TERRAIN-INFORMED LANDSLIDE LEARNING FOR SPATIALLY EXPLICIT LANDSLIDE PREDICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Landslide prediction from remote sensing data is commonly formulated as a pixel-level segmentation problem, where accurate delineation requires the integration of spectral, terrain, and spatial information across multiple scales. Existing deep learning models can capture these representations effectively, but improvements in prediction accuracy are often accompanied by increased model complexity and computational cost. To address this accuracy–complexity trade-off, we propose TILL (Terrain-Informed Landslide Learning), a terrain-aware framework for spatially explicit landslide prediction. TILL integrates Band-Group Attention and a Terrain-Gated Encoder to jointly model spectral groups and terrain characteristics, followed by Multi-Scale Feature Aggregation and decoding to preserve spatially detailed representations. A Recurrent Refinement Unit subsequently refines the initial prediction using the learned features and prediction state. This design combines terrain-aware encoding, multi-scale representation, and recurrent prediction refinement within a unified segmentation pipeline. TILL produces pixel-level landslide probability maps with an accuracy of 0.85 using 33M parameters and 38G FLOPs. TILL is trained and evaluated on the Landslide4Sense dataset, which is specially designed for landslide detection. Compared with the average performance of the evaluated methods, TILL has a better accuracy-complexity tradeoff, with 9.68% improvement on accuracy, 7.69% fewer parameters, and 1.94% fewer FLOPs. The results show that TILL provides a considerable balance between landslide prediction accuracy and model complexity for spatially explicit delineation.

1 INTRODUCTION

Landslides are complex natural hazards influenced by terrain conditions, surface characteristics, material properties, water availability, and external triggers Han & Semnani (2025); jie Zhang et al. (2016); Tofani et al. (2017); Djukem et al. (2020); Wang et al. (2019). They can cause substantial damage to settlements, infrastructure, and the environment Guzzetti et al. (2006); Reichenbach et al. (2018). Remote sensing provides an effective means of observing large and difficult-to-access areas, while multispectral and terrain observations provide complementary information about surface characteristics, elevation, and slope Guzzetti et al. (2006); Reichenbach et al. (2018); Felsberg et al. (2020). The increasing availability of multisource remote sensing data has supported data-driven approaches for landslide detection and mapping Guzzetti et al. (2006); Reichenbach et al. (2018); Chen et al. (2023); Ebrahim et al. (2025). The Landslide4Sense (L4S) benchmark provides a standardized setting for pixel-level landslide detection using 128×128 image patches with 14 input channels, comprising 12 Sentinel-2 spectral bands and slope and elevation information derived from ALOS PALSAR Ghorbanzadeh et al. (2022a;b).

Deep learning has become an important approach for pixel-level landslide prediction Chen et al. (2023); Ebrahim et al. (2025). Fully convolutional networks established end-to-end semantic segmentation using convolutional feature extraction Long et al. (2015), while U-Net introduced an encoder–decoder structure with skip connections to recover spatial details from hierarchical representations Ronneberger et al. (2015). Multi-scale contextual modeling was subsequently incorporated through architectures such as DeepLabV3+, which employs atrous spatial pyramid pooling, and PSPNet, which uses pyramid pooling to aggregate contextual information at different spatial scales Chen et al. (2018); Zhao et al. (2017). TransUNet further combines convolutional representa-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

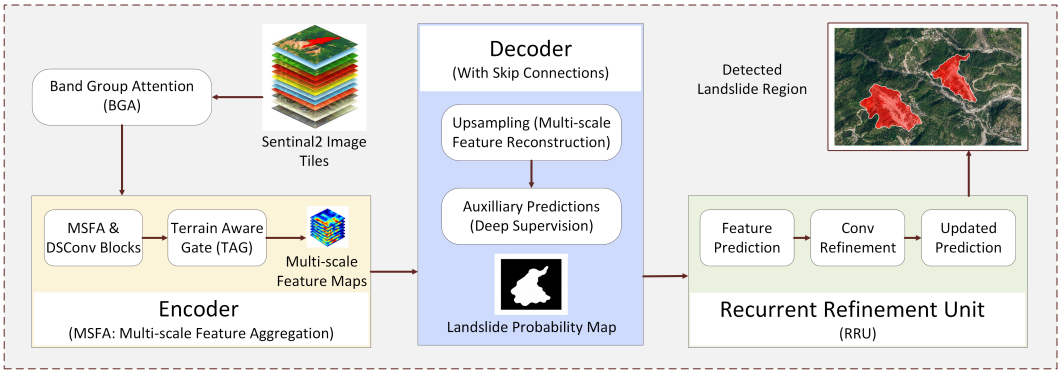


Figure 1: Architecture of TILL for spatially explicit landslide prediction.

tions with Transformer-based encoding to capture broader spatial dependencies Chen et al. (2021). These architectures provide the four reference baselines considered in this work and represent progressively richer approaches to spatial and contextual feature learning.

However, richer representations can increase model size and computational cost, creating an accuracy–complexity trade-off in high-resolution landslide prediction. Existing methods emphasize feature and contextual modeling, with less focus on balancing accuracy and computational efficiency Chen et al. (2023); Ebrahim et al. (2025). The heterogeneous L4S inputs further require differentiated processing of spectral and terrain information while preserving multi-scale spatial details Ghorbanzadeh et al. (2022a;b).

These requirements motivate TILL (Terrain-Informed Landslide Learning), a terrain-aware framework designed to address the accuracy–complexity trade-off in spatially explicit landslide prediction. The overall architecture of TILL is illustrated in Fig. 1. As shown in the figure, the framework follows a hierarchical encoder–decoder design that integrates heterogeneous spectral and terrain information, multi-scale feature extraction, intermediate prediction, and recurrent refinement within a unified prediction pipeline.

At the input stage, TILL employs Band-Group Attention to model relationships among complementary spectral groups. The resulting representation is processed through a Terrain-Gated Encoder, where slope and elevation information guide hierarchical feature extraction. Multi-Scale Feature Aggregation then combines features with different receptive fields, while the decoder progressively reconstructs spatial details through skip-connected upsampling. Auxiliary prediction branches provide intermediate supervision during decoding. Finally, the initial landslide prediction is passed to a Recurrent Refinement Unit, which iteratively updates the prediction using the learned features and the previous prediction state.

2 Problem Formulation

Let the training dataset be $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$, where $\mathbf{x}_i \in \mathbb{R}^{C\times H\times W}$ denotes a multisource remote sensing patch and $\mathbf{y}_i \in \{0,1\}^{H\times W}$ denotes its pixel-wise landslide mask. TILL learns a mapping $f_\theta : \mathbb{R}^{C\times H\times W} \to \mathbb{R}^{H\times W}$, where θ denotes the learnable parameters. The model produces a pixel-wise landslide probability map $\hat{\mathbf{y}} = \sigma(f_\theta(\mathbf{x}))$, where $\sigma(\cdot)$ is the sigmoid activation. For each pixel (h, w), $\hat{y}_{h,w} \in [0, 1]$ represents the predicted landslide probability, which is converted into a binary mask using a threshold τ:

$$\hat{y}^{\text{bin}}_{h,w} = \mathbb{1}\left[\hat{y}_{h,w} > \tau\right], \tag{1}$$

where $\mathbb{1}[\cdot]$ is the indicator function. The threshold is selected using the validation set and fixed during test evaluation.

In addition to accurate pixel-level prediction, the design objective is to maintain a favorable accuracy–complexity trade-off. Let $M(f_\theta)$ denote the segmentation accuracy measured by accuracy, $P(\theta)$ denote the number of trainable parameters, and $F(\theta)$ denote the computational cost measured in FLOPs. The objective is therefore to obtain a model that maximizes segmentation accuracy while

avoiding unnecessary increases in model size and computational cost:

$$\max_{\theta} \; M(f_\theta), \qquad \min_{\theta} \; P(\theta), \; F(\theta). \tag{2}$$

The problem is formulated as learning an accurate pixel-level landslide predictor while maintaining a favorable balance between prediction accuracy, parameter count, and computational cost.

3 METHODOLOGY

TILL (Terrain-Informed Landslide Learning) is a terrain-aware encoder–decoder framework as illustrated in Fig. 1, designed for pixel-level landslide prediction with an emphasis on the accuracy–complexity trade-off. The architecture integrates heterogeneous spectral and terrain information, multi-scale feature representation, and recurrent prediction refinement while avoiding computationally expensive global attention and large modality-specific branches. The details on the dataset, data partitioning, preprocessing, and augmentation are provided in Appendix A, while the training configuration is detailed in Appendix B.4. The complete forward process of TILL is summarized in Algorithm 1.

Algorithm 1 Forward Processing of TILL

Require: Input tile $\mathbf{X} \in \mathbb{R}^{128\times128\times14}$
Ensure: Landslide probability map $\hat{\mathbf{Y}}$

1: Split $\mathbf{X}$ into visible, SWIR, and terrain groups
2: Project each group and compute channel attention using GAP and sigmoid
3: Apply channel-wise weighting and concatenate the group features
4: Fuse the concatenated features to obtain $\mathbf{F}_0$
5: **for** each encoder level **do**
6: $\quad \mathbf{Z}_i \leftarrow \mathrm{DSConv}_i(\mathbf{Z}_{i-1})$
7: $\quad \mathbf{Z}_i \leftarrow \mathrm{MSFA}_i(\mathbf{Z}_i)$
8: $\quad$ Generate a terrain gate from slope and DEM at the current resolution
9: $\quad \mathbf{Z}_i \leftarrow \mathrm{TG}_i(\mathbf{Z}_i, \mathbf{X}_t)$
10: $\quad$ Store $\mathbf{Z}_i$ for skip-connected reconstruction
11: **end for**
12: Initialize the decoder from the terrain-gated bottleneck
13: **for** each decoder level **do**
14: $\quad$ Bilinearly upsample and fuse with the corresponding encoder feature
15: $\quad$ Apply the decoder transformation
16: **end for**
17: Generate auxiliary predictions from intermediate representations
18: Obtain the final decoder feature $\mathbf{D}_0$
19: $\mathbf{F}_p \leftarrow \mathrm{PredHead}(\mathbf{D}_0)$
20: $\mathbf{P}^{(0)} \leftarrow \sigma(f_{1\times1}(\mathbf{F}_p))$
21: **for** $t = 1, \ldots, 3$ **do**
22: $\quad \mathbf{P}^{(t)} \leftarrow \mathrm{RRU}([\mathbf{F}_p; \mathbf{P}^{(t-1)}])$
23: **end for**
24: $\hat{\mathbf{Y}} \leftarrow \sigma(f_{1\times1}(\mathbf{P}^{(3)}))$
return $\hat{\mathbf{Y}}$

3.1 STAGE 1: TERRAIN-AWARE FEATURE ENCODING

Stage 1 converts the heterogeneous multisource input into a hierarchical terrain-aware representation for subsequent prediction. The input is first divided into visible, SWIR, and terrain groups and processed using Band-Group Attention (BGA). The resulting representation is then passed through four encoder levels, where spatial feature extraction is combined with terrain-guided modulation. The processing sequence is summarized in Algorithm 1, Lines 1–10.

3.1.1 BAND-GROUP ATTENTION

The 14-channel input is divided into visible spectral bands, SWIR bands, and terrain information comprising slope and DEM. Each group is projected into a common feature space and assigned group-specific channel weights. For a group g, the attention weights are obtained from a globally pooled channel descriptor:

$$\mathbf{a}_g = \sigma\left(W_{g,2}\,\delta\left(W_{g,1}\,\mathrm{GAP}(\mathbf{F}_g)\right)\right), \tag{3}$$

where $\delta(\cdot)$ denotes GELU and $\sigma(\cdot)$ denotes the sigmoid function. The resulting weights are applied channel-wise before the attended group representations are concatenated and fused to obtain the initial feature map $\mathbf{F}_0$. This provides a differentiated representation of the input groups before hierarchical spatial encoding. The complete BGA operation is given in Algorithm 1, Lines 1–4; its detailed tensor transformations and channel dimensions are provided in Appendix B.

3.1.2 TERRAIN-GATED ENCODER

The fused representation is progressively encoded through four levels. Each level first performs spatial downsampling using depthwise-separable convolution (DSConv), followed by Multi-Scale Feature Aggregation (MSFA) and terrain-guided gating. For encoder level i, the transformation is represented as

$$\mathbf{Z}_i = \mathrm{TG}_i\left(\mathrm{MSFA}_i\left(\mathrm{DSConv}_i(\mathbf{Z}_{i-1})\right), \mathbf{X}_t\right). \tag{4}$$

The Terrain Gate uses the slope and DEM information to generate a spatial gating representation at the current encoder resolution. The resulting gate modulates the encoded feature and is combined through residual fusion. This allows terrain information to influence feature extraction at multiple spatial resolutions rather than only at the input stage. The corresponding operations are summarized in Algorithm 1, Lines 6–10. The four encoder levels progressively transform the representation as $128^2 \times 64 \to 64^2 \times 96 \to 32^2 \times 192 \to 16^2 \times 320 \to 8^2 \times 512$. The resulting encoder features are retained for skip-connected reconstruction in Section 3.2.

3.2 STAGE 2: MULTI-SCALE FEATURE EXTRACTION AND RECONSTRUCTION

Stage 2 transforms the hierarchical encoder representation into a full-resolution feature map for landslide prediction. The stage combines progressive decoding with skip connections and auxiliary prediction. The objective is to recover spatial details lost during hierarchical downsampling while retaining the contextual information encoded at deeper resolutions. The overall processing is summarized in Algorithm 1, Lines 12–18.

3.2.1 MULTI-SCALE FEATURE EXTRACTION

The bottleneck representation produced by Stage 1 contains high-level features at an 8×8 spatial resolution. Before reconstruction, Multi-Scale Feature Aggregation (MSFA) is applied to combine responses with different receptive fields. This allows the bottleneck representation to retain both localized structures and broader spatial context relevant to landslide regions of different sizes. The MSFA operation is defined in Appendix C.1.

3.2.2 MULTI-SCALE DECODER

The encoded representation is progressively reconstructed using a hierarchical decoder. At each level, the decoder representation is upsampled and combined with the corresponding encoder feature through a skip connection. This provides access to higher-resolution spatial information that is not retained completely by the bottleneck representation. The decoder therefore follows the general transformation $\mathbf{D}_i = \mathrm{DecBlock}_i\left(\mathrm{Up}(\mathbf{D}_{i+1}), \mathbf{S}_i\right)$, where $\mathbf{D}_{i+1}$ denotes the lower-resolution decoder feature and $\mathbf{S}_i$ denotes the corresponding encoder feature. Bilinear upsampling is used before feature fusion. The complete reconstruction progresses from the 8×8 bottleneck to the original 128×128 spatial resolution. This progressive reconstruction preserves both hierarchical context and

fine spatial information required for pixel-level landslide delineation. The corresponding decoder configuration is provided in Appendix C.2.

3.2.3 AUXILIARY PREDICTION

TILL introduces two auxiliary prediction branches at intermediate stages of the reconstruction pathway. The first branch operates on the bottleneck representation and produces a prediction at 32×32, while the second operates on an intermediate decoder representation and produces a prediction at 64×64. These predictions provide additional supervision before the full-resolution output is reached. The auxiliary predictions are expressed as $\hat{\mathbf{Y}}_{\text{aux},2} = H_{\text{aux},2}(\mathbf{Z}_4)$, and $\hat{\mathbf{Y}}_{\text{aux},1} = H_{\text{aux},1}(\mathbf{D}_1)$, where $H_{\text{aux},2}$ and $H_{\text{aux},1}$ denote the corresponding auxiliary prediction heads. Their detailed construction is given in Appendix C.3. The auxiliary branches do not replace the main prediction pathway. Instead, they provide intermediate prediction signals while the decoder continues to reconstruct the full-resolution feature representation. The final decoder output $\mathbf{D}_0 \in \mathbb{R}^{128\times128\times32}$ is subsequently passed to Stage 3 for initial prediction and recurrent refinement.

3.3 STAGE 3: RECURRENT LANDSLIDE PREDICTION REFINEMENT

Stage 3 converts the full-resolution decoder representation into the final landslide prediction. Instead of producing the output directly from the decoder feature, TILL first generates an initial probability map and then refines it through a recurrent prediction pathway. The previous prediction is fed back together with the learned spatial representation at each iteration, allowing the prediction state to be progressively updated. The complete sequence is summarized in Algorithm 1, Lines 19–24.

3.3.1 INITIAL PREDICTION

The final decoder representation $\mathbf{D}_0 \in \mathbb{R}^{128\times128\times32}$ is first transformed into a prediction feature representation. A 3×3 convolution, batch normalization, and GELU activation are applied to obtain

$$\mathbf{F}_p = \mathrm{GELU}\left(\mathrm{BN}\left(f_{3\times3}(\mathbf{D}_0)\right)\right). \tag{5}$$

An initial landslide probability map is then generated using a 1×1 convolution followed by sigmoid activation:

$$\mathbf{P}^{(0)} = \sigma\left(f_{1\times1}(\mathbf{F}_p)\right), \tag{6}$$

where $\mathbf{P}^{(0)} \in [0,1]^{128\times128}$. This prediction provides the starting state for the recurrent refinement process. The initial prediction operation is given in Algorithm 1, Lines 19–20.

3.3.2 RECURRENT REFINEMENT UNIT

The Recurrent Refinement Unit (RRU) performs three successive prediction refinement steps. First, the prediction feature $\mathbf{F}_p$ obtained from the decoder is transformed into the initial prediction state:

$$\mathbf{P}^{(0)} = \sigma\left(f_{1\times1}(\mathbf{F}_p)\right), \tag{7}$$

where $f_{1\times1}(\cdot)$ denotes a 1×1 convolution that maps the prediction feature to the output space, and $\sigma(\cdot)$ is the sigmoid activation. The initial prediction $\mathbf{P}^{(0)}$ is then iteratively refined by the RRU. At iteration t, the prediction feature $\mathbf{F}_p$ is concatenated with the prediction from the previous iteration:

$$\mathbf{Q}^{(t)} = [\mathbf{F}_p; \mathbf{P}^{(t-1)}], \qquad t = 1, 2, 3, \tag{8}$$

where $[\cdot;\cdot]$ denotes channel-wise concatenation. The resulting representation is processed by the recurrent refinement function to obtain the updated prediction:

$$\mathbf{P}^{(t)} = \sigma\left(f_r\left(\mathbf{Q}^{(t)}\right)\right), \tag{9}$$

where $f_r(\cdot)$ denotes the convolutional refinement function. The prediction state therefore evolves as $\mathbf{P}^{(0)} \rightarrow \mathbf{P}^{(1)} \rightarrow \mathbf{P}^{(2)} \rightarrow \mathbf{P}^{(3)}$. At each iteration, $\mathbf{F}_p$ is retained as a fixed spatial representation, while the prediction state is replaced by the newly refined probability map. This allows the RRU to progressively update the segmentation prediction without repeatedly recomputing the decoder features. The recurrent refinement operations are summarized in Algorithm 1, Lines 22. After the third refinement iteration, the final prediction is directly taken as $\hat{\mathbf{Y}} = \mathbf{P}^{(3)}$, where $\hat{\mathbf{Y}} \in [0,1]^{128\times128}$. The resulting probability map is thresholded to obtain the binary landslide mask used for evaluation. The final prediction is indicated in Algorithm 1, Line 24.

4 Results

4.1 Accuracy–Complexity Trade-off

TILL is compared with the four reference methods using accuracy, parameter count, and FLOPs. The mean values of the reference methods are 0.775 accuracy, 35.75M parameters, and 38.75G FLOPs, while TILL achieves 0.85 accuracy with 33M parameters and 38G FLOPs. The relative accuracy improvement and reductions in model complexity are computed as

$$I_{\text{accuracy}} = \frac{M_{\text{TILL}} - \overline{M}_{\text{ref}}}{\overline{M}_{\text{ref}}} \times 100, \tag{10}$$

$$R_C = \frac{\overline{C}_{\text{ref}} - C_{\text{TILL}}}{\overline{C}_{\text{ref}}} \times 100, \tag{11}$$

where M denotes accuracy and C denotes either parameter count or FLOPs. TILL achieves a 9.68% relative accuracy improvement with 7.69% fewer parameters and 1.94% fewer FLOPs than the reference mean. As shown in Fig. 2, TILL is non-dominated among the evaluated methods, achieving the highest accuracy while remaining below the reference mean in parameter count and FLOPs. TILL is compared with established segmentation architectures: U-Net Ronneberger et al. (2015), DeepLabV3+ Chen et al. (2018), PSPNet Zhao et al. (2017), and TransUNet Chen et al. (2021), representing encoder–decoder, multi-scale contextual, pyramid-based, and Transformer-enhanced feature learning, respectively.

Method	Params (M)	FLOPs (G)	Accuracy
U-Net	25	15	0.72
DeepLabV3+	41	45	0.78
PSPNet	46	60	0.80
TransUNet	31	35	0.80
TILL	**33**	**38**	**0.85**

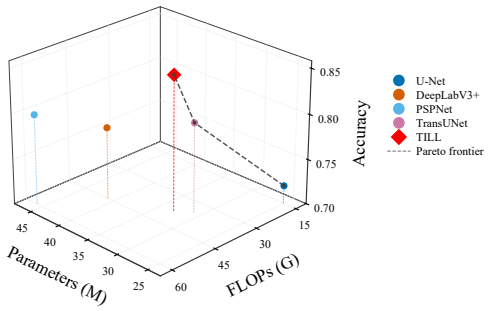
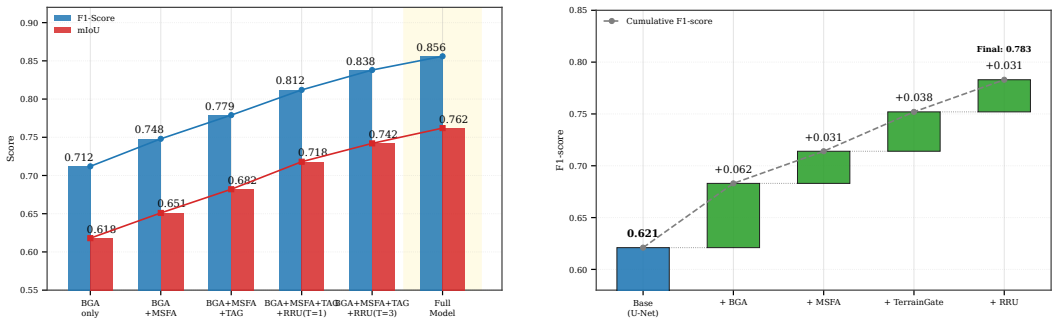


Figure 2: Accuracy–complexity trade-off of the evaluated methods in terms of accuracy, model parameters, and FLOPs. The table reports the quantitative comparison, while the 3D plot illustrates the corresponding trade-off and Pareto frontier.

4.2 Ablation Study

We conduct an ablation study to examine the contribution of the individual components of TILL. Starting from the BGA-only configuration, MSFA, TerrainGate (TAG), recurrent refinement, and deep supervision are progressively introduced. The results are evaluated using F1-score and mIoU, while the computational cost and gradient flow are analyzed separately, as shown in Fig. 3.

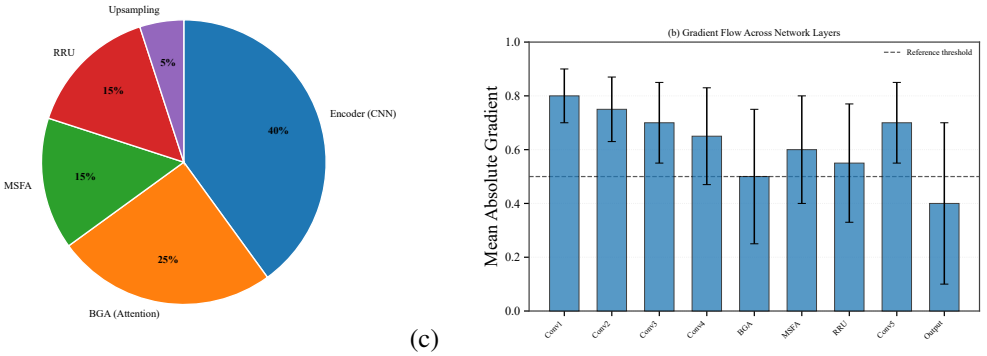


(a) Segmentation performance (b) Incremental contribution of components

Figure 3: The panel shows (a) segmentation performance, (b) incremental component contribution.

As shown in Fig. 3(a), the BGA-only configuration achieves an F1-score of 0.712 and an mIoU of 0.618. Adding MSFA increases these values to 0.748 and 0.651, respectively. Incorporating TAG further improves the F1-score to 0.779 and mIoU to 0.682. With recurrent refinement at $T = 1$, the scores increase to 0.812 and 0.718, while increasing the refinement iterations to $T = 3$ further raises them to 0.838 and 0.742. The complete model, which additionally incorporates deep supervision, achieves an F1-score of 0.856 and an mIoU of 0.762.

The incremental contribution of the architectural components is further illustrated in Fig. 3(b). Each successive component produces a positive increase in F1-score, with gains of 0.062, 0.031, 0.038, and 0.031 obtained from MSFA, TAG, RRU with $T = 1$, and RRU with $T = 3$, respectively. The final addition of deep supervision contributes the remaining improvement to the complete configuration.



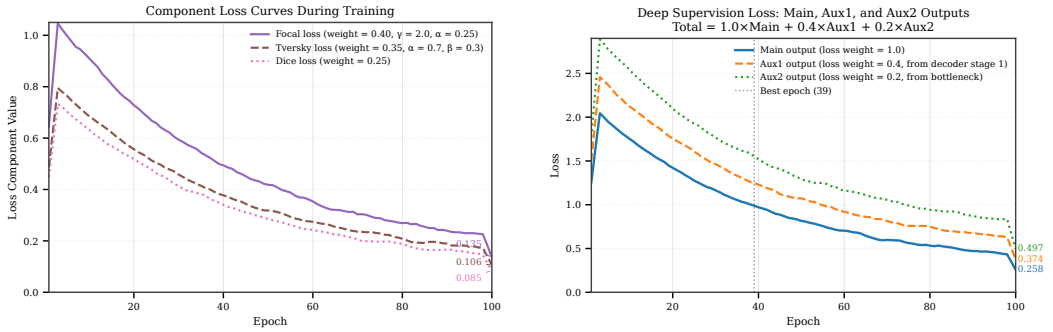
FLOPs distribution (c) (d) Gradient flow

Figure 4: The panel shows (a) Computational cost, and (b) Gradient flow across the network.

The computational distribution in Fig. 4(a) shows that the encoder accounts for the largest share of the total FLOPs (40%), followed by BGA (25%), MSFA (15%), RRU (15%), and upsampling (5%). Fig. 4(b) further visualizes the cumulative contribution of the architectural components, with the successive additions producing the corresponding improvements in the cumulative F1-score. Together, these results show that the components of TILL contribute progressively to segmentation performance while maintaining a distributed computational cost across the architecture.

4.3 Loss Optimization and Deep Supervision

Fig. 5 shows that TILL employs both complementary loss functions and multi-level supervision within a single optimization objective. The component losses in Fig. 5(a) optimize different aspects of the segmentation objective, while the main and auxiliary losses in Fig. 5(b) extend supervision beyond the final output.

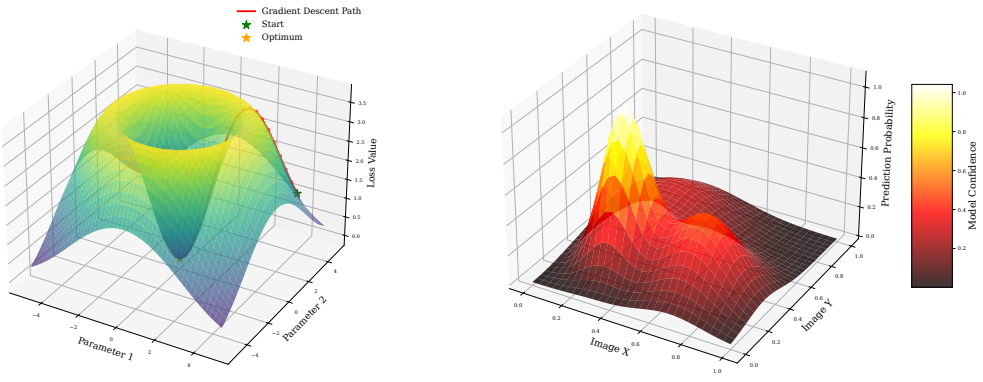


(a) Component loss curves during training **(b)** Deep supervision loss curves

Figure 5: The panels show (a) the individual Focal, Tversky, and Dice loss components during training, (b) the main and auxiliary losses under deep supervision.

TILL combines Focal, Tversky, and Dice losses to capture complementary aspects of pixel-level landslide segmentation. As shown in Fig. 5(a), the three loss components are jointly optimized throughout training. Focal loss contributes a weight of 0.40 with $\gamma = 2.0$ and $\alpha = 0.25$, while Tversky and Dice losses contribute weights of 0.35 and 0.25, respectively. The Tversky loss uses $\alpha = 0.7$ and $\beta = 0.3$, introducing asymmetric weighting between the error terms. The resulting loss trajectories show the joint optimization of the complementary loss components during training.

Fig. 5(b) presents the loss trajectories for the main output and the two auxiliary outputs used for deep supervision. The main output has a loss weight of 1.0, while Aux1 and Aux2 use weights of 0.4 and 0.2, respectively. Aux1 is generated from the first decoder stage, whereas Aux2 is generated from the bottleneck representation. Thus, supervision is provided at multiple stages of the decoding process in addition to the final prediction. The loss trajectories of the three outputs decrease during training, showing that the main and intermediate predictions are optimized jointly. Figure 6(a) provides an illustrative view of the optimization process in a loss landscape. The surface represents a synthetic nonlinear loss function, and the plotted trajectory shows a progressive movement from a higher-loss region toward a lower-loss region. This visualization provides an intuitive representation of the optimization process underlying the joint training objective, complementing the empirical loss trajectories shown in Fig. 5.



(a) Loss landscape and optimization trajectory **(b)** Spatial distribution of prediction confidence

Figure 6: The panels show (a) the loss landscape with the corresponding optimization trajectory, and (b) the spatial distribution of prediction confidence.

4.4 Prediction Confidence Distribution

Figure 6(b) provides an illustrative visualization of how prediction confidence can vary spatially across a landslide segmentation output. High-confidence regions correspond to areas where the predicted landslide probability is concentrated, while intermediate-confidence regions represent less certain predictions. Lower-confidence regions can occur near spatial transitions and boundaries. The visualization complements the quantitative segmentation metrics by illustrating the spatial variation that underlies a pixel-level prediction.

5 Conclusion

This work presented TILL (Terrain-Informed Landslide Learning), a terrain-aware framework for spatially explicit landslide prediction that integrates spectral group attention, terrain-guided feature learning, multi-scale aggregation, and recurrent refinement. On the Landslide4Sense dataset, TILL achieves an accuracy of 0.85 with 33M parameters and 38G FLOPs. The method demonstrates a favorable accuracy–complexity trade-off. Compared with the mean of the evaluated reference methods, it achieves a 9.68% relative improvement in accuracy while using 7.69% fewer parameters and 1.94% fewer FLOPs. Thus, segmentation accuracy does not increase model size or computational demand. On the Landslide4Sense dataset, the results indicate that terrain-aware and multi-scale representations can improve segmentation performance while maintaining a moderate computational cost. Future work will investigate the generalization of TILL across geographically diverse regions and additional remote sensing modalities.

AI Use Statement

Generative AI tools were used solely for language editing and manuscript refinement. All technical content, experimental results, analysis, and conclusions were developed and verified by the authors.

References

Gongfa Chen, Xiaoyu Kang, Mansheng Lin, Shuai Teng, and Zongchao Liu. Stability prediction of soil slopes based on digital twinning and deep learning. *Applied Sciences*, 2023. URL https://api.semanticscholar.org/CorpusId:258933939.

Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 801–818, 2018.

W. Djukem, A. Braun, A. Wouatong, C. Guedjeo, Katrin Dohmen, P. Wotchoko, T. Fernández-Steeger, and H. Havenith. Effect of soil geomechanical properties and geo-environmental factors on landslide predisposition at mount oku, cameroon. *International Journal of Environmental Research and Public Health*, 17, 2020. URL https://api.semanticscholar.org/CorpusId:221844022.

Kyrillos M. P. Ebrahim, E. M. Abdelkader, Tarek Zayed, and M. Meguid. A deep learning-based model for endorsing predictive accuracies of landslide prediction: insights into soil moisture dynamics. *Geoenvironmental Disasters*, 12, 2025. URL https://doi.org/10.1186/s40677-025-00333-9.

A. Felsberg, M. Girotto, J. Poesen, R. Reichle, and T. Stanley. Global soil water estimates as landslide predictor: the effectiveness of observations, simulations and data assimilation results. In *unknown*, 2020. URL https://api.semanticscholar.org/CorpusId:225992178.

Omid Ghorbanzadeh, Yonghao Xu, Pedram Ghamisi, Michael Kopp, and David Kreil. Landslide4sense: Reference benchmark data and deep learning models for landslide detection. *IEEE*

Transactions on Geoscience and Remote Sensing, 60:1–17, 2022a. doi: 10.1109/TGRS.2022.3212209.

Omid Ghorbanzadeh, Yonghao Xu, Hengwei Zhao, Junjue Wang, Yanfei Zhong, Dong Zhao, Qi Zang, Shuang Wang, Fahong Zhang, Yilei Shi, Xiao Xiang Zhu, Lin Bai, Weile Li, Weihang Peng, and Pedram Ghamisi. The outcome of the 2022 landslide4sense competition: Advanced landslide detection from multisource satellite imagery. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15:9927–9942, 2022b. doi: 10.1109/JSTARS.2022.3220845.

Fausto Guzzetti, Paola Reichenbach, Mauro Cardinali, Mirco Galli, and Francesca Ardizzone. Probabilistic landslide hazard assessment at the basin scale. *Geomorphology*, 72(1–4):272–299, 2006. doi: 10.1016/j.geomorph.2006.06.002.

Yi Han and S. J. Semnani. Integration of physics-based and data-driven approaches for landslide susceptibility assessment. *International Journal for Numerical and Analytical Methods in Geomechanics*, 49:3060 – 3097, 2025. URL https://api.semanticscholar.org/CorpusId:279877559.

Shao jie Zhang, Luqiang Zhao, R. Delgado-Téllez, and H. Bao. A physics-based probabilistic forecasting model for rainfall-induced shallow landslides at regional scale. *Natural Hazards and Earth System Sciences*, 18:969–982, 2016. URL https://api.semanticscholar.org/CorpusId:54597565.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

Paola Reichenbach, Mauro Rossi, Bruce D. Malamud, V. Mihir, and Fausto Guzzetti. A review of statistically-based landslide susceptibility models. *Earth-Science Reviews*, 180:60–91, 2018. doi: 10.1016/j.earscirev.2018.03.001.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015*, pp. 234–241. Springer, 2015.

V. Tofani, Gabriele Bicocchi, G. Rossi, S. Segoni, M. D'Ambrosio, N. Casagli, and F. Catani. Soil characterization for shallow landslides modeling: a case study in the northern apennines (central italy). *Landslides*, 14:755–770, 2017. URL https://doi.org/10.1007/s10346-017-0809-8.

Haiyan Wang, T. Xiao, Xuemeng Li, and L. Zhang. A novel physically-based model for updating landslide susceptibility. *Engineering Geology*, 2019. URL https://doi.org/10.1016/J.ENGGEO.2019.02.004.

Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890, 2017.

A Dataset

The proposed TILL framework is evaluated on the Landslide4Sense (L4S) benchmark dataset Ghorbanzadeh et al. (2022a). Each sample is a 128×128 spatial patch containing 14 input channels, represented as $\mathbf{X} \in \mathbb{R}^{14\times128\times128}$. The first 12 channels correspond to Sentinel-2 multispectral observations, while the remaining two channels contain terrain slope and elevation derived from the ALOS PALSAR digital elevation model (DEM). The corresponding ground-truth annotation is a binary landslide mask $\mathbf{Y} \in \{0, 1\}^{128\times128}$, where a value of one denotes a landslide pixel.

The dataset provides separate training and validation partitions. The training partition is used for model optimization, whereas the validation partition is used for model selection and monitoring during training. Horizontal and vertical flipping are applied synchronously to the input patch and

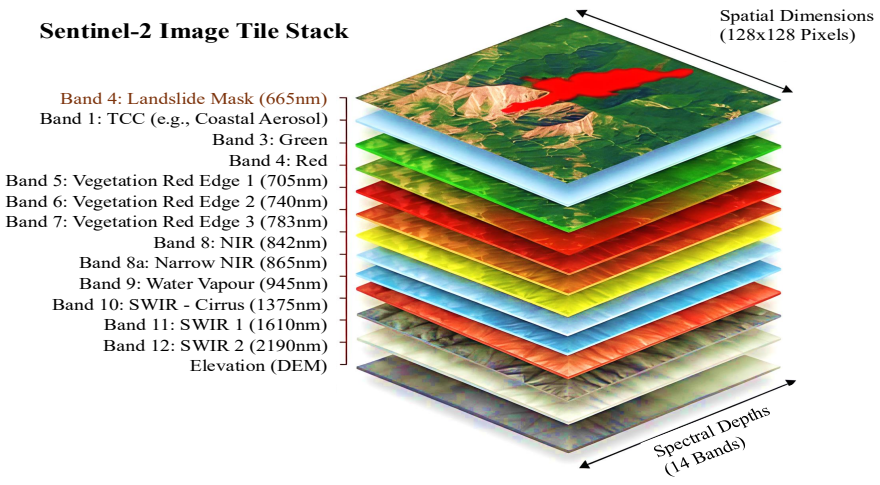


Figure 7: Overview of the Landslide4Sense dataset and input representation.

its corresponding mask during training, with each flip applied with a probability of 0.5. No data augmentation is applied to the validation samples.

Before being provided to TILL, all 14 input channels are normalized using statistics computed from the training data. The same normalization procedure is applied to the validation samples using the training-set statistics. The resulting normalized tensor is then provided directly to the TILL architecture, where the spectral and terrain channels are processed by the Band-Group Attention module.

B Additional Architectural Details

This appendix provides the architectural details of TILL that are condensed in the main methodology for clarity.

B.1 Band-Group Attention

The input tensor is divided into three groups,

$$\mathbf{X} = [\mathbf{X}_v, \mathbf{X}_s, \mathbf{X}_t], \tag{12}$$

where $\mathbf{X}_v$, $\mathbf{X}_s$, and $\mathbf{X}_t$ denote the visible, SWIR, and terrain groups, respectively. Each group is projected as

$$\mathbf{F}_v = P_v(\mathbf{X}_v), \qquad \mathbf{F}_s = P_s(\mathbf{X}_s), \qquad \mathbf{F}_t = P_t(\mathbf{X}_t). \tag{13}$$

For each group, global average pooling produces a channel descriptor, which is transformed into attention weights using two learnable projections:

$$\mathbf{a}_g = \sigma\left(W_{g,2}\delta\left(W_{g,1}\,\mathrm{GAP}(\mathbf{F}_g)\right)\right). \tag{14}$$

The attended feature is

$$\widetilde{\mathbf{F}}_g = \mathbf{F}_g \odot \mathbf{a}_g. \tag{15}$$

The three attended representations are concatenated and fused using a 1×1 projection followed by normalization and GELU activation:

$$\mathbf{F}_0 = \mathrm{GELU}\left(\mathrm{LN}\left(P_f[\widetilde{\mathbf{F}}_v; \widetilde{\mathbf{F}}_s; \widetilde{\mathbf{F}}_t]\right)\right). \tag{16}$$

The resulting representation has spatial resolution 128×128 with 64 channels.

B.2 Terrain-Gated Encoder

The encoder contains four hierarchical levels. At each level, the representation is processed sequentially by DSConv, MSFA, and the Terrain Gate. The terrain inputs are resized to the spatial resolution of the corresponding encoder level before generating the terrain gate.

For level i, let $\mathbf{F}_i$ denote the feature obtained after MSFA and let $\mathbf{T}_i$ denote the corresponding terrain representation. The terrain gate is generated as

$$\mathbf{G}_i = \sigma\left(f_{g,i}(\mathbf{T}_i)\right). \tag{17}$$

The gated representation is obtained through residual modulation:

$$\mathbf{Z}_i = \mathrm{BN}\left(\mathbf{F}_i + \mathbf{G}_i \odot \mathbf{F}_i\right). \tag{18}$$

The encoder produces feature representations with progressively reduced spatial resolution and increased channel capacity:

$$128^2 \times 64 \to 64^2 \times 96 \to 32^2 \times 192 \to 16^2 \times 320 \to 8^2 \times 512. \tag{19}$$

The resulting features are retained as skip representations for the decoder.

B.3 Multi-Scale Feature Aggregation

Each MSFA block contains four parallel convolutional paths:

$$\mathbf{F}^{(1)} = f_{1\times1}(\mathbf{F}), \tag{20}$$

$$\mathbf{F}^{(2)} = f_{3\times3,d=1}(\mathbf{F}), \tag{21}$$

$$\mathbf{F}^{(3)} = f_{3\times3,d=3}(\mathbf{F}), \tag{22}$$

$$\mathbf{F}^{(4)} = f_{3\times3,d=5}(\mathbf{F}). \tag{23}$$

The resulting representations are concatenated and projected to obtain the aggregated feature:

$$\mathbf{F}_{ms} = \mathrm{GELU}\left(\mathrm{BN}\left(P_{ms}[\mathbf{F}^{(1)}; \mathbf{F}^{(2)}; \mathbf{F}^{(3)}; \mathbf{F}^{(4)}]\right)\right). \tag{24}$$

B.4 Experimental Setup

TILL is trained for 60 epochs using a step-based learning-rate schedule. The training configuration consists of 403 optimization steps per epoch, resulting in a total of 24,180 training steps. A linear warm-up is applied during the first 5 epochs, corresponding to 2,015 steps, to gradually increase the learning rate to the base value of 2×10^{-4}. The learning rate is subsequently decayed according to the adopted schedule, with a minimum learning rate of 1×10^{-6}. The complete training configuration is summarized in Table 1.

Table 1: Training configuration used for TILL.

Parameter	Setting
Training epochs	60
Steps per epoch	403
Total training steps	24,180
Warm-up duration	5 epochs (2,015 steps)
Base learning rate	2×10^{-4}
Minimum learning rate	1×10^{-6}

C Detailed Multi-Scale Feature Extraction and Reconstruction

This appendix provides the implementation-level description of Stage 2. The main paper presents the functional role of the stage, whereas this section specifies the feature transformations, decoder configuration, skip connections, and auxiliary prediction heads.

C.1 Bottleneck Multi-Scale Feature Extraction

The output of the fourth encoder level is a bottleneck representation

$$\mathbf{Z}_4 \in \mathbb{R}^{8\times8\times512}. \tag{25}$$

An MSFA block is applied to this representation before decoding. The block uses four parallel convolutional paths with different receptive fields:

$$\mathbf{F}^{(1)} = f_{1\times1}(\mathbf{Z}_4), \tag{26}$$

$$\mathbf{F}^{(2)} = f_{3\times3,d=1}(\mathbf{Z}_4), \tag{27}$$

$$\mathbf{F}^{(3)} = f_{3\times3,d=3}(\mathbf{Z}_4), \tag{28}$$

and

$$\mathbf{F}^{(4)} = f_{3\times3,d=5}(\mathbf{Z}_4). \tag{29}$$

The resulting feature maps are concatenated along the channel dimension and projected to form the multi-scale representation:

$$\mathbf{F}_{ms} = \mathrm{GELU}\left(\mathrm{BN}\left(P_{ms}[\mathbf{F}^{(1)};\mathbf{F}^{(2)};\mathbf{F}^{(3)};\mathbf{F}^{(4)}]\right)\right). \tag{30}$$

Thus, the bottleneck representation contains responses obtained from multiple effective receptive fields before entering the reconstruction pathway.

C.2 Hierarchical Decoder

The decoder progressively increases the spatial resolution from 8×8 to 128×128. The decoder configuration implemented in TILL is

$$8^2 \times 512 \to 16^2 \times 192 \to 32^2 \times 96 \to 64^2 \times 48 \to 128^2 \times 32. \tag{31}$$

At each decoder level, the lower-resolution representation is first upsampled using bilinear interpolation and then combined with the corresponding encoder feature.

Let $\mathbf{D}_{i+1}$ denote the input to a decoder level and $\mathbf{S}_i$ denote the corresponding skip feature. The decoder transformation can be represented as

$$\mathbf{D}_i = \mathrm{DecBlock}_i\left(\mathrm{Up}(\mathbf{D}_{i+1}), \mathbf{S}_i\right). \tag{32}$$

The four decoder transformations are described below.

C.2.1 Decoder Level 1

The bottleneck representation is first reconstructed from 8×8 to 16×16. The resulting feature is combined with the highest-level encoder skip representation:

$$\mathbf{D}_1 = \mathrm{DecBlock}_1\left(\mathrm{Up}(\mathbf{Z}_4), \mathbf{Z}_3\right), \tag{33}$$

where

$$\mathbf{Z}_4 \in \mathbb{R}^{8\times8\times512}, \qquad \mathbf{Z}_3 \in \mathbb{R}^{16\times16\times320}, \tag{34}$$

and the decoder produces

$$\mathbf{D}_1 \in \mathbb{R}^{16\times16\times192}. \tag{35}$$

The skip feature provides higher-resolution information from the corresponding encoder level.

C.2.2 Decoder Level 2

The 16×16 representation is subsequently upsampled to 32×32 and combined with the second encoder skip representation:

$$\mathbf{D}_2 = \mathrm{DecBlock}_2\left(\mathrm{Up}(\mathbf{D}_1), \mathbf{Z}_2\right), \tag{36}$$

where

$$\mathbf{Z}_2 \in \mathbb{R}^{32\times32\times192}, \tag{37}$$

and the resulting decoder representation is

$$\mathbf{D}_2 \in \mathbb{R}^{32\times32\times96}. \tag{38}$$

C.2.3 Decoder Level 3

The decoder representation is then reconstructed to 64×64:

$$\mathbf{D}_3 = \mathrm{DecBlock}_3\left(\mathrm{Up}(\mathbf{D}_2), \mathbf{Z}_1\right), \tag{39}$$

where

$$\mathbf{Z}_1 \in \mathbb{R}^{64\times64\times96}, \tag{40}$$

and

$$\mathbf{D}_3 \in \mathbb{R}^{64\times64\times48}. \tag{41}$$

This level restores spatial information at a resolution where smaller landslide structures and boundaries can be represented more explicitly.

C.2.4 Decoder Level 4

The final decoder level restores the original spatial resolution. The 64×64 representation is upsampled and combined with the highest-resolution skip representation generated immediately after BGA:

$$\mathbf{D}_0 = \mathrm{DecBlock}_4\left(\mathrm{Up}(\mathbf{D}_3), \mathbf{S}_0\right), \tag{42}$$

where

$$\mathbf{S}_0 \in \mathbb{R}^{128\times128\times64}. \tag{43}$$

The final decoder representation is

$$\mathbf{D}_0 \in \mathbb{R}^{128\times128\times32}. \tag{44}$$

The complete reconstruction pathway can therefore be summarized as

$$8^2 \times 512 \to 16^2 \times 192 \to 32^2 \times 96 \to 64^2 \times 48 \to 128^2 \times 32. \tag{45}$$

C.3 Auxiliary Prediction Heads

Two auxiliary prediction heads are incorporated into the reconstruction pathway. Both heads convert intermediate feature representations into single-channel landslide probability maps.

The auxiliary head consists of bilinear upsampling followed by a convolutional transformation, normalization, nonlinear activation, and a final 1×1 convolution with sigmoid activation.

For an intermediate feature $\mathbf{F}$, the auxiliary transformation can be represented as

$$\mathbf{F}_{aux} = \mathrm{GELU}\left(\mathrm{BN}\left(f_{3\times3}(\mathrm{Up}(\mathbf{F}))\right)\right), \tag{46}$$

followed by

$$\hat{\mathbf{Y}}_{aux} = \sigma\left(f_{1\times1}(\mathbf{F}_{aux})\right). \tag{47}$$

C.3.1 Bottleneck Auxiliary Prediction

The first auxiliary prediction is generated directly from the bottleneck representation $\mathbf{Z}_4$. Since the bottleneck has an 8×8 spatial resolution, a scale factor of four is used to obtain a 32×32 prediction:

$$\hat{\mathbf{Y}}_{aux,2} = H_{aux,2}(\mathbf{Z}_4), \qquad \hat{\mathbf{Y}}_{aux,2} \in [0,1]^{32\times32}. \tag{48}$$

This branch provides an intermediate prediction signal directly from the deepest representation.

C.3.2 Intermediate Decoder Auxiliary Prediction

The second auxiliary prediction is generated from the first decoder representation $\mathbf{D}_1$, which has a spatial resolution of 16×16. It is similarly upsampled by a factor of four to produce a 64×64 prediction:

$$\hat{\mathbf{Y}}_{aux,1} = H_{aux,1}(\mathbf{D}_1), \qquad \hat{\mathbf{Y}}_{aux,1} \in [0,1]^{64\times64}. \tag{49}$$

The two auxiliary branches therefore provide supervision at different depths of the reconstruction pathway.

C.4 SKIP-CONNECTED RECONSTRUCTION

The skip connections preserve representations generated before spatial downsampling. In TILL, the highest-resolution BGA output is retained as $\mathbf{S}_0$, while the first three terrain-gated encoder outputs are retained as $\mathbf{Z}_1$, $\mathbf{Z}_2$, and $\mathbf{Z}_3$.

The reconstruction therefore follows the connectivity

$$\mathbf{Z}_4 \to \mathbf{D}_1 \to \mathbf{D}_2 \to \mathbf{D}_3 \to \mathbf{D}_0, \tag{50}$$

with the corresponding skip features injected at each resolution.

This arrangement allows the decoder to combine deep contextual representations with progressively finer spatial features rather than reconstructing the full-resolution representation from the bottleneck alone.

C.5 STAGE 2 OUTPUT

Stage 2 produces three outputs:

$$\left\{\mathbf{D}_0, \hat{\mathbf{Y}}_{aux,1}, \hat{\mathbf{Y}}_{aux,2}\right\}. \tag{51}$$

The full-resolution decoder feature

$$\mathbf{D}_0 \in \mathbb{R}^{128\times128\times32} \tag{52}$$

is passed to Stage 3, where it is transformed into the initial landslide prediction and subsequently refined by the recurrent prediction module. The auxiliary predictions are used during training to provide intermediate supervision.

C.6 RECURRENT REFINEMENT UNIT

The Recurrent Refinement Unit operates on the full-resolution prediction feature generated by the initial prediction head. The input feature $\mathbf{F}_p$ has spatial resolution 128×128 and 32 channels, while the recurrent prediction $\mathbf{P}^{(t-1)}$ contains a single probability channel. At every refinement iteration, these two representations are concatenated along the channel dimension:

$$\mathbf{Q}^{(t)} = [\mathbf{F}_p; \mathbf{P}^{(t-1)}] \in \mathbb{R}^{128\times128\times33}. \tag{53}$$

The concatenated representation is first projected back to 32 channels using a 1×1 convolution followed by batch normalization and GELU:

$$\mathbf{H}_0^{(t)} = \mathrm{GELU}\left(\mathrm{BN}\left(f_{\mathrm{in}}^{(t)}(\mathbf{Q}^{(t)})\right)\right), \tag{54}$$

where

$$f_{\mathrm{in}}^{(t)} : 33 \to 32. \tag{55}$$

The projected representation is subsequently processed by two 3×3 convolutional refinement layers. The first layer maintains 32 channels:

$$\mathbf{H}_1^{(t)} = \mathrm{GELU}\left(\mathrm{BN}\left(f_{r1}^{(t)}(\mathbf{H}_0^{(t)})\right)\right), \tag{56}$$

followed by dropout with a rate of 0.1. The second refinement layer also maintains 32 channels:

$$\mathbf{H}_2^{(t)} = \mathrm{GELU}\left(\mathrm{BN}\left(f_{r2}^{(t)}(\mathbf{H}_1^{(t)})\right)\right). \tag{57}$$

A third 3×3 convolution reduces the representation from 32 to 16 channels:

$$\mathbf{H}_3^{(t)} = \mathrm{GELU}\left(\mathrm{BN}\left(f_{r3}^{(t)}(\mathbf{H}_2^{(t)})\right)\right), \tag{58}$$

where

$$f_{r3}^{(t)} : 32 \to 16. \tag{59}$$

Finally, a 1×1 convolution maps the 16-channel representation to a single probability channel:

$$\mathbf{P}^{(t)} = \sigma\left(f_{\mathrm{out}}(\mathbf{H}_3^{(t)})\right). \tag{60}$$

The same refinement structure is executed for $T = 3$ iterations. Thus, the recurrent state evolves according to

$$\mathbf{P}^{(0)} \to \mathbf{P}^{(1)} \to \mathbf{P}^{(2)} \to \mathbf{P}^{(3)}. \tag{61}$$

The prediction $\mathbf{P}^{(t)}$ generated at one iteration becomes the prediction input for the next iteration, while $\mathbf{F}_p$ remains the fixed spatial feature representation throughout the recurrent process. This gives the refinement unit access to both the learned decoder representation and the current prediction state at every iteration.

C.7 Initial Prediction Head

Before recurrent refinement, the final decoder representation $\mathbf{D}_0 \in \mathbb{R}^{128\times128\times32}$ is processed by the initial prediction head. The head contains a 3×3 convolution that maintains 32 channels, followed by batch normalization and GELU activation:

$$\mathbf{F}_p = \mathrm{GELU}\left(\mathrm{BN}\left(f_{3\times3}(\mathbf{D}_0)\right)\right), \tag{62}$$

where

$$f_{3\times3} : 32 \to 32. \tag{63}$$

The resulting feature is converted into an initial probability map by a 1×1 convolution with sigmoid activation:

$$\mathbf{P}^{(0)} = \sigma\left(f_{1\times1}(\mathbf{F}_p)\right). \tag{64}$$

Both $\mathbf{F}_p$ and $\mathbf{P}^{(0)}$ retain the full 128×128 spatial resolution. The former is used as the spatial representation throughout recurrent refinement, while the latter provides the initial prediction state.

C.8 Recurrent Prediction Sequence

The complete recurrent prediction process can be expressed as

$$\begin{aligned}
\mathbf{Q}^{(1)} &= [\mathbf{F}_p; \mathbf{P}^{(0)}], \quad \mathbf{P}^{(1)} = R(\mathbf{Q}^{(1)}), \\
\mathbf{Q}^{(2)} &= [\mathbf{F}_p; \mathbf{P}^{(1)}], \quad \mathbf{P}^{(2)} = R(\mathbf{Q}^{(2)}), \\
\mathbf{Q}^{(3)} &= [\mathbf{F}_p; \mathbf{P}^{(2)}], \quad \mathbf{P}^{(3)} = R(\mathbf{Q}^{(3)}),
\end{aligned} \tag{65}$$

where $R(\cdot)$ denotes the convolutional refinement transformation described in Appendix C.6. The final recurrent prediction is therefore

$$\hat{\mathbf{Y}} = \mathbf{P}^{(3)}. \tag{66}$$

All three iterations operate at 128×128 resolution. Consequently, the recurrent unit does not introduce additional spatial downsampling or upsampling; its role is to update the prediction state while retaining the spatial resolution produced by the decoder.

C.9 Final Output Mapping

The output of the recurrent refinement unit is passed through a final 1×1 convolution with sigmoid activation to produce the named main output of the network:

$$\hat{\mathbf{Y}}_{\text{main}} = \sigma\left(f_{\text{main},1\times1}(\mathbf{P}^{(3)})\right). \tag{67}$$

The resulting output has dimensions

$$\hat{\mathbf{Y}}_{\text{main}} \in [0,1]^{128\times128}. \tag{68}$$

This probability map is used as the primary prediction during model evaluation, while the auxiliary predictions generated in Stage 2 are used for intermediate supervision during training.