Synthetic Data Generation with Unity for Semantic Segmentation of Levee Encroachments

Anav Katwal[1] Ayon Dey[1] Abdullah Bin Naeem[1] Abdullah Al Redwan Newaz[1] Md Tamjidul Hoque[1]

[1]Department of Computer Science, Louisiana State University New Orleans
Corresponding email: thoque@lsuneworleans.edu

Abstract – While encroachments pose high risk for a levee system's integrity, detecting and segmenting levee encroachments in an image is challenged by the lack of real training data and the fact that they are defined by their position relative to the levee rather than their shape, texture or form. We present a synthetic data generation pipeline using Unity, aimed to address the data scarcity of levee encroachments. 3D virtual environments were developed in Unity to simulate diverse landscapes under varying environmental conditions, and encroachment criterion was modeled within the framework. With this system, a synthetic dataset of labeled RGB images was generated, capturing a variety of encroachment scenarios. To assess the validity of the generated synthetic dataset, selected deep learning models for semantic segmentation were trained on the dataset and evaluated. The results were compared with the same models trained on a real dataset of levee encroachments. The comparison showed that models trained on the synthetic dataset achieved better mean Intersection over Union (mIOU) scores and accuracy than those trained on the limited real dataset. This underscores the viability of virtual environments as scalable and controllable sources of training data, contributing to the advancement of automated levee monitoring systems.

Index Terms – Encroachment, Levee monitoring, Semantic segmentation, Synthetic data generation, Unity

I. INTRODUCTION

Levees are man-made earthen embankments constructed to control, contain, or divert the flow of water to reduce the risk of flooding. Multiple of these levees, along with other structures like flood walls, drainage systems, and closures make a levee system, and their construction relies on sound engineering practices [1]. The issues that can make levee systems vulnerable to failure are known as levee deficiencies and include problems like erosion, animal burrows, cracks, ruts, seepages, sand boils, vegetations, and encroachments.

While levee monitoring is a crucial step for ensuring its effective operation, manual inspection can be expensive in terms of cost and time. Automating such task is realized by technologies like remote sensing, Unmanned Aerial Vehicles (UAVs), and Unmanned Ground Vehicles (UGVs) [2]. Recent development in Deep Learning (DL) techniques have added to the convenience and accuracy in the task of levee monitoring [3], [4]. But detecting and segmenting any object in an image or video with a DL model requires a large amount of labeled data.

There have been previous works that deal with detecting and segmenting deficiencies like sand boils [5], ruts, seepages [2], [3], etc. However, limited research is present for addressing levee encroachments compared to other deficiencies. Any physical structure or activity on, over, through, or under a levee system that can compromise its functionality or accessibility during maintenance and flood response activities is called a levee encroachment and can include structures like houses, fences, walls, roads, gazebos, utility poles, etc., objects like trash bins, tables, benches, trees, and large bushes, and activities like farming and excavation [6]. To quantify the criteria of encroachment, several standards propose that the area within 3 meters (10 feet) from the levee's toe is considered a levee easement area and any illegal structure on this area is an encroachment [6], [7], [8].

Acquiring a large real dataset for semantic segmentation of levee encroachment requires both manpower and time. This includes collecting images and videos of levee systems with encroachments followed by curating and annotating these data. The whole process requires a large capital in terms of human resources and time [9]. For tasks like semantic segmentation, it is highly challenging to collect accurate annotations due to human error and time constraints. Additionally, real datasets, especially those collected from a single source can have low diversity which affects the model's performance and cause overfitting [10].

On the contrary, synthetic dataset can be used to capture real world's data distribution while avoiding the issues with real data. A big challenge in detecting an

encroachment is that an encroachment is not just defined by its shape, size or texture but also by their distant to the levee. For a Computer Vision (CV) model to detect encroachment, this criterion must be reflected in the dataset that the model is trained on. This paper presents a synthetic data generation pipeline using Unity, a game development platform, leveraging virtual 3D environments to model relationship between levee systems and objects surrounding them. More specifically, we designed 3D levee structures in Unity, automatically placed diverse objects (ones that are likely to be found as levee encroachments) and capture samples from the scene. Our contributions are as follows:

- A synthetic data generation framework capable of producing diverse images and pixel accurate masks of levee encroachment using Unity.
- A structured domain randomization suite that ensures diversity in the generated synthetic dataset while modeling encroachment criteria and relationship between the background and foreground objects.
- Benchmark comparison of segmentation models trained on the synthetic vs. real data.
- Empirical evidence supporting that synthetic data outperforms limited real data for this task.

The remainder of this paper is organized as follows. Section II reviews related work on existing levee inspection systems, computer vision architectures used for semantic segmentation of levee deficiencies and synthetic data generation technologies for computer vision. Section III describes our synthetic dataset generation pipeline including scene development, domain randomization, and automated annotation. Section IV presents our experimental setup, evaluation metrics and results comparing models trained on synthetic vs. real data. Section V discusses the implications, limitations and broader applicability of the findings. Section 0 concludes the paper.

II. Related Works

A. Levee Inspection and Monitoring

Levee monitoring is done in numerous ways, including visual inspection, using tools like Levee Inspection Systems, or some geophysical methods like Electrical Resistivity Tomography to identify potential leakage paths. Manual inspection of levee encroachments involves human personnel visiting levee system sites to ensure that no illegal structure is present in the levee easement area.

With the advancement of remote sensing and unmanned aerial vehicle (UAV) technologies, automating the task of inspection and monitoring has gained some traction. Using time series Interferometric Synthetic-Aperture Radar (InSAR) levee deformation can be detected within millimeter (mm) level of precision with low cost and increased monitoring frequency compared to manual inspection [11]. Reference [12] uses Digital Elevation Models (DEMs) to detect levees across land masses which aids flood modeling and risk assessment. Multimodal UAV acquisition allows for better detection and segmentation of deficiencies like crack and seepage by leveraging state of the art Deep Learning (DL) methods [13].

Inspecting levee systems to detect and segment deficiencies in images can be done with deep learning approaches. Reference [5] leverages Convolutional Neural Networks (CNNs) with transfer learning to segment sand boils, one of the most common deficiencies that indicate soil erosion that have high impact on the system's structural integrity [14]. A similar approach can be found for detecting and segmenting levee rutting in inspection images [15]. Reference [2] leverages thermal images from UAVs to detect levee seepages with deep learning methods, more specifically Mask Region-based CNNs. However, not much can be found on automating the task of inspecting levees for encroachments and real dataset to train CV models is too scarce to get reasonable performance in segmentation task.

B. Synthetic Data for Computer Vision

Using synthetic data to capture real world data distribution can help avoid the problems that real data suffer with. Various types of Synthetic Data Generation (SDG) models are used to produce data that are close to their real-life counterpart, and they can be classified by their underlying architecture and the modality of output.

Generative Artificial Intelligence (AI) is a popular method for producing synthetic data and even this field has several sub-methods depending on the architecture type that the model implements. Some methods leverage Autoencoders and Recurrent Neural Networks (RNNs) to generate datasets of handwritten digits, music, video frames and image captions [16], [17], [18], [19]. Generative Adversarial Network (GAN) is another model that's been ubiquitous for SDG, particularly for the tasks of image generation with labels [20], audio generation [21], generating realistic images from hand drawings [22]. The field of Computed Tomography (CT) imagery saw great impact due to GANs and style transfer since it addressed a major issue of privacy that real data faced [23].

Transformer is a popular architecture underlying many generative AI models in the present day. Image Transformer [24] proves that generative models based on self-attention can be used for the task of Super Resolution. Additionally, recent development in diffusion models have made it possible to generate image datasets in applications that lack diversity. Use of diffusion models for both the decoder and prior shows higher quality output with more efficiency [25]. Moreover, generative AI used for SDG use more than just one architecture. For instance, DRAW [17] incorporates architecture from both an RNN and an Autoencoder to generate handwritten digits. Deep Convolutional GAN (DCGAN) [26] implements CNN for unsupervised learning with the GAN framework which can

be utilized to multiple applications ranging from medical image data generation to reconstructing objects from edge maps [27].

A variety of generative AI methods are suitable for SDG. Nevertheless, they come with their own problems that make them unsuitable for generating images of levee encroachments. Firstly, these models suffer from problems that are inherent to their architecture. For example, models that implement GANs encounter issues such as mode collapse and training instability, which is due to the generator being able to exploit shortcuts and difficulty in balancing the discriminator and the generator [28]. Other architectures like transformers and autoencoders have high computational costs which also stems from the architecture itself. But a common issue with these generative AI methods is their requirement of high amount of training data that brings us to the same problem that real data suffers from [29]. Additionally, for the field of levee deficiencies like encroachments, the SGD model needs to have a high level of controllability. While some generative AI models offer this, they are limited to a smaller-scaled features like facial expressions, age of human portraits, shape, angle, and color of objects in the image, or trivial attributes like artistic style [30]. Generating realistic synthetic data of large-scale outdoor scenes with structures like levees, lakes, rivers, canals, etc., and controlling the positions of objects based on their distance to these structure with generative AI is not possible till date, which brings us to explore alternatives like 3D virtual environments.

Simulation software platforms like Blender, Unity and Unreal provide tools to build complex 3D spaces with intricate objects to create 3D virtual environments, along with modeling physics rules to mimic interactions between objects as in real world. With recent advancement of Graphical Processing Units (GPUs), rendering images of a virtual environment is realizable in reasonable amount of time and quality.

3D virtual environments for SGD find their application in CV tasks ranging from low-level to high-level problems. Datasets generated from them are used for optical flow estimation, object tracking, video stabilization, motion analysis, and robot navigation. Synthetic datasets like MPI-Sintel [31] and Monkaa [32] leverage animated movies in Blender to generate optical flow data which were later used for evaluating optical flow estimation algorithms [33]. ShapeNet [34] provides a collection of about 3 million simple objects from online 3D model repositories and contains objects of a wide range of semantic categories like chairs, benches, airplanes, etc. The dataset also provides annotations across a variety of semantic information such as language-related annotations, geometric annotations, functional annotations and physical annotations.

There is a plethora of synthetic dataset that deal with segmentation tasks. SceneNet RGB-D dataset [35] covers 5 million RGB-D image samples from synthetic indoor scenes containing a variety of objects like beds, chairs, sofas, paintings, walls, etc. The authors demonstrated that CNNs pre-trained on their dataset performed better compared to the same models trained on a generic real dataset like ImageNet. Similarly, GTAV dataset [36] and GTA 5 Crowd Counting (GCC) dataset [37] utilize the popular video game Grand Theft Auto V and its realistic virtual environment and objects to generate large synthetic datasets of outdoor scenes for tasks like semantic segmentation, crowd counting, depth estimation, boundary detection, etc.

Similarly, datasets like Virtual KITTI [38] and SYNTHIA [39] use Unity to create a large 3D virtual environments of cities for semantic segmentation of objects on roads to aid autonomous driving agents. The authors utilized the large developer community of Unity and available 3D models in its asset store. To detect blockages in culverts, authors of reference [40] also use Unity to generate synthetic images of culvert openings with both clear and blocked cases. The primary motivation for their use of synthetic dataset was the lack of real one.

Using a simulation software like Unity has a major advantage of being able to attain pixel perfect annotation along with the image. Attaining a similar level of accurate annotations with real images suffers from high cost in terms of time and money [41]. Another advantage is the possibility of Domain Randomization. Since synthetic datasets are generated from a virtual world and cover its data distribution, it can get quite challenging to generalize the model to real data. To solve this problem, the synthetic data distribution can be widened with added diversity through randomness such that the model trained on it can work well on real data distribution [33]. To achieve this, various approaches have been used. Reference [42] randomly tweaks the parameters of simulation such as lighting, pose, textures, etc. even if the results are unrealistic. The authors found that this approach led to better model performance for object detection compared to using only real data. Similar results were attained by references [43] and [44]. While the previous works used unrealistic randomization approaches, structured domain randomization [45] considers the structure and context of the scene and uses a realistic range to tweak the parameters.

While there have been recent developments in the field of SDG, there aren't much work related to the semantic segmentation of levee deficiencies let alone encroachments. This paper is motivated by this research gap along with the fact that the problem of levee encroachments lacks real data.

C. Semantic Segmentation Models

U-Net [46] is a popular deep CNN initially developed for the segmentation of biomedical images. It is based on the encoder-decoder architecture where the encoder is used to extract features through convolution and pooling while contracting the input, and the decoder is used to reconstruct spatial resolution with transposed convolution. The major

reason for its success and significance in the segmentation tasks was its implementation of skip connections which lead to good results with low amount of training data. Expanding on the architecture of U-Net, Attention U-Net [47] added attention gates to the skip connection that added weights to the features extracted by the encoder and emphasized salient regions in the features. Similarly, MultiResUNet [48] improved the U-Net architecture by reducing the semantic gap between encoder and decoder skip connections. The authors did this by using a MultiRes block in place of plain convolution layers that use multiple filter sizes instead of the fixed 3x3 filter size. Secondly, they implemented Res paths instead of direct concatenation in the skip connection which does further processing accounting for the semantic gap between the respective encoder and decoder that the path connects. Lastly, U-Net++ (Nested U-Net) [49] improves the skip connections by adding nested, dense and deeper pathways between encoder and decoder. U-Net and its variations are widely used to detect and segment levee deficiencies from an image.

III. Synthetic Data Generation

Synthetic datasets were generated in Unity using the Perception package [50], an open-source toolkit built on the Unity engine designed for dataset generation. The Randomizer system was used to introduce controlled variation in scene parameters like camera position, object placement, etc. Perception's Semantic Segmentation Labeler was used to generate pixel accurate binary masks with each rendered frame, eliminating the manual annotation cost associated with real datasets. The dataset was exported in SOLO format which organizes the rendered frames and annotation metadata into sequentially numbered folders containing per-frame JSON annotation files. For compatibility with standard segmentation training pipeline, we extracted the RGB images and corresponding segmentation masks from each sequence and reorganized them into a flat paired folder structure with all images stored in a single directory and corresponding masks stored in another directory. The generated datasets are publicly available, see section VII : Data Availability Statement.

A. Scene Development in Unity

To generate the synthetic data with Unity, 3D scenes with levee, a water body like a lake or a river, background objects like houses, roads, etc. were developed. In total 2 distinct scenes were developed, one with a lake and another with a river/canal as the water body, along with varying levee dimensions and 3D models of houses, roads, vegetations, etc. The terrain game object was the fundamental structure used in all the scenes. Unity does not have any units in terms of distance, so 1 unit in the scene was considered to be 1 meter in real life. This enabled the levee structures and 3D models to be in proportion to one another. Due to Unity's restriction on terrain modification below 0 meters altitude, the terrain was elevated by 20 meters to accommodate the underground direction for water bodies. Therefore, in all the scenes, 20 meters is considered to be the ground level. To create the levee structures, brush tool along with *raise* and *smooth* functionality were used. To ensure diversity in levee structures, distinct levee heights of 6 meters and 10 meters were used in different scenes.

The terrains were rendered with diverse grass textures sourced from the Unity Asset Store [51]. Then different 3D models (also sourced from the Unity Asset Store) of houses, roads, poles, fences, and vegetation were strategically placed adjacent to the levee structures to simulate a realistic scenario. The placement of these objects was made to ensure that they would not be considered as encroachments but rather as background objects. The models and their placement strategy used for encroachments are discussed in subsections C and D of section III respectively.

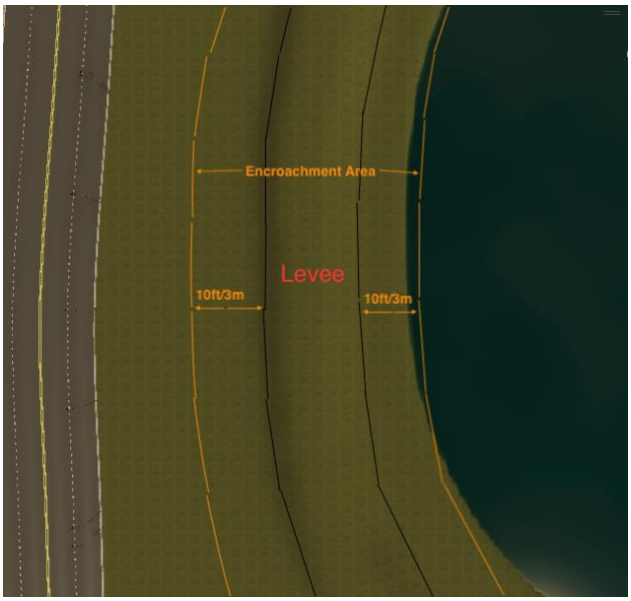


FIGURE 1: TOP VIEW OF A SCENE IN UNITY WITH A LEVEE, A WATER BODY AND A ROAD. LEVEE TOE IS MARKED WITH BLACK LINES AND ENCROACHMENT AREA IS MARKED WITH ORANGE LINE.

B. Encroachment Criteria in the Scene

To model the encroachment criteria in 3D scenes, we adhere to the standard used in real-world scenario. Any structure present on or within 3 meters of the levee toe is considered an encroachment. This criterion is the basis of classification of objects into encroachment or non-encroachment class. Figure 1 demonstrates this criterion and its implementation in the scenes. The automatic labeling pipeline was designed to consider this criterion

when it produces the masks for a rendered frame and is further discussed in subsection E.

C. 3D Models for Levee Encroachment

3D models of different objects prone to be found as levee encroachments were used. Initially, simple objects like benches, walls, fences, trash bins, etc. were developed with Blender, a 3D modeling and rendering application. Utilizing the polygonal modeling feature in Blender, which involves manipulating mesh vertices, edges, and faces complex 3D shapes were created the objects mentioned. To add realism and diversity, multiple materials (even for the same objects) were created. Subsequently, these models were exported in "fbx" format to be integrated to the virtual environments in Unity.

Additionally, a set of 3D models was imported from Unity's asset store. These models represented objects including, but not limited to concrete pipes, trees, shrubs, grasses, gazebo, wooden furniture, residential houses and roads.

D. Domain Randomization

The Perception package offers Randomization class that can be inherited to implement randomization tasks that can occur before, during, or after the simulation. Randomizers were used to set and change parameters such as camera position and rotation, and object position and rotation for each frame.

To sample camera position and rotation, two strategies were implemented. Firstly, a list of camera position and rotations was defined, and a simple integer sampler was used to sample a random position and rotation combination from the list. This approach ensured variance through a simple method. The second approach involved modeling different vehicle behaviors since the dataset tries to replicate real data from inspection vehicles. Three classes of vehicles were modeled: Aerial Vehicles (AV) that include UAVs; Ground Vehicles (GV) that include cars and trucks commonly used for levee inspection; and lastly Surface Vehicles (SV) which includes boats, ships and Unmanned Surface Vehicles (USVs). To model the camera positions and rotations for each vehicle class distinct sets of start and end points were defined in the scene for each class. We then sample multiple points (number defined by the user) between the start and end points for each vehicle and randomly offset these points to add variations in the path. Furthermore, for the height, a normal sampler was used with varying mean and standard deviation values for each vehicle aligning with the real-life specifications of the vehicle class. The number of samples from each class of vehicles was also user-defined.

Similarly, we defined an object placement randomizer to sample positions for the 3D models to be placed in the scene. Two distinct Uniform Samplers were created for the two horizontal axes, while a Constant Sampler was defined for the vertical axis. The limits for the Uniform Samplers and the constant value for the Constant Sampler were user-defined. For each sampled position, Ray Casting was used to determine the elevation of the terrain at that location. This methodology ensured that the 3D models were placed right on the terrain surface, avoiding instances of floating or being situated beneath the ground.

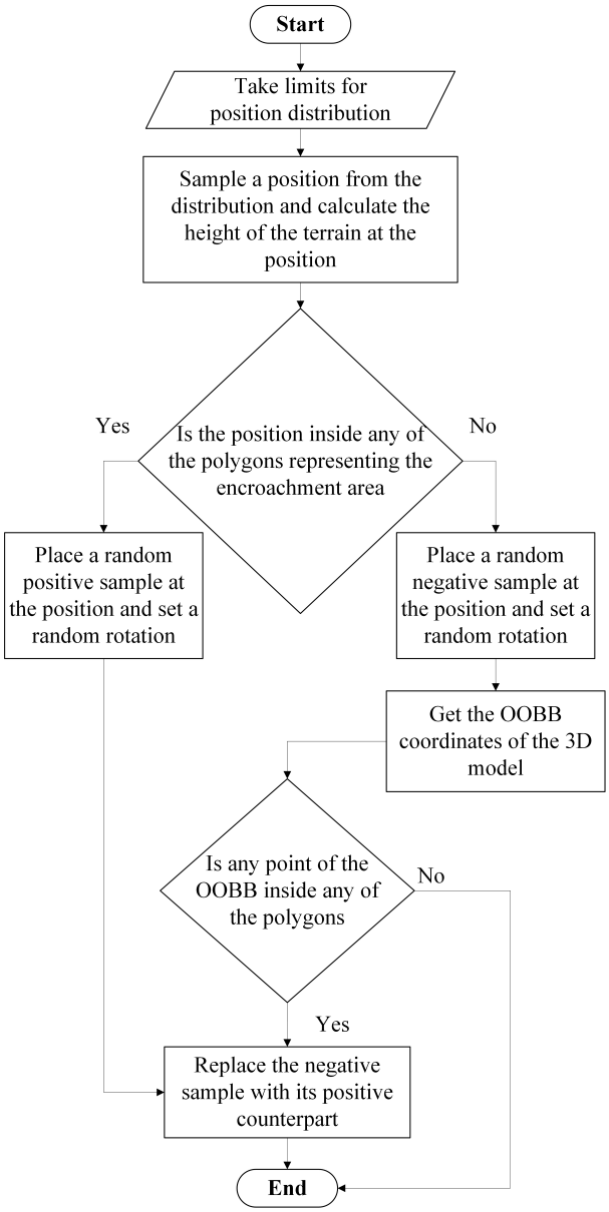


FIGURE 2: FLOW CHART REPRESENTING THE PROCESS OF PLACING AN OBJECT IN THE SCENE, DETERMINING IF IT IS A POSITIVE OR A NEGATIVE SAMPLE, AND REPLACING A NEGATIVE SAMPLE IF ANY PORTION OF THE OBJECT LIES INSIDE THE ENCROACHMENT AREA.

The degree of controllability and structure used in the implemented domain randomization strategy enabled customization and ensured that the generated dataset closely resembled real-world characteristics while

introducing the necessary level of diversity through randomness.

E. Automatic Labeling

Perception package provides different Labelers among which the Semantic Segmentation Labeler was used. A semantic segmentation label config was created that included two different labels: white label for encroachment and green label for levee terrain. After these labels were assigned to the respective 3D models, the Perception Camera component could recognize what objects need to be labeled in the generated mask during rendering. Given that the same object could be classified as an encroachment or not based on its proximity to the levee, two distinct sets of models were created for all the objects: positive samples with the encroachment label associated with the model, and negative samples without encroachment label. While it is straightforward to assign labels to 3D models in Unity, it is challenging to determine which objects placed in the scene are encroachments.

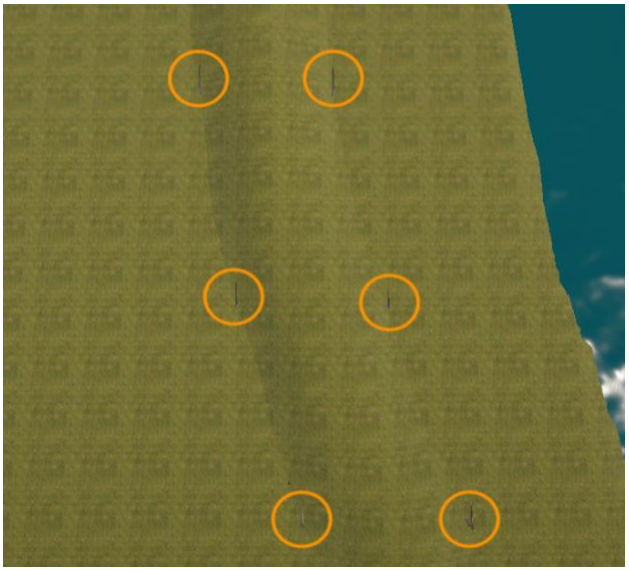


FIGURE 3: TOP VIEW OF A UNITY SCENE WITH LEVEE MARKER MODELS CIRCLED IN ORANGE

Following the sampling of object positions in the scene as outlined in subsection D, the framework evaluates whether each point meets the encroachment criterion. This evaluation was done by providing the randomizer with information regarding the levee's toe through levee marker points. These are discrete points on the toe of the levee (each side) that are uniformly distributed in the scene. A 3D model of lighting pole was used as a visual representation for the user to evaluate the points and make changes if the scenes were modified. Figure 3 shows the top view of a scene with the levee marker models circled in orange. The marker models were disabled during rendering. Once the maker reference was given to the randomizer, their positions were duplicated and offset by 3 meters in the x-axis. Then for each side of the levee, these points were taken to create polygons to represent the encroachment are. The Ray Casting algorithm (or Even-Odd rule) was used to determine whether any object lie inside or outside the polygons. If a sampled position lied inside the polygon, a random positive sample model would be instantiated at that position else a negative sample model would be instantiated.

Given that the randomizer positions 3D models based on their center points, it is possible that a negative sample situated close to the boundary of the encroachment area may have a portion of it inside the area. To check this, the Object-Oriented Bounding Box (OOBB) of the negative sample model placed within the scene was derived from its Axis-Aligned Bounding Box (AABB). Then each point of the OOBB was evaluated to see if they lie inside the polygons defining the encroachment area. If any one of them did, the negative sample was replaced by the exact positive sample conserving the same rotation. The entire process of sampling positions, placing objects and checking if they're encroachment or not is described in the flow chart in Figure 2.

In addition to the binary encroachment/background labeling scheme used for our primary experiment, we also generated a multiclass label variant distinguishing levee structure, encroachment, and background. Details on the methodology used to generate these labels and their results are presented in Appendix A and Appendix B respectively.

F. Lighting and Environment

To have diversity in terms of background, various light and environment setups were developed. These setups differ in parameters such as emission, volumetrics, shadows, and more. To enhance realism, four time of day setups were developed. Two distinct mid-day lighting and environment setups were created to mimic the sun at its highest point. These setups had the highest lux values with neutral color temperature and differed in their directional light appearance. Additionally, two setups for dawn lighting and cloudy lighting were developed introduce additional variance.

In order to optimize rendering time, these setups were used without randomization since randomly changing the setup during simulation adds computational overhead. Consequently, a single simulation run was done using one specific lighting and environment configuration. Subsequently, a different setup was used for the next run and so on.

IV. EXPERIMENTS & RESULTS

Table 1 illustrates the details on the two sets of datasets generated with the two distinct labeling schemes along with their environment and vehicle model setups, and Figure 4 shows some generated dataset samples.

To validate the effectiveness of the synthetic dataset, we trained different CNN models on the generated dataset and compared their performance with the same CNN architectures trained on a limited real dataset. Due to the lack of real images, a real dataset of only 20 images was available among which 16 were used for training and 4 for testing. The use of validation set was for early stopping. All the labels for the real dataset were manually annotated. To keep the comparison impartial, the same 4 real images were used to test both set of models.

The four CNN architectures used were U-Net, Attention U-Net, U-Net++ and Multi Res U-Net. All the models were trained on Nvidia A100 GPUs with a combination of dice loss and binary cross-entropy loss as the loss function. For the optimizer, Adam was used with an initial learning rate of 0.0004 and momentum values of 0.9 and 0.99 along with batch size of 16. Additionally, a reduced learning rate on plateau scheduler was implemented with reduction factor of 0.6 and patience of 6 epochs. Finally, the input size was set to 512 x 512 pixels to save on training time while maintaining details in the input.

TABLE 1: DIFFERENT SETUPS USED TO GENERATE SYNTHETIC DATASET FOR LEVEE ENCROACHMENT

Dataset	Environment	Vehicle Model	Number of samples
Synthetic dataset with labeled encroachment only	Day time 1	AV	100
		GV + SV	100
	Day time 2	AV	100
		GV + SV	100
	Cloudy	AV	100
		GV + SV	100
	Dawn	AV	100
		GV + SV	100
			Total = 800
Synthetic dataset with labeled encroachment and levee	Day time 1	AV	100
		GV + SV	100
	Day time 2	AV	100
		GV + SV	100
	Cloudy	AV	100
		GV + SV	100
	Dawn	AV	100
		GV + SV	100
			Total = 800

The evaluation metrics used for comparison include four complementary methods: Dice coefficient, mean Intersection over Union (mIoU), binary accuracy and balanced accuracy. Dice coefficient and mIoU are standard for semantic segmentation task [52], [53] particularly in domains with class imbalance since pixel-wise accuracy alone is biased towards majority class [54].

Additionally, an ablation study examining the effect of data augmentation on both the real and generated synthetic dataset was performed. 26 augmented variants per original images (both real and synthetic) using a diverse set of geometric, photometric, and noise-based transformations (full list in Appendix D) were generated and all four models were retrained on the augmented real and synthetic datasets.

The test results of all the models are given in Table 2. The *percentage point (pp)* increase is the absolute increase in the metric values for the two compared setups, in this case real vs synthetic as opposed to relative increase that tends to exaggerate the results.

TABLE 2: COMPARISON RESULT OF ALL THE MODELS' EVALUATION METRICS ON REAL VS SYNTHETIC BASE (WITHOUT AUGMENTATION) DATASET ALONG WITH THE PERCENTAGE POINT (PP) INCREASE WITH SYNTHETIC COMPARED TO REAL DATASET

Model	Metric	Real	Synthetic	Increase (pp)
U-Net	Dice	0.2345	0.4843	**+0.2498**
	mIoU	0.4339	0.5064	**+0.0725**
	Bin. A	0.7059	0.7118	**+0.0059**
	Bal. A	0.5719	0.6685	**+0.0966**
Attention U-Net	Dice	0.1866	0.6399	**+0.4533**
	mIoU	0.3658	0.5731	**+0.2073**
	Bin. A	0.6648	0.7426	**+0.0778**
	Bal. A	0.5196	0.7428	**+0.2232**
U-Net++	Dice	0.1133	0.6178	**+0.5045**
	mIoU	0.3396	0.5542	**+0.2146**
	Bin. A	0.6579	0.7336	**+0.0757**
	Bal. A	0.5000	0.7077	**+0.2077**
MultiResUNet	Dice	0.5605	0.6846	**+0.1241**
	mIoU	0.3697	0.6125	**+0.2428**
	Bin. A	0.6678	0.7695	**+0.1017**
	Bal. A	0.4999	0.7581	**+0.2582**

All the models tested performed consistently better on synthetic data compared to real data on all evaluation metrics even when tested against a real test dataset. The largest gains were observed in Dice coefficient (12 to 50 percentage points) and mean IoU (7 to 24 percentage points) indicating substantially improved segmentation quality and boundary precision. Similarly, Table 3 illustrates the percentage point change across all metrics with augmentations on training data for real and synthetic datasets. Synthetic-trained models with augmentations had consistent degradation in performance when compared to those without augmentations while the opposite is true for the real-trained models.

V. DISCUSSION

The use of synthetic dataset resulted in better segmentation performance for levee encroachments compared to real dataset likely due to the limited size and

diversity of the real dataset. The performance gain with the generated dataset supports the viability of synthetic data for detecting and segmenting levee encroachments and broadly for infrastructure monitoring tasks where real annotated data is scarce.

TABLE 3: PERCENTAGE POINT (PP) CHANGE IN ALL EVALUATION METRICS DUE TO AUGMENTING REAL AND SYNTHETIC TRAINING DATASET

Model	Metric	Real Δ (aug-base, pp)	Synthetic Δ (aug-base, pp)
U-Net	Dice	+0.1242	-0.0864
	mIoU	+0.0173	-0.0467
	Bin. A	-0.0254	-0.0269
	Bal. A	+0.0293	-0.0652
Attention U-Net	Dice	+0.0922	-0.0878
	mIoU	+0.0516	-0.0461
	Bin. A	+0.0049	-0.0310
	Bal. A	+0.0444	-0.0551
U-Net ++	Dice	+0.1967	-0.1470
	mIoU	+0.0910	-0.0441
	Bin. A	+0.0181	-0.0001
	Bal. A	+0.0778	-0.0560
MultiResUNet	Dice	-0.1432	-0.1729
	mIoU	+0.1141	-0.1039
	Bin. A	+0.0517	-0.0703
	Bal. A	+0.1177	-0.0848

Additionally, the models trained on the real dataset overfit its limited samples. Training curves (Appendix C) further support this interpretation as the real-trained models exhibit classic overfitting behavior, with validation loss increasing despite continued decrease in training loss throughout the epochs. This was mostly pronounced in U-Net++, likely reflecting its greater parameter capacity compared to the size of the real training dataset. Models also trained for fewer epochs on real data, consistent with early stopping triggered by overfitting. In contrast, synthetic trained models showed more conventional convergence behavior with gradual validation loss decrease suggesting that the synthetic dataset with its larger size and higher diversity enabled more stable generalization during training.

Among the four architectures, MultiResUNet performed the best likely due to its use of residual paths instead of plain skip connections, which improve gradient flow and better reconcile encoder and decoder features across wide range of scales. It is also worth noting that U-Net++ and Attention U-Net saw the most improvement in performance when trained on synthetic dataset compared to real dataset, which supporting further that these more architecturally complex models highly benefit from the larger, more diverse synthetic training dataset. While Dice coefficient, mIoU and balanced accuracy improved substantially with synthetic data, binary accuracy improved only moderately. This was expected due to the imbalanced nature of the pixel classes and background pixels dominating the image area, which resulted in those pixels already being predicted correctly in most cases regardless of the training data.

The ablation study examining the effect of data augmentation revealed an asymmetric pattern between training data sources. Augmentation improved performance for real-trained models in most cases, consistent with the typical role of augmentation in mitigating overfitting on small datasets. However, it reduced performance for synthetic-trained models across all four architectures and four metrics. We attribute this asymmetry to a key difference in how diversity was introduced for each data source. The synthetic dataset gets its diversity from physically grounded domain randomization, whereas the augmentation pipeline applied generic, semantically agnostic transformations including spatial distortion and orientation changes that can produce visually or physically implausible scenes inconsistent with real-world levee geometry. In contrast, the real dataset suffered from severe data scarcity (16 training images), even imperfect augmented variations likely still provided a net benefit by exposing the model to additional variations it lacked. Despite the degradation in performance, synthetic-augmented models continued to outperform their real-trained counterparts in nearly all model-metric combinations, indicating that the core advantage of synthetic dataset for this task is robust even under suboptimal augmentation choices.

While developing the SDG pipeline required substantial upfront investment in scene design and configuration, this was a one-time cost that does not scale with dataset size. Once established, the pipeline can generate additional labeled images at near-zero marginal cost, in contrast to real data collection and annotation, where the cost and time scale roughly linearly with each additional image.

Despite the promising results, it is important to point towards the limitations of this research. Firstly, the generated dataset is modest in scale with only 800 images and was generated from a limited set of Unity scene configurations which may have constrained the diversity of encroachment types, vegetations and terrains represented. Secondly, the study did not explore fine-tuning the synthetic data trained models on the real dataset which could further improve performance by closing the domain gap. Third, the real test set consisted of only 4 images due to the scarcity of annotated dataset. The small test set size means that the results may have been sensitive to the specific characteristics of those images and validation on a larger real test set would strengthen confidence in the findings. Finally, while our SDG pipeline incorporated structured domain randomization across camera and object position and rotation, and lighting and environment

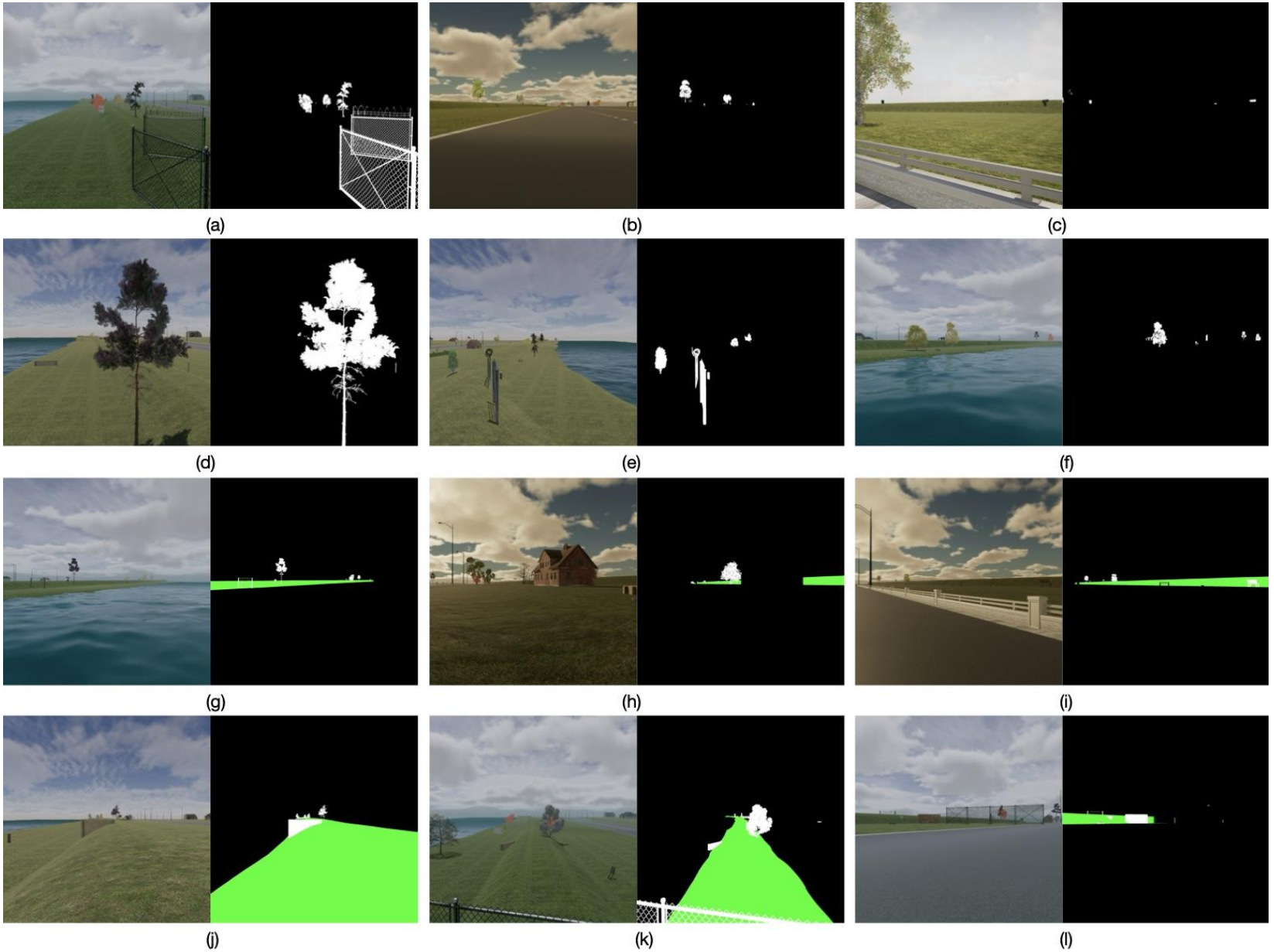


FIGURE 4: SAMPLES FROM THE GENERATED SYNTHETIC DATASET. (A-F): IMAGES AND MASKS WITH ONLY ENCROACHMENT LABELED, (G-L): IMAGES AND MASKS WITH BOTH ENCROACHMENT AND LEVEE LABELED.

parameters, a systematic ablation experiment was outside the scope of this study and would help clarify which variations contribute most to sim-to-real transfer for future pipeline development. A similar study could also be done for the data augmentation pipeline to find if any or a combination of transformations benefit the synthetically generated dataset.

VI. Conclusion

The tasks of semantic segmentation of levee encroachments suffers from lack of real data which is a big challenge in realizing a CV model to be used for automatic levee inspection and monitoring. Moreover, the available dataset lacks the diversity needed to train deep CNNs so that they perform with a reasonable accuracy. We provide a robust SDG pipeline focusing on diverse 3D environments and show that the generated dataset performs better than the limited real dataset available. The comparison evaluation between the synthetic and real dataset was done with U-Net architecture-based models where real images were used for the test dataset supporting that the models trained on the generated synthetic dataset can even perform better on real-world images compared to the limited real dataset.

VII. Data Availability Statement

The synthetic datasets generated and used in this study are publicly available in IEEE DataPort [55], [56].

VIII. References

[1] "Appendix E: NFIP Regulations," *Natl. Flood Insur. Program Regul.*, [Online]. Available: https://www.fema.gov/pdf/floodplain/nfip_sg_appendix_e.pdf

[2] B. Chen, Q. Duan, and L. Luo, "From manual to UAV-based inspection: Efficient detection of levee seepage hazards driven by thermal infrared image and deep learning," *Int. J. Disaster Risk Reduct.*, vol. 114, p. 104982, Nov. 2024, doi: 10.1016/j.ijdrr.2024.104982.

[3] M. Panta *et al.*, "Application of Deep Learning for Segmenting Seepages in Levee Systems," *Remote Sens.*, vol. 16, no. 13, p. 2441, Jan. 2024, doi: 10.3390/rs16132441.

[4] X. Zhao, H. Zhang, P. Wang, Q. Ren, and D. Zhang, "Improving efficiency and accuracy of levee hazard detection with deep learning," *Comput. Geosci.*, vol. 187, p. 105593, May 2024, doi: 10.1016/j.cageo.2024.105593.

[5] M. Panta, Md. T. Hoque, K. N. Niles, J. Tom, M. Abdelguerfi, and M. Falanagin, "Deep Learning Approach for Accurate Segmentation of Sand Boils in Levee Systems," *IEEE Access*, vol. 11, pp. 126263–126282, 2023, doi: 10.1109/ACCESS.2023.3330987.

[6] U.S. Army Corps of Engineers (USACE), "Operating and Maintaining a Levee," *Natl. Levee Saf. Guidel.*, vol. 9.

[7] U.S. Army Corps of Engineers (USACE), "Levee Owner's Manual for Non-Federal Flood Control Works." Mar. 2006. [Online]. Available: https://www.mvr.usace.army.mil/Portals/48/docs/EC/LSP/LeveeOwnersManual.pdf

[8] "Reclamation District No. 2074 - Levee Encroachment Standards," Apr. 2019.

[9] *The National Children's Study 2014: An Assessment*. Washington, D.C.: National Academies Press, 2014, p. 18826. doi: 10.17226/18826.

[10] R. Alfred, J. Leo, and S. F. Kaijage, "Optimizing dataset diversity for a robust deep-learning model in rice blast disease identification to enhance crop health assessment across diverse conditions," *Smart Agric. Technol.*, vol. 10, p. 100726, Mar. 2025, doi: 10.1016/j.atech.2024.100726.

[11] I. E. Özer, F. J. van Leijen, S. N. Jonkman, and R. F. Hanssen, "Applicability of satellite radar imaging to monitor the conditions of levees," *J. Flood Risk Manag.*, vol. 12, no. S2, p. e12509, 2019, doi: 10.1111/jfr3.12509.

[12] D. N. Khanh *et al.*, "Mapping the World's River Levees: A Hyper-Resolution Levee Database Based on Digital Elevation Models," *Geophys. Res. Lett.*, vol. 52, no. 11, p. e2024GL114121, 2025, doi: 10.1029/2024GL114121.

[13] H. Gu *et al.*, "Visible-infrared imagery and deep learning for UAV-based automated levee hazard detection," *Comput.-Aided Civ. Infrastruct. Eng.*, vol. 47, p. 100081, Jul. 2026, doi: 10.1016/j.cacaie.2026.100081.

[14] J. A. Schaefer, T. M. O'Leary, and B. A. Robbins, "Assessing the Implications of Sand Boils for Backward Erosion Piping Risk," pp. 124–136, Jun. 2017, doi: 10.1061/9780784480724.012.

[15] Padam Jung Thapa, "Toward Robust Semantic Segmentation in Levee Infrastructure Toward Robust Semantic Segmentation in Levee Infrastructure Monitoring: Enhancing Accuracy with High-Fidelity Synthetic Data Monitoring: Enhancing Accuracy with High-Fidelity Synthetic Data and Ensemble Learning," *Univ. New Orleans Theses Diss.*, 2025, [Online]. Available: https://scholarworks.uno.edu/td/3231

[16] Y. Bengio, L. Yao, G. Alain, and P. Vincent, "Generalized Denoising Auto-Encoders as Generative Models," Nov. 11, 2013, *arXiv*: arXiv:1305.6663. doi: 10.48550/arXiv.1305.6663.

[17] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "DRAW: A Recurrent Neural Network For Image Generation," May 20, 2015, *arXiv*: arXiv:1502.04623. doi: 10.48550/arXiv.1502.04623.

[18] M. Germain, K. Gregor, I. Murray, and H. Larochelle, "MADE: Masked Autoencoder for Distribution Estimation," Jun. 05, 2015, *arXiv*: arXiv:1502.03509. doi: 10.48550/arXiv.1502.03509.

[19] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription," Jun. 27, 2012, *arXiv*: arXiv:1206.6392. doi: 10.48550/arXiv.1206.6392.

[20] M. Frid-Adar, E. Klang, M. Amitai, J. Goldberger, and H. Greenspan, "Synthetic Data Augmentation using GAN for Improved Liver Lesion Classification," Jan. 08, 2018, *arXiv*: arXiv:1801.02385. doi: 10.48550/arXiv.1801.02385.

[21] C. Donahue, J. McAuley, and M. Puckette, "Adversarial Audio Synthesis," Feb. 09, 2019, *arXiv*: arXiv:1802.04208. doi: 10.48550/arXiv.1802.04208.

[22] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros, "Generative Visual Manipulation on the Natural Image Manifold," Dec. 16, 2018, *arXiv*: arXiv:1609.03552. doi: 10.48550/arXiv.1609.03552.

[23] A. Krishna and K. Mueller, "Medical (CT) image generation with style," in *15th International Meeting on Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine*, S. Matej and S. D. Metzler, Eds., Philadelphia, United States: SPIE, May 2019, p. 101. doi: 10.1117/12.2534903.

[24] N. Parmar *et al.*, "Image Transformer," Jun. 15, 2018, *arXiv*: arXiv:1802.05751. doi: 10.48550/arXiv.1802.05751.

[25] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical Text-Conditional Image Generation with CLIP Latents," Apr. 13, 2022, *arXiv*: arXiv:2204.06125. doi: 10.48550/arXiv.2204.06125.

[26] A. Radford, L. Metz, and S. Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," Jan. 07, 2016, *arXiv*: arXiv:1511.06434. doi: 10.48550/arXiv.1511.06434.

[27] A. Bauer *et al.*, "Comprehensive Exploration of Synthetic Data Generation: A Survey," Feb. 01, 2024, *arXiv*: arXiv:2401.02524. doi: 10.48550/arXiv.2401.02524.

[28] T. Miyato and M. Koyama, "cGANs with Projection Discriminator," Aug. 15, 2018, *arXiv*: arXiv:1802.05637. doi: 10.48550/arXiv.1802.05637.

[29] M. Goyal and Q. H. Mahmoud, "A Systematic Review of Synthetic Data Generation Techniques Using Generative AI," *Electronics*, vol. 13, no. 17, p. 3509, Jan. 2024, doi: 10.3390/electronics13173509.

[30] S. Wang, Y. Du, X. Guo, B. Pan, Z. Qin, and L. Zhao, "Controllable Data Generation by Deep Learning: A Review," *ACM Comput. Surv.*, vol. 56, no. 9, pp. 1–38, Oct. 2024, doi: 10.1145/3648609.

[31] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A Naturalistic Open Source Movie for Optical Flow Evaluation," in *Computer Vision – ECCV 2012*, vol. 7577, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., in Lecture Notes in Computer Science, vol. 7577. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 611–625. doi: 10.1007/978-3-642-33783-3_44.

[32] N. Mayer *et al.*, "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 4040–4048. doi: 10.1109/CVPR.2016.438.

[33] S. Nikolenko, "Synthetic Data for Deep Learning," Sep. 25, 2019, *arXiv*: arXiv:1909.11512. doi: 10.48550/arXiv.1909.11512.

[34] A. X. Chang *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Dec. 09, 2015, *arXiv*: arXiv:1512.03012. doi: 10.48550/arXiv.1512.03012.

[35] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: Can 5M Synthetic Images Beat Generic ImageNet Pre-training on Indoor Segmentation?," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice: IEEE, Oct. 2017, pp. 2697–2706. doi: 10.1109/ICCV.2017.292.

[36] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," Aug. 07, 2016, *arXiv*: arXiv:1608.02192. doi: 10.48550/arXiv.1608.02192.

[37] Q. Wang, J. Gao, W. Lin, and Y. Yuan, "Learning from Synthetic Data for Crowd Counting in the Wild," Mar. 08, 2019, *arXiv*: arXiv:1903.03303. doi: 10.48550/arXiv.1903.03303.

[38] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "Virtual Worlds as Proxy for Multi-Object Tracking Analysis," May 20, 2016, *arXiv*: arXiv:1605.06457. doi: 10.48550/arXiv.1605.06457.

[39] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, USA: IEEE, Jun. 2016, pp. 3234–3243. doi: 10.1109/CVPR.2016.352.

[40] U. Iqbal, J. Barthelemy, W. Li, and P. Perez, "Automating Visual Blockage Classification of Culverts with Deep Learning," *Appl. Sci.*, vol. 11, no. 16, p. 7561, Aug. 2021, doi: 10.3390/app11167561.

[41] S. Vadineanu, D. M. Pelt, O. Dzyubachyk, and K. J. Batenburg, "Reducing Manual Annotation Costs for Cell Segmentation by Upgrading Low-Quality Annotations," *J. Imaging*, vol. 10, no. 7, p. 172, Jul. 2024, doi: 10.3390/jimaging10070172.

[42] J. Tremblay *et al.*, "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Salt Lake City, UT: IEEE, Jun. 2018, pp. 1082–10828. doi: 10.1109/CVPRW.2018.00143.
[43] J. Borrego, A. Dehban, R. Figueiredo, P. Moreno, A. Bernardino, and J. Santos-Victor, "Applying Domain Randomization to Synthetic Data for Object Category Detection," Jul. 16, 2018, *arXiv*: arXiv:1807.09834. doi: 10.48550/arXiv.1807.09834.
[44] J. Tobin *et al.*, "Domain Randomization and Generative Models for Robotic Grasping," Apr. 03, 2018, *arXiv*: arXiv:1710.06425. doi: 10.48550/arXiv.1710.06425.
[45] A. Prakash *et al.*, "Structured Domain Randomization: Bridging the Reality Gap by Context-Aware Synthetic Data," Aug. 18, 2020, *arXiv*: arXiv:1810.10093. doi: 10.48550/arXiv.1810.10093.
[46] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," May 19, 2015, *arXiv*: arXiv:1505.04597. doi: 10.48550/arXiv.1505.04597.
[47] O. Oktay *et al.*, "Attention U-Net: Learning Where to Look for the Pancreas," May 20, 2018, *arXiv*: arXiv:1804.03999. doi: 10.48550/arXiv.1804.03999.
[48] N. Ibtehaz and M. S. Rahman, "MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation," *Neural Netw.*, vol. 121, pp. 74–87, Jan. 2020, doi: 10.1016/j.neunet.2019.08.025.
[49] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, and J. Liang, "UNet++: A Nested U-Net Architecture for Medical Image Segmentation," Jul. 18, 2018, *arXiv*: arXiv:1807.10165. doi: 10.48550/arXiv.1807.10165.
[50] Unity Technologies, *Unity Perception Package*. (2020). [Online]. Available: https://github.com/Unity-Technologies/com.unity.perception
[51] "Unity Asset Store." Accessed: Oct. 15, 2025. [Online]. Available: https://assetstore.unity.com
[52] Z. Wang and B. Dai, "RankSEG-RMA: An Efficient Segmentation Algorithm via Reciprocal Moment Approximation," Oct. 17, 2025, *arXiv*: arXiv:2510.15362. doi: 10.48550/arXiv.2510.15362.
[53] D. Jha *et al.*, "Kvasir-SEG: A Segmented Polyp Dataset," Nov. 16, 2019, *arXiv*: arXiv:1911.07069. doi: 10.48550/arXiv.1911.07069.
[54] A. A. Taha and A. Hanbury, "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool," *BMC Med. Imaging*, vol. 15, p. 29, Aug. 2015, doi: 10.1186/s12880-015-0068-x.
[55] A. Katwal, "Synthetic Levee Encroachments Dataset - Binary Segmentation (Unity Generated)." IEEE DataPort. doi: 10.21227/5WB9-QG93.
[56] A. Katwal, "Synthetic Levee Encroachments Dataset - Multi-Class Segmentation (Unity Generated)." IEEE DataPort. doi: 10.21227/M80X-7E94.
[57] A. Buslaev, A. Parinov, E. Khvedchenya, V. I. Iglovikov, and A. A. Kalinin, "Albumentations: fast and flexible image augmentations," *Information*, vol. 11, no. 2, p. 125, Feb. 2020, doi: 10.3390/info11020125.

APPENDIX A

Levee Labeling Methodology

Unlike the encroachment objects in the scene that are distinct 3D models and apart from the terrain or other game objects, the levee structure is not a single model. It is a part of the ground terrain game object that was manipulated in the vertical axis with the terrain tools to create an elongated hump along the border of ground and the water body. Therefore, assigning the levee label to the whole terrain game object means that the masks would have both the levee, and the ground labeled as a levee. As per the standard defined in subsection II-A, we have defined 20 meters to be the ground altitude and the terrain above this altitude is the levee. So, in order to correctly label a levee, the terrain section above 20 meters had to be separated from the terrain section at and below 20 meters.

For convenience and reusability, we created a separate window in Unity where the user could select the terrain game object and divide it into two different terrains: a levee terrain and a non-levee terrain. This was achieved with a C# class that inherits from the Editor Window class that

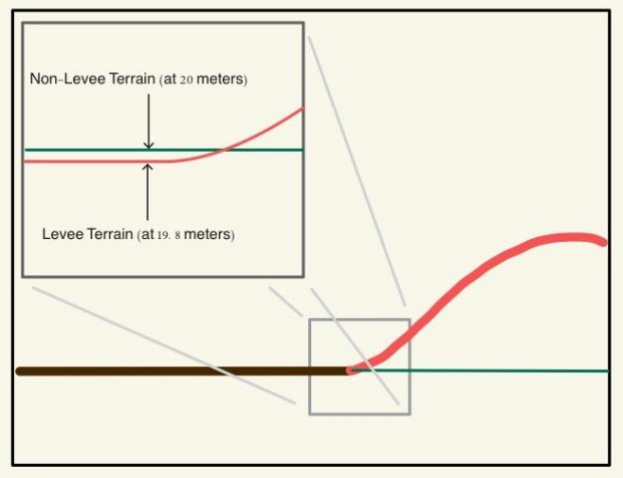


FIGURE A-1: TERRAIN CROSS-SECTION DEMONSTRATING THE DIFFERENCE BETWEEN LEVEE AND NON-LEVEE TERRAINS.

offers several features to build a custom Graphical User Interface (GUI) in Unity. A script was then developed in C# that could take the selected terrain game object and extract all the information about the terrain, more

specifically the height map. The height map was then normalized into the range of 0-1 and the terrain was split into two parts from a threshold height of 20 meters, which was also normalized based on the factor used for the height map normalization. The height values above the threshold were used to construct the levee terrain and the height values below the threshold were used to construct the non-levee terrain. To preserve the previous details like textures, the extracted information from the original terrain game object was used.

Figure A-2 shows the flow chart of the custom window developed to split the levee terrain game object. It is worth noting that while constructing the non-levee terrain the terrain height at all the location where the height was above threshold was set to the threshold height of 20 meters, while for the levee terrain all the location where the height was at and below the threshold height the new height was set slightly lower than the 20 meters threshold height of 19.8 meters. This was done so that when the labeler was assigned to the levee terrain, the non-levee terrain would occlude the levee terrain on ground areas in the generated masks, and only the terrain above 20 meters would be labeled as levee. This subtle difference that generates clean masks of levee is demonstrated in Figure A-1.

The levee labeler combined with the encroachment labeler enables us to get multiclass labels in the generated masks. As shown in Figure 4, the multiclass masks have black colored pixels representing background class, white colored pixels representing encroachment class and green colored pixels representing levee class.

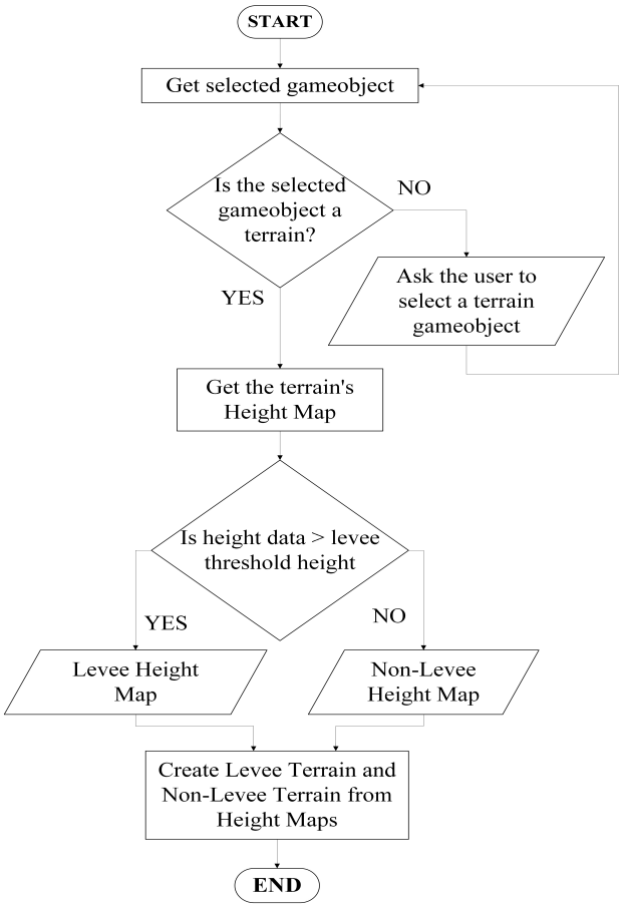


FIGURE A-2: FLOW CHART FOR SEPARATING A TERRAIN GAME OBJECT INTO LEVEE AND NON-LEVEE TERRAINS

APPENDIX B

Multiclass experiment results

To train the models with multiclass labels (background, encroachment and levee), the final layer of all the models was modified to output probabilities with three different classes. These modifications include adding 2 more filters in their final layer, making the total the number 3 and changing the activation function in the final layer from Sigmoid to SoftMax. The SoftMax function ensures that the probabilities of each pixel belonging to one of the 3 classes have a sum of 1. Additionally, the loss function was modified to have a combined loss of Dice loss and Sparse Categorical Entropy as opposed to the combination of Dice loss and Binary Cross Entropy used for binary class segmentation. While the same metrics were used, they were modified to take multiclass inputs.

Table B- gives a comparison of the models trained on synthetic multiclass labels with and without augmentations. The same augmentations used for binary class segmentation were used for multiclass segmentation as well and are described in

Table D-1: All the transformations applied to the real and synthetic images for augmentation.

TABLE B-1: EVALUATION RESULT OF MODELS TRAINED ON SYNTHETIC MULTICLASS SEGMENTATION DATASET WITH AND WITHOUT AUGMENTATIONS

Models	Metrics	Base multiclass dataset	Augmented multiclass dataset
U-Net	Dice	0.3769	0.3645
	mIoU	0.3003	0.2985
	Bin. A.	0.6655	0.7342
	Bal. A.	0.5622	0.5459
Attention U-Net	Dice	0.3783	0.3747
	mIoU	0.2903	0.3039
	Bin. A.	0.6669	0.7472
	Bal. A.	0.5496	0.5337
U-Net++	Dice	0.3956	0.3762
	mIoU	0.3083	0.3089
	Bin. A.	0.6761	0.7425
	Bal. A.	0.5692	0.5514
MultiResUNet	Dice	0.4144	0.3982
	mIoU	0.3301	0.3172
	Bin. A.	0.6547	0.7659
	Bal. A.	0.5892	0.5480

APPENDIX C

Training Curves

The training curves of all the models trained on synthetic and real datasets of levee encroachments are shown in Figure C-1 and Figure C-2 respectively. Models trained on synthetic data show a more conventional training trend with validation loss decreasing gradually alongside training loss. In contrast, models trained on real data show validation loss that increases as training loss continues to decrease indicating overfitting, consistent with the limited size of the real training set. Real-trained models also converged over substantially fewer epochs than synthetic-trained data due to overfitting that triggered early stopping.

Synthetic-trained models show some spikes in validation loss during training which decrease in amplitude as the training progresses and stabilize toward later epochs. We attribute these spikes to the structured domain randomization used in synthetic dataset generation since validation batches may contain varying combinations of randomized lighting, viewpoint and scene compositions. Validation loss can fluctuate epoch-to-epoch depending on which combinations of randomized parameters are represented in a given batch even as the model's overall generalization continues to improve.

APPENDIX D

Augmentation ablation study details

The paper also focuses on other attempts of expanding a dataset's distribution, more specifically the method of augmenting an image with various transformations. We performed an offline augmentation on both the real and synthetic dataset using the Albumentations library [57] . In total 26 different transformations and combinations were done on a single original image. All the transformations and their details are provided in Table D-1. This expanded the real training dataset from 16 to 432 samples and the synthetic training dataset from 800 to 21,600 samples. Unless otherwise noted transformations were applied with probability 1.0; two exceptions were Random Grid Shuffle (p=0.1) and the combined transformation block, which applies a sequence of transforms with individually specified probabilities. Grayscale conversion was performed directly via OpenCV color-space conversion rather than an Albumentations transform. Lastly, all output images and masks were resized to fixed resolution of 512x512 prior to saving. Figure D-1 shows the samples of each transformation used in the augmentation pipeline.

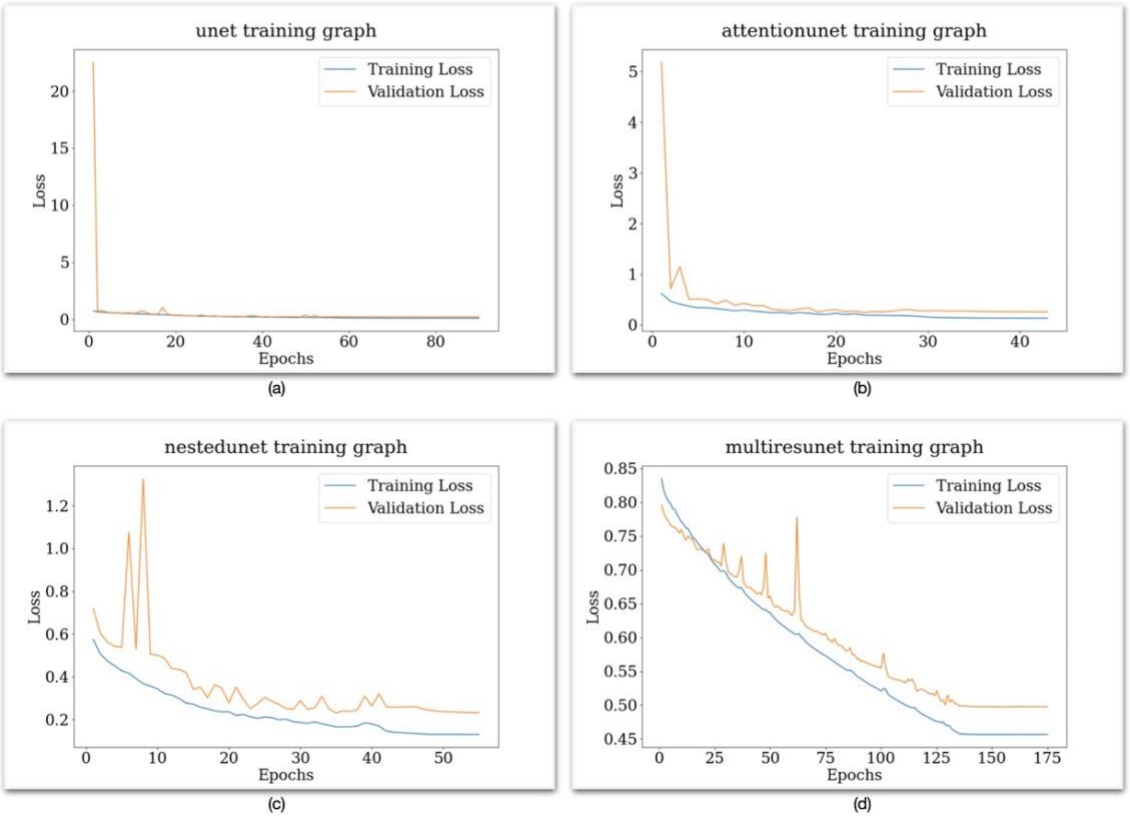


FIGURE C-1: TRAINING CURVES OF MODELS TRAINED ON BINARY SEGMENTATION SYNTHETIC DATASET OF LEVEE ENCROACHMENTS

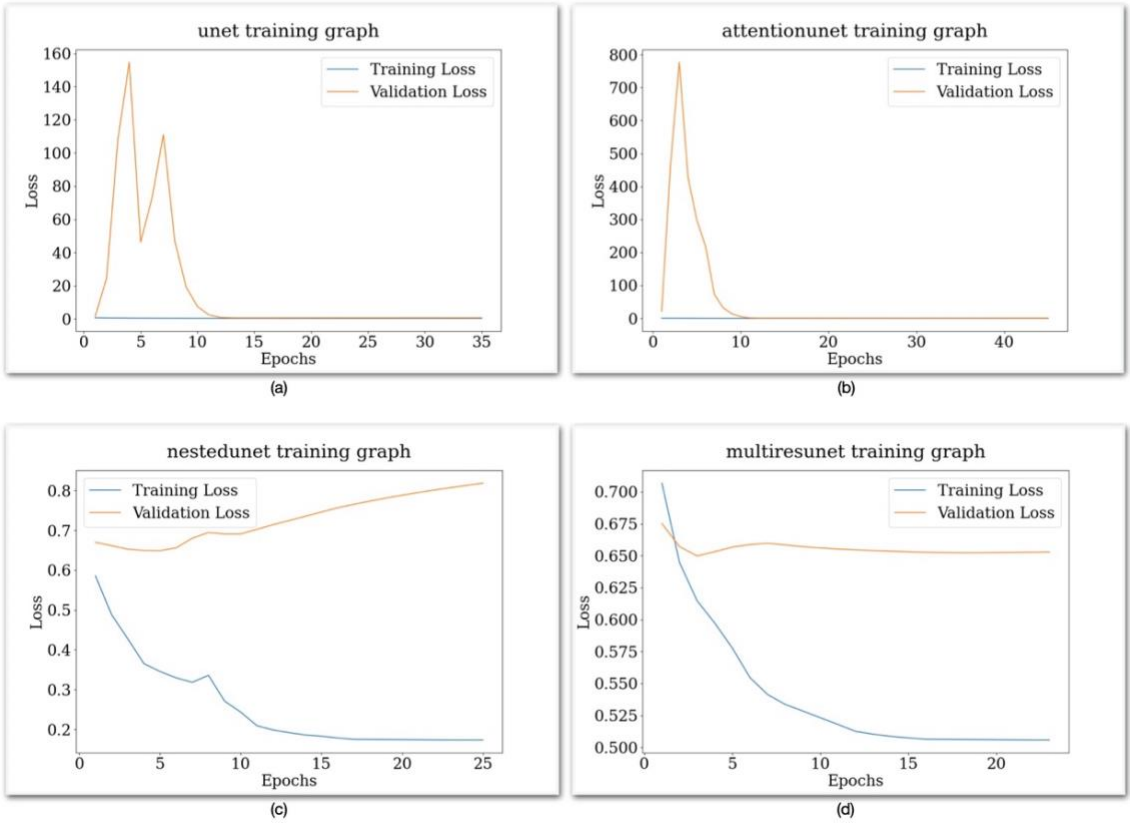


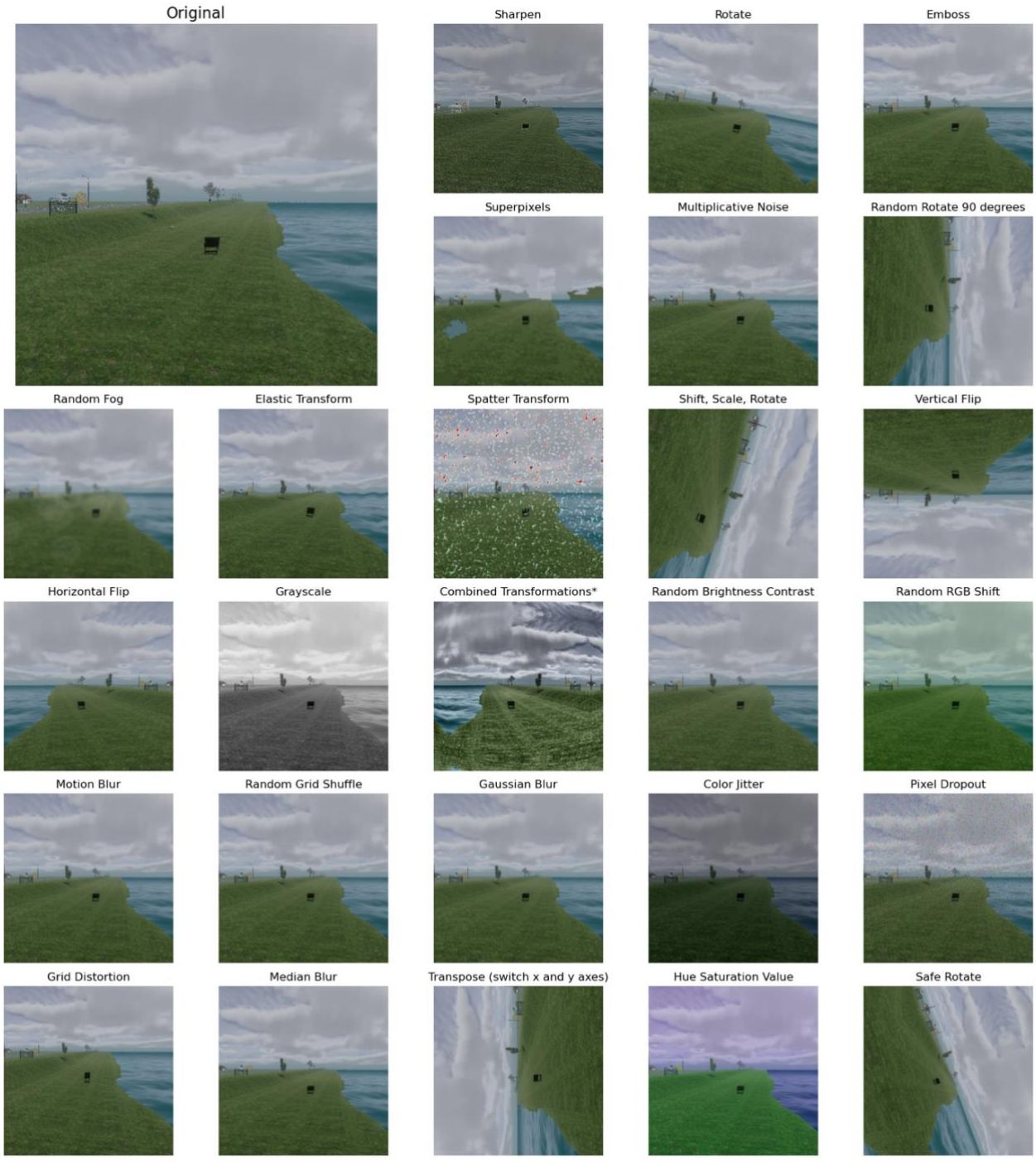
FIGURE C-2: TRAINING CURVES OF MODELS TRAINED ON BINARY SEGMENTATION REAL DATASET OF LEVEE ENCROACHMENT

TABLE D-1: ALL THE TRANSFORMATIONS APPLIED TO THE REAL AND SYNTHETIC IMAGES FOR AUGMENTATION

No.	Transformation	Description	Parameters
1.	Sharpen	Increases edge contrast	alpha=(0.4-0.7, lightness=(0.5-1.0)
2.	Rotate	Rotates image by a random angle	limit = ±60°
3.	Emboss	Applies embossing filter	default
4.	Superpixels	Replaces regions with superpixel averages	default
5.	Multiplicative Noise	Multiplies pixel values by random noise factor	per_channel=True, elementwise=True
6.	Random Rotate 90	Rotates images by a random multiple of 90°	default
7.	Random Fog	Simulates fog effect	fog_coef=(0.4-0.7)
8.	Elastic Transform	Applies elastic spatial distortion	alpha=120, sigma=120x0.08, alpha_affine=120x0.05
9.	Spatter	Simulates rain/mud spatter effect	default
10.	Shift, Scale Rotate	Combined affine transform	default
11.	Vertical Flip	Flips image along horizontal axis	default
12.	Horizontal Flip	Flips image along vertical axis	default
13.	Grayscale	Converts image to grayscale	via cv2.cvtColor
14.	Combined Transform	Sequential: Vertical flip, Random Rotate 90, one of {Elastic Transform, Grid Distortion, Optical Distortion}, CLAHE, Random Brightness/Contrast, Random Gamma	*See note below
15.	Random Brightness Contrast	Randomly adjusts brightness and contrast	brightness_limit=0.3, contrast_limit=0.3, brightness_by_max=True

16.	Random RGB shift	Randomly shifts RGB channel values independently	r_shift=25, g_shift=25, b_shift=20
17.	Motion Blur	Simulates motion blur	blur_limit=3
18.	Random Grid Shuffle	Randomly shuffles grid cells of the image	grid=(16, 16), p=0.1
19.	Gaussian Blur	Applies Gaussian blur	blur_limit=3, sigma_limit=0.8
20.	Color Jitter	Randomly perturbs brightness, contrast, saturation, hue	brightness=0.4, contrast=0.3, saturation=0.3, hue=0.4
21.	Pixel Dropout	Randomly sets pixels to zero	dropout_prob=0.04, per_channel=True
22.	Grid Distortion	Distorts image along a grid pattern	default
23.	Median Blur	Applies median filtering	blur_limit=5
24.	Transpose	Swaps image rows and columns (x/y axes)	default
25.	Hue Saturation Value	Randomly shifts hue, saturation, value channels	default
26.	Safe Rotate	Rotates image while preserving full content (no cropping/border loss)	limit=±75°

*Note: For the combined transform, the sub-transform probabilities are: Vertical Flip (p=0.5), Random Rotate 90 (p=0.5), one of three distortions (p=0.8, internally Elastic Transform p=0.5 / Grid Distortion p=0.5 / Optical Distortion p=1), CLAHE (p=0.8), Random Brightness Contrast (p=0.8), Random Gamma (p=0.8).



* Combined transformations of vertical flip, random rotate 90 degrees, one of (elastic transform, grid distortion, optical distortion), CLAHE, random brightness contrast, random gamma

FIGURE D-1: SAMPLES OF ALL THE TRANSFORMATIONS APPLIED TO THE REAL AND SYNTHETIC IMAGES FOR DATA AUGMENTATION