RMAE-ProGRess: Advancing Semantic Segmentation in Unstructured Environments

Supplementary Material

7. Implementation Details

We provide the detail implementation details in this section. In our benchmarking, different models have different configurations. For the configurations and pretrained models, please refer to the official `GitLab` repository at https://gitlab.com/coeaiml/rmae-progress.

Table 10. Detailed training and configuration settings used for RMAE8L-ProGRess on the RELLIS-3D.

Category	Configuration
Data Processing	
Dataset	RELLIS-3D
Crop Size	512×512
Normalization	Mean = [123.675, 116.28, 103.53], Std = [58.395, 57.12, 57.375]
Backbone (RMAE-8L)	
Model Type	ViT-MAE
Layers / Heads	8 layers, 12 heads
Embedding Dim	768
Patch Size	16×16
Drop Path Rate	0.1
Output Indices	[1, 3, 5, 7]
Neck (Feature to Pyramid)	
Type	Feature2Pyramid
Rescales	[4, 2, 1, 0.5] # Rescales to `(128×128, 64×64, 32×32 and 16×16)`
Decoder (ProGRess)	
Input Channels from Encoder	[768, 768, 768, 768]
Decoder Channels	768
PLF	Enabled (with Self-Fusion)
LCAR	Enabled (Residual only at f_4)
BFF	Enabled
Interpolation	Nearest Neighbor
Loss	CrossEntropyLoss
Training Configuration	
Optimizer	AdamW (β=(0.9, 0.999), lr=1e-4, decay=0.05)
Layer-wise Decay	0.65 across 8 RMAE layers *[Each layer LR shown in Fig. 3]*
LR Scheduler	Linear warm-up (0–1500 iters), PolyLR to 160k iters
Iterations	160K
Batch Size	2 (train), 1 (test)
Seed	596752934
Evaluation & Testing	
Inference	Sliding window (stride = 341×341)
Metric	mIoU, mAcc, aAcc

Reproducibility Notes

- Our implementation and experiments are carried out using the `OpenMMLab MMSegmentation` framework [7]. Table 10 summarizes the architectural details and training configuration used in RMAE8L-ProGRess on the RELLIS-3D dataset. All other configurations are provided in the `GitLab` repository.
- All experiments and benchmarking were conducted on NVIDIA GTX 1080Ti (11 GB), RTX 1080Ti (12 GB), NVIDIA A100 (40 GB) GPUs depending on the size and memory requirements of the models.
- Mixed-precision training was employed using enabling faster training and reduced memory usage.
- A fixed random seed (596752934) was used for all experiments involving our proposed model to ensure reproducibility. Minor variations in results may still occur due to CUDA-level nondeterminism. For all other benchmarked models, original seeds are preserved and documented in the training configuration files in the `GitLab` repository.

Layerwise Learning Rate Fig. 3 illustrates the layerwise learning rate decay strategy employed during training of our RMAE-ProGRess model. In transformer models, lower layers typically encode generic features such as edges and textures, while higher layers capture more task-specific semantics [2, 9]. Since our encoder is based on a ViT-Base architecture pretrained using the MAE strategy [16], we aim to preserve the semantic richness and feature integrity learned across its layers.

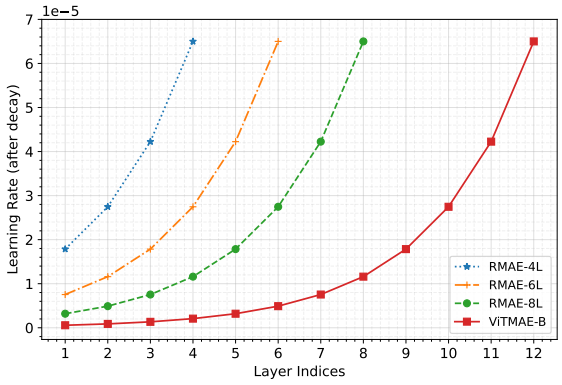

Figure 3. Layerwise Learning Rate for ViTMAE-B and RMAE variants. LR changes accordingly based on the number of layers.

Let η_0 be the base learning rate (`1e-4` in our case), and T be the total number of transformer layers, and $L = T + 2$ is the addition of the first patch embedding layer and decoding layer. The learning rate η_ℓ for layer ℓ is scaled as:

$$\eta_\ell = \eta_0 \cdot \lambda^{L-\ell-1} \tag{12}$$

where $\lambda \in (0, 1)$ is the decay rate (we use $\lambda = 0.65$).

8. Per-class IoU performance

Table 11 reports the per-class IoU on the RELLIS-3D dataset. ViTMAE-ProGRess consistently outperforms the strongest baseline, Swin-UPerNet, across a wide range of challenging categories. In classes characterized by reflective surfaces and irregular boundaries, our method significantly improves *water* segmentation from 23.08% to

Table 11. Extended form of Table 4 with all classwise IoU performance reported for all methods in RELLIS-3D dataset. Best/second/third IoU performance are shown in **bold**/underline/*italic* respectively.

Methods	Backbone	grass	tree	pole	water	sky	vehicle	object	asphalt	building	log	person	fence	bush	concrete	barrier	puddle	mud	rubble	mIou
ISANet [19]	ResNetV1c-101	75.5	73.69	5.69	0	96.6	25.66	52.15	1.6	*27.25*	9.1	38.39	0.89	58.85	66.79	51.55	55.37	35.46	57.17	40.65
FCN [30]	ResNetV1c-101	86.92	75.47	6.14	0	96.78	28.14	44.8	10.08	25.35	0.52	63.15	25.31	71.46	70.34	33.27	26.37	36.77	41.31	41.23
ENCNet [43]	ResNetV1c-101	79.39	80.09	7.83	0	96.94	23.61	49.71	3.8	16.35	1.16	56.86	14.81	65.82	81.44	45.15	55.34	42.68	50.91	42.88
ANN [48]	ResNetV1c-101	84.68	78.38	*10.42*	0	96.77	31.22	45.81	0.87	9.21	3.05	30.4	28.72	70.51	81.61	50.56	48.91	41.04	65.56	43.21
Segmenter [31]	ViT-B16	82.72	80.51	0.14	24.53	96.54	20.35	27.42	5.41	1.34	7.3	75.92	13.46	68.81	81.36	38.33	69.51	35.81	65.91	44.19
PSANet [45]	ResNetV1c-101	84.25	78.41	10.46	0	96.59	35.25	52.21	0	13.57	0.56	64.96	21.05	70.3	82.35	54.79	36.15	40.61	59.12	44.48
DeeplabV3 [4]	ResNetV1c-101	87.7	79.41	3.85	0	97.02	26.8	50.81	6.59	6.67	5.16	68.61	15.64	73.79	82.13	50.73	69.64	39.61	61.09	45.85
GCNet [3]	ResNetV1c-101	81.97	78.88	8.4	0	96.75	33.67	46.17	0	19.73	3.06	72.29	29.66	67.75	83.32	47.55	55.12	41.42	64.13	46.11
OCRNet + FCN [42]	HRNetV2-W48	89.08	81.07	**10.96**	0	96.84	26.57	*52.38*	11.16	8.37	0	59.02	22.28	77.22	80.72	51.32	64.19	39.87	66.8	46.55
DeepLabV3+ [5]	ResNetV1c-101	87.66	80.54	4.59	2.38	97.01	32.79	39.94	33.6	22.99	3.34	46.87	22.37	74.01	82.93	42.11	68.65	39.35	64.43	46.98
DPT [28]	ViT-B16	86.6	78.3	0.56	0	96.02	35.45	46.92	14.01	25.29	2.4	77.36	20.65	72.65	80.97	44.12	70.84	40.48	**70.52**	47.95
ViT-UPerNet [37]	ViT-B16	*87.75*	78.4	1.63	0	96.54	33.12	44.13	21.28	20.54	0	71.78	23.21	73.48	82.81	44.94	73.82	44.15	66.96	48.03
SETR [46]	ViT-L16	86.76	82.68	1.36	8.46	96.92	39.53	48.96	21.34	12.44	6.76	86.99	34.19	74	83.25	50.89	76.29	42.87	66.4	51.12
Segformer [38]	MiT-B5	84.92	79.01	2.69	0	96.88	45.6	51.45	26.69	27.99	4.97	88.97	32.56	70.22	83.77	55.27	67.77	41.19	67.14	51.51
Twins-UPerNet [6]	Twins-SVT	85.58	**83.49**	3.52	0.39	97	42.3	47.8	26.44	26.19	8.82	85.02	33.78	72.56	83.98	54.63	70.74	39.22	*67.62*	51.62
GSCNN [32]	-	84.95	78.52	6.90	0.94	97.02	**46.51**	**54.64**	44.18	11.47	2.92	90.31	**41.86**	70.33	83.82	55.12	71.49	*45.52*	66.03	52.92
Swin-UPerNet [24]	Swin-B-W7	**90.31**	82.35	3.4	23.08	96.93	41.27	50.63	30.33	**37.57**	8.65	87.47	32.49	**78.85**	83.29	*55.61*	61.24	37.58	68.36	53.86
ProGRess (Ours)	RMAE-4L	85.75	81.21	5.83	31.29	96.99	31.69	41.1	52.99	12.68	*9.55*	84.04	30.05	72.22	83.69	55.98	76.4	41.72	65.02	53.23
ProGRess (Ours)	RMAE-6L	86.79	*83.27*	1.83	43.96	97.06	32.32	43.9	58.5	13.4	9.4	86.99	36.71	74.08	84.38	55.42	78.51	45.7	66.03	*55.46*
ProGRess (Ours)	RMAE-8L	87.38	83.47	2.87	*41.74*	**97.1**	37.22	48.84	**67.04**	14.86	**10.97**	*89.77*	37.7	*74.92*	*84.35*	**56.98**	**79.41**	**46.49**	67.41	57.14
ProGRess (Ours)	ViTMAE-B	85.88	82.9	2.42	**49.07**	*97.03*	*42.63*	52.66	*57.38*	22.59	10.64	**91.01**	*37.4*	73.32	**85.2**	54.23	*77.87*	43.68	67.41	**57.41**

Table 12. Per-class IoU on the RUGD dataset. Best/second/third IoU performance are shown in **bold**/underline/*italic* respectively.

Methods	Backbone	dirt	sand	grass	tree	pole	water	sky	vehicle	container	asphalt	gravel	building	mulch	rock-bed	log	person	fence	bush	sign	rock	concrete	picnic-table	mIoU
ISANet [19]	ResNetV1c-101	0	2.15	69.55	77.11	13.02	2.56	79.85	36.71	0.32	5.28	29.6	59.99	47.86	1.13	25.7	0	6.17	18.92	0	0	0	0	21.63
FCN [30]	ResNetV1c-101	0	3.2	65.67	77.94	8.4	10.4	79.65	50.96	0.43	17.04	22.67	67.82	41.55	0.12	25.09	0.0	16.9	2.7	0.0	2.71	10.34	14.03	23.53
ENCNet [43]	ResNetV1c-101	0	10.12	60.28	79.34	6.86	1.95	79.35	48.5	0.08	7.84	34.25	61.69	37.85	2.33	37.72	0	39.71	*33.52*	0	0	0	0	24.61
ANN [48]	ResNetV1c-101	0	7.41	69.36	79.45	13.82	3.47	79.08	34.77	0.18	3.15	32.8	67.19	51.44	18.99	32.36	0	21.89	20.11	0	8.76	5.51	18.04	25.81
Segmenter [31]	ViT-B16	0.01	35.68	71.91	85.84	20.93	40.28	81.74	66.62	0	21.19	24.48	76.43	64.69	9.98	*53.18*	0	47.69	**40.56**	0	*15.48*	87.75	59.24	41.08
GCNet [3]	ResNetV1c-101	0	3.05	71.79	73.75	6.96	5.73	79.61	44.94	0.68	5.02	28.2	63.55	50.97	5.87	26.86	0	3.32	15.98	0	0	0	0	22.1
OCRNet + FCN [42]	HRNetV2-W48	0	17.77	72.23	81.44	7.08	5.96	81.93	32.12	0.09	19.21	22.17	63.5	55.85	2.27	32.81	0	16.13	29.93	0	12.6	4.42	40.03	27.16
DeepLabV3+ [4]	ResNetV1c-101	0	10.33	70.65	78.59	13.85	14.72	81.95	54.27	0.08	21.52	24.61	66.64	52.47	0	36.3	0	47.27	19.68	0	5.35	3.99	56.99	29.97
ViT-UPerNet [37]	ViT-B16	*2.48*	**40.27**	**75.67**	81.29	4.91	24.71	77.59	66.6	0.52	30.76	34.01	76.96	56.41	14.22	52.29	0	59.21	32.67	5.82	15.83	91.47	62.89	41.21
Segformer [38]	MiT-B5	0	*37.64*	70.71	82.13	21.6	**45.98**	83.18	67.64	5.84	**33.34**	33.78	77.91	60.87	*25.97*	53.7	0.67	58.8	18.22	12.18	10.88	90.07	*70.15*	43.69
Twins-UPerNet [6]	Twins-SVT	0.21	25.17	*73.81*	80.16	18.08	24.08	83.2	66.45	2.17	3.29	29.61	73.38	**67.33**	27.13	52.39	*0.22*	47.74	20.58	0	0	0	0	31.59
Swin-UPerNet [24]	Swin-B-W7	3.36	36.25	73.55	81.78	14.83	25.31	83.93	*72.16*	2.54	21.54	*34.77*	77.74	59.79	**27.71**	48.9	0.04	60.88	16.9	**14.39**	**16.79**	75.09	70.38	41.76
ProGRess (Ours)	RMAE-4L	0.54	26.96	74.25	80.47	13.7	21.99	85.67	57.44	1.29	28.74	35.35	74.36	48.2	3.07	43.18	0	51.31	18.04	0	6.8	86.3	63.05	37.3
ProGRess (Ours)	RMAE-6L	0.56	30.34	72.35	82.09	*28.91*	7.53	*86.03*	66.16	1.82	27.72	**36.61**	*78.58*	49.26	9.13	48.56	0	57.34	17.18	0.66	10.73	87.27	64.51	39.24
ProGRess (Ours)	RMAE-8L	**3.9**	38.64	73.28	*83.27*	35.82	8.55	86.25	73.72	*3.97*	*29.65*	32.77	79.42	57.55	18.27	50.32	0	*60.25*	26.56	5.32	12.68	*90.8*	67.05	*42.64*
ProGRess (Ours)	ViTMAE-B	0.18	36.31	71.21	**87.49**	**40.14**	*30.03*	**86.51**	**78.51**	**6.34**	**33.34**	30.04	**80.13**	*63.66*	18.17	**55.18**	**1.14**	**63.52**	36.15	*9.84*	12.75	**91.95**	**71.22**	**45.63**

49.07% (+26%). Among relatively rare classes in off-road environments, *asphalt* shows a substantial gain from 30.33% to 57.38% (+27%), while *concrete* improves from 83.29% to 85.20% (+2%). For categories with ambiguous boundaries, *puddle* increases from 61.24% to 77.87% (+17%), and *mud* from 37.58% to 43.68% (+6%).

Table 12 presents per-class IoU results on the RUGD dataset. Here, ViTMAE-ProGRess again surpasses the strongest baseline, SegFormer, across diverse terrain and object categories. For vegetation classes with ambiguous boundaries, *bush* improves from 18.22% to 36.15% (+18%), while *tree* increases from 82.13% to 87.49% (+5%) and *grass* from 70.71% to 71.21% (+1%). Among object-centric categories in unstructured natural scenes, *vehicle* improves from 67.64% to 78.51% (+11%), *fence* from 58.80% to 63.52% (+5%), and *picnic-table* from 70.15% to 71.22% (+1%). For rare classes, *pole* shows a substantial improvement from 21.60% to 40.14% (+19%), while *building* increases from 77.91% to 80.13% (+2%).

Overall, these results demonstrate the robustness of the ProGRess decoder in capturing fine-grained semantic boundaries, improving performance on rare categories, and resolving ambiguities across complex, unstructured terrains.

9. Computational Complexity Analysis

In this section, we provide a comprehensive analysis of the computational costs associated with our proposed model with RMAE-4L variant as the representative encoder. While our model achieves strong accuracy, understanding the computational bottlenecks is crucial for future optimization efforts.

Table 13. GFLOPS breakdown of our proposed model blocks.

Model Component	FLOPS (G)	Percentage
RMAE-4L	36.1	28.2%
F2P	14.5	11.3%
ProGRess	77.5	60.5%
Total	**128.1**	**100%**

9.1. Overall Model Breakdown

Table 13 presents the GFLOPS distribution across the three main components of our architecture. The total computational cost is 128.1 GFLOPS per inference with 512×512 image resolution. The analysis reveals that the ProGRess decoder accounts for the majority of operations at 77.5 GFLOPS (60.5%), while the RMAE-4L backbone consumes 36.1 GFLOPS (28.2%) and the F2P module requires 14.5 GFLOPS (11.3%). This breakdown clearly identifies

the ProGRess decoder as the primary computational bottleneck affecting inference speed.

9.2. ProGRess Decoder Analysis

To further investigate the computational bottleneck, Table 14 decomposes the ProGRess decoder into its constituent components. The BFF module has the highest decoder's computational cost at 39.0 GFLOPS, accounting for 50.3% of all decoder operations. The PLF module without self-fusion contributes 25.4 GFLOPS (32.8%), while the LCAR module adds 12.8 GFLOPS (16.5%).

Table 14. GFLOPS breakdown of the ProGRess decoder blocks.

Decoder Component	FLOPS (G)	Percentage
PLF (w/o self-fusion)	25.4	32.8%
Self-fusion	0.3	0.4%
LCAR	12.8	16.5%
BFF	39.0	50.3%
Total	**77.5**	**100%**

9.3. Optimization Strategies

Based on our computational analysis, we identify three potential optimization directions to improve computational efficiency while maintaining model accuracy:

- **Channel reduction before BFF.** Given that the BFF module is computationally costly, reducing the number of feature channels prior to BFF execution could significantly decrease computational costs.
- **Spatial downsampling with upsampling.** Applying pooling operations to reduce spatial dimensions before BFF execution, followed by bilinear or learned upsampling, could also provide substantial computational savings.
- **Depthwise separable convolutions.** Replacing standard Conv2D operations in the fusion modules with depthwise separable convolutions offers a well-established approach to reducing FLOPS.

10. Comparison with real-time models

Although Table 4 shows that our model is highly competitive in terms of parameter count and FLOPS when compared with general-purpose segmentation architectures, its efficiency is lower relative to specialized real-time models. As summarized in Table 15, all four ProGRess variants achieve substantially higher mIoU and mAcc on both RELLIS-3D and RUGD, clearly outperforming existing real-time baselines in segmentation quality. However, models such as STDC [10] exhibit significantly lower FLOPS and remain the most efficient among all compared approaches. This observation reflects the fact that our RMAE-ProGRess model is designed primarily for accuracy rather than strict real-time operation. In above section 9, we discussed the computational breakdown to understand which block of our model contributes to the higher FLOPS highlighting the research directions regarding the prospects of real-time efficiency of our model.

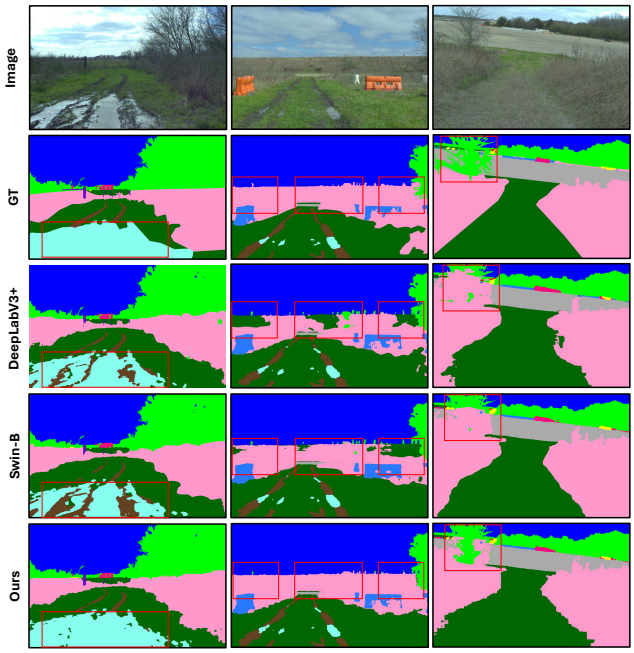

Figure 4. Comparison of the qualitative results of Swin-B Transformer, DeepLabV3+ and our Progress with RMAE-8L encoder. Red squares highlight the areas where our model perform better.

11. Qualitative Results

We qualitatively compare our proposed model against two strong baselines: Swin-B [24], representing transformer-based methods, and DeepLabV3+[5], representing CNN-based approaches, as selected from Table 4. As shown in Fig. 4, both Swin-B and DeepLabV3+ exhibit notable misclassifications in challenging regions of the segmentation map. In contrast, our ProGRess model with RMAE-8L encoder produces more accurate and consistent predictions that better align with the ground truth (GT), highlighting its superior generalization. Fig. 5 shows more qualitative results in some challenging situations. Distinguishing between *puddle* and *mud* remains difficult due to their ambiguous appearance and shared visual characteristics, yet our model handles these cases effectively. Similarly, the *bush*, *tree*, and *grass* classes are predicted with good clarity. These results highlight the strength of our model in handling unstructured scenes characterized by irregular and semantically ambiguous boundaries.

On the other hand, the per-class IoU results indicate that the model still struggles with certain categories. As

Table 15. Comparison of our proposed methods with the real-time models on RELLIS-3D and RUGD test sets. Bold values indicate the variants of our model outperforming other models, and underscore shows the performance of our model competitive to other models.

Methods	Backbone	Aux. Head	Params ↓ (M)	vRAM (MB) ↓	RELLIS-3D			RUGD		
					FLOPS (G) ↓	mIoU ↑	mAcc ↑	FLOPS (G) ↓	mIoU ↑	mAcc ↑
STDC [10]	STDCNet2	FCN	**12.6**	**54.8**	**11.8**	47.97	59.01	**23.5**	36.91	50.34
BiseNetV1 [40]	ResNetV1c-101	FCN	78.23	311.9	118.4	48.35	58.81	-	-	-
PIDNet [39]	PIDNet-L	-	37.31	152.7	34.5	50.46	63.16	68.9	35.13	51.81
DDRNet [17]	DDRNet	-	20.3	88.8	17.9	50.85	61.15	35.9	34.90	51.05
ProGRess (Ours)	RMAE-4L	-	46.47	221.9	128.1	**53.23**	63.04	256.3	**37.30**	50.31
ProGRess (Ours)	RMAE-6L	-	60.74	296.0	145.9	**55.46**	**66.02**	291.8	**37.67**	50.81
ProGRess (Ours)	RMAE-8L	-	75.02	365.0	163.6	**57.14**	**68.53**	327.3	**41.34**	**53.73**
ProGRess (Ours)	ViTMAE-Base	-	103.56	509.4	199.1	**57.41**	**69.21**	398.3	**45.63**	**57.80**

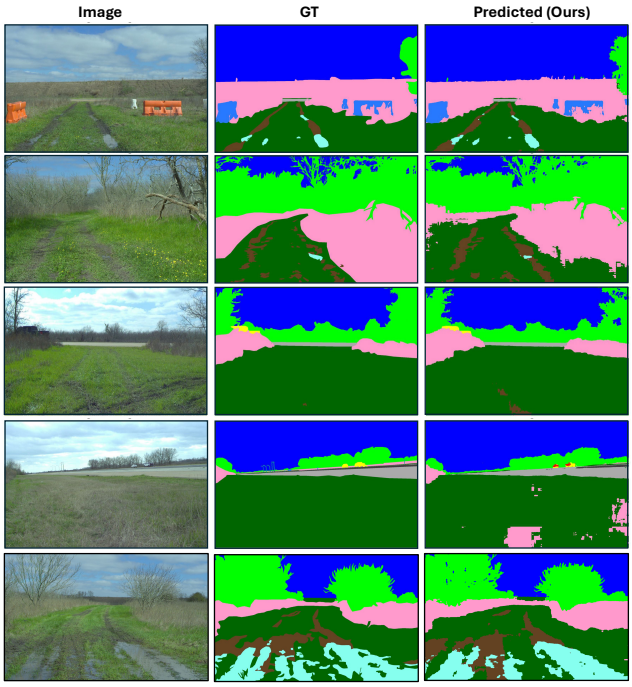

Figure 5. More qualitative results of our RMAE8L-Progress model on RELLIS-3D.

shown in Table 11, one notable example is the *pole* class, where RMAE-8L and ViTMAE achieve only 2.87% and 2.42% IoU, respectively. Similarly, the *fence* class attains IoU scores of 37.7% and 37.4%, which, although higher than those for *pole*, remain much lower compared to other classes. This suggests that while some correct predictions are made, the model still suffers from a considerable number of false negatives. These observations are illustrated qualitatively in Fig. 6, where failure cases are highlighted with red boxes. In the first and fourth images (from top to bottom), the model predicts only partial regions of the *fence* class. In the second image, confusion between *grass* and *bush* is evident, as these categories exhibit subtle visual differences. Although the model demonstrates clear distinctions between these classes in Fig. 4 and Fig. 5, the features in these particular instances are significantly more ambiguous. Furthermore, in the third image, the *pole* class is not detected at all. Overall, these failure cases highlight important areas for future improvement, particularly in handling extremely ambiguous regions.

12. Relationship with FPN-style Networks

While our ProGRess decoder shares the multi-scale fusion paradigm with FPN-style architectures [23, 37], it introduces several critical innovations specifically designed for Vision Transformer backbones to enhance semantic segmentation accuracy and efficiency:

- **Leapwise Feature Extraction:** Unlike FPN [23] and UPerNet [37] which extract features from consecutive CNN stages, our PLF employs leapwise sampling of non-consecutive RMAE layers (e.g., {1, 3, 5, 7} in RMAE-8L). This leverages semantic diversity without redundant computations, crucial in ViT-based architectures where adjacent layers are highly correlated.
- **Explicit Pyramid Construction:** FPN-style methods leverage CNN backbones that naturally produce multi-scale features with distinct spatial and channel dimensions at each stage, forming an implicit pyramid through architectural design. In contrast, RMAE outputs features from all sampled layers with identical dimensions. We therefore employ F2P module, which transforms these uniform features into a proper spatial pyramid through upsampling and downsampling. This explicit construction provides greater control over pyramid characteristics compared to FPN's architecturally-determined scales.
- **Self-Fusion:** The deepest feature map corresponds to the highest-level semantic representations. We enhance these representations through self-fusion as shown by Eq. 1, which concatenates the feature with itself, doubling the input channels to the fusion convolution and enabling richer semantic transformations before top-down propagation. FPN-style methods lack this self-enhancement mechanism, applying only simple lateral connections at the deepest level.

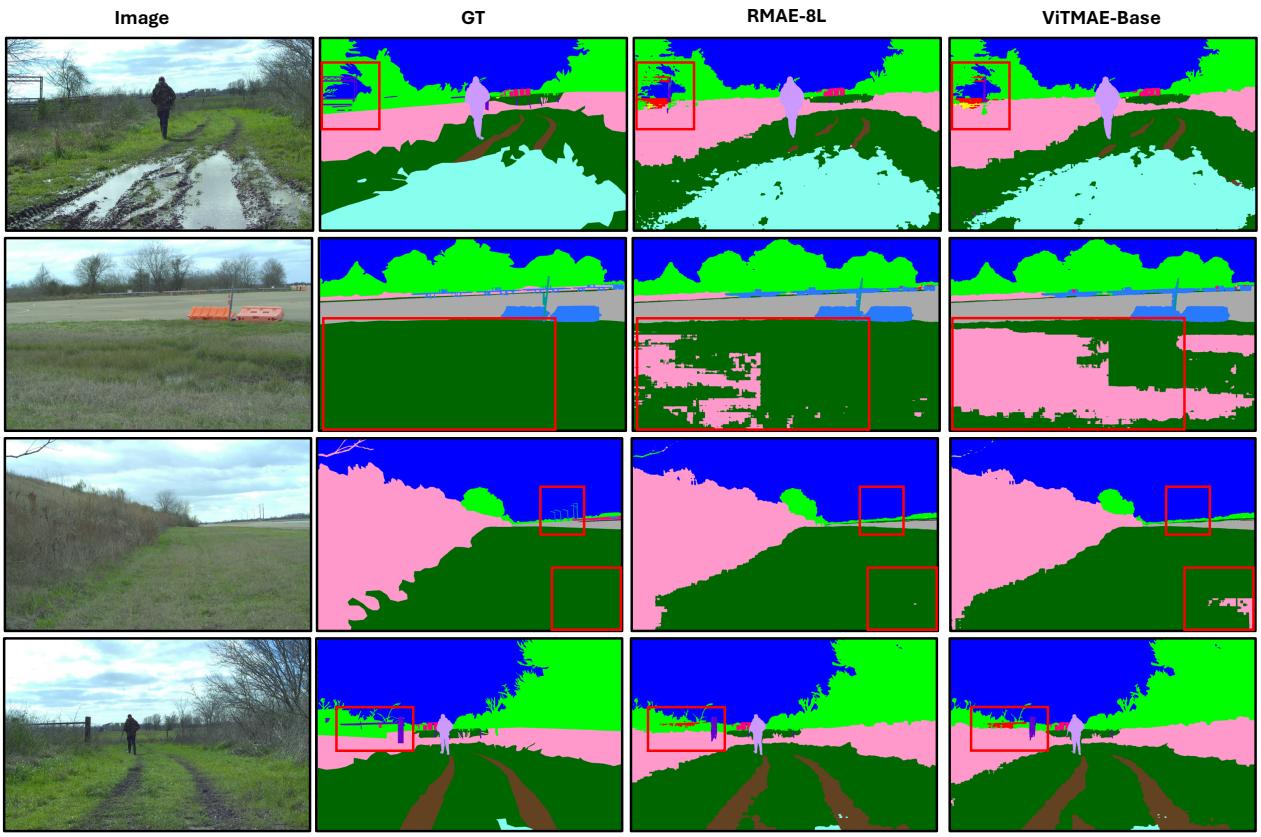

Figure 6. Qualitative results showing failure cases on RELLIS-3D using RMAE-8L and ViTMAE-Base encoders with our ProGRess decoder. Red squares highlight the failure cases.

- **LCAR Refinement:** In FPN-style architectures, fused features are directly aggregated or used for prediction. Our PLF incorporates LCAR at every pyramid level after fusion, enabling adaptive channel-wise recalibration. This attention mechanism emphasizes discriminative features while suppressing noise, significantly improving feature quality before the final BFF stage.

These design choices of the ProGRess decoder consistently outperform FPN-style decoder in both accuracy and parameter efficiency, e.g. UPerNet, across multiple encoders as shown in ablations studies (Table 6).

13. Evaluation on Cityscapes

To check the efficacy of our approach on structured scenes, we train our best model on Cityscapes dataset [8]. Table 16 compares our ViTMAE-ProGRess model on the Cityscapes validation set with the prior work in the literature. We collect these performance reports on the table from the respective research work. Our model achieves a competitive 78.08% mIoU, outperforming some standard ResNet-101 baselines (FCN and PSANet), while falling below DeepLabV3 with ResNet-101 backbone and SETR-PUP with ViT-L backbone. Despite our model is tuned specifically for the unstructured settings, these results indicate its potential to generalize to urban environments as well.

Table 16. Results on Cityscapes validation set.

Method	Backbone	mIoU
FCN [30]	ResNet-101	76.61
PSANet [45]	ResNet-101	77.14
ProGRess (Ours)	ViTMAE	78.08
DeepLabV3 [4]	ResNet-101	79.30
SETR-PUP [46]	ViT-L	82.15