RGB-TO-INFRARED IMAGE TRANSLATION FOR MILITARY AIRCRAFT USING CYCLEGAN - PROJECT FINAL REPORT

Bengisu YÜCEL
Department of Computer Engineering
Hacettepe University
Ankara, Turkiye
bengisuyucel@hacettepe.edu.tr

The codebase of this project link is https://github.com/bengisuycl/CMP719_TERM_PROJECT
Bengisu Yucel. Cyclegan-infragan wandb results. https://api.wandb.ai/links/bengisuycl1-hacettepe-university/fbrj9ikt.

1 INTRODUCTION

Infrared (IR) imaging plays a critical role in modern defense systems by enabling target detection, tracking, and situational awareness under poor visibility conditions such as nighttime, fog, smoke, and camouflage environments. Unlike conventional RGB imaging, which captures reflected visible light, infrared sensors detect emitted thermal radiation, allowing them to operate effectively in challenging scenarios where visual cues are degraded or absent. Consequently, IR-based systems are widely deployed in surveillance, reconnaissance, and guidance applications, particularly for airborne military platforms such as fighter jets, drones, and helicopters. However, a significant bottleneck in the development and deployment of AI-driven IR vision systems is the lack of large-scale,labeled, and high-quality infrared datasets specific to military applications. Most existing IR datasets focus on civilian use cases—autonomous driving (e.g., FLIR ADAS), pedestrian detection, or agricultural monitoring—and are therefore illsuited for training models in defense contexts. Moreover, collecting and annotating military-grade IR imagery is constrained by confidentiality, operational security, and limited access to thermal sensors and flight data. In this context, synthetic data generation emerges as a viable solution to augment scarce IR datasets. Particularly, Generative Adversarial Networks (GANs) have demonstrated remarkable success in generating photorealistic images across modalities. Among them, CycleGAN has proven especially powerful in unpaired image-to-image translation tasks, where aligned training pairs (e.g., RGB-IR) are unavailable. CycleGAN leverages a dual generator-discriminator setup and a cycle consistency loss to learn bidirectional mappings between two domains using unpaired data. This property makes it ideal for cross-modal translation tasks such as RGB-to-IR conversion, where paired training images are infeasible to obtain. The objective of this project is to utilize CycleGAN for the translation of visiblespectrum (RGB) images of military aircraft into the infrared (IR) domain, without requiring pixel-wise correspondences between domains. The ultimate goal is to enable the generation of synthetic IR imagery that retains the semantic structure of the input RGB images while accurately capturing the stylistic and thermal characteristics of real IR imagery. Through this work, aiming to contribute a reproducible pipeline for synthetic infrared data generation in military vision systems, potentially reducing reliance on expensive hardware and limited datasets while enabling rapid prototyping and simulation based training for AI models.

2 MODEL DESCRIPTION

The model is taken baseline of CycleGAN architecture. CycleGAN is a type of Generative Adversarial Network (GAN) designed for unpaired image-to-image translation e.g. Horse2zebra or apple2orange. CycleGAN is a type of Generative Adversarial Network (GAN) designed for unpaired image-to-image translation. CycleGAN comprises two generators and two discriminators

to enable bidirectional, unpaired image-to-image translation. Generators use a ResNet-based encoder–transformer–decoder structure adapted from Johnson et al.**?** and Zhu et al.Zhu et al. (2017).

On the generator architecture, the Initial convolution: 7×7, stride 1 , Downsampling: Two 3×3 convolutions stride 2 (d128, d256), Transformer: 6–9 residual blocks (R128), For 128×128 inputs: 6 blocks, For $\geq$ 256×256: 9 blocks, Upsampling: Two fractional-strided convolutions (u128, u64)and the Final convolution: 7×7 to produce 3-channel RGB output All convolutions use instance normalization and ReLU, plus reflection padding to reduce artifact. And the Discriminator Architecture, CycleGAN employs 70×70 PatchGAN discriminators: convolutional networks that classify each patch as real/fake C64 $\rightarrow$ C128 $\rightarrow$ C256 $\rightarrow$ C512 $\rightarrow$ output Typical architecture (Ck = 4x4 Conv, stride 2, LeakyReLU, no BatchNorm initially).

InfraGAN Özkanoğlu & Ozer (2022) introduces, loss functions and structural consistency held on differently with Adversarial GAN loss, L1 pixel loss for realism in IR domain and Structural similarity (SSIM) loss to preserve edges and structure between visible and IR images. In InfraGAN, while exact patch size details aren't fully specified, it's implied that a similar multi-scale PatchGAN or standard discriminator is used, augmented specifically with SSIM-focused training in the generator stage. The figure 1 shows the my architecture.

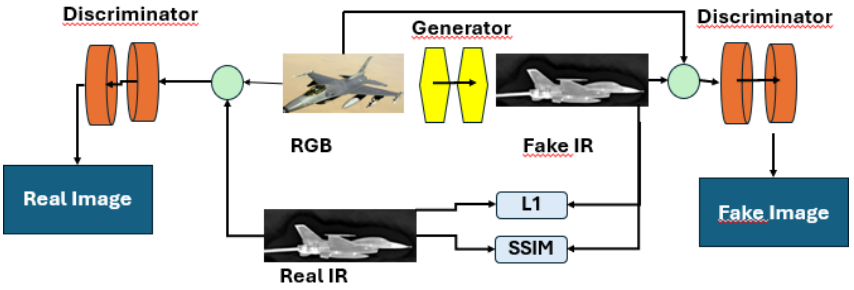

Figure 1: The rgb2ir GAN Model

Network Components:

- Generator (U-Net based):
  - Downsampling layers are Convolution + InstanceNorm + ReLU
  - Bottleneck as Residual Blocks
  - Upsampling layers: Transposed Convolutions + Skip Connections
  - Final activation: Tanh
- Discriminator:
  - Classifies each image patch (70×70) as real or fake
  - Uses Spectral Normalization for training stability
- Activation Functions:
  - LeakyReLU in discriminators
  - ReLU in generators
  - Sigmoid for final discriminator output

2.1 Dataset Preparation

- RGB Images: 35410 images collected from Kaggle (Military Aircraft Detection) and video frames extracted from military.com footage.The dataset is used here mil.
- IR Images: 270 images from Roboflow IR dataset.
- Images resized to 256x256, normalized to [-1, 1].
- No alignment or pairing between RGB and IR images.

2.2 GENERATOR ARCHITECTURE

The generator network is designed based on a classical encoder–residual–decoder architecture, commonly used in image-to-image translation tasks such as CycleGAN. The model takes an input RGB image and outputs a transformed RGB image, with pixel values normalized in the range of [-1, 1] using a Tanh activation function.

- **Encoder:**The encoder consists of a sequence of convolutional layers that progressively downsample the input image while increasing the number of feature channels. It begins with a convolution layer using a large 7×7 kernel to capture broad spatial context, followed by Instance Normalization and ReLU activations. Downsampling is performed with two successive 3×3 convolution layers with stride 2, resulting in feature maps with 256 channels and reduced spatial dimensions.
- **Residual Blocks:** The core of the generator is composed of nine residual blocks. Each block contains two convolutional layers with 3×3 kernels, each followed by Instance Normalization and a ReLU activation in the first layer. The residual connections help preserve spatial information and enable effective feature transformation without significant degradation of the original input features.
- **Decoder:** The decoder reconstructs the output image from the encoded representation by progressively upsampling the feature maps through transposed convolutional layers. Each upsampling step is followed by Instance Normalization and ReLU activation. Finally, a convolutional layer with a 7×7 kernel reduces the feature map to three channels, corresponding to the RGB output image, followed by a Tanh activation to scale the output pixel values.
- **Normalization:** Instance Normalization is employed throughout the network, as it has proven effective in style transfer and domain adaptation tasks by normalizing feature statistics per-instance, which stabilizes training and improves output quality.

This architecture balances the depth and capacity needed for complex image translation tasks with mechanisms to maintain spatial details through residual connections, making it well-suited for domain conversion problems such as visible-to-infrared image translation.

2.3 LOSS FUNCTIONS

- **Adversarial Loss:** Encourages the generator to produce realistic images that can fool the discriminator. The discriminator tries to correctly classify real images from the target domain and fake images generated by the generator. Formally, the discriminator loss is: $L_D = -E_y[logD(y)] - E_x[log(1 - D(G(x)))]$
  and the generator loss is: $L_G = -E_x[log_D(G(x))]$
  where x is a source domain image, y is a target image, G(x) is the generated image, and D(.) outputs the probability of the image being real.
- **Cycle Consistency Loss:** ensures that if an image is translated from the source domain to the target domain and then back again, the resulting image should be close to the original. Given two generators G: $X \longrightarrow Y$ and F: $Y \longrightarrow X$, this loss is defined as:
  $L_{cycle} = E_x[||F(G(x)) - x||_1 + E_y[||G(F(y)) - y||_1]]$
  This L1 loss encourages the model to preserve important content and structure through the cycle.
- **Identity Loss:** It is used to prevent generators from altering images that are already in the target domain. This helps preserve colors and styles. It is defined as: $L_{identity} = E_x[||F(G(x)) - x||_1 + E_y[||G(F(y)) - y||_1]]$
  where applying the generator to images from the target domain should yield the same images.
- **Structural Similarity Index (SSIM) Loss:** It measures the structural similarity between the original and generated images, reflecting perceptual quality. The SSIM index ranges from 0 (completely different) to 1 (identical). The loss is expressed as:
  $L_{SSIM} = 1 - SSIM(x, \hat{x})$
  Minimizing this loss increases the structural similarity and helps preserve image details that L1 or L2 losses might miss. the same images.

- **Histogram Loss:**It compares the intensity distributions of the generated image and the target image by measuring the difference between their histograms. This helps preserve global brightness and contrast, which are important for realistic image translation. It is defined as:
  $L_{hist} = \sum_{i=1}^{N}(H_1(\hat{y}) - H_i(y))^2$
  Where Hi is the value of the ith bin in the normalized histogram of the image.

2.4 Training Configuration

- Batch size = 1 due to GPU limitations.
- Training conducted on Google Colab Pro with CUDA stability fixes.
- SSIM was used for validation at every epoch.

3 Experimental results and comparisons

The source code with histogram loss is available at gan (a), while results without histogram loss can be found at gan (c). The main GAN model implementation is shared at gan (b).

The code and training is handled in Google Colab Pro notebook,with GPU Google Colab Pro T4 GPU.The setup has composed of two generators and two discriminators, trained jointly using multiple loss functions designed to ensure adversarial learning, cycle consistency, identity preservation, and structural similarity. By integrating these losses, the model not only learns to generate realistic translations but also preserves fine details and structural information essential for the target domain. The following subsections describe each component and loss function in detail. In my generator Architecture Diagram has; The Detailed Architecture is shown in figure 2

According to the model,
Downsampling : 3x3 Convs with stride 2 (from 64 → 128 → 256 channels)
9 Residual Blocks: each with 256 channels (For your case, you've used 9 like the full CycleGAN model for 256×256 images)
Upsampling: ConvTranspose2D with stride 2 (from 256 → 128 → 64 channels)
Final Conv2D: 7×7 kernel back to 3-channel output
Activation: Tanh for pixel value normalization to [1, 1]
InstanceNorm everywhere (better for style transfer tasks than BatchNorm)

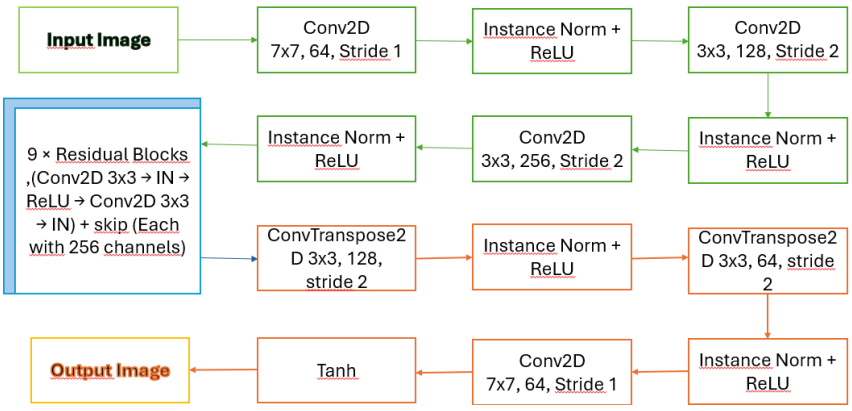

Figure 2: Caption

On my project, I add Add SSIM loss to generator training, adjust generator loss weighting hyperparameters, Use little paired images , keep identity loss based on experiment.SSIM, L1, PSNR used as evaluation metrics.

- Input Image: The network starts with an image, typically with 3 color channels (e.g., RGB).

- Initial Convolution Block: A convolutional layer increases the number of channels from 3 to 64, using a larger kernel (7x7) to capture broader features. Padding ensures the spatial dimensions remain the same. Instance Normalization and ReLU activation follow.
- Downsampling Block: Two sequential convolutional layers with a stride of 2. Each of these layers effectively halves the spatial dimensions (width and height) of the feature maps while doubling the number of channels (from 64 to 128, then 128 to 256).
- Instance Normalization and ReLU follow each convolutional layer. At the end of this block, the feature maps are significantly smaller spatially but have a high number of channels (256).
- Residual Blocks: A series of number of resudial blocks (defaulting to 9) Residual Blocks are stacked. Each Residual Block maintains the number of channels (256) and the spatial dimensions. Their purpose is to further process the extracted features and add depth to the network without losing information due to simple sequential layers.
- Upsampling Block: Two sequential ConvTranspose2d (also known as deconvolution or fractional-strided convolution) layers are used. These layers effectively double the spatial dimensions of the feature maps in each step, gradually restoring the image to its original size. The number of channels is halved in each upsampling step (from 256 to 128, then 128 to 64), and instance normalization and ReLU follow each transpose convolution.
- Output Layer: A final convolutional layer with a kernel size of 7 reduces the channels back to 3 (for an RGB image). A Tanh() activation function is applied. Tanh squashes the output values to a range of [-1, 1], which is common for generated image pixels (as images are often normalized to this range).
- Output Image: The generated image, which ideally has the same spatial dimensions and channel count as the input image. The model builds upon the CycleGAN architecture to enable image-to-image translation between the visible and infrared domains without requiring paired datasets.

3.1 Experimental Results

As shown in Figure 3, the infrared (IR) image is synthetically generated from the input RGB image using a CycleGAN-based framework. The unpaired image-to-image translation approach enables cross-domain conversion without Histogram Loss.

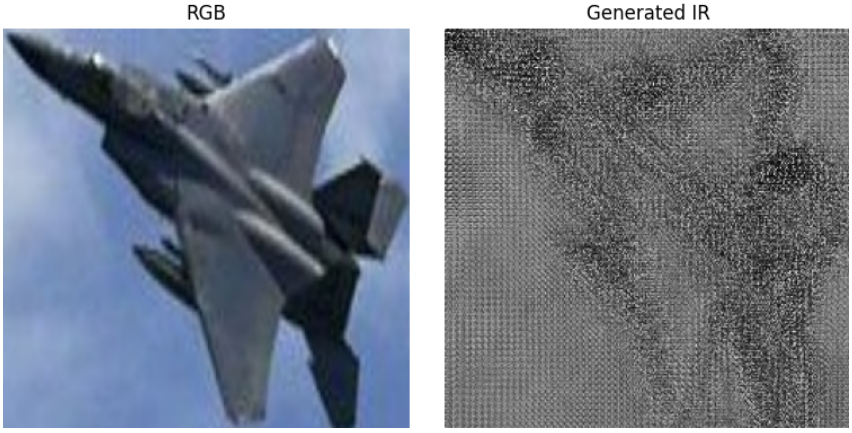

Figure 3: RGB and Generated IR Image without Histogram Loss

As shown in Figure 4, the infrared (IR) image is synthetically generated from the input RGB image using a CycleGAN-based framework. The unpaired image-to-image translation approach enables cross-domain conversion with Histogram Loss.

As shown in Figure 5, Figure 6 and Figure 7, adding histogram loss improved perceptual realism by enforcing intensity distribution consistency between generated and real IR images. As ex-

Figure 4: RGB to IR Conversion with Histogram Loss

pected, generator losses increased initially, but over iterations, the generator learned to match not only pixel-wise structures (via SSIM and L1) but also histogram profiles, resulting in enhanced quality IR translations. Generally stays between 0.05 and 0.35 throughout, with brief peaks up to 0.42 around batches where the generator was performing unusually well or poorly.Mostly very low (around 0.001–0.008), showing the discriminator easily classifies real vs fake in the RGB domain, likely because RGB images are high-quality and structured.

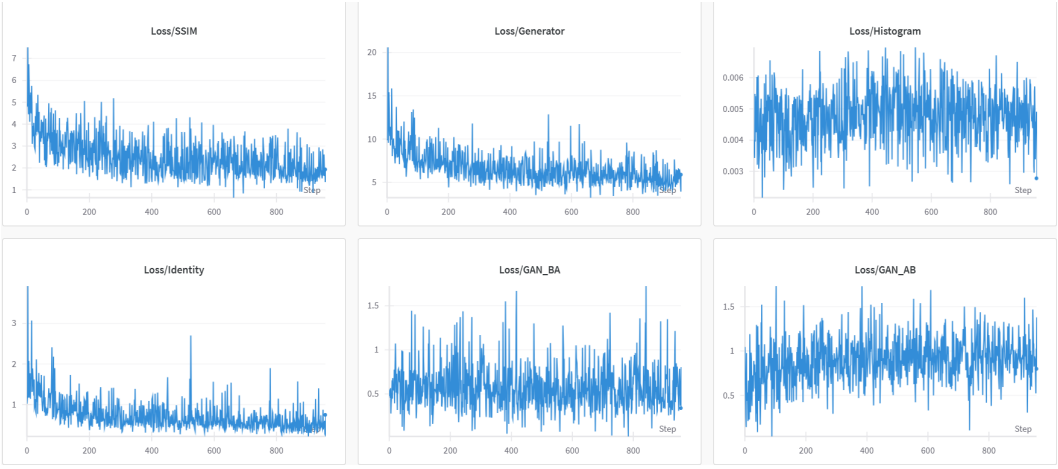

Figure 5: The Output Graphs w/ histogram Loss added

As shown in Figure 6, Discriminator Losses (D_A and D_B) were relatively low and stable. D_B loss was very low (mostly under 0.01) — indicating the discriminator quickly adapted to distinguish RGB images, likely because the generator's initial IR translations still had RGB-like features. D_A loss varied more (0.05–0.40) since generating realistic IR images is a harder, ill-posed problem without exact paired ground-truths.

As shown in Figure 6 and 7 taken from Wandb, without histogram loss the experiments shows us Generator Loss ($Loss_G$) fluctuated typically between $\approx$ 2.5 and 5.5 across batches, which is expected due to adversarial dynamics. You can view in, You can view in WandB Yücel. `https://api.wandb.ai/links/bengisuycl1-hacettepe-university/fbrj9ikt`. In this link, website that refences is given, the output visualization epoch graph, loss graphs of discriminator and generators in both comparison of with histogram losses or without histogram losses are reported. Also step by step output figures (generated IR, RGB, original RGB) are also shown.

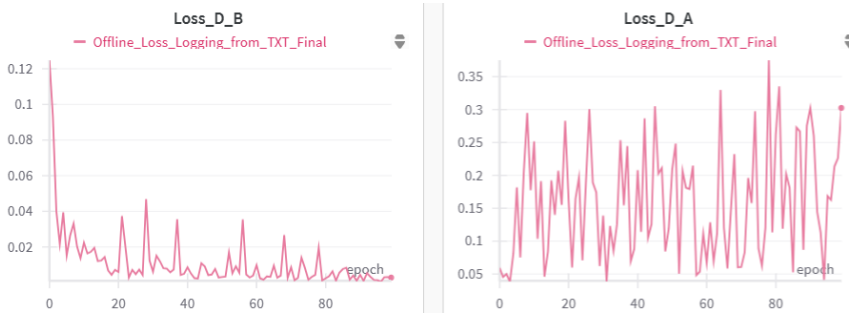

Figure 6: The Output Graph of Loss Functions (without Histogram Loss

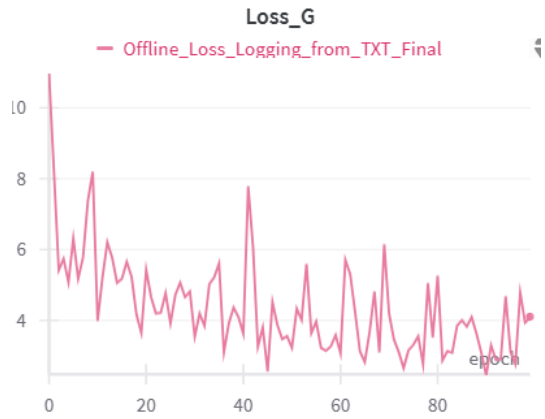

Figure 7: The Output Graph of Total Generator Loss Functions (without Histogram Loss)

3.2 Conclusion

This study confirms the feasibility of using CycleGAN for unpaired RGB-to-IR translation in the defense domain. Despite challenges in dataset quality and training stability, the model produces visually and structurally plausible IR images. Future work will explore diffusion models, transformer-based generators, and synthetic IR augmentation for real-time defense applications. This project effectively demonstrates the implementation of a CycleGAN architecture for image-toimage translation, specifically from RGB to IR. Key aspects include careful data preparation, a robust generator architecture with residual blocks, the crucial adversarial, cycle consistency, and identity losses, and the smart integration of SSIM loss for perceptual quality. The use of offline logging with Weights and Biases ensures reliable experiment tracking and visualization, which is essential for debugging and understanding the complex training dynamics of GANs. While I assumed the Discriminator class was present, its absence in the provided snippet highlights the importance of complete code for reproducibility.

As said in the original paper CycleGAN, Zhu et al. (2017) Nonetheless, in many cases completely unpaired data is plentifully available and should be made use of. This paper pushes the boundaries of what is possible in this “unsupervised” setting.

4 Challenges and Discussions

Project implementation is done by two different GAN network on special dataset with modifications and improve your own network. Modifications improved to specialized variant of GAN-driven image translation, tailored for visible-to-infrared conversion to better preserve structural details and reduce edge distortion. The challenges are, Setting up the environment appropriate with the Infra-GAN network dependencies is a big challenge. I had an error that GPU memory is not enough to

work at custom dataset data loading. Set an environment of the requirements by ubuntu wsl is installed to run the GAN network but still does not work so I ended up trying use and enhance github repo model architecture and fully set new model architecture from zero. Another challenge was Domain Mismatch: The absence of aircraft IR images required using proxy datasets. Later stages may include domain adaptation or fine-tuning if such data becomes available. Another challenge was, implementing histogram and SSIM loss modules requires adaptation from existing code and testing their effect through ablation experiments. On the dataset collection part, I tried a lot to find paired IR – RGB aircraft dataset however, I could not. Also since IR images in military aircraft is very rare. I cut images IR camera aircraft video.

REFERENCES

Source code of infragan with histogram loss. https://colab.research.google.com/drive/1oj1gUpLZteSdCEJIRHhMaYLjpxS7lsa?usp=sharing, a.

Source code of gan. https://colab.research.google.com/drive/1Kq4TS-3L2WREulPoMr7j5Cvt0KrbqxiH, b.

Source code of output result of without histogram loss. https://colab.research.google.com/drive/1SgfBaev85wCxBofBiJhaZwo3lAK1l8V?usp=sharing, c.

Military aircraft detection dataset. https://www.kaggle.com/datasets/a2015003713/militaryaircraftdetectiondataset. Accessed: 2025-06-21.

Bengisu Yücel. Cyclegan-infragan wandb results. URL https://api.wandb.ai/links/bengisuycl1-hacettepe-university/fbrj9ikt.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2232. IEEE, 2017.

Mehmet Akif Özkanoğlu and Sedat Ozer. Infragan: A gan architecture to transfer visible images to infrared domain. *Pattern Recognition Letters*, 155:69–76, 2022. ISSN 0167-8655. doi: https://doi.org/10.1016/j.patrec.2022.01.026. URL https://www.sciencedirect.com/science/article/pii/S0167865522000332.