Exploring GAN Variants for Balancing Imbalanced Datasets

Implementation and comparison of Vanilla GAN and advanced variants to generate synthetic minority-class samples, with evaluation of classification performance using different metrics.

Fares Mohammed-Saleem Hamdi Hatahet

Course: Special Topics in Artificial Intelligence

Course Instructor: Dr. Yousef Sanjalawe

26th of May 2025

Table of contents

1. Problem Statement

Machine learning models are not an all-in-one box, they are affected by many factors, the most important one is data. Every model is only as good as the data it was trained on. In many practical scenarios, datasets are not evenly distributed across classes, this is known as the class imbalance problem. For example, in a fashion classification task, there might be thousands of samples of common items like shirts or sneakers, but only a few instances of rare items like ankle boots or coats, this skewed distribution leads to a bias during training, where models become overly confident in predicting the majority classes and fail to recognize or learn the characteristics of underrepresented ones. Class imbalance is not just a data inconvenience; it directly impacts the reliability and fairness of machine learning models. When left unaddressed, it can lead to poor recall for minority classes, which is a huge problem in critical fields such as medical diagnosis or fraud detection, where the minority class often represents the event of interest.

Generative Adversarial Networks (GANs) have shown great promise in synthesizing realistic data samples. This project focuses on leveraging different versions of GANs to generate synthetic images for the minority class in an intended imbalanced version of the FashionMNIST dataset, a commonly used benchmark in image classification tasks, then evaluate and compare the performance of a classifier after being trained on balanced new versions of the dataset generated by the GANs.

The main objective this experiment is to evaluate and compare the performance of three different GAN variants: Vanilla GAN, Deep Convolutional GAN (DCGAN), and Wasserstein GAN with gradient penalty (WGAN). By generating additional samples for the underrepresented class using each of these models, in order to balance the dataset and train a more robust classifier. The effectiveness of each GAN is assessed based on how well it improves classification metrics such as accuracy, precision, recall, and F1-score on the balanced versus imbalanced datasets.

2. Description of Dataset & Imbalance Analysis

The dataset used for this project is the FashionMNIST dataset, a very commonly used dataset for benchmarking for image classification tasks, it consists of 28x28 pixels grayscale images of clothing items, there are 10 different classes, including T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. The dataset has two separated sets, a training set of 60,000 images and a test set of 10,000 images, with 6,000 training samples and 1,000 test samples for each class under normal, balanced conditions.

To simulate a real-world scenario where data is not evenly distributed, an artificial imbalance was introduced into the training set. Specifically, the number of training samples for the chosen class, which is class 1 (trousers) was reduced to just 20% of its original size, keeping only 1,200 examples instead of 6,000. This represents a severe class imbalance, since the other classes remain unaffected, each with their full set of 6,000 training samples.

Two distinct methods were used to generate this imbalance:

1. Local Dataset Manipulation: Preprocessed, imbalanced datasets were created and saved separately. These datasets were later loaded on the fly during model training, allowing for quicker experimentation and consistent class distributions.
2. Imbalancer Utility Class: A custom class was implemented to apply class imbalance dynamically, by giving it a dataset, the desired class to be imbalanced and the keep ratio. This method offered greater flexibility, especially during development, and made it faster and easier to experiment with different imbalance ratios.

To understand the nature of the new imbalanced dataset, bar plots were generated to visualize the number of samples per class in the training set, as shown in Figure 1. The imbalance skewed the dataset in favor of the majority classes, since a single class is being reduced severely, with trousers being

heavily underrepresented. This level of imbalance is not just visually apparent but statistically significant, which can easily result in biased models, classifying it as an extreme imbalance scenario.

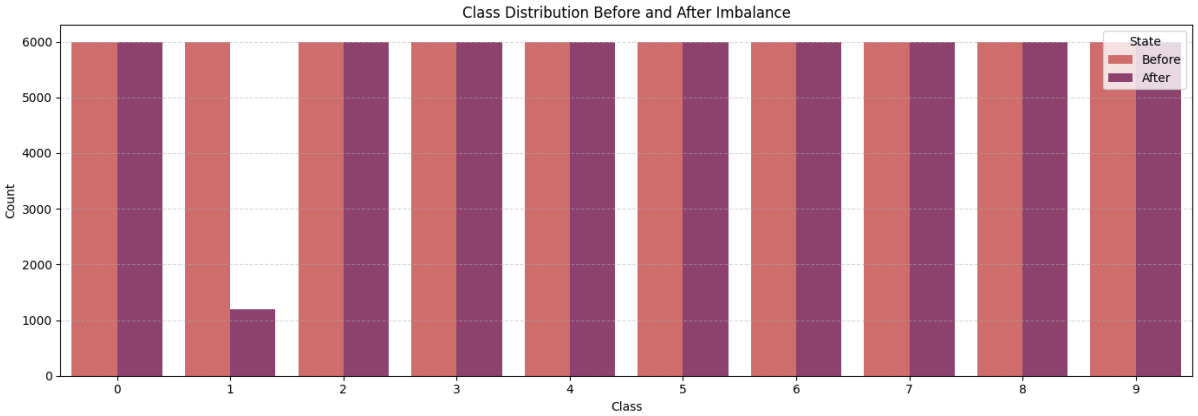


Figure 1: Class distribution before and after imbalancing

Such an imbalance may lead to several issues in model training. Most notably, the classifier tends to become biased toward the majority classes. While overall accuracy may appear high, this can be misleading because the model often fails to accurately forecast the minority class. This leads to low recall and precision for class 1, potentially harming the model's ability to generalize and may cause it to ignore patterns unique to the underrepresented class, and lead to ineffectiveness in most of the applications where accurate identification of all classes is crucial.

By introducing generative models like GANs to synthesize new samples for the minority class, this project aims to counter these issues and evaluate whether synthetic augmentation can truly enhance classification performance in an imbalanced setting.

3. Details of GAN Architectures & Training

All models were implemented in PyTorch and trained for 500 epochs using CUDA-enabled GPUs. Each GAN aimed to generate synthetic samples for the minority class in imbalanced datasets. A noise vector of size 100 was used as input across all models.

3.1 Vanilla GAN

The Vanilla GAN consists of simple fully connected layers. The generator maps the 100-dimensional noise vector to the image space using two linear layers with a LeakyReLU activation followed by a Tanh function to scale outputs between -1 and 1. The discriminator mirrors this simplicity, using two linear layers with LeakyReLU and a Sigmoid activation to predict whether an input image is real or fake. Training used Binary Cross Entropy Loss and the Adam optimizer with a learning rate of 0.0003. Images were flattened to vectors of size 1024 (for 32×32 resolution), and training was performed using a batch size of 64.

3.2 Deep Convolutional GAN (DCGAN)

The DCGAN introduces a more robust architecture using deep convolutional layers. The generator employs a series of transposed convolution layers , each followed by Batch Normalization and ReLU activations, ending with a Tanh output layer. This structure progressively upsamples the input noise into 32×32 images, which were later cropped to 28×28 for dataset compatibility. The discriminator uses standard convolutional layers followed by Batch Normalization and LeakyReLU activations, gradually downsampling the input image to a single scalar prediction. Unlike the vanilla GAN, DCGAN used Binary Cross Entropy with logits loss provided by Pytorch to combine the sigmoid layer and binary cross entropy in a numerically stable form. The optimizer was Adam with a learning rate of 0.0002, and the batch size was changed to 128.

3.3 Wasserstein GAN with Gradient Penalty (WGAN-GP)

The Wasserstein GAN (WGAN) follows the same deep convolutional architecture as DCGAN but replaces the discriminator with a “critic” and uses a different training objective. It eliminates the sigmoid activation and uses the Wasserstein loss function, which provides smoother gradients and more stable training. To enforce the Lipschitz constraint required by Wasserstein distance, a gradient penalty term is added during training. The critic is updated multiple times (typically five) for each generator update. The optimizer is Adam with the same learning rate (0.0002), and a gradient penalty coefficient of $\lambda = 10$ is used. This model also outputs 32×32 images.

Throughout training, several challenges were addressed. Mode collapse, especially with Vanilla GAN, was partially mitigated using label smoothing and noise injection to real images during discriminator training. Training instability was handled by tuning learning rates and choosing loss functions suited for each architecture. The PneumoniaMNIST dataset (224×224) was initially considered but ultimately excluded due to its high resolution and the excessive training time it would require within project constraints.

Generated sample images for each GAN variant are included in Figure 2, Figure 3 and Figure 4 shown below, showing qualitative improvements across architectures. Vanilla GAN produced low-detail, pixelated outputs and struggled with diversity. DCGAN generated sharper and more structured images, while WGAN showed the best stability and visual realism in outputs across epochs.

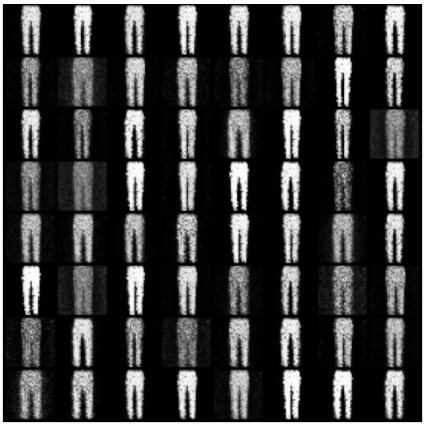


Figure 2: Vanilla GAN results

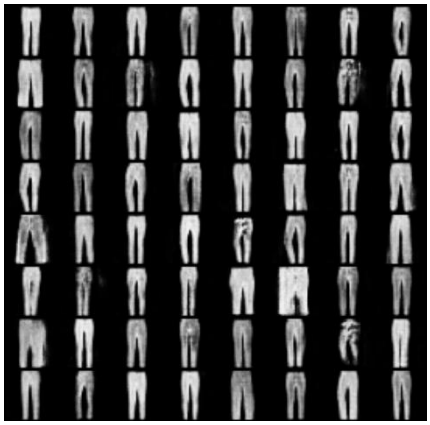


Figure 3: DCGAN results

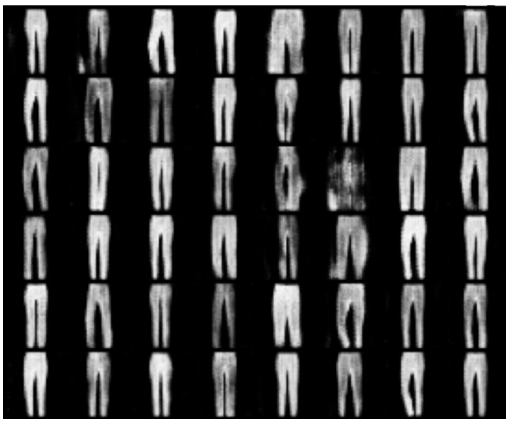


Figure 4: WGAN-GP results

4. Classifier Setup and Evaluation

For classification, a pre-trained ResNet-18 model was used following a transfer learning approach. The model was initialized with weights learned from the ImageNet dataset, and its original layers were kept frozen to preserve their learned representations. Only the final classification layer was replaced and fine-tuned for binary classification, ensuring that updates during training affected only this new layer while the rest of the network remained unchanged.

The classifier was trained under four different scenarios: using the original imbalanced dataset as a baseline; using a dataset balanced with synthetic samples from a Vanilla GAN; using a dataset balanced with DCGAN-generated samples; and using a dataset balanced with WGAN-GP outputs. In all cases, testing was performed on the original (unaltered) dataset to ensure a consistent and fair evaluation across all scenarios. The optimizer used a learning rate of 0.0001 and a weight decay of 0.00001.

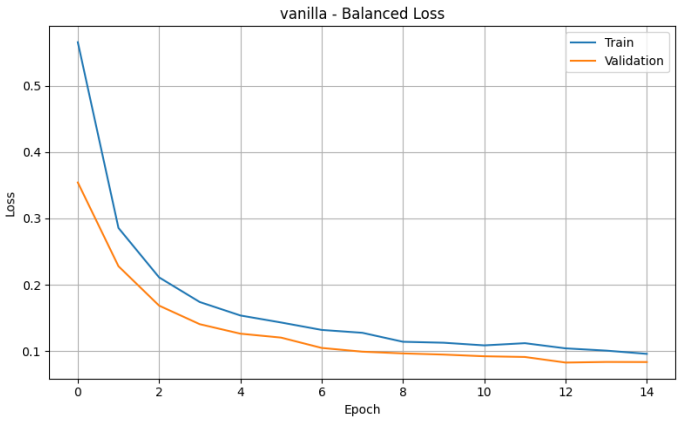


Figure 4: Vanilla GAN epochs vs. loss

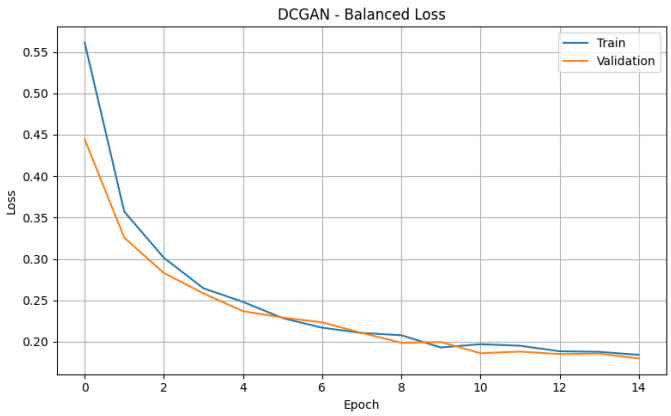


Figure 5: DCGAN epochs vs. loss

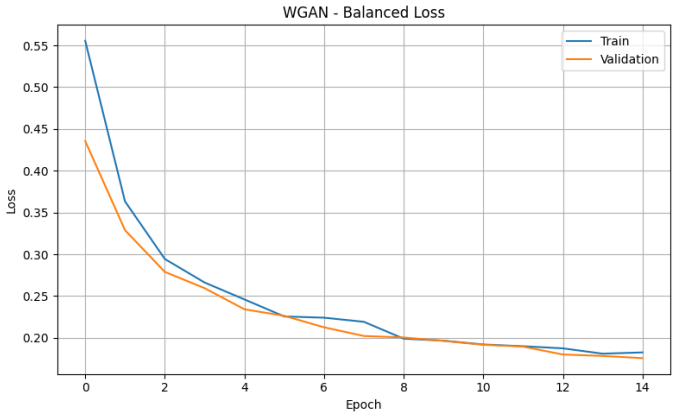


Figure 6: WGAN-GP epochs vs. loss

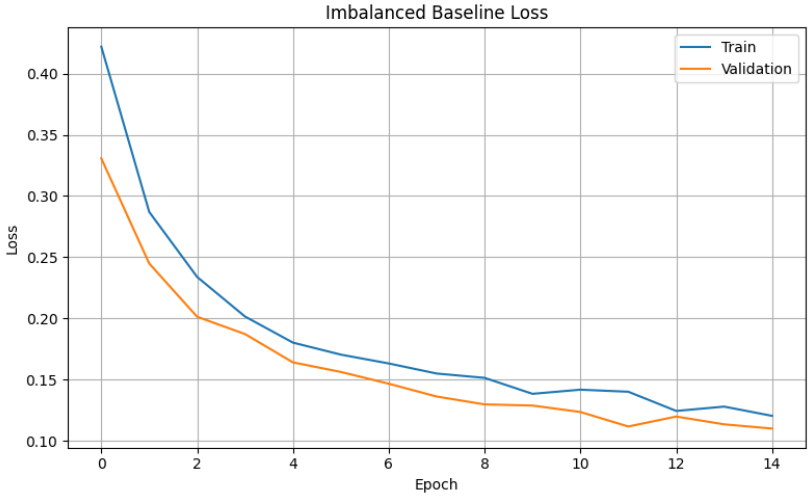


Figure 7: Baseline epochs vs. loss

To evaluate the performance, multiple metrics were used, including accuracy, precision, recall, F1-score, AUC, and confusion matrices. These metrics were selected to provide a well-rounded assessment, particularly because accuracy alone is insufficient in imbalanced data settings. Metrics like recall and F1-score are especially crucial in tasks that have imbalanced nature of data, where correctly identifying the minority class can be more important than overall accuracy. The AUC and confusion matrices further illustrate the model's ability to distinguish between the two classes and provide insight into false positive and false negative rates.

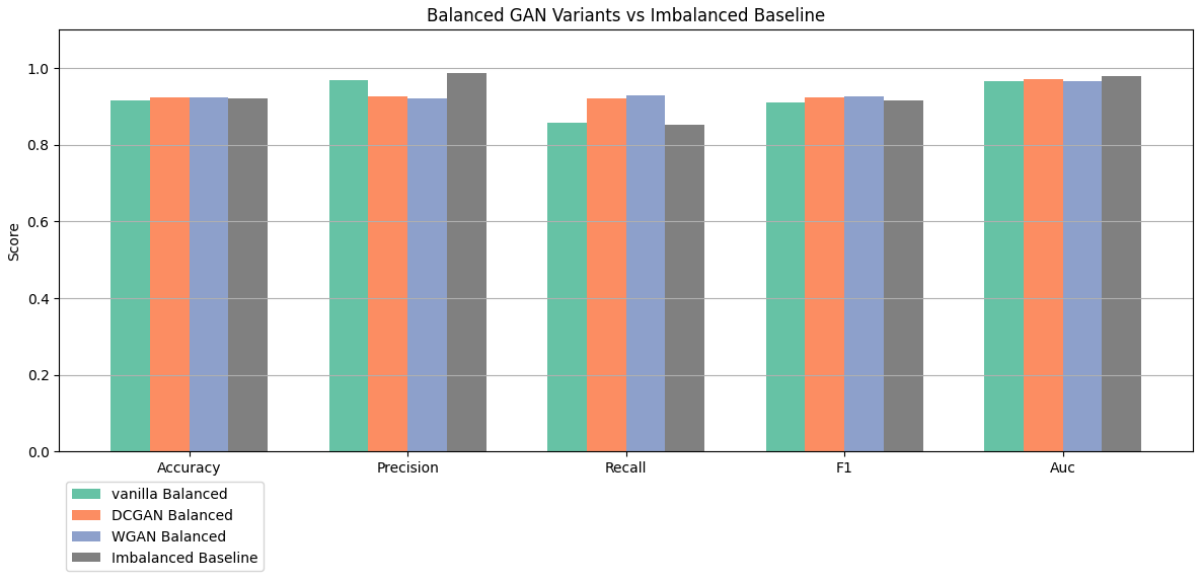
5. Results & Comparisons

This section presents the performance results of the classifier across all four training scenarios. Comparative tables summarize key evaluation metrics for each setup, showing both absolute differences and percentage changes relative to the baseline (imbalanced) dataset. Following the table, a bar plot offers a visual comparison across the same metrics to make performance differences more interpretable.

Metric	Baseline	Vanilla Balanced	Absolute Difference	Performance Shift
Accuracy	0.920	0.915	-0.006	-0.65%
Precision	0.987	0.969	-0.018	-1.81%
Recall	0.852	0.856	+0.004	+0.47%
F1-Score	0.915	0.909	-0.006	-0.60%
AUC	0.978	0.966	-0.012	-1.21%

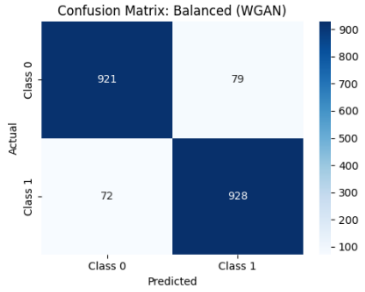
Metric	Baseline	DCGAN Balanced	Absolute Difference	Performance Shift
Accuracy	0.920	0.923	+0.002	+0.22%
Precision	0.987	0.925	-0.063	-6.34%
Recall	0.852	0.920	+0.068	+7.98%
F1-Score	0.915	0.922	+0.008	+0.84%
AUC	0.978	0.97	-0.008	-0.85%

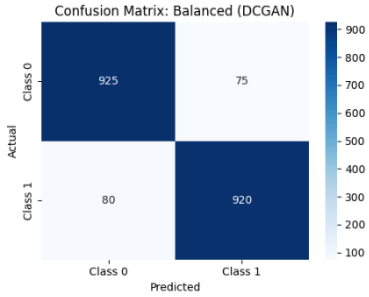
Metric	Baseline	WGAN-GP Balanced	Absolute Difference	Performance Shift
Accuracy	0.920	0.925	+0.004	+0.43%
Precision	0.987	0.922	-0.066	-6.66%
Recall	0.852	0.928	+0.076	+8.92%
F1-Score	0.915	0.925	+0.010	+1.11%
AUC	0.978	0.965	-0.013	-1.30%

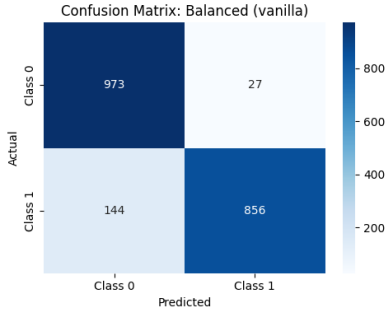


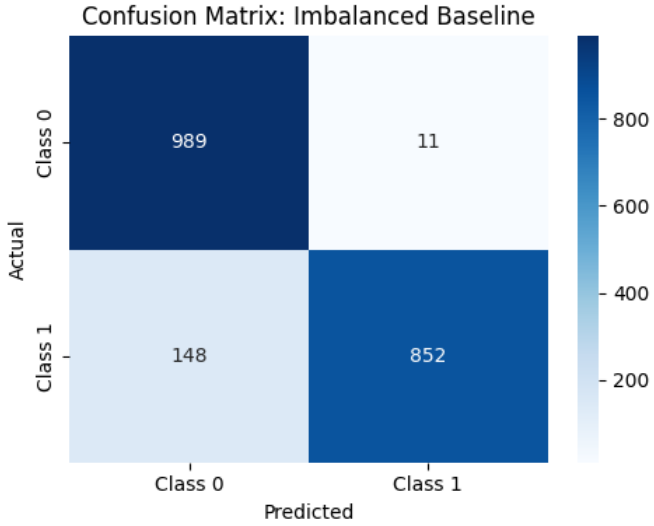
In comparing the Vanilla GAN-balanced dataset with the baseline, results show a slight drop in most metrics such as accuracy, precision, F1-score, and AUC. However, recall improved marginally, indicating a slight advantage in detecting minority class samples despite a minor overall performance decrease. For the DCGAN-balanced dataset, the results reveal a notable improvement in recall 7.98%, suggesting enhanced detection of minority class samples. F1-score and accuracy also showed slight gains. However, precision dropped significantly, indicating an increased rate of false positives, this trade-off may be acceptable in domains where recall is more critical. The WGAN-GP-balanced dataset yielded the best recall performance, with an 8.92% improvement over the baseline. It also had the highest gains in F1-score and accuracy among all scenarios. Similar to DCGAN, this came at the cost of a decrease in precision, but the overall balance between metrics makes WGAN-GP the most promising method among those tested.

Together, these results demonstrate that GAN-based balancing can improve minority class sensitivity, especially when using more advanced GAN variants like WGAN-GP, although the trade-off with precision must be carefully considered depending on the application.









6. Observations and Conclusions

This project explored the use of Generative Adversarial Networks (GANs) to address the challenge of class imbalance in image classification tasks. By generating synthetic samples for the underrepresented class in the FashionMNIST dataset, the goal was to enhance the classifier's ability to generalize without relying solely on real-world data augmentation.

The results highlight several key observations. First, although the baseline classifier trained on imbalanced data achieved relatively high overall accuracy, it struggled in terms of recall, indicating poor performance on the minority class. Introducing synthetic samples via GANs proved effective in improving this. Among the three GAN variants used, the more advanced models, DCGAN and WGAN-GP, demonstrated a noticeable benefit in recall and F1-score, especially in the case of WGAN-GP, which showed the most consistent improvements across multiple metrics. However, this improvement came with a trade-off: both DCGAN and WGAN-GP caused a drop in precision, likely due to the increased number of false positives introduced by the synthetic data. This suggests that while GAN-based augmentation can increase sensitivity to the minority class, it may do so at the cost of specificity. The Vanilla GAN, although able to slightly improve recall, did not result in overall performance improvement and suffered from unstable training and poor sample quality. One important takeaway is that the quality and structure of the generated images play a crucial role in the effectiveness of synthetic augmentation. More sophisticated GAN architectures like WGAN-GP, which stabilize training and maintain diversity in outputs, yield better results than simpler models.

In conclusion, using GANs to rebalance datasets shows promise, particularly for improving recall in imbalanced classification problems. Among the models tested, WGAN-GP offered the best overall performance improvement, albeit with some trade-offs. These findings suggest that when used thoughtfully, GAN-based approaches can be a valuable tool for enhancing model robustness in imbalanced learning scenarios. Future work could explore hybrid strategies that combine synthetic augmentation with sampling techniques or extend the approach to higher-resolution datasets given sufficient computational resources.