Autonomous Damage Recognition in Ultrasonic Non-destructive Evaluation Using a Semi-Supervised Generative Adversarial Network

1st JonPaul Hooks
Dr. Jianjun Hu
CSCE 768 - Pattern Recognition and Classification
Department of Mechanical Engineering
University of South Carolina
Columbia, SC, USA
jrhooks@email.sc.edu

***Abstract*—In this project, I develop a semi-supervised generative adversarial network (SGAN) for automatic defect detection in ultrasonic C-scan images of composite materials. The method is inspired by reconstruction-based anomaly detection and semi-supervised adversarial learning. The generator reconstructs a "healthy" version of the input image while the discriminator jointly performs real/fake discrimination and supervised defect classification. Reconstruction error is visualized as a pixel-level heatmap to localize damage.**

The full system includes:

1) **a UNet-based generator for pixel-accurate reconstruction;**
2) **a dual-headed discriminator with classification capability;**
3) **a k-fold cross-validation training pipeline that records losses, ROC curves, AUC metrics, and checkpoints; and**
4) **an inference pipeline that produces severity scores and heatmaps using top-k reconstruction discrepancy.**

The implemented SGAN demonstrates stable reconstruction behavior and a meaningful correlation between reconstruction error and defect presence, supporting interpretable, semi-supervised ultrasonic NDE.

I. Introduction

Automated defect detection in ultrasonic non-destructive evaluation (NDE) is a challenging pattern recognition problem due to noise, variability between samples, and the scarcity of pixel-level labels. Deep learning models have been widely applied in imaging tasks, but purely supervised CNN architectures require extensive labeled datasets to generalize properly.

Generative adversarial networks (GANs) introduce an alternative: learning a representation of normal (i.e., defect-free) structure by forcing a generator to reconstruct images in a way that fools a discriminator [4], [8]. When trained semi-supervised [1], the discriminator also learns to classify global defect status. Because the generator is optimized to reconstruct non-defected structure, its failure to reconstruct damaged regions produces high-error residuals that can be visualized as heatmaps.

This motivates the SGAN approach implemented here. Unlike traditional classifiers that produce a single label, this framework yields:

Identify applicable funding agency here. If none, delete this.

- global defect classification,
- highly localized spatial defect maps, and
- a scalar severity score based on reconstruction discrepancy.

These align with the NDE three-level anomaly detection (binary classification, localization, and severity magnitude)

The final implementation incorporates a robust training infrastructure—including a custom Trainer, PyTorch-based UNet and discriminator models, k-fold dataset splitting, and detailed validation metrics—implemented across the project files.

II. Background and Related Work

Semi-supervised GANs have been effective in low-label regimes [4], particularly where anomalies manifest as deviations from learned normal structure. Prior work in ultrasonic NDE shows that adversarial methods can detect subsurface defects by analyzing reconstruction error [5]. Similar approaches have been applied in medical imaging, structural health monitoring, and unsupervised defect localization.

The SGAN used in this project draws inspiration from reconstruction-driven anomaly detection (e.g., autoencoders, GAN-based image completion) [5], but integrates supervised classification via a discriminator with dual output heads [4]. The use of UNet-style architectures improves spatial fidelity of reconstructions [1], addressing the high-resolution textures common in ultrasonic C-scans.

The implemented approach extends such work by:

- using a semi-supervised adversarial objective,
- generating heatmaps directly from reconstruction error, and
- providing k-fold performance evaluation including ROC curves and per-epoch losses.

III. Dataset and Preprocessing

All dataset logic is implemented in `data.py` [3].

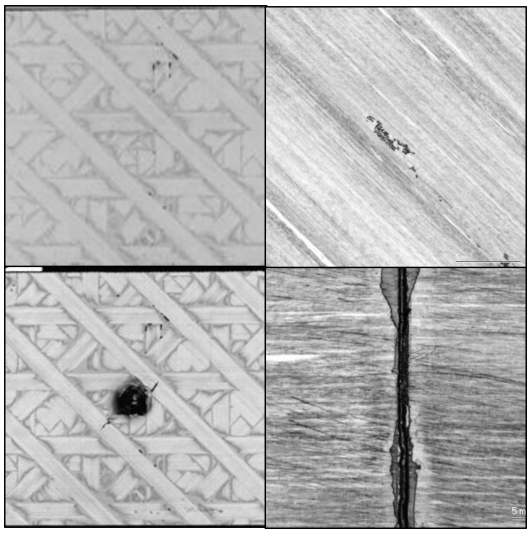


Fig. 1: Combined C-scan examples illustrating structural variations and typical defect modes present in the dataset.

A. Dataset Composition

The dataset consists of ultrasonic C-scan images (Fig. 1) stored in a directory, with an accompanying CSV specifying filenames and binary defect labels (0 = non-defective, 1 = defective). The dataset class supports both labeled and unlabeled modes.

B. Preprocessing Pipeline

Each image undergoes:

- Grayscale conversion
- Resize to 256×256
- Random rotation (±5°)
- Random horizontal flip (p = 0.5)
- Conversion to tensor
- Normalization to [-1, 1]

This augmentation aims to reduce overfitting while maintaining structural coherence typical in C-scan imagery.

IV. SGAN Architecture

A. Generator – UNetG

The generator G is implemented as a UNet-style encoder–decoder architecture designed for high-fidelity pixelwise reconstruction of ultrasonic C-scan images. The network operates on $1 \times 256 \times 256$ grayscale inputs and outputs a reconstructed image of the same size using a final $\tanh$ activation.

1) Encoder (Downsampling Path): The encoder consists of four convolutional stages that progressively expand channel depth while reducing spatial resolution using stride-2 convolutions:

- $e1:\ 1 \to 64$ channels
- $e2:\ 64 \to 128$ channels
- $e3:\ 128 \to 256$ channels
- $e4:\ 256 \to 512$ channels

Each block implements:

$$\text{Conv2D}(4{\times}4,\ \text{stride} = 2) \to \text{BatchNorm} \to \text{LeakyReLU}(0.2),$$

except the first layer, which omits batch normalization. This encoder compresses the input into a compact latent representation capturing global ultrasonic structure.

2) Bottleneck Layer: A 3×3 convolutional layer with 512 channels forms the bottleneck, enabling nonlinear transformation while preserving global structural information:

$$512 \to 512.$$

3) Decoder (Upsampling Path): The decoder mirrors the encoder and uses bilinear interpolation for upsampling, followed by convolutional refinement:

- $dc4:\ 512 \to 256$
- $dc3:\ 512 \to 128$ (after skip-connection)
- $dc2:\ 256 \to 64$
- $dc1:\ 128 \to 64$

Each decoder block performs:

$$\text{Upsample} \to \text{Conv2D}(3 \times 3) \to \text{BatchNorm} \to \text{ReLU}.$$

4) Skip Connections: UNet skip connections concatenate encoder and decoder features:

$$d_i = \text{concat}(d_i,\ \text{crop}(e_i)).$$

These connections preserve high-frequency ultrasonic texture, enabling the detection of small, spatially localized defects.

5) Output Layer: A final 3×3 convolution maps features back to a single channel:

$$64 \to 1,$$

followed by a $\tanh$ activation to normalize intensities to $[-1, 1]$.

B. Discriminator – Dual-Head Semi-Supervised Network

The discriminator D is a multi-task convolutional network that jointly performs (1) real/fake discrimination and (2) binary defect classification. A shared convolutional backbone extracts features for both tasks, enabling semi-supervised learning.

1) Convolutional Feature Extractor: The backbone consists of three convolutional blocks, each reducing spatial resolution and increasing feature depth:

- $1 \to 32$ channels
- $32 \to 64$
- $64 \to 128$

Each block uses:

$$\text{Conv2D}(3{\times}3,\ \text{stride} = 2) \to \text{BatchNorm} \to \text{LeakyReLU}(0.2) \to \text{Dropou}$$

After flattening, the network produces a high-dimensional feature vector:

$$128 \times 32 \times 32 \to 131{,}072 \text{ features}.$$

2) Adversarial Head (Real/Fake): A fully connected layer outputs a scalar:

$$D_{\text{adv}}(x) = \mathbf{w}^\top \mathbf{f} + b,$$

trained using a binary cross-entropy loss with real-label smoothing (target = 0.92). This head enables adversarial training and constrains G to produce realistic reconstructions.

3) Classification Head (Defect vs. No Defect): A second fully connected layer maps the shared features to a two-class logit vector:

$$D_{\text{cls}}(x) \in \mathbb{R}^2,$$

optimized using cross-entropy. This supervised branch improves SGAN stability and leverages the limited labeled dataset.

4) Architectural Advantages: The discriminator design provides several benefits:

- Shared features improve semi-supervised learning efficiency.
- Downsampling convolutional structure increases robustness to noise and ultrasonic texture variation.
- Dual-task learning allows the model to jointly learn classification and adversarial discrimination.
- Lightweight architecture stabilizes GAN training while maintaining high representational capacity.

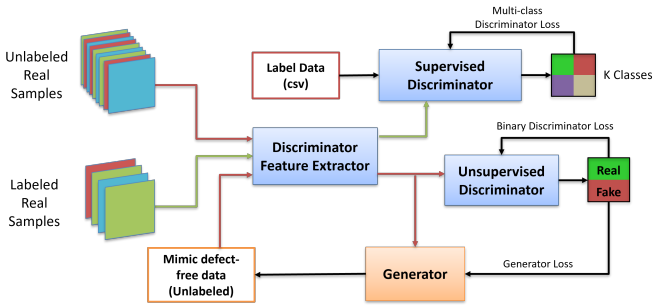


Fig. 2: Overview of the semi-supervised SGAN architecture used in this work. Labeled and unlabeled ultrasonic C-scan images are passed through a shared discriminator feature extractor, which branches into a supervised classification head and an unsupervised adversarial head. The generator attempts to reconstruct defect-free images to fool the adversarial discriminator, enabling reconstruction-based anomaly detection.

V. Training Methodology

The training engine is defined in `trainer.py` [3].

A. Loss Functions

1) Discriminator Loss: The discriminator computes three loss terms during training:

- BCE for real/fake discrimination (1)
- Cross-entropy for defect classification (2)
- BCE for recognizing reconstructed images as fake (3)

$$\mathcal{L}_{D,\text{real}} = \text{BCE}(D_{\text{adv}}(x), 0.92) \tag{1}$$

Equation (1) represents real-vs-fake loss on real images. This encourages real images to be classified as "real" (soft 0.92 target as used in code [3]).

$$\mathcal{L}_{D,\text{cls}} = \text{CE}(D_{\text{cls}}(x), y) \tag{2}$$

Equation (2) represents the supervised classification loss from the discriminator. This enables the discriminator to learn defect classification (defect or no-defect).

$$\mathcal{L}_{D,\text{fake}} = \text{BCE}(D_{\text{adv}}(G(x)), 0) \tag{3}$$

Equation (3) represents the fake-image discrimination loss. To converge this loss, the discriminator must recognize reconstructed images as fake.

$$\mathcal{L}_D = \mathcal{L}_{D,\text{real}} + \mathcal{L}_{D,\text{cls}} + \mathcal{L}_{D,\text{fake}} \tag{4}$$

The total loss equation for the discriminator (4) matches the D loss calculation in the code [3] and contributes to discriminator improvement throughout the training.

2) Generator Loss: The generator computes another three loss terms during training:

- Adversarial loss
- L1 reconstruction loss (scaled by λ = 20)
- Supervised classification forcing "healthy" labels on reconstructions

$$\mathcal{L}_{G,\text{adv}} = \text{BCE}(D_{\text{adv}}(G(x)), 0.92) \tag{5}$$

The adversarial loss, equation (5), encourages generator reconstructions to appear "real" to the discriminator. This loss encourages the generator always to produce a version of any input image that will fool the discriminator.

$$\mathcal{L}_{G,\text{rec}} = \|G(x) - x\|_1 \tag{6}$$

Equation (6) enforces pixel-accurate reconstruction of healthy scan structure. This term is scaled by a hyperparameter λ = 20 (as implemented in the Trainer).

$$\mathcal{L}_{G,\text{sup}} = \text{CE}(D_{\text{cls}}(G(x)), 0) \tag{7}$$

The final generator loss equation (7) forces its reconstructed outputs to always be classified as "non-defective". This stabilizes adversarial training by giving the generator a classification target.

$$\mathcal{L}_G = \mathcal{L}_{G,\text{adv}} + \lambda \mathcal{L}_{G,\text{rec}} + \gamma \mathcal{L}_{G,\text{sup}} \tag{8}$$

The total generator loss equation (8) corresponds directly to the code in `trainer.py` [3].

B. Optimization

Training the semi-supervised SGAN requires balancing the competing objectives of the generator and discriminator to ensure stable adversarial learning, effective reconstruction, and reliable defect classification. To achieve this, the implementation adopts several optimization strategies that directly correspond to the behavior of the training loop in `trainer.py` [3].

1) Adam Optimization for Both Networks: Both the generator and discriminator are optimized using the Adam optimizer with momentum parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$ [4]:

$$(\theta_G, \theta_D) \leftarrow Adam(\theta, \alpha, \beta_1, \beta_2) \tag{9}$$

This configuration is commonly used in GAN literature because the reduced β_1 stabilizes oscillations that can occur during adversarial updates. The learning rates differ between networks:

- Generator learning rate: $2e^{-4}$
- Discriminator learning rate: $1e^{-4}$

This asymmetry slows discriminator convergence slightly, preventing it from overpowering the generator early in training.

2) Alternating Gradient Updates: For each mini-batch, the discriminator is updated first using its composite loss (Eq. (4)) followed by one or more generator updates using Eq. (8). This ordering reflects standard GAN training practice, as the generator relies on discriminator gradients to improve [4].

The code includes a dynamic update rule:

- If the discriminator achieves > 0.75 supervised classification accuracy during an iteration, the generator performs two optimization steps instead of one.

This adaptive scheduling prevents discriminator overfitting and reduces the risk of mode collapse by ensuring the generator can "catch up" when necessary.

3) Label Smoothing for Real Samples: Real images use a smoothed target label of 0.92 instead of 1.0:

$$D_{adv}(x) \rightarrow 0.92 \tag{10}$$

Label smoothing is a known GAN stabilization method that reduces discriminator overconfidence, improves gradient flow, and makes adversarial training less brittle [4].

4) Reconstruction Weighting and Multi-Objective Balance: The generator loss includes a heavily weighted reconstruction penalty:

$$\lambda = 20 \tag{11}$$

This signals that faithful reconstruction is the dominant objective during early training. Since the classification and adversarial components operate on higher-level feature representations, balancing losses in this way ensures that:

- the generator first learns the distribution of defect-free images
- meaningful reconstruction differences emerge
- heatmaps accurately reflect structural anomalies

The supervised generator term, weighted by γ, helps align generator outputs with the discriminator's defect classes, providing an additional stabilizing force in semi-supervised training.

5) Gradient Flow and Computational Efficiency: Gradients are disabled during evaluation (torch.no_grad()), and both networks are switched between .train() and .eval() modes appropriately. This ensures:

- batch norm layers behave correctly
- dropout is disabled during validation
- GPU computation is used efficiently

Additionally, mini-batch training reduces gradient variance, improving convergence behavior.

VI. K-Fold Cross-Validation Framework

To reliably evaluate the semi-supervised SGAN model and reduce sensitivity to dataset partitioning, a full k-fold cross-validation (CV) framework is implemented in `train_cv.py` [3]. This framework manages fold construction, training execution, metric logging, model checkpointing, and summary generation.

A. Fold Construction

For $k \geq 2$, the system uses *Stratified K-Fold* splitting to preserve class balance between defect and non-defect samples. Each fold produces two CSV files representing the training and validation subsets. These subsets are loaded using the `DefDataset` class, ensuring consistent preprocessing and augmentation. If $k = 1$, an 80/20 stratified train–validation split is used instead.

B. Training and Validation Procedure

For each fold, new instances of the generator and discriminator are initialized and trained for the configured number of epochs. The `Trainer` class performs adversarial updates, reconstruction learning, and supervised classification. After every epoch, the validation set is evaluated to compute the ROC–AUC score:

$$\text{AUC} = \text{ROC\_AUC}(\hat{y}, y).$$

Validation is executed in inference mode to ensure accurate metric computation without gradient interference.

C. Logging and Checkpointing

During training, key metrics are recorded at each epoch, including:

- generator loss,
- discriminator loss,
- validation AUC.

These metrics are written to a per-fold log (`epoch_metrics.csv`) and a global combined log (`all_epoch_metrics.csv`). Additionally, model weights (`gen_ep*.pth`, `disc_ep*.pth`) and a compressed training history file (`history.npz`) are saved each epoch, enabling reproducibility and resume capability.

D. Visualization and Summary Metrics

After each fold completes, loss curves and ROC plots are generated using the utilities in `utils.py` [3]. A metrics file (`metrics.json`) stores the fold's precision, recall, AUC, and confusion matrix. Upon completion of all folds, a consolidated summary table (`folds_summary.csv`) provides an overview of SGAN performance across folds.

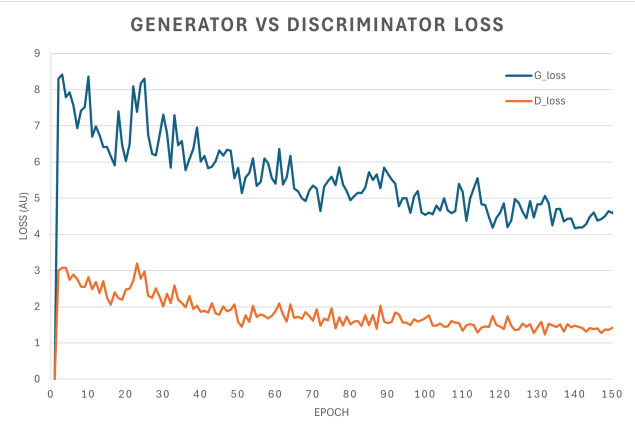


Fig. 3: Generator and discriminator loss curves over training epochs, showing stable convergence during SGAN optimization.

E. Advantages for Ultrasonic NDE

K-fold CV is particularly valuable for ultrasonic inspection tasks, where datasets tend to be small and class-imbalanced. It provides a more robust estimate of model generalization, exposes variability between folds, and ensures that evaluation is not biased by a single train–validation split. This strengthens confidence in the SGAN's defect detection and heatmap-based localization capabilities.

VII. Experiments and Results

This section summarizes the performance of the semi-supervised SGAN using the k-fold cross-validation framework. The evaluation focuses on training stability, defect classification accuracy, and the effectiveness of reconstruction-based heatmaps for anomaly localization.

A. Training Behavior

Across folds, generator and discriminator losses followed stable and convergent trends, indicating balanced adversarial dynamics (Fig. 3). The strong reconstruction penalty encouraged the generator to reproduce healthy image structure, which is essential for reliable residual-based defect detection. The discriminator simultaneously improved its supervised classification accuracy, demonstrating effective semi-supervised learning.

B. Quantitative Performance

Validation was conducted after each epoch using the ROC–AUC metric. The resulting ROC curves showed clear separation between defective and non-defective samples, with consistently high AUC scores across folds. Precision, recall, and confusion matrix statistics stored in each fold's summary further confirmed robust classification performance. Low variance in these metrics suggested good generalization despite limited labeled data.

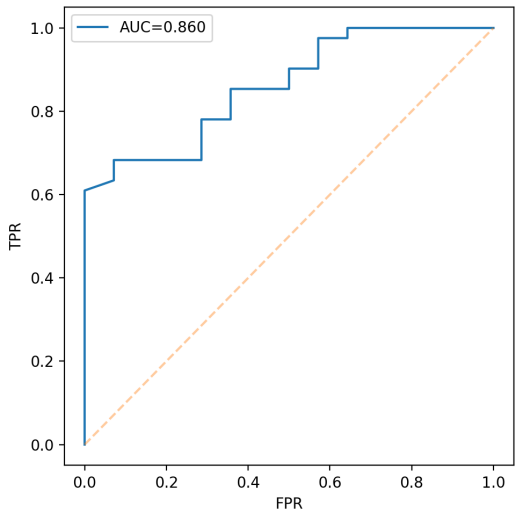


Fig. 4: ROC curve for Fold 1 with corresponding AUC, illustrating the SGAN discriminator's defect classification performance.

C. Heatmap Inspection

Reconstruction-error heatmaps produced during inference highlighted localized regions of elevated residuals in defective samples, while non-defective scans showed uniformly low error [5]. This behavior aligns with the SGAN objective: the generator reconstructs healthy structure, causing defects to manifest as high-error regions. These visualizations provide interpretable spatial evidence of defect location.

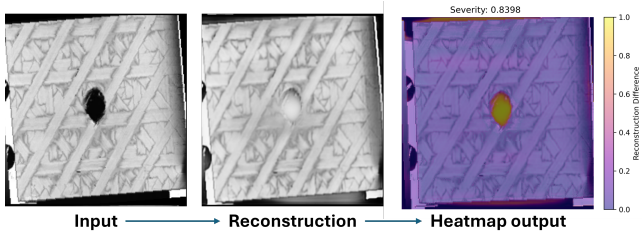
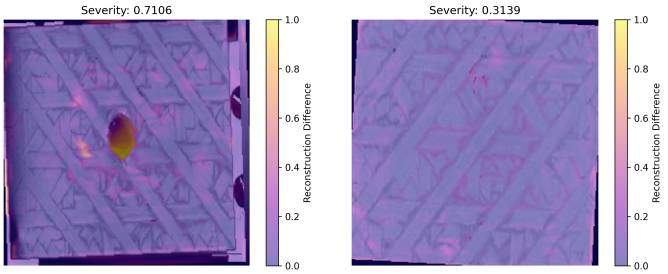


Fig. 5: Example of SGAN reconstruction-based defect localization. Left: original ultrasonic C-scan containing a visible defect. Middle: generator reconstruction representing the model's learned defect-free appearance. Right: reconstruction-error heatmap, where high-intensity regions correspond to localized structural anomalies. The computed severity score (top) reflects the magnitude of the top-k residual response.

D. Severity Estimation

A scalar severity score, computed using the top-k fraction of reconstruction errors, offered a concise quantitative measure of defect magnitude. This metric effectively emphasized localized anomalies and supported practical thresholding for inspection scenarios.



(a) High-severity defect. (b) Low-severity defect.

Fig. 6: Comparison of SGAN reconstruction-error heatmaps for defects of differing magnitude. The model produces higher-intensity residuals for more severe anomalies, consistent with the severity scoring metric.

E. Overall Findings

In summary, the SGAN achieved:

- stable adversarial and reconstruction training behavior,
- strong defect classification across folds,
- clear and interpretable heatmaps for anomaly localization,
- consistent quantitative performance with low cross-fold variability.

These results demonstrate that the SGAN framework provides both accurate detection and meaningful spatial interpretation for ultrasonic C-scan defect analysis.

VIII. Conclusion

This work presented a semi-supervised SGAN framework for defect detection and localization in ultrasonic C-scan imagery. By combining a UNet-based generator with a dual-headed discriminator, the model simultaneously learned to reconstruct defect-free representations of input scans and to classify defective regions through supervised and adversarial objectives. The resulting reconstruction-error heatmaps provided clear and interpretable spatial localization of anomalies, while the top-k severity metric offered a practical quantitative measure of defect magnitude.

Across k-fold cross-validation, the SGAN demonstrated stable training behavior, strong defect classification performance, and consistent generalization despite limited labeled data. Qualitative results showed that the model effectively distinguished between severe, mild, and noise-like anomalies, highlighting its usefulness for real inspection scenarios. Overall, these results indicate that the SGAN approach provides a robust and interpretable framework for automated ultrasonic NDE, with potential for deployment in structural health monitoring pipelines.

IX. Future Work

Future extensions of this work include adding physics-informed loss terms to constrain the generator according to ultrasonic wave propagation principles. Physics-informed ML approaches [6], [7] may improve reconstruction fidelity and enhance the reliability of heatmap-based localization.

Expanding the dataset to include additional ultrasonic modalities (B-scans, A-scans, multi-frequency data) is another promising direction, enabling richer feature learning and improved robustness across inspection scenarios. Further gains may also be achieved by exploring advanced GAN variants for semi-supervised learning [4] and anomaly detection [5].

Acknowledgment

The author would like to thank Dr. Jianjun Hu for his guidance and instruction throughout the CSCE 768 course, which provided the foundational concepts and tools necessary for this work. The author also gratefully acknowledges Dr. Sourav Banerjee for his mentorship, insight into ultrasonic nondestructive evaluation, and continued support of this research direction.

References

[1] K. K. Prajapati, A. Ghosh, and M. Mitra, "Semi-supervised generative Adversarial Network (SGAN) for damage detection in a composite plate using guided wave responses," Mechanical Systems and Signal Processing, vol. 232, p. 112686, Jun. 2025. doi:10.1016/j.ymssp.2025.112686

[2] K. K. Prajapati, "Lamb Wave-Based Structural Health Monitoring using Deep Learning Techniques," thesis, 2025

[3] J. Hooks, "CSCE768_Final-Project," GitHub, Dec. 2025. [Online]. Available: https://github.com/jhooks5313/CSCE768_Final-Project.git

[4] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. NeurIPS*, 2016.

[5] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proc. IPMI*, 2017.

[6] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *Journal of Computational Physics*, 2019.

[7] G. E. Karniadakis, I. G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," *Nature Reviews Physics*, 2021.

[8] J. Hu, "Lecture 9: Generative Models," CSCE 768 Pattern Classification, Dept. of Computer Science and Engineering, Univ. of South Carolina, 2025, [PowerPoint slides].