mentioned in the result and analysis section this proved to be not very effective.

3.8 Style Transfer

We apply style transfer to the output of GAN models to improve the resemblance of synthetic images with respect to the real images. We use a pre-trained VGG-19 model (pre-trained on ImageNet) to extract style and content features of the image. The synthetic image is passed through the intermediate layers 0, 5 and 10 of the VGG-19 model and the resulting output is used to calculate the total loss which

is a linear combination of style loss (weight = 10, alpha) and content loss (weight = 5, beta). The parameters alpha and beta can be tuned to control the degree of similarity. The total loss is minimized by an Adam optimizer with beta1 = 0.9 and beta2 = 0.999. The content image (fake image) and the style image (real image) is given to the model and the optimizer updates the content image to minimize the total loss every epoch. We used 500 epochs for the style transfer process.

3.9 Classifier model

To classify the images as either T1 or T1ce class, we used a classification model which consists of 2D convolution, ReLU and max pooling layers. We use 2 blocks each consisting of 2 Conv2D layers followed by a ReLU and a max pooling layer. Finally we flatten the output and pass it through 2 consecutive linear layers. We used cross entropy loss for training the model. We chose a batch size of 256 and image size of 64x64. We used Adam optimizer with learning rate of 1e-3 for 15 epochs for all the cases. Our case study was performed on 5 different proportions of real data and fake data which we used to train the model. These 5 cases are explained in detail in the next section.

4 Results and Analysis

4.1 Evaluation metrics: SSIM and PSNR

To quantitatively evaluate our approach we use Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index Measure (SSIM). We use these metrics to evaluate the performance of each of the individual GANs and the aggregated image. These metrics are also used to evaluate the performance after style transfer. We now formally define these metrics

PSNR: It denotes the ratio of maximum intensity value and present noise value. Maximum intensity in our images is 255 and the square root of the Mean Squared Error (MSE) can be used to measure noise. Therefore, PSNR is evaluated as follows:

$$PSNR = 20 \log_{10} MAX_I / \sqrt{MSE} \tag{5}$$

SSIM: It denotes the degree of similarity between two images. The closer the similarity the higher the SSIM. Two identical images have SSIM = 1. SSIM is evaluated using the following formula

$$SSIM = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{6}$$

Here, μ_x and μ_y are the mean intensity value of both the images, σ_x and σ_y are the standard deviation of the intensity values present in both the images. σ_{xy} is the covariance between the intensities of both the images. c_1 and c_2 are constants used to negate the weak denominator effect.

4.2 Generated Images

We run inference on the model and the trained weights to generate the fake images corresponding to a real image. For every real image we run 100 rounds of inference using random gaussian latent vectors, then, we select the fake image with the highest PSNR with respect to the real image. Based on our experiments, we got the highest mean PSNR across all the 369 image pairs for T1 and T1ce classes out of the 4 classes in the BraTS 2020 dataset. Therefore, we only used results for T1 and T1ce for analysis. Figure 3 shows sample generated images using individual models and after aggregation, for both the classes, note that these are the images after style transfer. Note that we have upscaled the images in the aforementioned figure from 64 x 64 resolution to 256 x 256 resolution in order to better visualize the differences between images without the images getting distorted. For this purpose we used the super resolution interface in the cv2 python library, EDSR x4 pretrained model was used with this interface to upscale the images dimensions by 4 times.

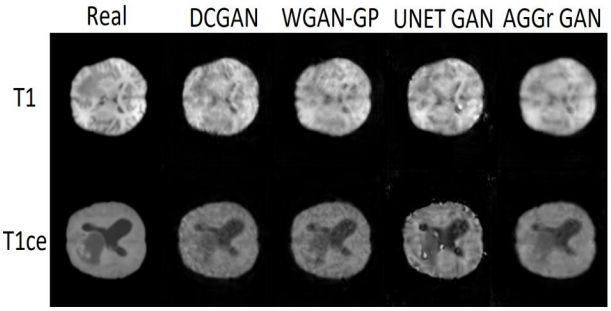


Fig. 3. Sample images generated by DCGAN, WGAN-GP, UNet GAN and AGGrGAN and their comparison with the real image. We only used images from T1 and T1ce modality in our experiments. Note that these images are upscaled from 64 x 64 resolution to 256 x 256 resolution using upscaling techniques from cv2 python library.

Figure 2 shows an ablation study of various components of the AGGrGAN model. Quantitatively judging, DCGAN with style transfer seems to be the best performing model with PSNR = 29.64 and SSIM = 0.87.

4.3 Training

In this subsection we analyze the generator and discriminator losses as well as PSNR and SSIM metrics as the model training progresses. We show the graphs for the best performing model that is DCGAN. Figure 4 shows the Generator and Discriminator losses as the training proceeds. It is evident that training the Generator of a GAN is difficult than training the Discriminator since the former loss is very volatile as compared to the latter. The Generator loss first decreases rapidly then starts decreasing steadily.

The PSNR value trend seems to agree with the above graph, it increases rapidly in the beginning followed by a steady increase as can be seen in Figure 5

The SSIM scores on the other hand continuously increase during training if we ignore the fluctuations caused due to instability in training a GAN. This can be seen in Figure 6. These results show that we could have run the DCGAN for more than 500 epochs, however, we couldn't do it due to resource constraints.

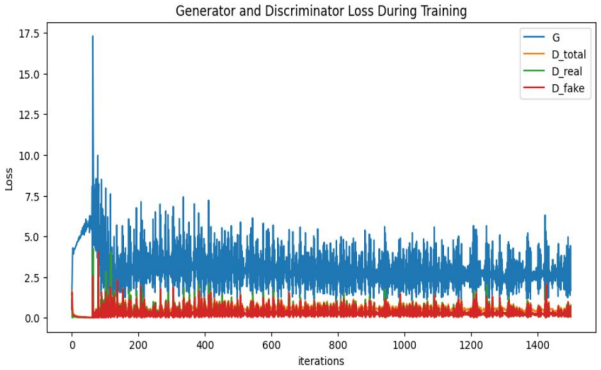


Fig. 4. Generator and Discriminator losses of DCGAN as training progresses. The blue curve denotes the Generator losses which rapidly decreases at first then steadily decreases as the training progresses. The Discriminator loss on the other hand is more or less constant during the training.

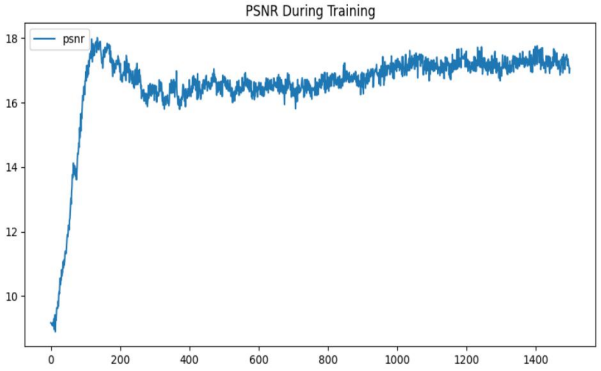


Fig. 5. PSNR score of the Generated images during training.

4.4 Classification using fake images

We conducted 5 case studies using different proportions of real and fake data for training the classifier. In each case 50% of the testing data was used for validation. Note that since T1 and T1ce images look different from each other (especially in terms of intensity), most of the cases result in a high accuracy score.

Case 1: We chose 80% of real images for training and 20% of real images for testing. This is our base case where we classified the images with 89.8% accuracy.

Case 2: We chose 100% of the fake images produced by WGAN-GP for training and 100% of real images for testing. We used this setup to qualitatively assess the quality of the fake images. Classification accuracy was 90.1% accuracy in this case.

Case 3: We used 50% of real images and 50% of fake images for training and 50% of fake images for testing. This is the most common scenario in the real world where we would use a mixture of real data with augmented data to boost model performance. Classification accuracy was 95% for this case which is considerably higher than Case 1.

Case 4: This is similar to Case 2 with the only change being that we used the fake images generated by DCGAN instead. Classification accuracy was 90% in this case which is approximately the same as Case 2.

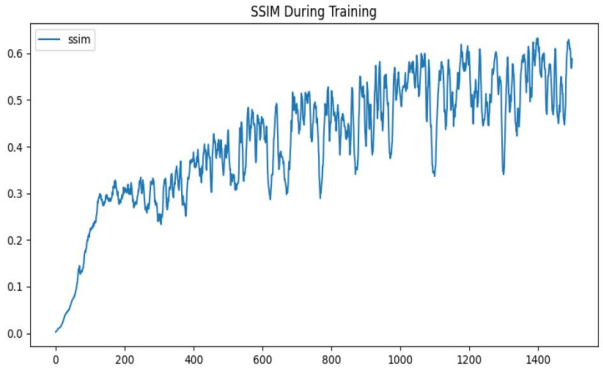


Fig. 6. SSIM score of the Generated images during training.

Case 5: We made a slight change to Case 1 by additionally adding 100% of fake images to the training data. This resulted in accuracy of 92% which is slightly higher than Case 1.

We can clearly see that in all the cases adding synthetic images in the training processes boosts the performance of the classification model.

5 Conclusion

S In this work, we showed a mechanism to create and aggregate synthetic image using the output of various GAN models. This allows us to learn features from the shared space of all the latent vectors. We explained each GAN model and the aggregation process in detail. We also performed style transfer on top of the fake images to add more internal details of the brain to the fake images to further increase the resemblance with the real images which is evident by a sharp increase in PSNR and SSIM scores. We performed an ablation study to understand the impact of each part of our workflow. The experimental setup was explained and we showed that DCGAN with style transfer resulted in the best performance. Finally, we trained a basic classifier with a mixture of real and fake data and show the boost in the performance of the classification model. Therefore, we quantitatively and qualitatively evaluated the output of our models and show that the GANs can produce a good quality MRI image of a brain tumor. This synthetic data is useful since there is no associated data handling risks.

References

[1] D. Mukherkjee, P. Saha, D. Kaplun, A. Sinitca, and R. Sarkar, "Brain tumor image generation using an aggregation of GAN models with style transfer," *Scientific Reports*, vol. 12, no. 1, Jun. 2022. [Online]. Available: https://doi.org/10.1038/s41598-022-12646-y

[2] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos, "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," *Scientific Data*, vol. 4, no. 1, Sep. 2017. [Online]. Available: https://doi.org/10.1038/sdata.2017.117

[3] B. H. Menze, A. Jakab, S. Bauer, J. Kalpathy-Cramer, K. Farahani, J. Kirby, Y. Burren, N. Porz, J. Slotboom, R. Wiest, L. Lanczi, E. Gerstner, M.-A. Weber, T. Arbel, B. B. Avants, N. Ayache, P. Buendia, D. L. Collins, N. Cordier, J. J. Corso, A. Criminisi, T. Das, H. Delingette, C. Demiralp, C. R. Durst, M. Dojat, S. Doyle, J. Festa, F. Forbes, E. Geremia, B. Glocker, P. Golland, X. Guo, A. Hamamci, K. M. Iftekharuddin, R. Jena, N. M. John, E. Konukoglu, D. Lashkari, J. A. Mariz, R. Meier, S. Pereira, D. Precup, S. J. Price, T. R. Raviv, S. M. S. Reza, M. Ryan, D. Sarikaya, L. Schwartz, H.-C. Shin, J. Shotton, C. A. Silva, N. Sousa, N. K. Subbanna, G. Szekely, T. J. Taylor, O. M. Thomas, N. J. Tustison, G. Unal, F. Vasseur, M. Wintermark, D. H. Ye, L. Zhao, B. Zhao, D. Zikic, M. Prastawa, M. Reyes, and K. V. Leemput, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Transactions on Medical Imaging*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015. [Online]. Available: https://doi.org/10.1109/tmi.2014.2377694

[4] S. Bakas, M. Reyes, A. Jakab, S. Bauer, M. Rempfler, A. Crimi, R. T. Shinohara, C. Berger, S. M. Ha, M. Rozycki, M. Prastawa, E. Alberts, J. Lipková, J. B. Freymann, J. S. Kirby, M. Bilello, H. M. Fathallah-Shaykh, R. Wiest, J. Kirschke, B. Wiestler, R. R. Colen, A. Kotrotsou, P. LaMontagne, D. S. Marcus, M. Milchenko, A. Nazeri, M. Weber, A. Mahajan, U. Baid, D. Kwon, M. Agarwal, M. Alam, A. Albiol, A. Albiol, A. Varghese, T. A. Tuan, T. Arbel, A. Avery, P. B., S. Banerjee, T. Batchelder, K. N. Batmanghelich, E. Battistella, M. Bendszus, E. Benson, J. Bernal, G. Biros, M. Cabezas, S. Chandra, Y. Chang, and et al., "Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the BRATS challenge," *CoRR*, vol. abs/1811.02629, 2018. [Online]. Available: http://arxiv.org/abs/1811.02629

[5] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014. [Online]. Available: https://arxiv.org/abs/1406.2661

[6] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.06434

[7] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017. [Online]. Available: https://arxiv.org/abs/1701.07875

[8] E. Schönfeld, B. Schiele, and A. Khoreva, "A u-net based discriminator for generative adversarial networks," 2020. [Online]. Available: https://arxiv.org/abs/2002.12655

[9] C. Han, H. Hayashi, L. Rundo, R. Araki, W. Shimoda, S. Muramatsu, Y. Furukawa, G. Mauri, and H. Nakayama, "GAN-based synthetic brain MR image generation," in *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE, Apr. 2018. [Online]. Available: https://doi.org/10.1109/isbi.2018.8363678

[10] D. Nie, R. Trullo, J. Lian, L. Wang, C. Petitjean, S. Ruan, Q. Wang, and D. Shen, "Medical image synthesis with deep convolutional adversarial networks," *IEEE Transactions on Biomedical Engineering*, vol. 65, no. 12, pp. 2720–2730, Dec. 2018. [Online]. Available: https://doi.org/10.1109/tbme.2018.2814538

[11] H. Emami, M. Dong, S. P. Nejad-Davarani, and C. K. Glide-Hurst, "Generating synthetic CTs from magnetic resonance images using generative adversarial networks," *Medical Physics*, vol. 45, no. 8, pp. 3627–3636, Jul. 2018. [Online]. Available: https://doi.org/10.1002/mp.13047

[12] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers, "Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning," *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285–1298, May 2016. [Online]. Available: https://doi.org/10.1109/tmi.2016.2528162

[13] R. C. Petersen, P. S. Aisen, L. A. Beckett, M. C. Donohue, A. C. Gamst, D. J. Harvey, C. R. Jack, W. J. Jagust, L. M. Shaw, A. W. Toga, J. Q. Trojanowski, and M. W. Weiner, "Alzheimer's disease neuroimaging initiative (ADNI): Clinical characterization," *Neurology*, vol. 74, no. 3, pp. 201–209, Dec. 2009. [Online]. Available: https://doi.org/10.1212/wnl.0b013e3181cb3e25

[14] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," 2016. [Online]. Available: https://arxiv.org/abs/1611.07004

[15] S. Sarkar, A. Kumar, S. Chakraborty, S. Aich, J.-S. Sim, and H.-C. Kim, "A cnn based approach for the detection of brain tumor using mri scans," *Test Engineering and Management*, vol. 83, p. 16580 – 16586, 06 2020.

[16] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[17] A. M. Saxe, J. L. McClelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," *arXiv preprint arXiv:1312.6120*, 2013.