Deep Learning Models in Predicting Arid Vegetation Image Dynamics Least Squares Mean Squared Generative Adversarial Network

Supervision: Dr. Soledad Villar
Report Composition: Meng-Hsuan (Michelle) Wu
Johns Hopkins University – Whiting School of Engineering

May 7, 2023

Abstract

The rising demand of autonomous vehicles, global warming prediction, and early cancer diagnosis have prompted the demand for image prediction research. The objective of this project is to explore the deep learning model's ability to mimic the Rietkerk's PDE model in generating the vegetation image dynamics. Therefore, the CNN-LSTM GAN model is constructed that extracts the key features of each consequtive time-point input images and establishes the temporal relationship across input images. The evaluation of the generated images is conducted by the discriminator, a multi-layers CNN model. This project proposes an innovative objective formula, the Least Squares Mean Squared Loss (LS-MSE) that modified from the traditional least squares loss. This proposed objective formula aims to directly pull the generated image data to the real data by adding the MSE term in the generator loss, calculating the average squared pixel differences between the generated and real images. In addition, the added MSE term may provide additional information for the CNN-LSTM model for the backpropagation stage. The results demonstrate that the CNN-LSTM trained under LS-MSE outperforms either the Wasserstein loss with gradient penalty (WGAN-GP) or the least squares loss (LSGAN). In addition, the LS-MSE GAN has a better performance when compared to the Pix2Pix GAN and the GRU-Unet.

1 Introduction

Image prediction has emerged as a prominent deep learning development in finance, ecology and medical field. The time series image prediction has been widely applied in predicting the stock returns [Jan20]. EfficientNet architecture is proven to be successful in identifying multiple cancer types, including breast cancer, lung cancer and brain cancer [HHL22]. The artificial intelligence is argued to play a significant role in the early cancer diagnosis [HHL22]. The urban land changes in Wuhan city is detected by a fully Atrous convolutional neural network (FACNN) that is composed of Astrous convolution encoder and pixel-based binary change map [ZWJL19]. These recent research has elevated the importance of image prediction as a prominent topic of interest for machine learning scientists.

Inspired by the project conducted by Zhang, where several series of pine tree growth images are examined using the CRNN model [Zha18], and the video classification task conducted by Ferlet [Fer19], a CNN-LSTM GAN model is constructed to mimic the vegetation dynamics images generated via the Rietkerk PDE model. The aim of this capstone project is introduce the construction of CNN-LSTM GAN model and this deep learning model's performances when trained in combination with 3 various loss formula. This project introduces an innovative loss formula, the Least Squares Mean Squared Error loss (LS-MSE GAN), which has the best performance than the other two loss formula, the Wasserstein loss with Gradient Penalty (WGAN-GP), and the Least Squares loss (LSGAN). In addition, the LS-MSE GAN has the best performance when compared to the Pix2Pix GAN and the GRU-Unet.

2 Method

2.1 Data Generation

The dynamics of the vegetation density is explored using the Rietkerk's PDE model, which is composed of a series of differential equations relating to surface water, water absorbed into the soil and the vegetation density [VYH+22]. Using the code developed by the authors, 100 series of vegetation dynamics dataset is produced, and within each series, 100 snapshots are generated, where the default runtime is set to be 200 days. For each generation, the randomly chosen Rietkerk's parameters values are saved in a json file. When the vegetation density is lower than 0.001, the vegetation is declared to be dead and a json file of the mean of vegetaion density, the mean of surface water density and the mean of soil water density is saved in a json file.

Each image is normalized by dividing each pixel value by 255 before proceeding to the training phase. The deep learning model used in this project is motivated by the generative adversarial networks (GANs), where the CNN layers, the LSTM layers and the transposed CNN layers control the generator, and the discriminator is formed by a multi-layers convolutional network. The objective of the generator is to construct realistic images to deceive the discriminator, whereas the motivation of the discriminator is to enhance its ability to distinguish the real images and the generated images. The generator and the discriminator are trained simultaneously and against each other.

2.2 Generator: CNN-LSTM

The generator (G) is a deep learning model composed of an encoder and a decoder. Several model parameters have been tuned. The detailed construction of the model with the best performance is demonstrated in Figure 1. The encoder takes in $t_k \sim t_m$ consecutive time points images as input images, which includes a convolution neural network, followed by a 4 layers long short term memory recurrent neural network. The aim of the encoder is to extract the key features, preserve the spatial dependencies and establish the temporal relationship between each consecutive time-point figure. The decoder takes in the output (z) from the LSTM layers, which is composed of a transposed convolution neural network

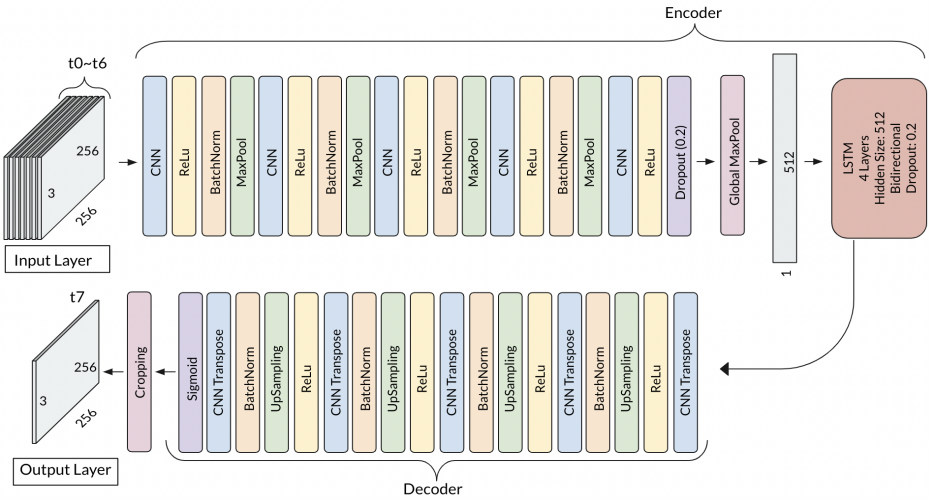

Figure 1: The construction of the generator, CNN-LSTM . The generator includes an encoder and a decoder, where the encoder is composed of CNN layers, followed by a 4 layers bidirectional LSTM, and the decoder is a multi-layers transposed CNN.

that aims to produce the generated image at time-point t_{m+1}. The optimizer used for the generator is the RMSprop, which is an optimizer with adaptive learning rate, where the learning rate is set to be 0.00001. The adapting learning rate originates from dividing the learning rate of a weight by a moving average of the weight's recent gradients, which ensures faster convergence [Bus18].

2.2.1 Encoder

Convolutional layers are proven to have the ability to extract spatial and geometrical information of the image data and are successful in classification task [JMS+18]. Therefore, five convolutional layers are used to extract features of each given images. Each convolutional layer is followed by the activation function, Rectified Linear Unit (ReLU), where $ReLU(x) = max(0, x)$. Although the ReLU is a non-linear function, the Universal Approximation Theorem states that any continuous function can be approximate by a neural network with one hidden layer with a sufficient number of units [Ye20]. Therefore, with the sufficeint number of ReLU activation functions, the model can approximate any linear or nonlinear functions. In addition, ReLU has the advantage of avoiding the diminishing gradient return problem that may be incurred by other activation functions, such as Sigmoid. To increase the generalization ability of the deep learning model, both the Batch Normalization layers and a dropout probability of 20% are added. The Encoder CNN is followed by a global max pooling to flatten out the array before fitting the array into the LSTM layers.

Four bidirectional LSTM layers are implemented to captured the temporal dependencies across the input image data. Long Short-Term Memory (LSTM) is a type of Recurrent Neural Networks that is introduced in 1997 [HS97], which tackles long-term dependencies in sequential data by enforcing the constant error flow and gated units mechanism to avoid the vanishing or exploding gradient problems occurred in the traditional RNN models. The

memory cells are composed of the input gates, the forget gates and the output gates, which regulate the information to be preserved at each stage. LSTM is successful in language translation [SVL14] and precipitation nowcasting [SCW+15]. Therefore, it is implemented following the CNN layers to capture the temporal relationship between input images. To increase the generalization ability, a dropout probability of 20% is enforced.

2.2.2 Decoder

The construction of the decoder is referenced from the generator used for DCGAN [Ink]. The decoder is a mirror version of the convolutional network within the Encoder (Figure 1). Given the image representation z outputted by the LSTM, the decoder aims to output the predicted image at t_{m+1}. Five transposed convolutional layers and upsampling layers are enforced in the Decoder, aiming to upsample the spatial features to the original image dimensions. Although transposed convolution is not the same as deconvolution, the transposed convolution reconstruct the spatial dimentions of the input. The upsampling layers are enforced to generate a higher resolution of the image representation, in which the nearest neighbor algorithm is used. In addition to transposed convolution layers and upsampling layers, the ReLU activation functions and the batch normalization layers are used throughout the decoder. Since the pixel values of the input images are between 0 and 1, a Sigmoid activation function is implemented to ensure the pixel values of the output image is also within the range of 0 and 1. The resulting images will have a dimension of $3 \times 1024 \times 1024$, which will subsequently be center-cropped to the size of $3 \times 256 \times 256$ prior to being fed to the discriminator. It should be noted that an alternative approach of configuring the decoder to generate the output images of $3 \times 256 \times 256$ without further cropping has been evaluated and found to result in inferior performance.

2.3 Discriminator

To evaluate the synthetic images generated from the CNN-LSTM model, a discriminator is created, which is a multi-layers CNN model. The purpose of the discriminator is to judge both the predicted images generated by the generator and the real images from the dataset.

The construction of the discriminator (D) is illustrated in Figure 2. The discriminator is composed of five convolutional layers, four leaky ReLU activation functions, followed by a Sigmoid activation function. The leaky ReLU function is proposed as it solves the dying ReLU problem, which occurs when the neuron is stuck in the negative side of the function and is always outputting 0, while simultaneously speeds up the training process [Liu17]. Instead of outputting 0 for all negative side, the leaky ReLU imposes a negative slope for the negative side. The negative slope used in this discriminator is 0.01. The discriminator outputs a score, where $D(\tilde{x})$ is for the predicted image and $D(x)$ is for the real image; both $D(\tilde{x})$ and $D(x)$ are between 0 and 1 since the last layer of the discriminator is a Sigmoid activation function. The Adam optimizer is used for the discriminator, which has the advantage of adaptive learning rate, where the learning rate is set to be 0.00001.

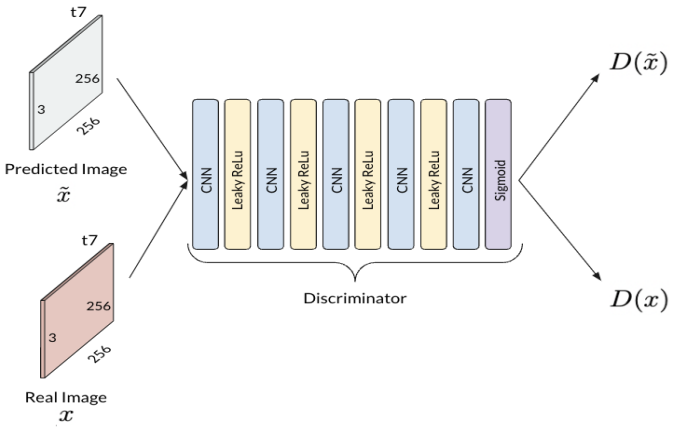

Figure 2: The construction of the discriminator. The discriminator critics both the predicted and the real images, respectively, and outputs a score between 0 and 1 for each image.

2.4 Objective Functions

After the discriminator outputting the scores for both the real and synthetic images, the scores will be inputting into the objective formula for the model to update its parameters during the backpropagation stage. This project analyze the model performance when using three various objective formula: the Wasserstein loss, the traditional least squares loss, and an innovative loss function– the least squares mean squared error loss. The result demonstrates that the least squares mean squared error loss has the best performance, comparing to the other two objective formula.

2.4.1 Wasserstein GAN with Gradient Penalty

The Wasserstein GAN (WGAN) is first introduced by Arjovsky, el., which modified from the traditional GAN loss function by introducing an Earth Mover (EM) Distance that measures the distance between the real data distribution P_r and the distribution of the parameterized density P_g [ACB17]. The authors introduced a critic, which is composed of multi-layers CNN. The critic acts similar as the discriminator in the GAN model as it outputs a score for both the predicted and the real images. The EM distance is argued to be continuous and differentiable everywhere, allowing the critic to be trained until optimality without mode collapse. When WGAN was first introduced, to enforced a Lipschitz constraint, a weight clipping method is proposed; however, this method was also recognized as a terrible way to enforce such constraint as it biases the critic to resort to simpler functions. Therefore, an alternative is introduced by Gulrajani, el., the WGAN with gradient penalty (WGAN-GP), which penalizes the norm of the gradient of the critic with respect to its output [GAA+17]. The WGAN-GP is argued to have a more stable training than the original weight clipping WGAN. Therefore, the WGAN-GP is examined in this project.

Let G represents the generator, the CNN-LSTM model, and D represents the discriminator, a multi-layers CNN. The discriminator will output a score between 0 and 1 for the both the generated image, $\tilde{x}$, where $\tilde{x} = G(z)$, and the real image, x. The objective functions used

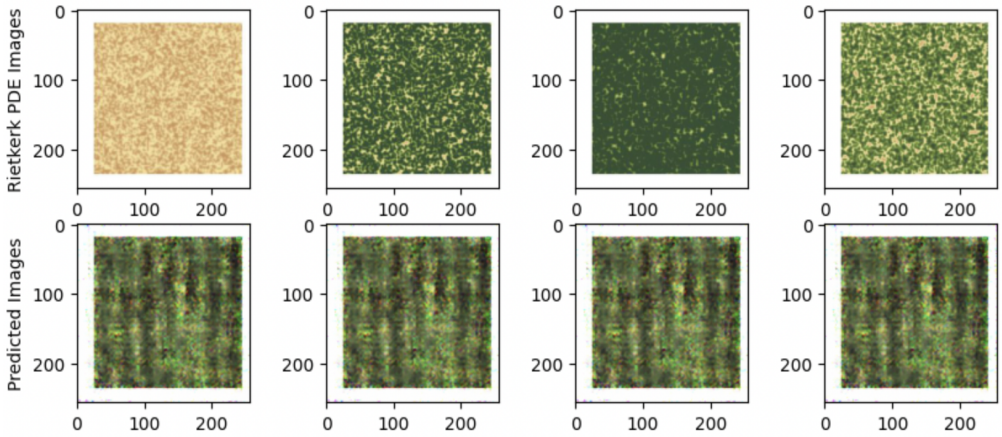

Figure 3: The graph shows the result when inputting t0 and t1 images into the trained CNN-LSTM to generate the t2 images. The CNN-LSTM model is trained with the Wasserstein loss functions with gradient penalty. The top row shows the real images at t2 for 4 random series and the bottom row demonstrates the predicted images at t2 for the CNN-LSTM model trained with Wasserstein loss with gradient penalty.

in WGAN-GP for the generator and discriminator are as followed,

$$\text{Generator: } \min_G V_{\text{WGAN-GP}} = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}[(||\nabla_{\hat{x}}D(\hat{x})||_2 - 1)^2]$$

$$\text{Discriminator: } \min_D V_{\text{WGAN-GP}} = -\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})]$$

$$\text{where } \hat{x} = \epsilon x + (1-\epsilon)\tilde{x}, 0 < \epsilon < 1, \lambda = 10$$

Figure 3 shows the result of the WGAN-GP. Compared to the real images in the top row, the WGAN-GP objective seems to have a significantly inaccurate prediction at t2. One of the contributing factors of the WGAN-GP disappointing performance may be that the evaluation relies heavily on the single score outputted by the discriminator, which might not produce sufficient information to evaluate both the images.

2.4.2 Least Squares GAN

Since the WGAN-GP delivers sub-optimal result, another objective function is analyzed, the Least Squares loss. The Least Squares Generative Adversarial Network (LSGANs) is introduced by Mao, el. in 2017, which is argued to solve the vanishing gradient problem exists in the Sigmoid cross entropy loss function used in regular GANs [MLX+17]. The authors discover that the vanishing gradients in the traditional GANs leads to the difficulties in updating the generator using the generated samples that are in the correct decision boundary but far away from the real data. Thus, the LSGAN is introduced to enhance the abilities of GANs to move the generated data toward the decision boundary by penalizing samples that are on the correct side of the decision boundary but far from the real data. Since LSGAN penalized based on the samples' distance to the decision boundary, it produces more gradients to update the generator.

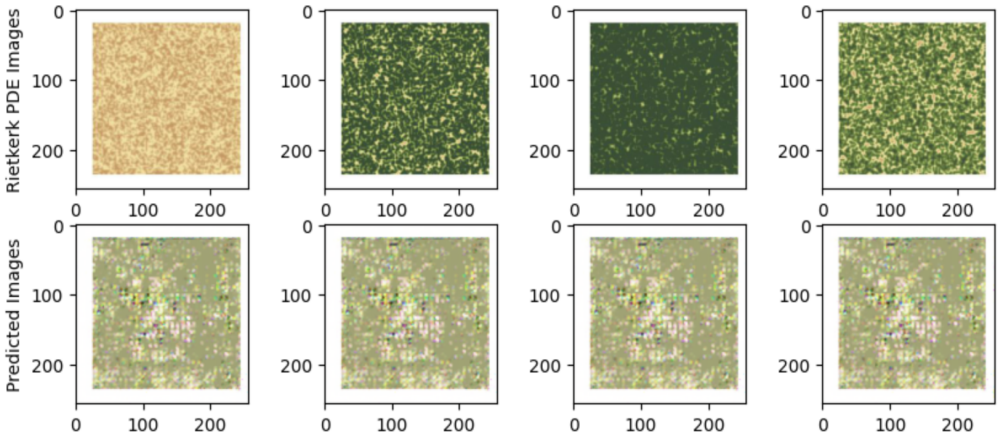

Figure 4: The graph shows the result of inputting t0 and t1 images into the trained CNN-LSTM to generate the t2 images. The CNN-LSTM model is trained with the least squares loss functions. The top row shows the real images at t2 for 4 random series and the bottom row demonstrates the predicted images at t2 for CNN-LSTM trained with the least squares loss.

The last layer of the discriminator is a Sigmoid activation function that outputs a score between 0 and 1. Thus, let 0 be the label of a fake image and 1 be the label of a real image. Let z represents the variables outputted from the LSTM layers, $\tilde{x}$ denotes the predicted images generated by the Decoder, where $\tilde{x} = G(z)$. The loss functions used in LSGAN is as followed,

$$\text{Generator: } \min_G V_{LSGAN}(G) = \frac{1}{2} E_{z \sim p_{z(z)}}[(D(\tilde{x}) - b)^2]$$

$$\text{Discriminator: } \min_D V_{LSGAN}(D) = \frac{1}{2} E_{x \sim p_{data(x)}}[(D(x) - b)^2] + \frac{1}{2} E_{z \sim p_{z(z)}}[(D(\tilde{x}) - a)^2]$$

$$\text{where } b = 1, a = 0, \tilde{x} = G(z)$$

Figure 4 demonstrates the result of of the CNN-LSTM model trained with the traditional least squared loss. Similar to the result of WGAN-GP, the bad performance may be that the evaluation heavily relies on the single score outputted by the discriminator, which might not capture sufficient information to evaluate the generated image data.

2.4.3 Least Squares Mean Squared Error GAN

In their paper of LSGANs, Mao, el. [MLX+17] suggests the future work to analyze the possibility of pulling the generated data directly to the real data instead of pulling the generated data to close to the decision boundary. Therefore, this project proposed a new loss function, the Least Squares and Mean Squared Generative Adversarial Networks (LS-MSE GAN). The LS-MSE loss modifies from the traditional least squares loss by introducing the means squared error term in the generator objective function, such that the pixel value differences between the generated and real images can be minimized. The LS-MSE loss functions are as followed,

$$\text{Generator: } \min_G V_{LSMSEGAN}(G) = \frac{1}{2} E_{z \sim p_{z(z)}}[(D(\tilde{x}) - b)^2] + \frac{1}{2} MSE_{pixel}(x, \tilde{x})$$

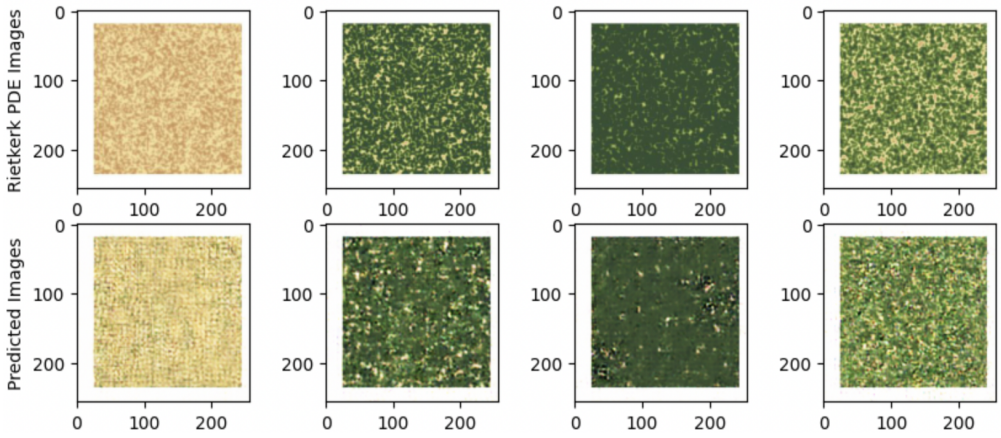

Figure 5: The graph shows the result of inputting t0 and t1 images into the trained CNN-LSTM to generate the t2 images. The CNN-LSTM model is trained with the least squares means squared error loss functions. The top row shows the real images at t2 for 4 random series and the bottom row demonstrates the predicted images at t2 for CNN-LSTM with least squares means squares loss.

$$\text{Discriminator: } \min_D V_{LSMSEGAN}(D) = \frac{1}{2}E_{x\sim p_{data(x)}}[(D(x)-b)^2] + \frac{1}{2}E_{z\sim p_{z(z)}}[(D(\tilde{x})-a)^2]$$
$$\text{where } b=1, a=0, \tilde{x}=G(z)$$

Under this objective, a successful generator can produce realistic images $\tilde{x}$, such that it tricks the discriminator into believing the predicted images are realistic by giving these images the label of 1, while minimizes each pixel values differences between the real x and generated images $\tilde{x}$, leading to the generator loss function to be 0. Similarly, a successful discriminator can distinguish the predicted and the real images by giving the x the label of 1 and $\tilde{x}$ the label of 0, also leading to the discriminator loss function to be 0.

Figure 5 demonstrates the result comparison between of the Rietkerk PDE images and the CNN-LSTM images at t2. The result shows a significant improvement of the LS-MSE objective function from using either the loss functions of WGAN-GP (Figure 3) or LSGAN (Figure 4). This improvement may originate from the increase information as the mean squared error is added to the generator loss function.

2.5 Training

After comparing the model's performance for generating t2 images under three different objective functions in Table 1, the LS-MSE GAN is selected. The 100 series in the dataset is splitted as followed: 80 series for training, 10 series for validation and 10 series for testing. To generate images at t_{m+1}, the model takes in 7 consequtive time-point images at $t_{m-6} \sim t_m$. Due to the limited computing power of Google Colab, instead of training through all remaining 93 time-steps iteratively, this project resorts to an efficient method by only training randomly selected 35 time-steps among the 93 time-steps.

The training is conducted by first randomly selecting 35 time-steps between $t_7 \sim t_{99}$ as the real output images' time-points of interest. After the output images' time-points are selected, the input images can be identified as being the images at the 7 consecutive time-steps prior to the output images' time-step. For instance, let t_k be one of the randomly selected output time-steps, the input images are identified as the 7 consecutive images between time-steps t_{k-7} and t_{k-1}. The training is completed after all of the 35 time-steps are trained.

3 Comparison Result

The metrics chosen for the evaluation are the mean squared error (MSE), the structural similarity index measure (SSIM), and the peak signal-to-noise ratio (PSNR). MSE evaluates the average squared difference between the pixel values of the real and generated images. The closer the MSE value to 0, the better the quality of the images. Although MSE is sensitive to outliers, it has the limitation as various perturbed images could result to the same MSE [Ryb18]. Therefore, two other metrics are included. The SSIM metric assesses the perceptual quality between the distorted real images and the generated images by referencing the properties of the human visual system. The SSIM metric evaluates the images using 3 key features: luminance, contrast and structure [Dat20]. The SSIM of 2 given images will be between -1 and 1 where -1 indicates that the 2 images are drastically different and 1 indicates that the 2 images are similar. The PSNR is derived from MSE, which evaluates the maximum possible intensity of the generated images and of the corresponding real images after distortion [NBS22]. The higher the PSNR denotes to better simulation. Through the evaluation via MSE, SSIM and PSNR, the most suitable objective formula and deep learning model can be identified.

3.0.1 Comparison of Objective Functions

Table 1 demonstrates the evaluation of the CNN-LSTM GAN model performance with three different objectives introduced in Section 2.4. The comparison is conducted by having the same 10 series of testing images at t0 and t1 act as input images, which pass though the same CNN-LSTM GAN model trained with 3 different objectives after 1000 epochs. The metrics evaluate the t2 images generated by the trained model when t0 and t1 images are given. The result shows that the CNN-LSTM GAN model with the Least Squares MSE objective functions has the lowest MSE, highest SSIM and highest PSNR, indicating that this objective formula produces a better result than the other two objective functions.

3.0.2 Comparison of Deep Learning Models

Table 2 illustrates the models' performance comparison of the LS-MSE GAN (constructed by Meng-Hsuan Wu), the Pix2Pix GAN (constructed by Atul Zacharias) and the GRU-Unet (constructed by Harry Lin). The evaluation scores are calculated by Harry Lin. Each deep learning model evaluates the model's performance on the same 10 testing series, within each series, there are 100 time-steps of images. However, the number of initial images put into each deep learning model is different: LS-MSE GAN takes in 7 images, ie. $t0 \sim t6$, Pix2Pix

Metrics	WGAN-GP	LSGAN	LS-MSE GAN
MSE	3520.77	4108.61	2304.13
SSIM	0.24	0.28	0.34
PSNR	12.75	11.80	15.06

Table 1: The table shows the CNN-LSTM model's performance when trained with three different objective functions: Wasserstein loss with gradient penalty, Least Squares loss and Least Squares Mean Squared loss. The metrics evaluate the model's performance when generating 10 series of images at t2 after inputting t0 and t1 images.

takes in 1 image, ie. $t0$, and GRU-Unet takes in 5 images ie. $t0 \sim t4$.

Comparing via the MSE, SSIM and PSNR values, the LS-MSE GAN produces the most stable quality of images across all time-steps. In addition, the SSIM and PSNR values shows that the LS-MSE GAN has the best performance comparing to the other two deep learning models. However, as time increases, the increase of MSE indicates that the error accumulates as the predicted images deviate from the real images; whereas, the SSIM and PSNR are both stable across all time-steps.

In addition, it is found that the increase of input images enhances the model's performance as the LSTM benefits from the increase information. Table 1 illustrates that when t0 and t1 images are used to generate t2 images, the MSE is 2304.13, the SSIM is 0.34 and the PSNR is 15.06. All these metrics improve when 7 consequtive images are given to produce a single images at t10 where the MSE decreases to 1387.16, SSIM increases to 0.52 and PSNR increases to 19.42 (Table 2). The increase input images allows the LSTM component within the LS-MSE GAN model to extract more sophisticated temporal relationship across images at different time-steps, and subsequently produce a more accurate generated images.

Models	Metrics	t10	t24	t49	t99	Overall
LS-MSE GAN	MSE	1387.16	3070.15	3458.52	3542.09	3208.27
	SSIM	0.52	0.59	0.59	0.59	0.59
	PSNR	19.42	18.65	18.26	18.14	18.54
Pix2Pix GAN	MSE	5426.73	4986.33	6208.69	7001.73	949.62
	SSIM	0.33	0.51	0.55	0.4	0.14
	PSNR	10.19	10.95	7.77	3.84	3.68
GRU-Unet	MSE	6264.02	9833.77	8875.79	7641.71	2125.08
	SSIM	0.24	0.6	0.54	0.54	0.13
	PSNR	9.64	3.58	3.49	4.42	1.11

Table 2: The table illustrates the deep learning models comparison: the LS-MSE GAN, Pix2Pix GAN and GRU-Unet. The metrics evaluate the performance of the 3 deep learning models when generating all time-points images after few initial images are given.

4 Conclusion

To replicate the Rietkerk PDE model that simulates the vegetation dynamic images, a CNN-LSTM GAN model is constructed, where the generator includes an encoder and a decoder components. The encoder is composed of 5 convolutional layers to extract the key features of each input images, and a 4 layers bidirectional LSTM to establish the temporal realtionship across the input images. The decoder is composed of 5 transposed convolutional layers that aims to construct the predicted images. To evaluate the quality of the generated images, a discriminator is established, which is a multi-layers CNN model, that outputs a score between 0 and 1. However, the single score outputted by the discriminator seems to struggle to effectively evaluate the quality of the generated images when using the two traditional objective formula: the Wasserstein loss with Gradient Penalty (WGAN-GP), and the Least Squares Loss (LS-GAN).

Therefore, this project proposes an innovative objective formula, the Least Squares Mean Squared Error Loss (LS-MSE GAN) that includes a MSE term in the generator objective that calculate the average squared difference between the real and generated images' pixel difference. This proposed objective formula aims to pull the generated data directly to the real data and generate more information for the deep learning model during the back-propagation process. The comparision results demonstrate that the CNN-LSTM model trained under the LS-MSE objective formula has the best performance when comparing to using the other two traditional loss formula. Moreover, comparing to the Pix2Pix GAN and the GRU-Unet, the LS-MSE GAN has the most satisfactory performance.

Acknowledgement

I want to thank Dr. Soledad Villar for providing her research materials, vegetation image dynamics Colab code, her insides, knowledge and support for me and my teammates in every bi-weekly meetings. In addition, I want to thank my teammates, Atul Zacharias and Harry Lin for sharing their Pix2Pix GAN and GRU-Unet models' construction and providing their knowledge and support in every weekly meeting.

References

[ACB17] Martin Arjovsky, Soumith Chintala, and Leon Bottou. Wasserstein gan. *arXiv:1701.07875*, 3, 2017.

[Bus18] Vitaly Bushaev. Understanding rmsprop – faster neural network learning, 2018. Accessed on April 3, 2023.

[Dat20] Pranjal Datta. All about structural similarity index (ssim): Theory + code in pytorch, 2020. Accessed on April 3, 2023.

[Fer19] Patrice Ferlet. Training a neural network with an image sequence — example with a video as input, 2019. Accessed on April 3, 2023.

[GAA+17] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. *arXiv:1704.00028*, 2017.

[HHL22] Benjamin Hunter, Sumeet Hindocha, and Richard W. Lee. The role of artificial intelligence in early cancer diagnosis. *Cancers(Basel)*, 14(6):1524, 2022.

[HS97] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):97–111, 1997.

[Ink] Nathan Inkawhich. Dcgan tutorial. Accessed on April 3, 2023.

[Jan20] Stefan Jansen. *Machine Learning for Trading*. Packt Publishing, 2020.

[JMS+18] Manjunath Jogin, Mohana, Madhulika M S, Divya G D, Meghana R K, and Apoorva S. Feature extraction using convolution neural networks (cnn) and deep learning. *3rd IEEE International Conference on Recent Trends in Electronics, Information Communication Technology (RTEICT)*, 2018.

[Liu17] Danqing Liu. A practical guide to relu, 2017. Accessed on April 3, 2023.

[MLX+17] Xudong Mao, Qing Liy, Haoran Xiez, Raymond Y.K. Laux, Zhen Wang, and Stephen Paul Smolleyk. Least squares generative adversarial networks. *arXiv:1611.04076v3*, 3, 2017.

[NBS22] Tereza Nečasová, Ninon Burgos, and David Svoboda. *Biomedical Image Synthesis and Simulation*. The MICCAI Society book Series. Academic Press, 2022.

[Ryb18] Oleg Rybkin. The reasonable ineffectiveness of pixel metrics for future prediction (and what to do about it), 2018. Accessed on April 3, 2023.

[SCW+15] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. *NIPS*, 1:802 – 810, 2015.

[SVL14] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *arXiv: 1409.3215*, 2014.

[VYH+22] Soledad Villar, Weichi Yao, David W. Hogg, Ben Blum-Smith, and Bianca Dumitrascu. Dimensionless machine learning:imposing exact units equivariance. *arXiv:2204.00887*, 2022.

[Ye20] Andre Ye. Finally, an intuitive explanation of why relus works, 2020. Accessed on April 3, 2023.

[Zha18] Zao Zhang. Image series prediction via convolutional recurrent neural networks with limited training data. *Binghamton University*, 2018.

[ZWJL19] Chi Zhang, Shiqing Wei, Shunping Ji, and Meng Lu. Detecting large-scale urban land cover changes from very high resolution remote sensing images using cnn-based classification. *IJGI*, 8(4):189, 2019.