Tropical Cyclone Rainfall Prediction via Infrared Imagery: A Comparative Study Using Autoencoders and Conditional Generative Adversarial Networks

Guilherme Santos[1] João Monteiro[1] Pedro Pinto[1]

1University of Aveiro, Portugal

Abstract—**This study addresses the challenge of estimating Passive Microwave Rainfall (PMR) from cost-effective Infrared (IR) satellite imagery, a crucial task for precipitation forecasting given PMR data sparsity. We systematically compare Autoencoders and Conditional Generative Adversarial Networks (cGANs), specifically Pix2Pix, on the TCIRRP dataset. Our findings show cGANs consistently achieve superior reconstruction accuracy and generalization. Autoencoder analysis reveals nuanced behavior: higher errors on sparse rainfall images (due to MSE sensitivity) and increased error spread on high-entropy images, indicating difficulty with complexity. Autoencoder optimization identified a 32-dimensional latent space, Linear activation, and a batch size of 64 (for efficiency) as optimal. While cGANs are more robust for this complex translation, comprehensive Autoencoder optimization provides critical insights into its limitations and potential, guiding future research into advanced generative models.**

Index Terms—**Precipitation, Autoencoders, GANs, Satellite Imagery, Forecasting.**

I. Introduction

Accurate precipitation forecasting is essential for various fields, from meteorology to disaster management. While Passive Microwave Rainfall (PMR) data offers high precision, its coverage is limited and costly. Infrared (IR) satellite imagery, however, provides extensive, affordable coverage but lacks direct quantitative rainfall correlation. Bridging this gap, learning to map IR imagery to detailed PMR data, is a critical challenge in remote sensing. Recent advancements in deep learning, particularly image-to-image translation, offer promising solutions. Prior work using models like Pix2Pix (a paired cGAN) and CycleGAN (an unpaired GAN) on datasets such as TCIRRP [1] has shown potential but also notable issues. Pix2Pix often blurs fine details or creates artifacts, while CycleGAN can distort rainfall intensities due to its unpaired nature. Some applications of Pix2Pix have also struggled with convergence [2]. This study directly compares two deep learning architectures: Autoencoders and Conditional Generative Adversarial Networks (cGANs), specifically the Pix2Pix framework. Our goal is to evaluate their effectiveness in learning the complex IR-to-PMR mapping, focusing on reconstruction fidelity, generalization, and overall suitability. We aim for a solution that accurately predicts rainfall while maintaining realistic spatial structures.

II. Dataset Description and Analysis

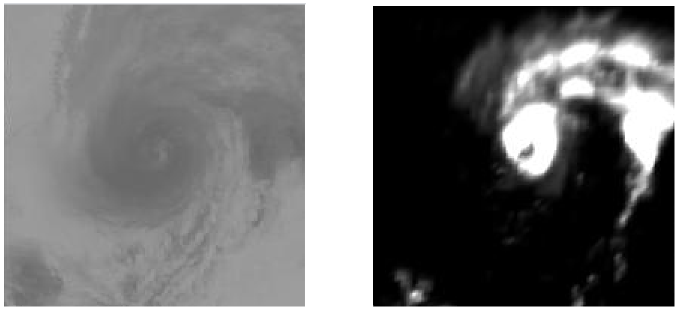
A. Dataset Description

The Tropical Cyclone IR-to-Rainfall Prediction (TCIRRP) dataset [1] is a benchmark dataset developed to facilitate the prediction of PMR Fig.1b from IR Fig.1a satellite imagery, specifically in the context of tropical cyclones (TCs). The TCIRRP dataset comprises over 70000 paired samples of IR and corresponding PMR images. Each sample represents a snapshot of a tropical cyclone, capturing the cloud-top brightness temperature from IR sensors and the associated rainfall distribution from passive microwave observations. The images are centered on the cyclone's core and standardized to a resolution of 201×201 pixels. The IR data in the dataset are sourced from geostationary satellites, offering high temporal resolution, while the PMR data are obtained from polar-orbiting satellites equipped with passive microwave sensors. The TCIRRP dataset is organized into training and testing subsets, facilitating the development and evaluation of deep learning models for IR-to-PMR translation. Its extensive size and paired structure make it particularly suitable for supervised learning approaches, such as conditional generative adversarial networks (cGANs) [3].

B. Statistical Analysis

In this section, we present three different analyses performed on each of the ground-truth images (PMR) in the dataset. Specifically, we (1) generate an 8×8 "average-heatmap" to identify where, on average, predicted rainfall is concentrated; (2) compute the Shannon entropy of each map to characterize its information content; and (3) measure the fraction of non-black pixels in each map to understand how much of each image contains rainfall.

1) Concentration Heatmap: Each PMR ground-truth image is first converted to a single-channel intensity array (0–255 scale), then cropped to a multiple of 8 in each dimension and downsampled to 8 × 8 by taking the mean over each non-



(a) Input: IR (b) Output: PMR

Fig. 1: TCIRRP dataset sample [1]

overlapping block. Denoting the downsampled array for image i as

$$\mathbf{R}^{(i)} = \left[R_{m,n}^{(i)}\right]_{m,n=1}^{8}, \tag{1}$$

we compute the element-wise mean across all N images:

$$H_{m,n} = \frac{1}{N}\sum_{i=1}^{N} R_{m,n}^{(i)}, \quad m,n = 1,\dots,8. \tag{2}$$

The resulting matrix constitutes the *average heatmap*, where each entry $H_{m,n}$ represents the mean predicted-rainfall intensity in cell (m,n) over the entire dataset. Figure 2 depicts this 8×8 heatmap in grayscale.

As shown in Fig. 2, the four central cells exhibit the highest mean intensity, whereas the perimeter cells are substantially darker. This "peak-in-the-center" pattern indicates that, on average, PMR predictions are strongly concentrated near the image center.

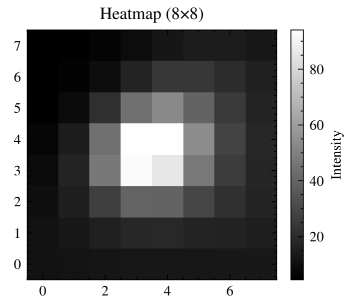


Fig. 2: 8x8 Heatmap

2) Entropy Analysis: For each image map $\mathbf{I}^{(i)}$, $n_k^{(i)}$ is defined as the number of pixels with intensity $k \in \{0,1,\dots,255\}$. The empirical probability mass function is given by:

$$p_k^{(i)} = \frac{n_k^{(i)}}{201\times 201}, \qquad \sum_{k=0}^{255} p_k^{(i)} = 1 \tag{3}$$

and the Shannon entropy of image i is:

$$H^{(i)} = -\sum_{k=0}^{255} p_k^{(i)} \log_2\left(p_k^{(i)}\right) \quad \text{[bits]}. \tag{4}$$

If a map is nearly all zeros (i.e., almost no rainfall), then $H^{(i)} \approx 0$. Conversely, if pixel intensities are uniformly distributed across many levels (i.e., a highly "textured" rainfall field), then

$$H^{(i)} \approx \log_2(256) = 8 \text{ bits}. \tag{5}$$

We computed $H^{(i)}$ for all N images and plotted a histogram with 30 bins spanning 0–7.5 bits (see Fig. 3). Approximately 700 maps have $H^{(i)} < 1$ bit, corresponding to nearly uniform with small complexity images. Similarly, very few maps exceed 6 bits, indicating very high variability across many intensity levels. The entropy distribution is roughly symmetric around the median, with a mild right skew.

These findings imply that low-entropy maps (i.e., nearly empty or containing a single rainfall cluster) are relatively rare but important to sample adequately, as they can lead to false-positive biases if underrepresented.

3) Rain-coverage fraction: We define the non-black fraction $w^{(i)}$ of each image as

$$w^{(i)} = \frac{\left|\{(x,y) : I^{(i)}(x,y) \neq 0\}\right|}{201\times 201}, \tag{6}$$

i.e., the proportion of pixels indicating any positive rainfall. Consequently, $1 - w^{(i)}$ is the fraction of pixels with zero rainfall. Figure 4 shows a histogram of $\{w^{(i)}\}$ using 60 bins over $[0, 0.967]$, where 0.967 is the maximum observed (indicating nearly full coverage by rainfall). Key statistics from Fig. 4 are:

- **Zero-Rain Maps:** Exactly 615 images have $w^{(i)} = 0.00$, meaning no positive-rainfall pixels.
- **Modal Range:** The highest bin occurs at $w \approx 0.35$, with approximately 2 200 maps in that interval. Thus, most images contain rainfall on roughly 35 % of the grid.
- **Right Tail:** Very few images have $w^{(i)} > 0.95$, indicating that there are only a small number of cases where almost the entire image predicts rain.

III. Data Preprocessing

High-quality and consistent input data are paramount for the performance and generalization of deep learning models. The preprocessing pipeline was designed to address the inherent challenges of diverse and often noisy meteorological satellite images. This ensures optimal data characteristics for training deep neural networks, enhancing model generalization and robustness to real-world variations. The detailed steps are as follows:

A. General Preprocessing

1) Input Splitting: Each input sample in the TCIRRP dataset [1] is provided as a single, horizontally concatenated image, containing both the IR image and PMR image. Specifically, the right half of the concatenated image represents the input (IR) image, and the left half corresponds to the target output (PMR), both in grayscale (a single channel). The images are loaded and subsequently split along the horizontal axis to separate the input-output pairs.

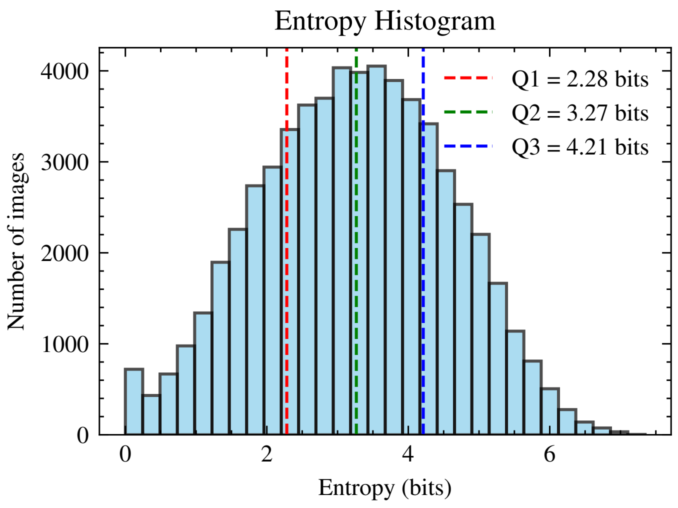


Fig. 3: Entropy Histogram

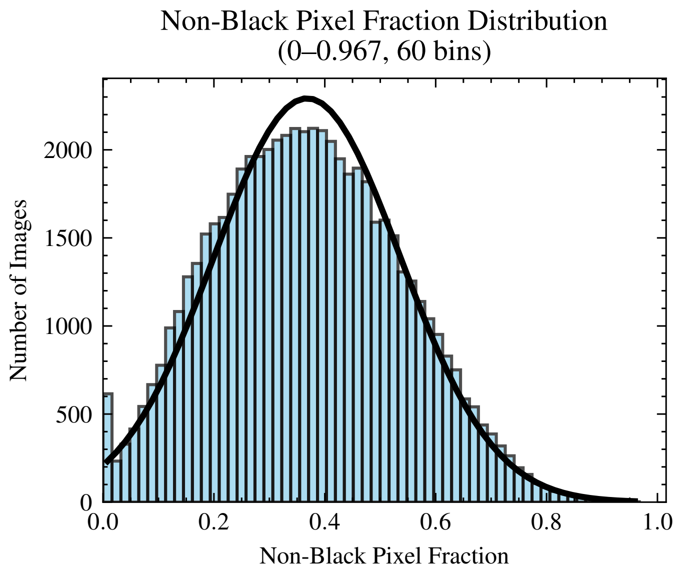


Fig. 4: Coverage Distribution

2) Resizing: Images were resized to a fixed resolution of x using bilinear interpolation, which yields smoother results and preserves spatial continuity better than nearest-neighbor methods. To mitigate edge effects and interpolation artifacts, `reflective padding` (symmetric padding) was applied prior to resizing. This technique reflects border pixels, preventing artificial discontinuities often seen with `zero-padding` or `constant-padding`.

3) Per-Image Normalization: To reduce intrinsic variability within each sample and optimize the stability of the model training process, each image was subjected to z-score normalization. This process was calculated independently for both input and target images, following the formulation:

$$x' = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

Here, x is the original pixel value, μ is the image's mean pixel value [0, 255], σ is its standard deviation, and ϵ (typically 10^{-8}) is a small constant to prevent division by zero. This standardization ensures each sample has zero mean and unit variance, crucial for accelerating model convergence and preventing gradient issues by reducing global brightness/contrast shifts.

B. Advanced Preprocessing (Training Only)

1) Data Augmentation: Data augmentation techniques were exclusively applied during the training phase to address significant overfitting, a primary challenge encountered in this project, and to enhance the model's generalization capability to unseen data. It is imperative that both input and target images are augmented identically to preserve spatial alignment and semantic correspondence between image pairs. The following transformations were applied:

- **Random Rotation (0-360°):** This operation simulates arbitrary orientations of meteorological structures, like satellites. The application of a full rotation (0-360°) is justified by the absence of absolute geolocation information in the dataset, implying that all spatial orientations are equally probable and relevant for the model.
- **Random Zoom ($\pm$10%):** Introduces subtle scale variability, allowing the model to learn features at different sizes and distances, thereby increasing its invariance to minor scale variations.
- **Random Contrast ($\pm$5%) and Brightness ($\pm$5%):** These variations simulate fluctuations in image acquisition conditions and sensor characteristics. By exposing the model to these variations, its robustness is significantly improved against differences in illumination and sensor calibration in real-world scenarios.

All data augmentation operations involving resizing or rotation utilized `reflective padding` to prevent the introduction of artifacts at the image borders. This choice is particularly critical in cloud formations, where the preservation of physical continuity is essential for data integrity and for the model's learning of significant patterns.

IV. Methodologies

A. Deep Learning Algorithms

In this study, we explore two deep learning architectures for the IR-to-PMR translation estimations: Autoencoders and Conditional Generative Adversarial Networks (cGANs). Each model offers unique advantages and challenges in the context of image-to-image translation tasks.

1) Autoencoders: Autoencoders are a type of unsupervised neural network designed to learn efficient, compressed representations of input data. They achieve this by first encoding the data into a lower-dimensional latent space and then decoding it back to its original form. This process forces the network to identify and capture the most significant features of the input [4]. The architecture of an autoencoder consists of two primary components:

- **Encoder:** This part of the network takes the raw input data and transforms it into a compressed, latent representation. Think of it as summarizing the essential information.
- **Decoder:** The decoder then takes this latent representation and attempts to reconstruct the original data. The goal here is to reproduce the input as accurately as possible from its compressed form.

Autoencoders can learn the underlying characteristics within IR images that are strongly correlated with rainfall patterns. By doing so, they can then reconstruct corresponding PMR images, effectively bridging the gap between these two data types. Their inherent simplicity and robust stability make them an possible good choice for tasks where minimizing the reconstruction error is paramount and computacional speed is mandatory.

2) Conditional Generative Adversarial Networks (cGANs): cGANs extend the traditional GAN framework by conditioning both the generator and discriminator on additional information, in this case, the IR images [3]. The architecture involves:

- **Generator**: Attempts to produce realistic PMR images conditioned on input IR images.

- **Discriminator**: Evaluates the authenticity of the generated PMR images, also conditioned on the same IR inputs.

This conditional setup enables cGANs to generate outputs that are not only realistic but also contextually aligned with the input data. The adversarial training encourages the generator to produce high-fidelity images that can deceive the discriminator, leading to more accurate and detailed PMR estimations.

3) Comparative Analysis: The choice between autoencoders and conditional Generative Adversarial Networks (cGANs) for image-to-image translation, such as IR-to-PMR, hinges on their distinct capabilities and limitations:

- **Autoencoders** offer simpler architectures and more stable training. They excel at capturing global patterns by minimizing reconstruction error, making them suitable when computational resources are limited or initial robustness is key. However, they are prone to overfitting and often struggle to generate fine-grained details, requiring very large and diverse datasets to produce high-fidelity outputs. This makes them less ideal when precise textural or nuanced feature generation is critical.
- **Conditional GANs**, conversely, are superior for synthesizing high-quality, perceptually realistic images with intricate details. Their adversarial training framework allows the generator to learn complex data distributions, producing outputs that are often indistinguishable from real data [3]. While cGANs are powerful for detail generation, their training is significantly more complex, computationally intensive, and susceptible to instability.

Given the paired nature of the TCIRRP dataset, cGANs are generally better suited for generating highly detailed PMR predictions due to their ability to leverage conditional information for precise, context-aware transformations.

4) Autoencoder Architecture: Our autoencoder model employs a deep, symmetric encoder-decoder architecture influenced by the U-Net paradigm [5], designed for efficient IR-to-PMR image estimation, very similar to the Generator presentend in the Pix2Pix Architecture at Figure 5. While not incorporating explicit skip connections as in a canonical U-Net, its structure prioritizes hierarchical feature learning and spatial reconstruction.

- **The Encoder** path acts as a contracting network, progressively extracting multi-scale features through three successive convolutional blocks. Each block consists of a Conv2D layer (kernel size 3, stride 2), followed by BatchNormalization and ReLU activation. The filter counts increase with depth (32, 64, 128), progressively downsampling the spatial dimensions while enriching feature representations. The final convolutional output is flattened and mapped to a 32-dimensional latent space via a Dense layer with ReLU activation, forming the compact image representation. The bottleneck dimensions are determined by the input image size and the downsampling factor.
- **The Decoder** path mirrors the encoder, functioning as an expansive network that reconstructs the image from the latent representation. It begins by projecting the 32-dimensional latent vector back to the bottleneck's spatial and channel dimensions using a Dense layer, BatchNormalization, and ReLU activation, followed by a Reshape operation. Subsequently, two Conv2DTranspose layers (kernel size 3, stride 2, with 64 and 32 filters, respectively, each followed by BatchNormalization and ReLU) perform learned upsampling, progressively restoring spatial resolution. The final Conv2DTranspose layer outputs a single-channel image at the original input dimensions, with its activation function linear chosen to match the target PMR image's pixel value range.

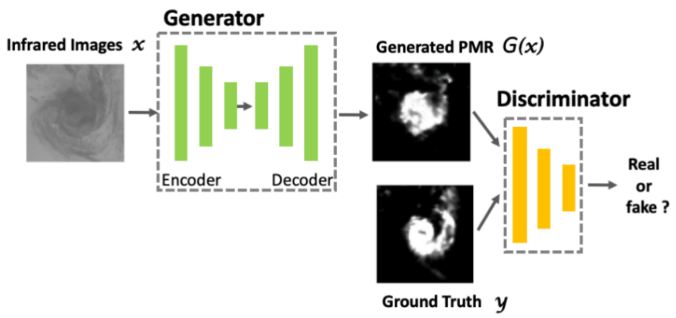


Fig. 5: Pix2Pix - GAN Architecture [2]

This architecture is optimized for learning a compact, meaningful representation of IR images and accurately translating them into PMR estimates by minimizing reconstruction error. The use of BatchNormalization throughout the network enhances training stability and accelerates convergence.

5) Generative Adversarial Network (GAN) Architecture: In exploring GAN architectures for the IR-to-PMR translation task, we considered both Pix2Pix and CycleGAN models. Pix2Pix, Figure 5, is a conditional GAN that requires paired training data, where each input image has a corresponding target image. It learns a mapping from input to output images and is well-suited for tasks where such paired datasets are available [3]. CycleGAN, on the other hand, is designed for unpaired image-to-image translation tasks. It employs a cycle consistency loss to ensure that the translated image can be converted back to the original image, even without paired training data [6]. While CycleGAN offers flexibility in scenarios lacking paired datasets, it may not achieve the same level of accuracy as Pix2Pix when paired data is available. Given that the TCIRRP dataset provides paired IR and PMR images, we opted for the Pix2Pix model [7].

B. Loss Function

In training our models for the IR-to-PMR translation task, we evaluated several loss functions to determine the most suitable for our regression problem. For a given input infrared image x_i, our models predict a corresponding PMR image $\hat{y}^i$, which is then compared against the actual (ground truth) PMR image y_i. The primary loss functions considered were Mean Squared Error (MSE), Mean Absolute Error (MAE), and Cross-Entropy.

1) Mean Squared Error (MSE): MSE is a widely used loss function for regression tasks. It calculates the average of the squares of the differences between predicted and actual values:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{7}$$

This loss function penalizes larger errors more severely due to the squaring term, which can be beneficial when large errors are particularly undesirable. Additionally, MSE is differentiable and convex, facilitating efficient optimization during training.

2) Mean Absolute Error (MAE): MAE computes the average of the absolute differences between predicted and actual values:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{8}$$

Unlike MSE, MAE treats all errors equally, making it more robust to outliers. However, its gradient is not continuous at zero, which can pose challenges for certain optimization algorithms.

3) Cross-Entropy: Cross-Entropy is primarily used for classification tasks, measuring the dissimilarity between two probability distributions. It is defined as:

$$\text{Cross-Entropy} = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) \tag{9}$$

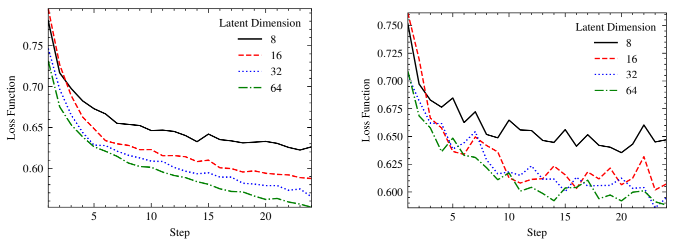
This loss function is not suitable for regression problems where the outputs are continuous values, as it assumes the target variables represent probability distributions.

4) Rationale for Choosing MSE: Given that our task involves predicting continuous rainfall values from infrared imagery, it is inherently a regression problem. MSE is particularly well-suited for such tasks, as it emphasizes larger errors and provides a smooth, differentiable surface for optimization. While MAE offers robustness to outliers, our dataset does not exhibit significant outlier issues, making MSE a more appropriate choice. Cross-Entropy, being tailored for classification, is not applicable in this context.

C. Parameters Optimization

1) Autoencoder Optimization: The effectiveness and generalization of these models are highly dependent on meticulous hyperparameter tuning. This section details the systematic optimization of key autoencoder parameters: latent dimension, batch size, and the last layer's activation function, utilizing the adaptive Adam optimizer for learning rate management.

a) Latent Dimension Optimization: The latent dimension serves as a critical bottleneck, representing the compressed, lower-dimensional encoding of input data [4]. An appropriately sized latent space forces the autoencoder to learn salient features by filtering noise, leading to efficient data representation. Empirical analysis, illustrated in Figure 6 (a) and (b), revealed a classic bias-variance trade-off. While training loss consistently decreased with increasing latent dimensions (e.g., 64 achieved the lowest), validation loss showed diminishing returns beyond a certain point (e.g., 16 or 32). This indicated that excessive capacity can lead to memorizing training-specific noise rather than improving generalization to unseen



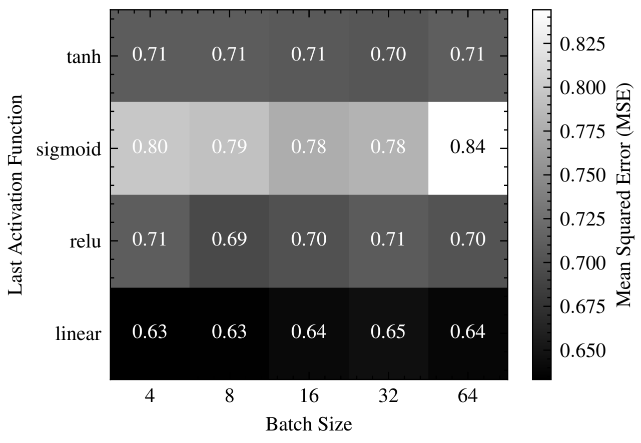
(a) Training MSE vs. Step (b) Validation MSE vs. Step

Fig. 6: Latent Dimension Optimization: Impact on Training and Validation MSE

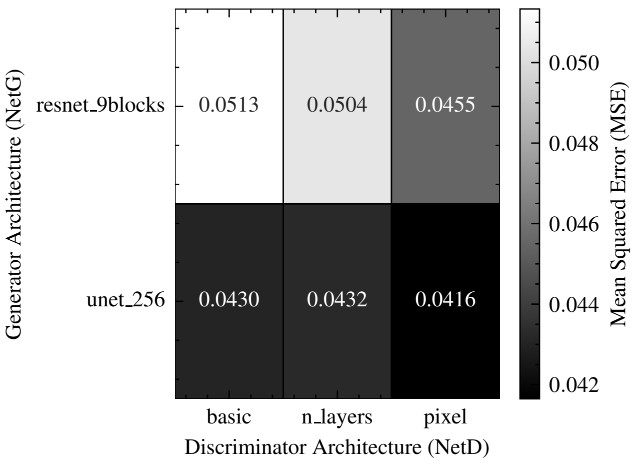
data. Based on this analysis, a latent dimension of 32 was selected. This choice provides a pragmatic balance, offering sufficient representational capacity for strong reconstruction performance without excessive model complexity or risk of overfitting to noise.

b) Batch Size and Last Layer Activation Function Optimization: Batch size, which defines the number of samples processed per weight update, influences gradient stability and convergence. Smaller batches introduce more gradient noise, potentially aiding escape from shallow local minima, but can slow convergence. Conversely, larger batches offer more stable gradient estimates and faster convergence per epoch, though they may converge to sharper minima that generalize less effectively [8]. Activation functions are crucial for introducing non-linearity in neural networks [8], enabling the learning of complex relationships. For the final layer, their importance is paramount as they directly define the model's output type and scale, which must align with the nature of the problem being solved. For instance, in regression tasks, the final activation function maps the network's internal representation to a continuous output, determining its range and distribution. Functions considered include ReLU $f(x) = max(0, x)$, Tanh $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$, and Sigmoid $f(x) = 1/(1 + e^{-x})$.

Empirical analysis, visualized in Figure 7a, displayed the Mean Squared Error for various combinations of batch sizes (4, 8, 16, 32, 64) and last layer activation functions (Linear, ReLU, Tanh, Sigmoid). Darker shades indicate lower MSE. The Linear activation function consistently achieved the lowest MSE values (e.g., 0.63 with batch sizes 4 and 8), significantly outperforming other functions. While a batch size of 4 yielded the absolute lowest MSE (0.63) for Linear, a batch size of 64 also showed comparable performance (0.64) and was chosen for its superior computational efficiency, offering a favorable trade-off for faster training time by processing more samples simultaneously. This highlights that for this regression task, a direct linear mapping in the output layer, combined with an optimized batch size, provides the best balance of accuracy and training speed. The Linear activation's superior performance is expected for continuous output, avoiding saturation issues common with Sigmoid or Tanh when target values extend beyond their fixed ranges. This suggests the network's preceding layers effectively transformed the input for linear mapping to PMR targets.



(a) Autoencoder



(b) Pix2Pix

Fig. 7: Average Mean Squared Error (MSE) in Parameters Optimization.

c) Learning Rate Optimization with Adam: The Adam (Adaptive Moment Estimation) optimizer was employed to manage learning rates [9]. Adam adaptively adjusts the learning rate for each network parameter, promoting faster and more stable convergence, especially with sparse gradients or noisy data. This adaptive nature significantly reduces the need for extensive manual tuning of a global learning rate, contributing to overall training efficiency and reproducibility.

d) Conclusion on Autoencoder Optimization: Systematic autoencoder hyperparameter optimization provided crucial insights for the IR-to-PMR translation task. Analysis of the latent dimension revealed an optimal balance between training fidelity and generalization at 32 dimensions. Concurrently, optimization of batch size and last layer activation empirically favored the Linear activation function with a batch size of 64. This configuration achieved a low MSE (0.64) with high computational efficiency, suggesting a direct linear mapping aligns best with the target rainfall data characteristics. While the autoencoder may not be the most performant model compared to more complex architectures for this task, its optimization demonstrates the feasibility of achieving reasonable and efficient performance. The Adam optimizer further simplified training by adaptively managing learning rates, enhancing convergence stability. These optimized hyperparameters significantly contribute to the autoencoder's ability to accurately reconstruct PMR images from IR inputs. Future work could explore advanced regularization, deeper latent space analysis, alternative autoencoder architectures, and more comprehensive hyperparameter search strategies.

2) GAN Optimization: The GAN hyperparameter tuning was carried out in two phases. In Phase 1, we employed *random search* to identify the optimal batch size, learning rate, and normalization scheme since it efficiently identifies promising regions of the hyperparameter space, especially when only a subset of parameters has a significant impact, and because exhaustively exploring all permutations would be prohibitively expensive in GPU time given the large number of parameters [8], [10]. In Phase 2, we fixed those three parameters at their best values and performed a *grid search* over netG and netD choices, as we did with Autoencoder, visualizing results with a heatmap.

a) Phase 1: To avoid biasing results toward any specific network architecture, we first defined ranges for every hyperparameter, including **batch size** ($\{1,\ 2,\ 4\}$), **learning rate** ($[1\times10^{-4}, 3\times10^{-4}]$), **normalization** ({batch, none, instance}), **netG** ({ResNet$_{\text{9blocks}}$, U-Net$_{256}$}), and **netD** ({basic, pixel, n_layers}). We then conducted 15 trials, each consisting of 20 epochs. In every trial, one combination of these five hyperparameters was chosen at random, the GAN was trained for 20 epochs, and the validation MSE was recorded. After completing all trials, we computed the average MSE for each parameter value across all trials and identified those settings yielding the lowest average MSE. Table I summarizes the Phase 1 search space and shows the best combination at the final column.

TABLE I: Phase 1 Random Search: Ranges and Best Values

Hyperparameter	**Range / Sampling**	**Best Value**
Batch size	$\{1,\ 2,\ 4\}$	4
Learning rate	$[1\times10^{-4},\ 3\times10^{-4}]$	2.98×10^{-4}
Normalization	{batch, none, instance}	batch

b) Phase 2: With batch size, learning rate, and normalization fixed with the best values of the previous phase, we exhaustively evaluated:

- **netG**: {ResNet$_{\text{9blocks}}$, U-Net$_{256}$}
- **netD**: {basic, pixel, n_layers}

We trained each combination for 20 epochs and recorded the MSE. Because network architecture has the largest impact on GAN performance and the total number of (netG, netD) combinations is small, grid search is the most appropriate method to be used [8], [11]. Figure 7b shows a heatmap of mean MSE against all the combinations of networks.

c) Conclusion on GAN Optimization: Systematic GAN hyperparameter optimization produced a configuration that balances training stability with high-frequency detail preservation for the IR-to-PMR task. In Phase 1, random search under a 15-trial budget identified a batch size of 4, learning rate 2.98×10^{-4}, and batch normalization as the best combination, minimizing validation MSE. Fixing these values, Phase 2 grid search over generator and discriminator architectures selected U-Net$_{256}$ (with `resize_and_crop` to 256×256) and a pixel-level ("PatchGAN") discriminator, which together preserved multi-scale contextual information and patch-level realism most effectively. These choices align with established GAN design principles, moderate batch sizes for stable updates, carefully tuned learning rates, batch normalization for consistent gradients, U-Net skip connections for detail retention, and PatchGAN for local fidelity, culminating in the lowest observed translation error. Future work could explore automated architecture search and alternative adversarial objectives to further improve fidelity [3], [11].

V. Results

A. Test Set Description

The full dataset, as provided in the original Kaggle source, contains a total of over 70000 samples whom are split into smaller chunks for faster evaluation [1]. For the general comparison between models, we used the entire 6000-sample test set. To better understand the generalization behavior of the Autoencoder specifically, we also created a custom division of the dataset into 8 subsets. This was done along two independent criteria:

- **Black-and-white ratio**: Four subsets were created based on the proportion of non-black pixels in each image. These subsets correspond to different levels of rainfall coverage, ranging from images with no rainfall (all black pixels) to those with near-complete coverage. The division into these subsets resulted in variant sizes, as it accounted for the natural distribution of non-black pixels.
- **Entropy**: The other four subsets were formed by grouping images according to their Shannon entropy, which quantifies the information content or complexity of the rainfall patterns. This categorization helps to isolate the Autoencoder's performance across different levels of visual complexity. To ensure a consistent comparative analysis, each of these entropy-based subsets contained the same number of images.

These subdivisions were used exclusively for a deeper analysis of the Autoencoder's behavior under different types of image characteristics. Each subset was divided with the same number of images.

B. General Comparison: Autoencoder vs GAN

For each input image, the models produced a reconstruction, and the Mean Squared Error (MSE) was computed between the reconstructed and the ground truth output. To assess the quality and stability of the reconstructions, we analyzed this distribution of values using two visualizations per model:

- A **boxplot**, to summarize the median, quartiles, and outliers in the MSE distribution.
- A **histogram** to examine the distribution's shape and central tendency.

As shown in Figure 8a, the Autoencoder exhibits higher MSE values overall, with a noticeable spread and numerous outliers. This indicates unstable and poor-quality reconstructions, likely due to overfitting or insufficient generalization.

In contrast, the GAN (Figure 8b) demonstrates significantly lower MSE values and its histogram follows a Gaussian-like shape with few extremes, suggesting more consistent and generalizable reconstructions.

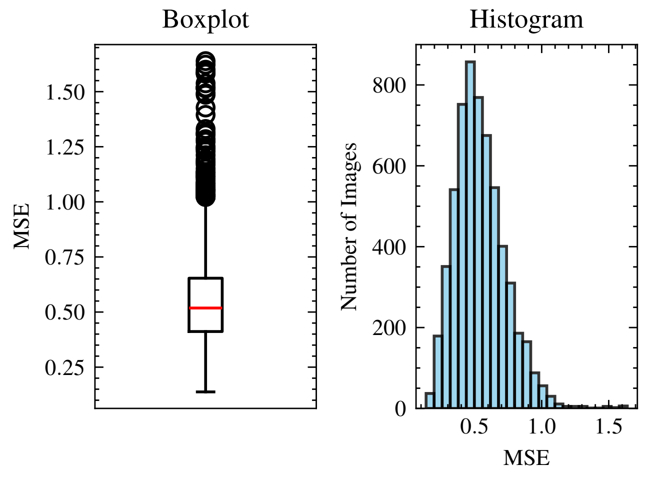
Preliminary Conclusion: The GAN consistently achieves lower reconstruction error across the entire test dataset, making it the more robust choice in this setting. The Autoencoder struggles to generalize, as was expected, given that autoencoders are designed for purposes where the underlying image structure does not change as dramatically, which prompts a deeper investigation in the next section.

C. Autoencoder Generalization Analysis

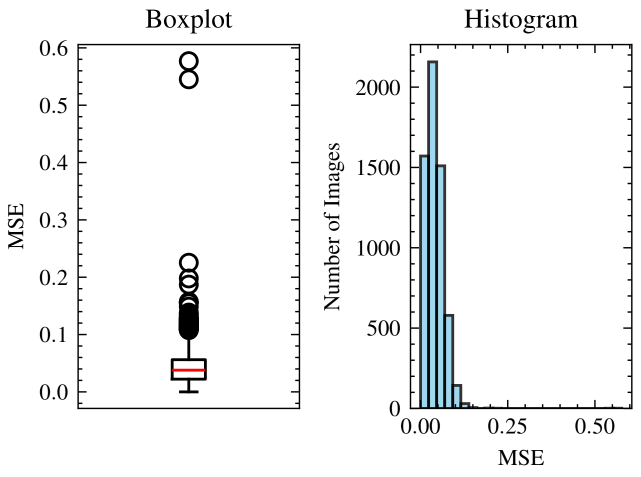
Given the Autoencoder's poor overall performance on the full test set, we analyzed its behavior on the eight previously defined subsets to evaluate its generalization capabilities. We trained and a specific subset. The distribution of reconstruction errors (MSE) across these subsets is presented in Figure 9. This figure offers a detailed view of the Autoencoder's generalization performance under varying image characteristics. We observe that the Autoencoder struggles consistently across all groups, with only slight improvements in subsets with simpler images. In particular:

- **Entropy-based Subsets:** Median MSE is slightly better or stable in mid-to-high entropy ranges ("2.28 to 3.27" and "3.27 to 4.21 entropy bits") compared to the lowest entropy subset ("0 to 2.28 entropy bits"). However, higher entropy subsets, especially "4.21 to inf entropy bits," show significantly wider error spreads and more high-MSE outliers. This indicates that while the model might occasionally perform well on complex images, its consistency and reliability degrade substantially with increasing information content.
- **Black-and-white Ratio Subsets:** The Autoencoder exhibits higher median MSE for predominantly black images ("0 to 0.05 not black pixels," though this subset has only 11 samples). Performance generally improves (lower median MSE) as the proportion of non-black pixels increases ("0.4 to 1 of not black pixels"). This counter-intuitive trend suggests the model struggles more with sparse rainfall (mostly black) than with denser patterns. This is likely due to MSE heavily penalizing deviations from zero in sparse regions; a missed prediction in few critical areas disproportionately increases MSE, while denser rainfall offers more opportunities for accurate predictions, leading to lower overall MSE despite some inaccuracies.

Overall, the analysis of these subsets reinforces the preliminary conclusion that the Autoencoder struggles to generalize effectively to the full spectrum of image complexities present in the TCIRRP dataset. While it can achieve lower errors on very simple, sparse images, its performance deteriorates significantly with increasing entropy and rainfall coverage,



(a) Autoencoder: MSE Boxplot and Histogram



(b) GAN: MSE Boxplot and Histogram

Fig. 8: Distribution of Mean Squared Error (MSE) values over 6000 test samples for the Autoencoder and the GAN.

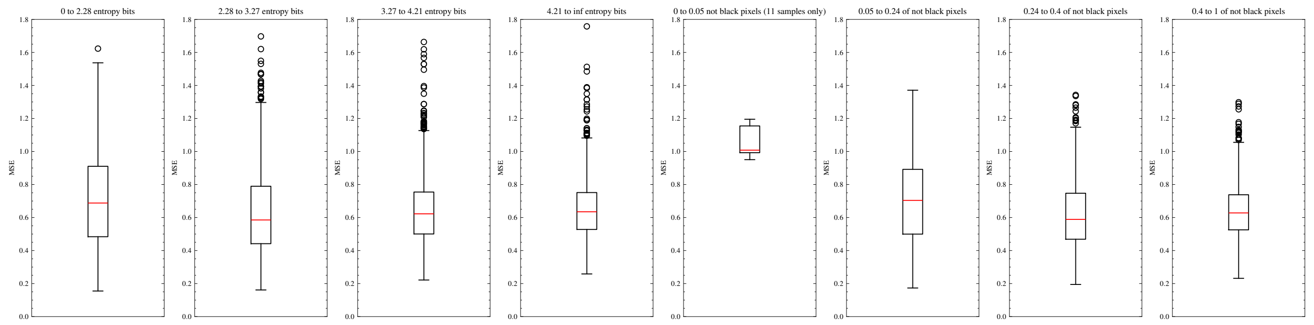


Fig. 9: MSE distribution across 8 Autoencoder models, each trained and tested on a different subset.

highlighting its limitations in capturing the nuanced patterns required for accurate PMR prediction.

VI. Conclusion

This study compared Autoencoders and Conditional Generative Adversarial Networks (cGANs) for estimating tropical cyclone rainfall from IR imagery. Our analysis clearly showed that cGANs were superior, providing higher accuracy and better generalization for this complex image translation task. While autoencoders proved more stable in training, they struggled with generalization, especially showing higher errors on images with sparse rainfall and increased inconsistency with more complex image patterns. Despite efforts to optimize their performance, cGANs consistently demonstrated a stronger ability to handle the intricate mappings required for accurate rainfall estimation. In summary, deep learning is a validated approach for meteorological image translation, with cGANs emerging as the more effective architecture for generating detailed and accurate rainfall predictions. Future work will explore advanced generative models, like diffusion models, for further enhancements.

References

[1] K. B. Dharun, "TCIRRP dataset for thermal to color infrared image translation," Kaggle, https://www.kaggle.com/datasets/kbdharun/tcirrp-dataset/data, 2020.

[2] F. Meng, T. Song, and D. Xu, "Tcr-gan: Predicting tropical cyclone passive microwave rainfall using infrared imagery via generative adversarial networks," *ArXiv*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246035212

[3] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *CVPR*, 2017, pp. 1125–1134.

[4] D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders," *arXiv preprint arXiv:2003.05991*, 2020. [Online]. Available: https://arxiv.org/abs/2003.05991

[5] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2015.

[6] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *ICCV*. IEEE, 2017, pp. 2242–2251.

[7] ——, "Pytorch-CycleGAN-and-Pix2Pix repository," GitHub, https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix, 2017.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[9] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017. [Online]. Available: https://arxiv.org/abs/1412.6980

[10] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," in *Journal of Machine Learning Research*, vol. 13, 2012, pp. 281–305.

[11] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *ICLR*, 2016. [Online]. Available: https://arxiv.org/abs/1511.06434