Project Description

This project, titled **Synthetic Aperture Radar (SAR) Image Generation based on Generative Adversarial Network (GAN) and Target Characteristics**, aims to use GAN-based generative deep learning models to generate small-scale SAR images of different categories with distinct characteristics. The author targets to find the most appropriate model for addressing this problem among the various GAN-based models by comparison.

Technical Background

Synthetic Aperture Radar (SAR) is widely applied in environment status monitoring, topographic features mapping, object tracking, and applications for military and security monitoring [1]. It has the following advantages [1]:

1. Can be measured in all weathers (Robust to weather conditions)
2. Potential to reach very high resolution

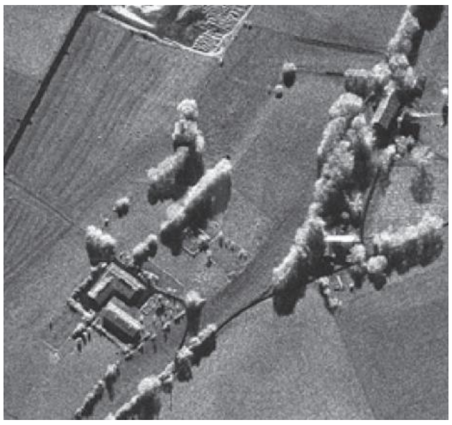
Fig 1. *A sample SAR image. Source: [1]*

It is natural to combine machine learning with SAR remote sensing. Such algorithms include image classification, semantic segmentation, object detection etc. can be adapted to SAR images to achieve target tasks [2]. However, the cost of acquiring SAR images is still high by the year of 2022 according to [3]. The challenge lies in the fact that machine learning models typically require a large amount of data to perform well, but obtaining a large amount of SAR images is cost-prohibitive. Hence, it is important for the researchers and engineers to find methods that is capable of generating photorealistic SAR images.

Generative deep learning is a subfield of machine learning that focuses on using deep learning models to generate new, synthetic data that is similar to a training dataset. These models can generate a wide range of data, including audio, text, and image. Typical deep generative models include Normalizing Flows, Variational AutoEncoders (VAE) and Energy-Based Models [4]. In 2014, *I. Goodfellow, Y. Bengio* and their colleagues [5] proposed the structure of Generative Adversarial Networks (GAN), which became the dominant model in generative deep learning in the following years. It has been extensively developed and have spawned numerous variants. GAN is known for its excellent performance and relatively simple structure, which makes it a suitable choice for expanding the SAR image database by generating deep-fake images.

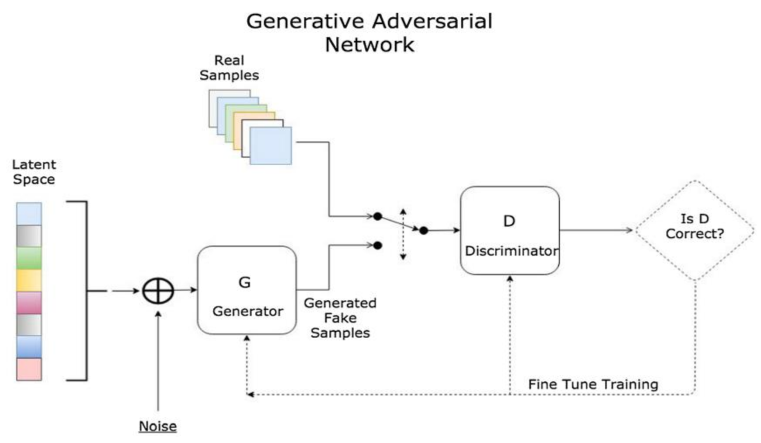

***Fig 2.** General structure of GAN. Source: [6]*

Fig 2 shows the structure of a standard GAN. Two neural networks: Generator and Discriminator, are trained simultaneously in GAN. Generator is responsible for generating images that are as similar as the real images as possible. Discriminator needs to identify whether an image is real or fake. Ideally, both networks become more powerful after each iteration under the idea of adversarial training. Generator can output photorealistic images after sufficient epochs of training [5].

There has been a number of studies focusing on generating SAR images by GAN. *S. Bhamidipati et al.* [7] used Deep-Convolutional-GAN (DCGAN) to generate SAR images. *M. Zhu et al.* [8] added a Local Interpretable Model-Agnostic Explanation (LIME) block to select those that possess the representative features from the SAR images generated by DCGAN. *Z. Cui et al.* [9] applied a Wasserstein Loss DCGAN with Gradient Penalty to generate SAR images, and filtered out those with low quality by a PCA-based Support Vector Machine and an Azimuth Filter. *S. Du et al.* [10] proposed a Multi-Constraint-GAN that specializes in imitating SAR images, and compared its performance with other models. GAN holds the potential of being combined with other deep learning models. *Q. Song et al.* [11] implemented an Adversarial AutoEncoder (AAE, the combination of VAE and GAN) for synthesizing SAR images. Although this field is not included as discussed in this project, it is worth mentioning that style transfer is another task that GAN can achieve. *M. Liu et al.* [12] designed a SiaNet-CycleGAN to transfer the 3DS models to SAR images.

Project Outline

- **Project Requirements**
  1. Hardware Requirements
     a) The author's programmable **Personal Computer** with high performance in parallel computing
  2. Software Requirements
     a) Programming Language: **Python** and **IPython**
     Python is an interpreted programming language, which is the optimal choice for this project, in which massive data processing and complex calculation workflow is involved. IPython is a variant of Python, which is suitable for lightweight program.
     b) Programming Platform: **JupyterLab** and **PyCharm**

The former is for writing IPython files, and the latter is for writing Python files.

c) Key Libraries: **PyTorch** and **Scikit-Learn**

Meta's PyTorch is an open-source library, and it is the most popular deep learning framework in academic field. PyTorch encapsulates many of the common functions in deep learning, making it easy to train neural networks. Scikit-Learn is an open-source library specialize in classic machine learning. Both libraries can be imported in Python.

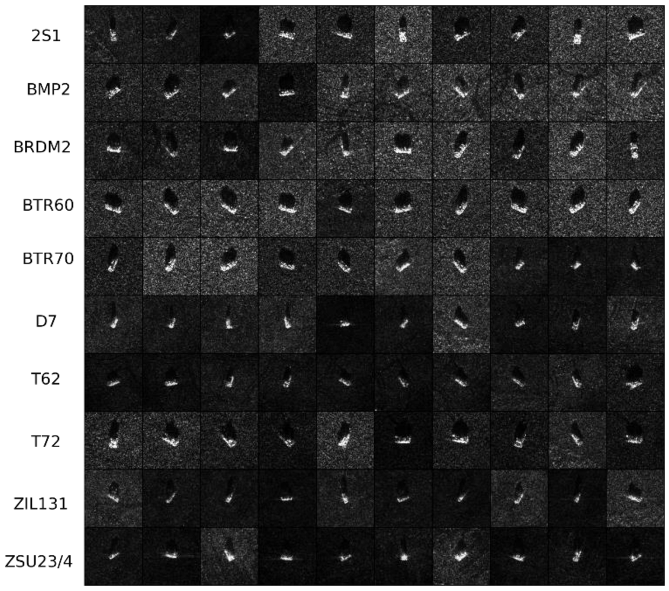

Fig 3. *A glance of MSTAR dataset.*

3. Database

   The database utilized in this project is **MSTAR (Moving and Stationary Target Acquisition and Recognition)**. *It consists of SAR images of 10 different types of military vehicles, with approximately 500 images per category.*

- **Algorithms Involved in this Project**

1. **Neural Networks + Back Propagation Algorithm**

   Using back propagation to optimize the parameters [13] in GAN to achieve better performance after each iteration is the core algorithm in this project.

2. Classic Machine Classification Algorithm (i.e., SVM, *k*-NN)
3. Image Processing Algorithm (i.e., RGB-to-Grayscale, Normalization)
4. Image Quality Evaluation Algorithm (i.e., FID Score, PSNR, SSIM)

Main Tasks and Targets

- **Project Targets**

  This project aims generate high-quality SAR images of 10 categories by GAN-based models.

- **Main Tasks**

1. **Initial design and implementation of different GAN-based models** **(Completed by Feb. 10th)**

   This is the project's central task which the author spends most of the time working on. Although the models that the author intends to implement are famous GAN-Based models that are proposed in highly cited papers, adjusting the structure and hyperparameters of these models to transform them into generative models capable of generating ***SAR images with high inter-category similarity*** *(see Fig 3, the images of different category are similar in MSTAR) remains* *a challenging task. The author firstly trains each GAN model on the shuffled, uncategorized MSTAR dataset to see the generalization ability of a model (model state 1).*

2. Modify the models to generate images of a given category (Completed around Mar. 10th)
   If a model in Task 1 performs well, it is modified to a conditional model or trained on each category of images rather than the entire dataset (model state 2).
3. Evaluate different models (Completed around Mar. 31st)
   This task is conducted throughout the project. Common evaluation includes the computing cost of training a model, the convergence speed, the quality of generated images and to what extent the generated images affect the classification accuracy, etc.
4. Fine-tune the GAN-based models (Completed around Apr. 11st)
   It is necessary to slightly adjust a well-performed model to achieve even better performance.

Measurable Outcomes

The outcomes of this project are expected to be in pure software level:

1. Several trained GAN-based models that can generate high-quality SAR images of 10 categories
2. Images generated by different models
3. Evaluations of different models

Work Done so far and Literature Review

Due to the limited length of the article, the author cannot present all the work that has been done.

- **Pre-processing of SAR images**

The author mapped the SAR images **from RGB into Grayscale** and **normalizes** them. The author believed it a wise choice to use one-channel grayscale images instead of three-channel RGB images. Due to the colour features of SAR images, turning them from RGB to grayscale was almost lossless, which cannot be distinguished by human eyes (see Fig 4.). This reduced the total parameters needed in training GANs. Normalization operation switched the range of each pixel from the interval [0,1] to [-1,1], which was a common trick for increasing the convergence speed of neural nets. The SAR image shape was (1,128,128), which represents *channel, width,* and *height* respectively.

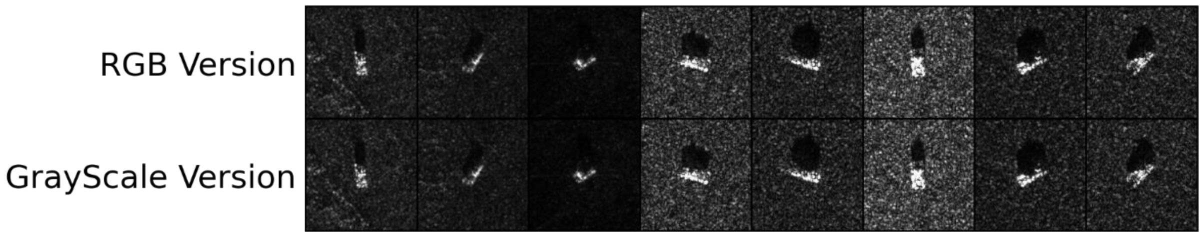

Fig 4. *No difference between RGB and Grayscale format for SAR images*

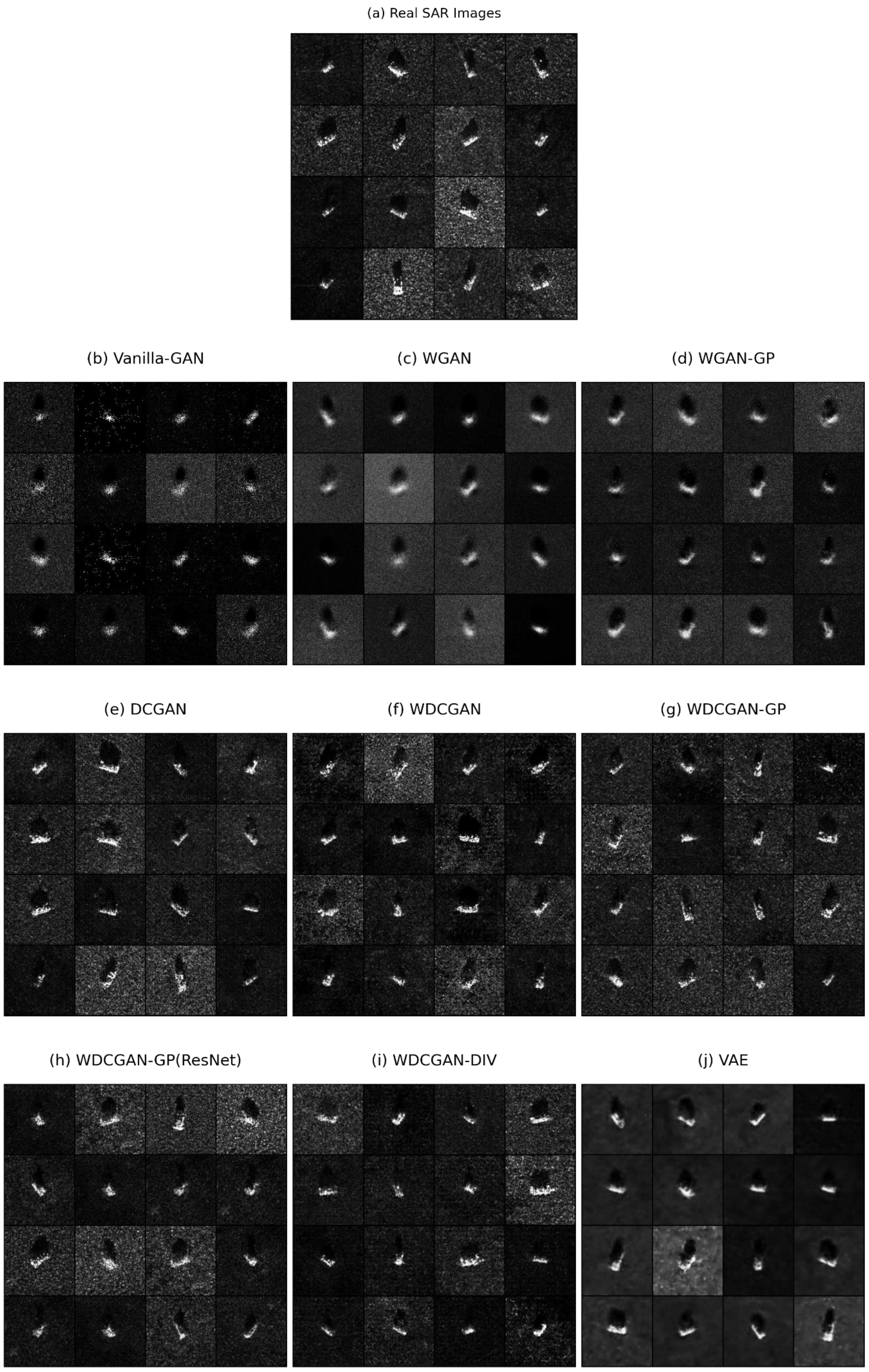

Fig 5. *A glance of SAR images generated by different models*

- **Implementation of GAN-based models**

  So far, the author has constructed 9 different generative models (See Table 1). The author only demonstrated the general performance of these models (means they are not conditional ones).

1.	Vanilla-GAN	2.	WGAN	3.	WGAN-GP
4.	DCGAN	5.	WDCGAN	6.	WDCGAN-GP
7.	WDCGAN-GP (ResNet)	8.	WDCGAN-DIV	9.	VAE

Table 1. *Models implemented by the author so far*

1. Vanilla-GAN

'Vanilla' meant 'Original' in machine learning, which meant this model was the original GAN model proposed in [5], 2014. Training a Vanilla-GAN was helpful for beginners to understand the working principles of GAN together with how network structure and hyperparameters affected the performance and convergence of GAN. The author spent a considerable amount of time to carefully design the model and fit it on SAR image dataset.

Network	Layers	Type	Number of output features	Normalization	Activation
Discriminator (D)	1	Fully Connected (FC)	1024	/	LeakyReLU
	2	FC	512	/	LeakyReLU
	3	FC	512	/	LeakyReLU
	4	FC	1	/	Sigmoid
Generator (G)	1	FC	64	BatchNorm	LeakyReLU
	2	FC	128	BatchNorm	LeakyReLU
	3	FC	512	BatchNorm	LeakyReLU
	4	FC	16384 ⇔ (1,128,128)	/	Tanh

Table 2. *Structure of Vanilla-GAN*

(a) General Objective Function:

$$\min_G \max_D V(D,G) = E_{x\sim P_r}[log\,D(x)] + E_{x\sim P_g}\big[log\big(1-D(x)\big)\big] = -2log2 + \min_G[2JSD\big(P_r\big|\big|P_g\big)]$$

$$JSD\ is\ Jensen\text{–}Shannon\ Divergence$$

(c) Objective Function for D: $Maximize\ E_{x\sim P_r}[log\,D(x)] + E_{x\sim P_g}\big[log\big(1-D(x)\big)\big]$

(d) Objective Function for G: $Maximize\ E_{x\sim P_g}[log\,D(x)]$

Formula 1. *Objective (Loss) functions of Vanilla-GAN. Source: [5][13]*

The author designed a Generator (G) and a Discriminator (D) that followed the structure in Table 2. G aimed to optimize Formula 1(b) and D aimed to optimize Formula 1(c) via back-propagation algorithm. Note that training a GAN consisting of only multi-layer perceptron and achieving merely satisfactory performance was a very hard task. For the GAN in Table 2, a change in the number of output features in D, a removal of Batch Normalization in G or adjusting the learning rate of optimizers would possibly cause one of the three issues: Vanishing Gradients, Mode Collapse or

Convergence Failure. In General, Vanilla-GAN had the worst generating image quality (see Fig 5(b)) and the lowest training stability.

2. Wasserstein-GAN (WGAN)

(a) General Objective Function:

$$W(P_r, P_g) = \sup_{\|f\|_{L\leq 1}} E_{x\sim P_r}[f(x)] - E_{x\sim P_g}[f(x)], W\ is\ Wasserstein\ (Earth - Mover)\ Distance$$

$$f(x)\ must\ be\ 1 - Lipschitz\ Continuous, \|f(a) - f(b)\| \leq \|a - b\|$$

(b) Objective Function for D: $Maximize\ E_{x\sim P_r}[D(x)] - E_{x\sim P_g}[D(x)]$

$$To\ make\ D(x)\ 1 - Lipschitz\ Continuous, D_{weights} \in [-c, c], c\ is\ a\ constant, usually\ c < 1$$

(c) Objective Function for G: $Maximize\ E_{x\sim P_g}[D(x)]$

Formula 2. *Objective (Loss) functions of WGAN. Source: [15]*

To overcome to instability issue in optimizing in training Vanilla-GAN, WGAN was proposed in [15]. When two data distribution do not overlap, Jensen–Shannon divergence (Formula 1) went to zero, which made GAN untrainable and fail to converge [15]. The introduce of Wasserstein distance overcame this issue, since it could measure the distance of two non-overlapping distributions [15].

Network	Layers	Type	Number of output features	Normalization	Activation
Discriminator	4	FC	1	/	/
(D)	The weights of the D network are restricted to the range [-0.01, 0.01] (weight clipping)				

Table 3. *Structure of WGAN, the rest part is the same to Table 2.*

The only structural change of WGAN compared to Vanilla-GAN was removing the Sigmoid activation in the final layer of D. Although there were still occasional failures in training WGAN, it was much stabler than Vanilla-GAN. The generated SAR images was in Fig 5(c), better than Vanilla-GAN.

3. Wasserstein-GAN with Gradient Penalty (WGAN-GP)

Objective Function for D: $Maximize\ E_{x\sim P_r}[D(x)] - E_{x\sim P_g}[D(x)] - \lambda E_{x\sim \chi}[\|\nabla_x D(x)\|_2 - 1]$

$$The\ last\ term\ is\ Gradient\ Penalty, aims\ to\ force\ D(x)\ to\ be\ 1 - Lipschitz\ Continuous$$

Formula 3. *Objective (Loss) functions for WGAN-GP, the rest is identical to Formula 2, Source: [16]*

Usually, when using weight clipping to train WGANs, the weights in D will always go close to the boundary value (In the author's case, -0.01 and 0.01) [16]. In this occasion, D's performance was limited by the boundary value. A poor D led to a poor G, which resulted in poor generating images. WGAN-GP replaced weight clipping by a gradient penalty in D's loss function to satisfy the 1-Lipschitz condition [16]. This improved version of WGAN tremendously enhanced the performance and the training stability of GAN. The structure of the WGAN-GP designed by the author was the same as that of WGAN in Table 3. This network converged in a short time at the very first training, which did not happen for the previous two GANs. The generated images were shown in Fig 5(d).

4. Deep Convolutional GAN (DCGAN)

In 2012, *A. Krizhevsky, I. Sutskever* and *G. Hinton* showed that deep neural network based on multiple convolution layers is better at image recognition than any other approaches to computer vision, and by a very large margin [17]. That moment, known as ImageNet moment, changed the whole AI field. In 2015, *A. Radford et al.* [18] showed that replacing the fully connected layers by convolution layers would significantly increase a GAN's performance.

Network	Layers	Type	Output feature map dimension	Kernel Size/Stride/Padding	Normalization	Activation
Discriminator (D)	1	Conv	1024	4/2/1	/	LeakyReLU
	2	Conv	512	4/2/1	BatchNorm	LeakyReLU
	3	Conv	256	4/2/1	BatchNorm	LeakyReLU
	4	Conv	128	4/2/1	BatchNorm	LeakyReLU
	5	Conv	64	4/2/1	BatchNorm	LeakyReLU
	6	Conv	1	4/1/0	/	Sigmoid
Generator (G)	1	Deconv	64	4/1/0	BatchNorm	ReLU
	2	Deconv	128	4/2/1	BatchNorm	ReLU
	3	Deconv	256	4/2/1	BatchNorm	ReLU
	4	Deconv	512	4/2/1	BatchNorm	ReLU
	5	Deconv	512	4/2/1	BatchNorm	ReLU
	6	Deconv	1	4/2/1	/	Tanh

Table 4. *Structure of DCGAN*

Above was the author's DCGAN structure. The objective functions of DCGAN were identical to those of Vanilla-GAN in Formula 1 [18]. The generated SAR samples of DCGAN were shown in Fig 5(e). Starting from this model, the quality of images produced by GAN progressed significantly. While the images generated by previous models were merely 'similar' to SAR images, the output of DCGAN was 'photorealistic'.

5. Wasserstein Deep Convolutional GAN (WDCGAN)

The author combined Wasserstein-distance with DCGAN in this attempt. Hence the objective functions of WDCGAN were the same as those of WGAN in Formula 2. While keeping the Generator network structure unchanged from Table 4, the author modified the Discriminator network structure as follows:

Network	Layers	Type	Output feature map dimension	Kernel Size/Stride/Padding	Normalization	Activation
	1	Conv	1024	4/2/1	LayerNorm	LeakyReLU
	2	Conv	512	4/2/1	LayerNorm	LeakyReLU
	3	Conv	256	4/2/1	LayerNorm	LeakyReLU
	4	Conv	128	4/2/1	LayerNorm	LeakyReLU

Discriminator (D)	5	Conv	64	4/2/1	LayerNorm	LeakyReLU
	6	Conv	1	4/1/0	/	/
	The weights of the D network are restricted to the range [-0.01, 0.01] (weight clipping)					

Table 5. *Structure of WGAN, the rest part remains the same to Table 4.*

The reason why the author replaced Batch Normalization with Layer Normalization was that the former can change the activations and therefore the 1-Lipschitz constraint may no longer be maintained during the iteration of the model, which might lead to convergence issues or unstable gradients [15]. However, in the author's experiments, it seems that WDCGAN is harder to converge than DCGAN. This might result from the weight clipping value. See Fig 5(f) for generated images.

6. Wasserstein Deep Convolutional GAN with Gradient Penalty (WDCGAN-GP)

This model was the combination of WGAN-GP and convolution layers. The objective functions were identical to those of WGAN-GP (Formula 3). Expect for replacing weight clipping operation by adding a gradient penalty term in the loss function of D, the rest of the model structure was the direct copy of WDCGAN (Table 4). This model took the shortest training time to converge so far, which meant that only after a small number of iterations, the model was able to generate photorealistic images with high diversity. WDCGAN-GP could perform well even when the author reduced the complexity of its structure, indicating that WDCGAN-GP has the potential to reach high performance.

7. WDCGAN-GP with Residual Networks (WGAN-GP(ResNet))

In 2015, *K. He et al.* [19] introduced residual connections to deep learning, which significantly improved the training stability and the performance, and influenced the design of deep neural networks in subsequent years [20].

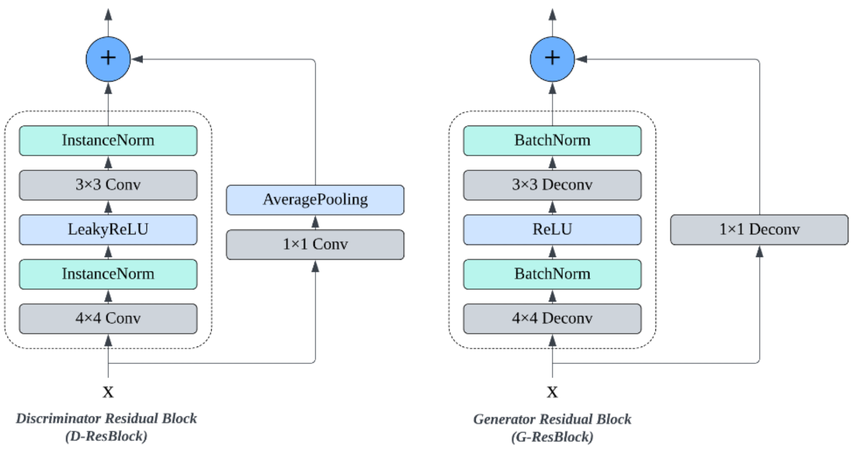

Fig 6. *The structure of D-ResBlock and G-ResBlock*

Fig 6 demonstrated the residual blocks that the author designed for the Generator and Discriminator used to generate SAR images. The central idea of residual networks was residual connection. A residual connection was a link between the input and output of a layer in a neural network that bypassed one or more intermediate layers. It allowed the input to be directly added to the output, rather than being transformed by the intermediate layers [19]. This operation helped to prevent the vanishing gradient problem [20]. There have been several papers, including [16][21], demonstrating the involvement of residual connections improve the performance of GAN.

Network	Layers	Type	Output feature map dimension	Kernel Size/Stride/Padding	Normalization	Activation
Discriminator (D)	1	Conv	32	3/1/1	InstanceNorm	LeakyReLU
	2	D-ResBlock	128	/	/	LeakyReLU
	3	D-ResBlock	128	/	/	LeakyReLU
	4	D-ResBlock	128	/	/	LeakyReLU
	5	D-ResBlock	64	/	/	LeakyReLU
	6	D-ResBlock	8	/	InstanceNorm	LeakyReLU
	7	Conv	1	4/1/0	/	/
Generator (G)	1	G-ResBlock	256	/	/	ReLU
	2	G-ResBlock	256	/	BatchNorm	ReLU
	3	G-ResBlock	128	/	/	ReLU
	4	G-ResBlock	128	/	BatchNorm	ReLU
	5	Deconv	64	4/2/1	BatchNorm	ReLU
	6	Deconv	16	4/2/1	BatchNorm	ReLU
	7	Deconv	1	4/2/1	/	Tanh

Table 6. *Structure of WDCGAN-GP(ResNet)*

The objective functions were the same as Formula 3. The training of WDCGAN-GP(ResNet) was highly stable, reflected in the smaller oscillation of the loss function values of both G and D, compared to other models, during the iteration process. The generated images are shown in Fig 5(h).

8. Wasserstein-Divergence for DCGAN (WDCGAN-DIV)

(a) General Objective Function: $W - div\left(P_r, P_g\right) \ = E_{x\sim P_r}[f(x)] - E_{x\sim P_g}[f(x)] + kE_{x\sim P_u}[\|\nabla f(x)\|^p]),$

$$with\ no\ 1 - Lipschitz\ constraint\ for\ f(x)$$

(b) Objective Function for D: $Maximize\ E_{x\sim P_r}[D(x)] - E_{x\sim P_g}[D(x)] + kE_{x\sim P_u}[\|\nabla D(x)\|^p])$

(c) Objective Function for G: $Maximize\ E_{x\sim P_g}[D(x)]$

Formula 4. *Objective (Loss) functions of WGAN-DIV. Source: [21]*

[21] proposed an improved WGAN by replacing W-distance by W-divergence, which not only preserved the property of Earth-Mover distance but also removed 1-Lipschitz constraint. Although WGAN-GP increased both the stability and performance of training WGANs, gradient penalty is an empirical formula lacking rigorous mathematical proof. In [21], it is proved mathematically that above objective function really works, and WGAN-GP is a subset of WGAN-DIV [21].

Due to the outstanding performance of DCGAN family shown previously, rather than designing and training a WGAN-DIV constructed on multilayer perceptron, the author only built it upon convolutional neural nets. The structure of WDCGAN-DIV the author designed is identical to that in Table 5. The generated samples are in Fig 5(i). However, so far, the training of this network was unstable, including convergence failure issue.

9. Variational Autoencoder (VAE)

VAE is a different type of generative model from GAN. The author regarded it as an extra experiment for this project. Unlike GAN, VAE is a mathematically elegant model, which is essentially a probabilistic model (maximum likelihood estimator) [22]. GAN, on the other hand, is an implicit generative model. It is hard to explain what GAN learns and how it learns from real data. VAEs consist of two parts: an encoder network and a decoder network. The encoder network maps the input data to a latent representation, which is typically a lower-dimensional space. The decoder network maps the latent representation back to the original space, allowing the VAE to reconstruct the input data [22].

$$\text{Objective Function: } Minimize - E_{x\sim P_r} E_{\emptyset}\, log\, p_{\theta}(x|z) + KLD(E_{x\sim P_r} p_{\emptyset}(z|x)||N(0,I)),$$
$$KLD\ is\ Kullback - Leibler\ Divergence$$

Formula 5. *Objective (Loss) function of VAE. Source: [22]*

The author will not show the structure of VAE due to the limitation of length of this report. In general, VAE has the blurred-generated-image issue, but can be avoided by combining it with GAN.

- **Comparison of different models**

Model Type	Peak Memory Usage	Training Time per Epoch	SSIM	Mean-Squared Error
Vanilla-GAN	1.6 GB	3.68 sec	0.0746	0.0324
WGAN	1.4 GB	3.80 sec	0.1047	0.0290
WGAN-GP	1.5 GB	3.93 sec	0.1039	0.0222
DCGAN	3.3 GB	8.33 sec	0.1187	0.0268
WDCGAN	3.2 GB	9.45 sec	0.1226	0.0259
WDCGAN-GP	3.9 GB	16.11 sec	0.1293	0.0258
WDCGAN-GP (ResNet)	11.7 GB	56.51 sec	0.1426	0.0242
WDCGAN-DIV	4.2 GB	19.04 sec	0.1354	0.0249
VAE	3.7 GB	5.93 sec	0.2397	0.0205

Table 7. *A comparison of the above generative models*

The parameters of neural network are stored in the memory of GPU (V-RAM). A higher peak memory usage indicates a more complex model. An epoch is the process of a model go through the entire dataset and back-propagates errors. SSIM measures the similarity of two batches of images and Mean-Squared Error (MSE) measures the average of the squared differences of them. The larger SSIM, the smaller MSE, the more similar. SSIM and MSE between real SAR images and generated SAR images were calculated and shown above.

Conclusions of Work Done so far

Before this project, the author has almost no prior knowledge about generative deep learning. Hence, the majority work done by the author so far focus on adjusting the structures and hyperparameters of different GAN-based models to make them really work in generating SAR images. During the vast number of experiments and attempts conducted, the author gradually understood the common process to define and train a GAN including some tricks, which seemingly make no sense mathematically but actually helps much. Training a neural network is a time-consuming work. The author usually sets epoch (one epoch means an entire go through of the database) to 200, and batch size to 64, in a train process. For the costliest WDCGAN-GP(ResNet), it took about 3.5 hours to finish even with a powerful GPU. However, the author had to constantly optimize the model parameters for better performance, so each model was at least modified and retrained for 5 times.

The author believes that the eye observation is the best evaluation method for the generated image quality. Although SSIM and MSE (Table 7) can, to some extent, reflect the similarity between real image and generated samples, but sometimes they are counter intuitive. For example, VAE got the highest SSIM and lowest MSE. Does this indicate that images from VAE is better than images from DCGAN (see Fig 5(e) and (j))? The answer is unequivocal: No. In previous papers about synthesizing SAR images [9][10], a common approach to measure the performance of a GAN is to find out how the generated samples affect the classification performance of SAR images, which will be included in future work of this project. On the other hand, SSIM is useful in tracking the performance progress of a model during training: If the value of SSIM is increasing, normally the performance of the model is positively progressing either.

Based on the results in Fig 5. The author concludes that GANs based on convolution layers are generally better than those based on fully connected layers in this specific project that aims to synthesis SAR images. Therefore, Vanilla-GAN, WGAN and WGAN-GP will not be further included in the rest of this project. As W-loss DCGANs, WDCGAN-GP and WDCGAN-DIV are more stable in training than WDCGAN, which will be abandoned. DCGAN takes longer time to converge and performs worse compared to the above three, but the author will include it in future plan as a comparison approach, because it uses original GAN's loss functions proposed in [5]. WDCGAN-GP(ResNet) is so far the best model that generates the most realistic SAR images with relatively high diversity, though it has the longest training time and the most complicated structure. The blurred image issue in VAE can be mitigated by adding a Discriminator that usually appears in GAN to the structure of VAE, to make it an Adversarial-VAE, which can be regarded as the combination of VAE and GAN. In summary, DCGAN, WDCGAN-GP, WDCGAN-GP(ResNet), WDCGAN-DIV and VAE will be further discussed and modified in the ongoing project.

Plan of Work to be Done.

Apart from the models discussed in this report, the author plans to implement another one to two GAN-based models. The future work of this project should mainly focus on:

1. Converting existing models into conditional models, or training models on each class of SAR images
2. Evaluating models based on classification performance

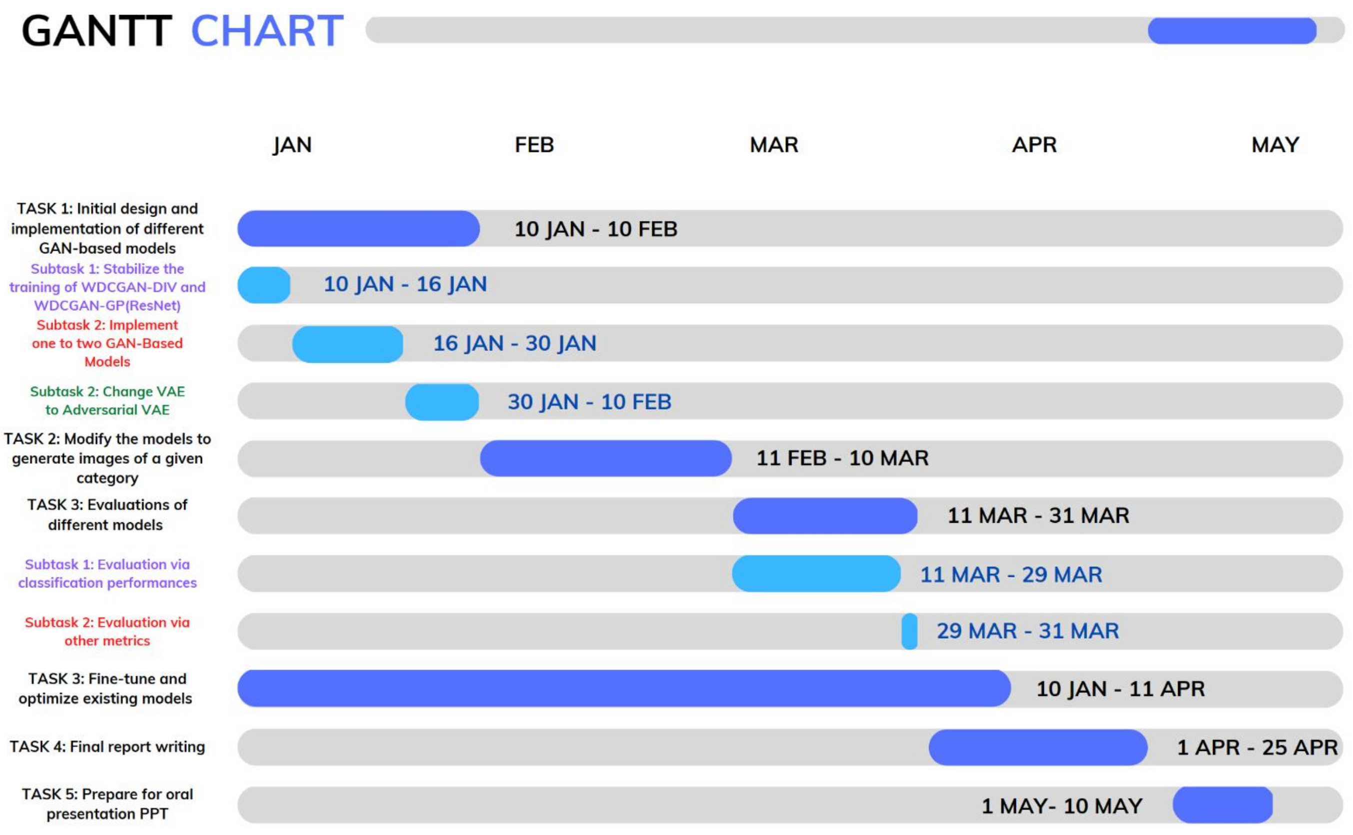

Fig 7. *Gantt chart of future work plans*

Acknowledgment and Notification

I would like to express my gratitude to *Prof. Zongyong Cui* and his graduate student for their exceptional guidance in successfully carrying out this project so far. This report is finished in Jan 10th, 2023, which suggests that the content of the report covers the work that the author has done before this specific date.

References

[1] A. Moreira, P. Prats-Iraola, M. Younis, G. Krieger, I. Hajnsek, and K. P. Papathanassiou, "A Tutorial on Synthetic Aperture Radar," *IEEE Geoscience and Remote Sensing Magazine,* vol. 1, no. 1, 2013.

[2] X. X. Zhu, S. Montazeri, M. Ali, Y. Hua, Y. Wang, L. Mou, Y. Shi, F. Xu, and R. Bamler, "Deep Learning Meets SAR: Concepts, Models, Pitfalls, and Perspectives," *IEEE Geoscience and Remote Sensing Magazine,* vol. 9, no. 4, pp. 143–172, 2021.

[3] L. S. Vailshery, "Geocento SAR Imagery Cost Worldwide 2022," *Statista*, 04-Mar-2022. [Online]. Available: https://www.statista.com/statistics/1293899/geocento-commercial-satellite-sar-imagery-resolution-cost-worldwide/. [Accessed: 05-Jan-2023].

[4] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks, "Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 44, no. 11, pp. 7327–7347, 2022.

[5] 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.

[6] P. Chaudhari, H. Agrawal, and K. Kotecha, "Data Augmentation Using MG-GAN for Improved Cancer Classification on Gene Expression Data," *Soft Computing,* vol. 24, no. 15, pp. 11381–11391, 2019.

[7] S. R. Bhamidipati, C. Srivatsa, C. Kanakapura Shivabasave Gowda, and S. Vadada, "Generation of SAR Images Using Deep Learning," *SN Computer Science,* vol. 1, no. 6, 2020.

[8] M. Zhu, B. Zang, L. Ding, T. Lei, Z. Feng, and J. Fan, "Lime-Based Data Selection Method for SAR Images Generation Using GAN," *Remote Sensing,* vol. 14, no. 1, p. 204, 2022.

[9] Z. Cui, M. Zhang, Z. Cao, and C. Cao, "Image Data Augmentation for SAR Sensor via Generative Adversarial Nets," *IEEE Access,* vol. 7, pp. 42255–42268, Mar. 2019.

[10] S. Du, J. Hong, Y. Wang, and Y. Qi, "A High-Quality Multicategory SAR Images Generation Method with Multiconstraint GAN for ATR," *IEEE Geoscience and Remote Sensing Letters,* vol. 19, pp. 1–5, 2022.

[11] Q. Song, F. Xu, X. X. Zhu, and Y.-Q. Jin, "Learning to Generate SAR Images with Adversarial Autoencoder," *IEEE Transactions on Geoscience and Remote Sensing,* vol. 60, pp. 1–15, 2022.

[12] M. Liu, L. Dong, X. Liu, M. Hui, W. Liu, and Y. Zhao, "Generating Simulated SAR Images Using Generative Adversarial Network," *Applications of Digital Image Processing XLI,* 2018.

[13] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Representations by Back-Propagating Errors," *Nature,* vol. 323, no. 6088, pp. 533–536, 1986.

[14] S. Chintala, "soumith/ganhacks: Starter from 'How to Train a GAN?" at NIPS2016," *GitHub,* 2020. [Online]. Available: https://github.com/soumith/ganhacks. [Accessed: 08-Jan-2023].

[15] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN", *arXiv e-prints,* 2017.

[16] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, "Improved Training of Wasserstein GANs", *arXiv e-prints,* 2017.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM,* vol. 60, no. 6, pp. 84–90, 2017.

[18] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", *arXiv e-prints,* 2015.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2016.

[20] A. Zhang, Z. Lipton, M. Li, and A. J. Smola, "Dive into Deep Learning", *arXiv e-prints,* 2021.

[21] J. Wu, Z. Huang, J. Thoma, D.Acharya, and L. V. Gool, "Wasserstein Divergence for GANs", *arXiv e-prints,* 2017.

[22] Kingma, D. P. and Welling, M., "Auto-Encoding Variational Bayes", *arXiv e-prints,* 2013.