IEEE Bigdata Cup 2024: Building Extraction Generalization Challenge (BEGC2024)

Team: DoubleY (Yi Jie Wong, Yin Loon Khor)

1. Description of Our Method and Tools Used:

Model Selection

YOLO series has been the *de facto* object detection model for real-time applications [1], [2], due to its inherently high inference speed and accuracy. With its latest instalment, the YOLO series is now equipped with instance segmentation capability, making it a suitable candidate for this work. Specifically, YOLOv8-seg employs an enhanced version of CSPDarknet53 as its backbone, incorporating cross-stage partial (CSP) connections to improve information flow between layers. The head consists of a series of convolutional layers that process feature maps from P3, P4, and P5 of the backbone. YOLOv8-seg also replaces the traditional YOLO neck with the C2f module, which extracts features at three distinct scales. Finally, three segmentation heads are used to identify objects of varying sizes—small, medium, and large—by leveraging the different feature sets. These heads use fully connected layers and anchor-free detection, allowing the model to adapt more effectively to various object shapes and sizes. Note that the choice of instance segmentation model does not affect the results of our method, as the our proposed approach is more data-centric rather than model-centric.

Transfer Learning

Next, we also perform a simple ablation study to validate the impact of transfer learning on training the building extraction model. Specifically, we compare the F1 score of the model in three settings: (i) **without pretrained weights**, (ii) with **COCO-segmentation pretrained weights**, and (iii) with **DOTAv1 Aerial Detection's pretrained weights**. Note that we only experiment with these three setups, as these pretrained weights are more accessible via Ultralytics' YOLOv8 framework [3].

Extra Dataset

We leverage the Microsoft Building Footprint (BF) dataset, a large-scale dataset containing 1.4 billion building footprints from around the world [4]. Note that we avoid using Japan building footprint data from the Microsoft BF dataset to prevent data leakage, as the aim of the challenge is to test the generalizability of our model with limited data from Japan. We extract building footprints from two regions: (1) **Redmond, Washington**, and (2) **Las Vegas, Nevada**. For simplicity's sake, we refer to the former as the Redmond Dataset and the latter as the Las Vegas Dataset. For the Redmond Dataset, we set the longitudinal range to be -122.16484503187519 to -122.06529607517405, and the latitudinal range to be 47.69090474454916 to 47.6217555345674. Meanwhile, we set the longitudinal range to be -115.31432742408262 to -114.98257204779121, and the latitudinal range to be 36.00372747612303 to 36.27297250862463 for the Las Vegas Dataset. Next, we randomly extract 3,000 satellite images from each region and acquire the corresponding building footprints as labels. With this, we ended up with two extra datasets which can be used along with the provided Building Extraction Generalization Challenge (**BEGC2024**) data as the training dataset. Figure 1 and 2 show examples of images sampled for Redmond dataset and Las Vegas dataset, respectively.

Diffusion Augmentation

In the previous section, we discussed how we leveraged existing open-source datasets to improve the diversity of the training dataset for our YOLOv8 segmentation model. However, such a method is not label-efficient and might fail when there is no suitable dataset available. In fact, our empirical results in Table III show that not all extra datasets might be useful for training. Hence, we propose to leverage diffusion augmentation, which is a technique that uses diffusion models to produce realistic images and enhance the diversity of the training dataset. Our goal is to use only the BEGC2024 training dataset to train a diffusion model that can generate realistic data as a form of augmentation, without requiring additional datasets. This method guarantees the label-efficiency of the training.

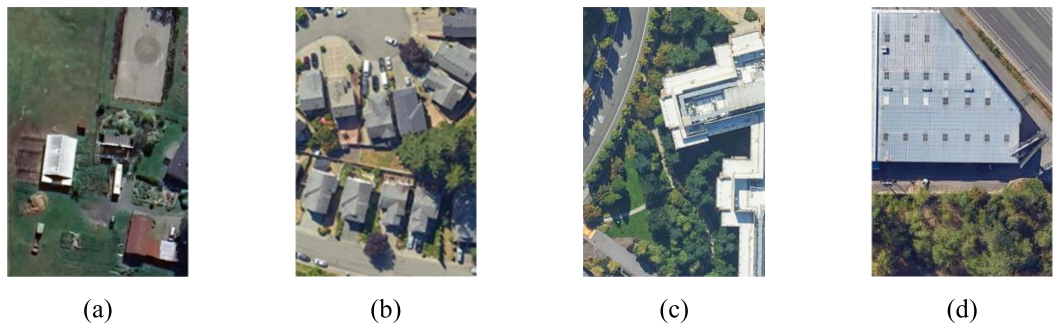

Fig 1. Example of images extracted from Microsoft BF dataset, sampled around Redmond, Washington.

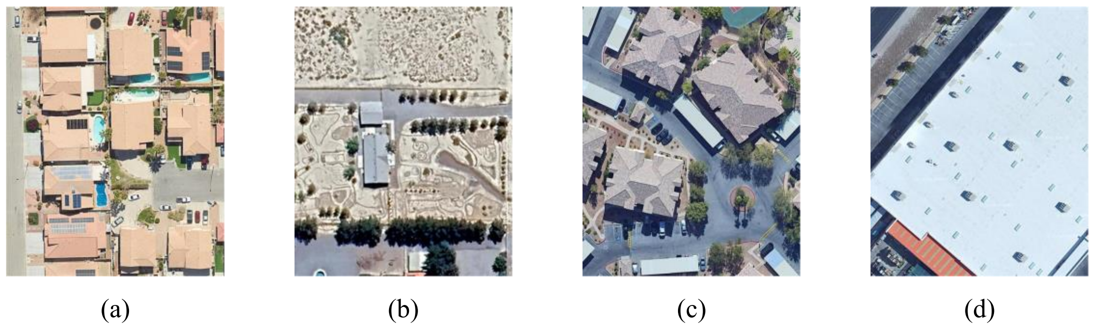

Fig 2. Example of images extracted from Microsoft BF dataset, sampled around Las Vegas, Nevada.

Figure 3 illustrates our diffusion augmentation pipeline. Firstly, we extract the segmentation labels (in polygon format) from the BEGC2024 training set and convert them into segmentation masks. Next, we utilize an open-source land-cover semantic segmentation model, trained on the LoveDA dataset [5], to perform segmentation on the BEGC2024 training images. Although the predicted segmentation masks may contain inaccuracies, our primary concern is the accurate identification of building locations. The remaining land cover classes serve to enhance the realism of the synthetic images. Therefore, we refine the predicted semantic segmentation masks using the building footprints from the BEGC2024 polygon labels. This process yields a usable segmentation mask, which can then be employed to train our diffusion model. Lastly, we train a segmentation-guided diffusion models which take the semantic segmentation mask as input, and generate a realistic image which follows the mask, similar to [6]. Specifically, we uses a Denoising Diffusion Implicit Models (DDIM) which takes the target image concatenated with the segmentation mask at each denoising step.

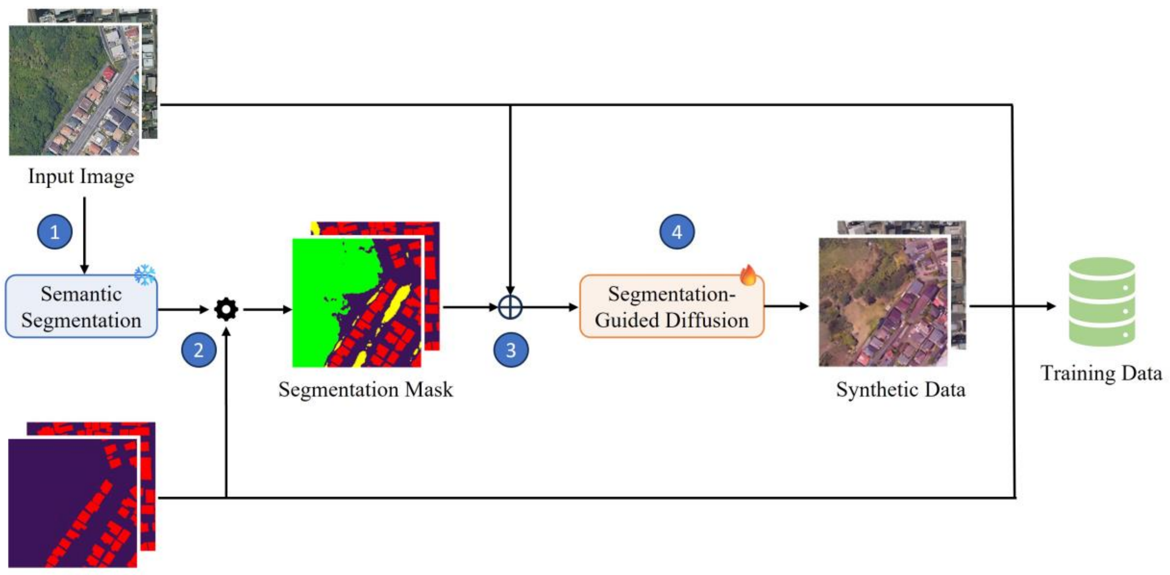

Fig. 3. The proposed diffusion augmentation pipeline. (1) Use pretrained segmentation model to generate semantic segmentation. (2) Refine the segmentation mask using the building polygon labels. (3) Concatenate input image with the semantic mask. (4) Train the segmentation-guided diffusion model using the concatenated inputs.

2. Evaluation of Our Method:

The YOLOv8 series includes several instance segmentation models, ranging from the smallest nano (n) variant to the largest extra-large (x) variant. We conducted multiple experiments to select the best YOLOv8 variant for our task, considering both the F1 score and model complexity. Additionally, we compared the performance of YOLOv8-based instance segmentation models with other state-of-the-art models, including YOLOv9, Mask R-CNN, and EfficientNet. All models were trained for 50 epochs with a 640 image size. During testing and submission, the confidence and NMS IoU thresholds were set to 0.20 and 0.70, unless stated otherwise. We also evaluated the F1-score of the models with a confidence threshold of 0.50, primarily to assess the models' confidence rather than for actual submission. After selecting the best model, we experimented with training the model by varying the pretrained weights and the training dataset.

3. Detailed Analysis of Results:

Model Comparison

Table I shows the comparison of different models in term of F1-score and model complexity. Note that we do not use other tricks (i.e. additional training dataset or diffusion augmentation) at this stage, as we keep the ablation study for these tricks in the following sections. Generally, we observe that the F1 score increases when scaling up the model from the smallest YOLOv8n-seg to the medium size YOLOv8m-seg. Notably, there is a significant jump in F1 score from the YOLOv8s-seg to the YOLOv8m-seg when evaluated in confidence threshold of 0.50, with the score improving from 0.535 to 0.592. Interestingly, the F1 score of YOLOv8m-seg is slightly lower than the smaller YOLOv8s-seg when setting the confidence threshold to 0.20. This observation suggest that the m-variant still has a bigger room of improvement compared to s-variant. Meanwhile, the largest YOLOv8x-seg variant has a lower F1 score than YOLOv8m-seg in both confidence threshold 0.50 and 0.20. This suggests that further improvements in F1 score beyond the m-variant may be minimal unless we enhance the quality of the training dataset or address generalizability issues. Note that all YOLOv8-seg variants are pretrained using the DOTAv1 Aerial Detection dataset. We also tested pretraining the models using COCO segmentation dataset, but found no significant difference, as shown in Table II.

For YOLOv9 segmentation models [7], only the c- and e-variants are provided, which correspond to the m- and x-variants of YOLOv8, respectively. To compare their performance, we can look at the "medium" size category models: YOLOv9c-seg and YOLOv8m-seg. YOLOv9c-seg has 27.9 million parameters and 159.4 giga FLOPS, while YOLOv8m-seg has 27.3 million parameters and 110.2 giga FLOPS. This means that although both models have a similar number of parameters, YOLOv9c-seg requires 1.45 times more FLOPS than YOLOv8m-seg. Despite this, YOLOv9c-seg has a significantly lower F1 score compared to YOLOv8m-seg's F1 in both confidence threshold level. Note that this may due to the different batch size used for YOLOv8

Table I: Comparison of (Pubic) F1-Score for Different YOLO Variants and Size

Model	Pretrained Weights	Batch Size	Params (M)	FLOPs (G)	Public F1-Score	
					Conf = 0.50	*Conf = 0.20*
YOLOv8n-seg	DOTAv1 Aerial Detection	16	3.4	12.6	0.510	0.645
YOLOv8s-seg		16	11.8	42.6	0.535	0.654
YOLOv8m-seg		16	27.3	110.2	0.592	0.649
YOLOv8x-seg		8	71.8	344.1	0.579	0.627
YOLOv9c-seg	COCO Segmentation	4	27.9	159.4	0.476	0.577
Mask R-CNN (MPViT-Tiny)	COCO Segmentation	4	17	196.0	-	0.596
EfficientNet-b0-YOLO-seg	ImageNet	4	6.4	12.5	-	0.560

Table II: Comparison of Transfer Learning using Different Pretrained weights

Model	Public F1 Score
YOLOv8m-seg	0.636
YOLOv8m-seg + COCO pretrained weights	0.649
YOLOv8m-seg + DOTAv1 pretrained weights	0.649

and YOLOv9, due to our GPU memory size constraint. However, this does not change the fact that YOLOv9 is hard to train and slower due to the high FLOPS, with no F1-score improvement.

For the performance baseline comparison, we also tried using Mask R-CNN with the MPViT backbone, as suggested in [8]. However, due to resource constraints, we were only able to test the smallest MPViT-Tiny variant for our Mask R-CNN. With this modification, Mask R-CNN achieved an F1 score of 0.596 at confidence threshold of 0.20, which is considerably lower than that of YOLOv8m-seg. However, the model has a surprisingly high FLOPS of 196 giga FLOPS, which is 1.78 times greater than that of YOLOv8m-seg. Hence, we argue that the substantial increase in FLOPS from YOLOv8m-seg to Mask R-CNN does not justify the incremental improvement in the F1 score.

Additionally, we tested replacing the YOLOv5 backbone with EfficientNet [9], a lightweight CNN model designed for low-computational-power devices. EfficientNet achieved its lightweight design and high inference speed by replacing parts of the convolutional layers with depthwise convolutional layers, which require fewer parameters. The caveat is that modern accelerators (i.e., GPUs) are not optimized for depthwise convolutional layers. As a result, we were only able to test the smallest EfficientNet-b0 variant as a proof of concept. Despite this, it achieved an F1 score of 0.560, which is surprisingly close to Mask R-CNN (with the MPViT-Tiny backbone). Note that This model is not pretrained with DOTAv1 and/or COCO segmentation dataset. We believe the performance could be further improved by pretraining and scaling up the EfficientNet backbone.

In short, we find that the YOLOv8 series is a strong candidate for instance segmentation tasks, offering the best tradeoff between FLOPS and F1 score. Specifically, we selected the m-variant (YOLOv8m-seg) as our preferred model. In the following sections, we will demonstrate how we can improve the performance of our instance segmentation model with datasets and other techniques.

Impact of Using Different Training Dataset

Next, we experimented with the performance of YOLOv8m-seg by varying the training dataset, as shown in Table III. Setup A represents the performance baseline, where YOLOv8m-seg was trained solely on the provided BEGC2024 training dataset. As expected, this setup resulted in the lowest F1-score, likely due to the lack of diversity in the training data. In Setup B, we trained YOLOv8m-seg using both the BEGC2024 training set and our Redmond dataset. This simple step of diversifying the training data led to a significant increase in the F1-score, from 0.649 to 0.660. Surprisingly, using the Las Vegas dataset resulted in an even higher public F1-score of 0.686, as shown in Setup C. We believe the reason why the Las Vegas dataset results in a greater improvement in F1 score is due to its greater semantic difference from the BEGC2024 training set, which helps enhance the model's ability to generalize in the test set. As shown in Figure 2, the buildings in the Las Vegas dataset have subtle differences compared to those in the BEGC2024 dataset, particularly in terms of

Table III: Comparison of (Public) F1-score for YOLOv8m-seg using Different Dataset

Setup	Dataset	Public F1 Score
A	BEGC 2024	0.649
B	BEGC 2024 + Redmond Dataset	0.660
C	BEGC 2024 + Las Vegas Dataset	0.686
D	BEGC 2024 + Diffusion Augmentation	0.672
E	BEGC 2024 + CutMix Dataset	0.650

rooftop colours and shapes. Additionally, the images in the Las Vegas dataset are mostly sampled from desert-like areas, which are not common in the BEGC2024 training set. In contrast, the Redmond dataset is more similar to the BEGC2024 training set. Not only do the building structures and colours look similar, but the backgrounds also resemble those of the BEGC2024 rural area, which is predominantly covered with forests or vegetation. This similarity explains why the performance improvement with the Redmond dataset is not as significant as with the Las Vegas dataset.

Surprisingly, the performance of the YOLOv8m-seg model trained with the BEGC2024 dataset using diffusion augmentation resulted in a considerably high F1 score of 0.672, as shown in Setup D. This F1 score is even higher than that of Setup B, which was trained with the Redmond dataset. This observation demonstrates that our diffusion augmentation method successfully created semantically different images that were sufficient to diversify the BEGC2024 training set. However, the F1 score of Setup D is still lower than that of Setup C, which was trained with the Las Vegas dataset. We observed that the advantage of the Las Vegas dataset over diffusion augmentation lies in its background diversity. Since our segmentation-guided diffusion model was trained solely on the BEGC2024 training set, it predominantly learned from the same underlying distribution. Even though the generated buildings are different, the backgrounds remain largely similar to those in the BEGC2024 training set. As a result, the Las Vegas dataset still offers higher diversity (i.e., desert backgrounds, different building shapes). Nonetheless, we have demonstrated the practicality of diffusion augmentation when additional datasets are not available. Figure 4 shows example of images generated via our diffusion augmentation pipeline using the segmentation-guided diffusion model.

We also tried CutMix augmentation to diversify the training dataset to improve generalization of our model. Similar to [10], we cut out buildings from the BEGC2024 dataset and placed them randomly into new background images. Specifically, we collected five different backgrounds, ranging from desert and beach to forest. When generating the CutMix dataset, we randomly selected one of these background images and pasted the building extracted from the original image. We applied this CutMix augmentation to each image in the BEGC2024 training set, resulting in 3,784 new images. However, we found this method to be less effective, achieving an F1 score of only 0.650, as shown in Setup E of Table III. The F1 score improvement was almost negligible compared to our baseline in Setup A. We believe this is due to the lack of variation in building structures, as we only changed the backgrounds. This highlights the importance of diversifying both building shapes and background textures to improve the model's generalization.

In short, we found that using additional datasets and diffusion augmentation can significantly improve the generalizability of our model, resulting in an improvement in the F1 score. Table IV presents a comparison of our methods with those of the 2nd and 3rd place entries on the Kaggle Leaderboard. Our YOLOv8m-seg model, trained on both the BEGC2024 training set and the Las Vegas dataset, achieved the highest F1 score on both the public and private leaderboards. Meanwhile, our model trained with the diffusion augmentation method performed slightly lower than the 2nd place entry but significantly higher than the 3rd place. This again highlights the robustness of our diffusion augmentation pipeline. At this time, the methods used by the 2nd place participants are unknown. However, it is likely that they utilized either a larger model or a larger dataset, as scaling laws typically lead to better performance. Nevertheless, the close F1 score between our model trained with diffusion augmentation and the 2nd place solution suggests that our proposed diffusion augmentation is both label-efficient and effective.

Table IV: Public and Private F1-Score of Our Methos

Dataset	Public F1 Score	Private F1 Score
BEGC 2024	0.64926	0.66331
BEGC 2024 + Redmond Dataset	0.65951	0.67153
BEGC 2024 + Las Vegas Dataset	**0.68627**	**0.70326**
BEGC 2024 + Diffusion Augmentation	0.67189	0.68096
2nd place	0.6813	0.68453
3rd Place	0.59314	0.60649

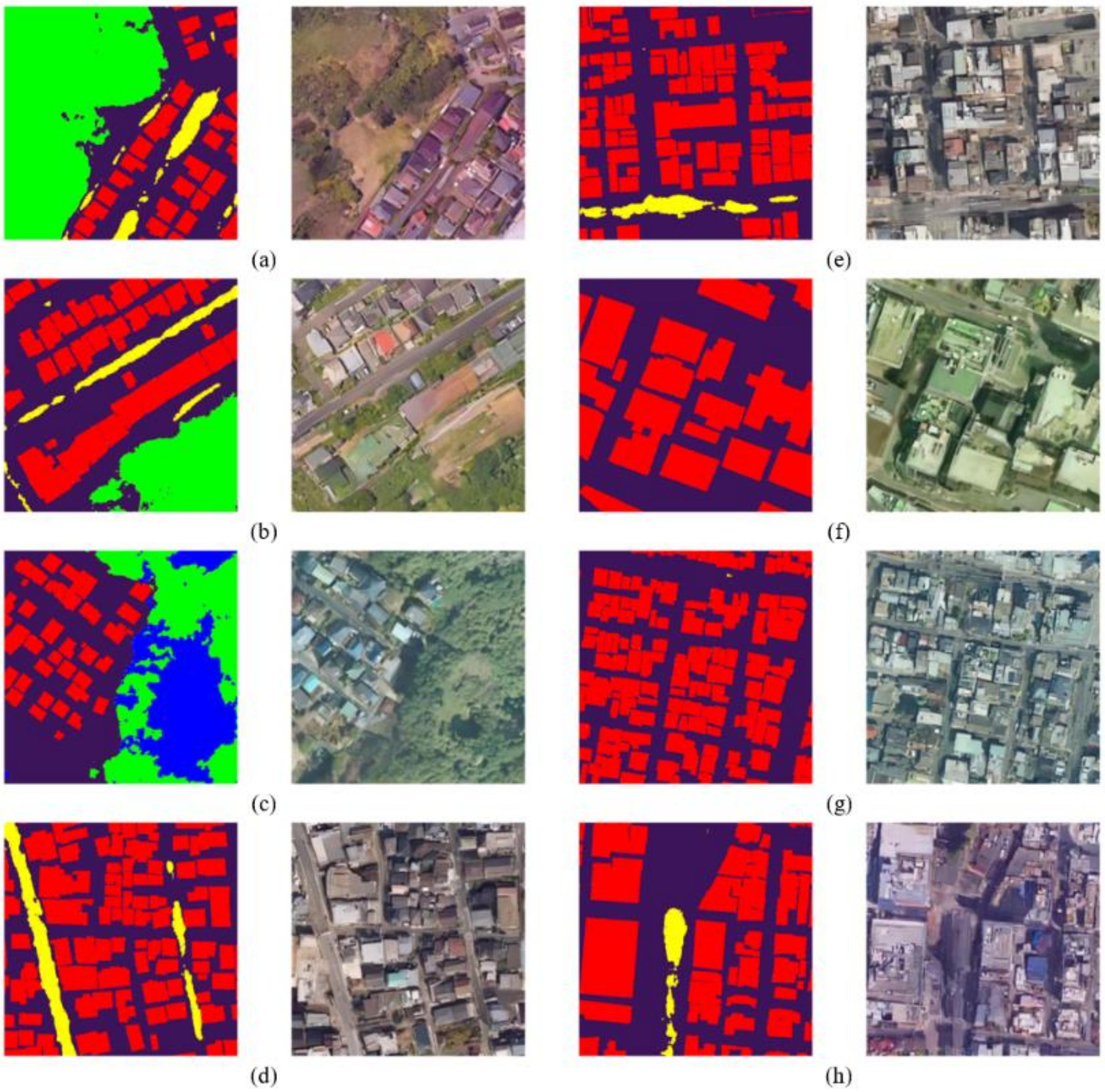

Fig. 4. Samples of synthetic images generated by the diffusion augmentation pipeline based on the semantic segmentation mask.

Varying the Intersection of Union (IoU) threshold for Non-Maximal Surpression (NMS) layer

NMS can be less effective at detecting small, densely packed objects, as it relies on IoU to suppress overlapping bounding boxes. In scenarios involving small and dense objects, the bounding boxes often overlap significantly, which can lead to the suppression of true positives. Examples of dense and small objects are illustrated in Figures 4d, 4e, and 4g. We can mitigate this issue by increasing the IoU threshold in the NMS layer to prevent unnecessary reduction of bounding boxes. We experimented by increasing the IoU threshold in the NMS layer of YOLOv8m-seg from the default 0.70 to 0.95, with increments of 0.05. Generally, we found that IoU thresholds of 0.90 and 0.95 work best compared to other threshold settings. Note that simply increasing the IoU threshold does not directly translate to better performance, as it may lead to an increase in false positives that should have been suppressed by the NMS layers. For instance, setting the IoU threshold between 0.75 and 0.80 is generally worse than the default 0.70 threshold, as shown in Tables V. Hence, our final submission is the YOLOv8m-seg model trained on the BEGC2024 and Las Vegas datasets, with the IoU threshold for NMS set to 0.95. In future works, we consider trying more advanced NMS variation including Attention based NMS [11] and Density-based NMS [12] to better mitigate this problem. Note that this should be the last resort to improve the model performance, as the main focus of the challenge is to address the generalizability of the model with limited domain-specific training data.

Table V: Private F1 Score for Different NMS IoU Threshold. Missing Values are Denoted with dash.

Dataset	NMS IoU Threshold					
	0.70	0.75	0.80	0.85	0.90	0.95
BEGC2024 + Redmond Dataset	0.672	0.677	-	-	0.748	0.866
BEGC2024 + Las Vegas Dataset	0.703	0.693	0.686	0.721	0.766	0.897
BEGC2024 + Diffusion Augmentation	0.681	-	0.694	0.711	0.751	0.887

4. Error Analysis, including examples of Failed Attempts:

As mentioned in earlier sections, we made various attempts to improve the generalization of our model from the training set to the test set. Some of the failed attempts included using different models, such as the more recent YOLOv9c model, Mask R-CNN with an MPViT-Tiny backbone, and EfficientNet-b0-YOLO-seg. A common pitfall with these models is that they are computationally expensive to train without a decent GPU. As a result, we either used smaller variants of these models or decreased the batch size, significantly limiting their potential. Additionally, we found that without pretrained weights from either the COCO segmentation or DOTA aerial detection dataset, the models performed poorly. This is evident from the poor performance of our EfficientNet-b0-based YOLO model, which was initialized with only ImageNet weights. We also attempted CutMix augmentation to generate synthetic data and diversify our training set. However, the lack of variation in building shapes and structures resulted in only a limited F1 score improvement. For more details, please refer to the previous sections.

5. Codes & Model Weights:

a) Code for YOLO-based Building Instance Segmentation:
https://github.com/yjwong1999/RSBuildingExtraction

b) Code for Diffusion Augmentation Training Pipeline:
https://github.com/yjwong1999/RSGuidedDiffusion

c) Las Vegas Dataset:
https://www.dropbox.com/scl/fi/qaqo2lqd7x7gxh521f8ce/LasVegas.zip?rlkey=j3oute8e9ia9yoa1hc85fw2ev&st=ug5qoefi&dl=0

d) Synthetic Data Generated by Diffusion Augmentation:
https://www.dropbox.com/scl/fi/slq3qcg0qhzpj9cc22ws4/generated_images.zip?rlkey=npgj3v4ki6o7sogrca742ubt3&st=ooqs8l0q&dl=0

e) Pretrained Model trained using BEGC2024 + Las Vegas Dataset:
https://www.dropbox.com/scl/fi/cdrl62i3mx9p82lqwpik5/yolov8m-seg_LasVegas.pt?rlkey=8ao7a5zz7xnqfd74deffprix2&st=rbtt6dfi&dl=0

f) Pretrained Model trained using BEGC2024 + Diffusion Augmentation:
https://www.dropbox.com/scl/fi/3et60ycgrefyqjz0rtj5b/yolov8m-seg_DiffusionAugmentation.pt?rlkey=z8hov0plgte3iqkop0k41ejnw&st=93xdy0pe&dl=0

Remark: We provided the link to our Colab notebooks in both our GitHub repository for easy reimplementation of our training pipeline! Our Colab notebooks will teach step by step how to train everything from scratch and/or use our pretrained models to reimplement our results. Feel free to check the Colab notebooks.

6. References

[1] Y. L. Khor, Y. J. Wong, M. L. Tham, Y. C. Chang, B. H. Kwan, and K. C. Khor, ‘Multi-Task YOLO for Vehicle Colour Recognition and Automatic License Plate Recognition’, *IEEE Conference on Evolving and Adaptive Intelligent Systems*, 2024, doi: 10.1109/EAIS58494.2024.10570013.

[2] Y. J. Wong, M. L. Tham, B. H. Kwan, E. M. A. Gnanamuthu, and Y. Owada, ‘An Optimized Multi-Task Learning Model for Disaster Classification and Victim Detection in Federated Learning Environments’, *IEEE Access*, vol. 10, pp. 115930–115944, 2022, doi: 10.1109/ACCESS.2022.3218655.

[3] Ultralytics, ‘ultralytics/ultralytics: NEW - YOLOv8 🚀 in PyTorch > ONNX > OpenVINO > CoreML > TFLite’. Accessed: Aug. 15, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics?tab=readme-ov-file

[4] ‘GlobalMLBuildingFootprints’. Accessed: Aug. 20, 2024. [Online]. Available: https://github.com/microsoft/GlobalMLBuildingFootprints/?tab=readme-ov-file

[5] J. Wang, Z. Zheng, A. Ma, X. Lu, and Y. Zhong, ‘LoveDA: A Remote Sensing Land-Cover Dataset for Domain Adaptive Semantic Segmentation’, Oct. 2021, Accessed: Sep. 11, 2024. [Online]. Available: https://arxiv.org/abs/2110.08733v6

[6] N. Konz, Y. Chen, H. Dong, and M. A. Mazurowski, ‘Anatomically-Controllable Medical Image Generation with Segmentation-Guided Diffusion Models’, Feb. 2024, Accessed: Aug. 31, 2024. [Online]. Available: https://arxiv.org/abs/2402.05210v4

[7] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, ‘YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information’, Feb. 2024, Accessed: Aug. 16, 2024. [Online]. Available: https://arxiv.org/abs/2402.13616v2

[8] S. Chen, Y. Ogawa, C. Zhao, and Y. Sekimoto, ‘Large-scale individual building extraction from open-source satellite imagery via super-resolution-based instance segmentation approach’, *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 195, pp. 129–152, Jan. 2023, doi: 10.1016/J.ISPRSJPRS.2022.11.006.

[9] M. Tan and Q. V. Le, ‘EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks’, *36th International Conference on Machine Learning, ICML 2019*, vol. 2019-June, pp. 10691–10700, May 2019, Accessed: Aug. 16, 2024. [Online]. Available: https://arxiv.org/abs/1905.11946v5

[10] Z. Li, F. Lu, J. Zou, L. Hu, and H. Zhang, ‘Generalized Few-Shot Meets Remote Sensing: Discovering Novel Classes in Land Cover Mapping via Hybrid Semantic Segmentation Framework’, Apr. 2024, Accessed: Sep. 12, 2024. [Online]. Available: https://arxiv.org/abs/2404.12721v1

[11] C. Guo, M. Cai, N. Ying, H. H. Chen, J. Zhang, and D. Zhou, ‘ANMS: attention-based non-maximum suppression’, *Multimed Tools Appl*, vol. 81, no. 8, pp. 11205–11219, Mar. 2022, doi: 10.1007/S11042-022-12142-5/METRICS.

[12] L. Rui, X. song Tang, and K. Hao, ‘DB-NMS: improving non-maximum suppression with density-based clustering’, *Neural Comput Appl*, vol. 34, no. 6, pp. 4747–4757, Mar. 2022, doi: 10.1007/S00521-021-06628-W/METRICS.

[13] Z. h.igawa, ‘IEEE Bigdata Cup 2024: Building extraction’, 2024, *Kaggle*. [Online]. Available: https://kaggle.com/competitions/building-extraction-generalization-2024