A MODEL-AGNOSTIC PHYSICS-GUIDED ADAPTER FOR FEW-SHOT TRANSFER OF COASTAL FLOOD PREDICTION MODELS TO UNSEEN REGIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Deep learning (DL) surrogates can produce high-resolution coastal flood maps orders of magnitude faster than physics-based hydrodynamic simulators, yet transferring them to new coastal regions remains costly, since generating target-region data for fine-tuning typically requires numerous time-consuming simulations. To tackle this bottleneck, we introduce the Physics Adapter (PA), a compact, architecture-agnostic adaptation interface that enables efficient few-shot transfer of flood prediction models across diverse coastal regions. PA predicts peak water level through a differentiable wet/dry response that compares terrain elevation against a learned water level, and blends this physics-structured prediction with a data-driven branch through a learned gate. Unlike physics-informed formulations, PA imposes no PDE-residual or conservation losses and instead exploits elevation as an architectural inductive bias, adding a negligible number of trainable parameters. We integrate PA into 12 heterogeneous models, spanning graph, convolutional, Transformer, state-space, depth-foundation and diffusion models, and evaluate them on two coastal regions with markedly distinct geometries, topographies, and shoreline protection configurations. The performance of PA is benchmarked against a no-physics baseline, full fine-tuning, and standard parameter-efficient fine-tuning (PEFT) methods, considering both within-region generalization to unseen sea level rise (SLR) values and between-region transfer. In low-shot regime (K=3), and averaged over all backbones and transfer settings, adding PA reduces root mean square error (RMSE) by 11.5% when only the output head is adapted on a frozen backbone, by 15.4% when combined with PEFT methods, and by 22.9% under full fine-tuning, compared to matched configurations without PA. Taken together, the findings of this work offer practitioners a concrete recipe for extending DL-based coastal flood predictors to new, data-scarce regions, thereby advancing scalable AI support for coastal adaptation planning.

1 INTRODUCTION

Climate adaptation-aware coastal protection planning requires repeated prediction of how peak water level (PWL) responds to candidate shoreline protection configurations under different sea level rise (SLR) and forcing conditions. Physics-based high-fidelity simulators, such as Delft3D (Lesser et al., 2004), can accurately simulate nearshore hydrodynamics, providing fine-grained estimates of depth, duration, and velocity of floods. However, due to prohibitively high computational cost, their direct adoption in large-scale coastal protection investigations, where each combination of protection configuration and SLR value requires a separate simulation, remains impractical (Jia et al., 2019). Prior studies (Hassan et al., 2026; Karapetyan et al., 2026; Bian et al., 2025) have demonstrated that learned surrogate models can dramatically reduce this computational burden by approximating the simulator's output, thereby replacing repeated hydrodynamic simulations with efficient inference.

Existing surrogate models have been typically trained and evaluated for one coastline or forcing condition, and their accuracy can degrade sharply when applied to a different coastal region or forcing conditions outside the training range (Sec. 5; see also Zhao et al., 2026). This hinders their practical deployment, since generating sufficient training data for every new setting entails additional hydrodynamic simulations and substantial compute. Moreover, how well existing surrogates trans-

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.

fer across coastlines and SLR conditions, and how many target simulations are necessary to attain satisfactory performance, remains largely unexamined.

In this paper, we investigate whether pretrained coastal flood prediction models can be adapted to new regions and SLR conditions from only a few target simulations, and whether this can be achieved in an architecture-agnostic manner. The problem is challenging for two reasons. First, domain shift arises in different forms: geographic transfer changes terrain, protection geometry, land-cover structure, and flood response, whereas SLR transfer affects the forcing within the same region. An effective adaptation mechanism must handle both from only a handful of labeled examples. Second, achieving this in an architecture-agnostic manner is difficult, since the surrogate models can span various learned representations, from graph and mesh networks to vision, state-space and diffusion models. As Lee et al. (2022) illustrate, with limited target data the choice of which parameters to update matters, and the best choice depends on the type of shift. Parameter-efficient fine-tuning (PEFT) methods, such as LoRA (Hu et al., 2021), BitFit (Zaken et al., 2022), and IA^3 (Liu et al., 2022a), restrict updates to selected parameters or inserted modules. These methods specify where and how a source model changes, but do not encode any flood-specific relation between PWL and the physical variables that remain observable in the target domain. In this context, we treat parameter efficiency and physical structure as distinct, potentially complementary components of adaptation.

A recent article by Daramola et al. (2026) argues that transferable coastal flood models require inductive biases reflecting the underlying hydrodynamic processes, rather than solely relying on statistical similarity between source and target regions. In line with this view, we anchor our approach on terrain elevation, which is readily available for coastal regions and is directly linked to inundation extent and dynamics. More concretely, we introduce a lightweight module, termed the *Physics Adapter* (PA), that combines elevation with the features of a pretrained surrogate model (hereafter, also referred to as backbone) to predict PWL. A physics-guided branch predicts PWL through a differentiable wet/dry response that compares terrain elevation against a learned water level, while a parallel data-driven branch captures effects that terrain alone does not explain. A learned gate combines the two predictions. As PA requires only backbone features and elevation, it can be attached to any architecture, with only the integration interface tailored to each backbone. Unlike physics-informed formulations such as PINNs (Raissi et al., 2019), PA does *not* embed the shallow-water or Navier Stokes equations, minimize PDE residuals, or enforce mass or momentum conservation. Terrain elevation thus serves PA as an *architectural* inductive bias rather than a *governing-equation constraint*.

We instantiate PA across 12 diverse backbones and evaluate it on two coastal regions with markedly different geometries, topographies, and shoreline protection configurations, namely the coastal city of Abu Dhabi (AD) and the San Francisco (SF) Bay Area. Our main contributions are as follows:

- **A lightweight, architecture-agnostic physics-guided adapter:** PA injects terrain-based physical structure into pretrained flood surrogates without PDE-residual or conservation losses, while adding a negligible number of trainable parameters.
- **Extensive Evaluation:** We evaluate twelve backbones spanning graph, dense-vision, state-space, foundation, and diffusion models under bidirectional cross-region transfer (SF↔AD) and within-region SLR transfer, comparing ten adaptation regimes (full fine-tuning, head-only adaptation, and three PEFT methods, each with and without PA) under a matched protocol.
- **Empirical evidence that physical structure complements parameter efficiency:** Averaged across backbones and transfer settings, every regime with PA outperforms every regime without it once a single target simulation is available, and adapting PA alone surpasses full fine-tuning without PA. At the level of individual backbones, a PA regime remains the best parameter-efficient choice for nine of the twelve models.

2 Related Work

Learned surrogates for flood prediction. For coastal domains, DL-based surrogates have been developed for predicting extreme storm surge under future climate scenarios (Longo et al., 2026; Rice et al., 2025; Gharehtoragh & Johnson, 2024), spatiotemporal flood dynamics (Bian et al., 2025), and tidal and riverine shallow-water dynamics (Rivera-Casillas et al., 2025). The CASPIAN framework (Karapetyan et al., 2026) and its follow-up (Hassan et al., 2026) predict PWL under shoreline protection for SF and AD across multiple SLR conditions, and their publicly released

dataset serves as the data source for this work. Most of these surrogates, however, are trained and evaluated within a single region. Only a few studies have examined how coastal surrogates transfer to regions unseen during training. Zhao et al. (2026), for instance, report zero-shot generalization of a storm-surge model to unseen bays along the *same coastline*. Similar efforts for urban and riverine flooding adapt neural-operator surrogates to new catchments or forcing conditions via transfer learning (Xu et al., 2025) or domain adaptation (Taghizadeh et al., 2025a). These studies, however, examine transfer within a single surrogate design. On the other hand, the present work investigates whether one physics-guided adaptation strategy, shared across multiple backbone model families, can recover target-domain performance from only a few labeled target scenarios, including across geographically and hydrodynamically distinct coastlines.

Physics-guided learning. Physical constraints can enter a learned surrogate through different mechanisms. PINNs impose governing equations through residual-based objectives (Raissi et al., 2019), and physics-informed neural operators combine operator learning with PDE constraints (Li et al., 2024). Flood-specific models can encode more domain-specific hydraulic structure. HydroGraphNet (Taghizadeh et al., 2025b) includes mass conservation in its training objective, whereas hydraulics-informed message passing derives graph interactions from shallow-water structure (Kazadi et al., 2024). Beyond flooding, physical structure can also be built into the architecture itself, as in ClimODE (Verma et al., 2024), which encodes advection within continuous-time weather dynamics. Closest to our setting, GeoAda-PINN (Zhu et al., 2026) freezes a PINN backbone and updates compact geometry-conditioned adapters to handle geometric changes. In contrast, the proposed adapter uses a lighter architectural inductive bias based on terrain elevation and a differentiable wet/dry response, without any governing-equation loss or conservation guarantee. Moreover, PA employs a consistent formulation across heterogeneous backbone families, whereas GeoAda-PINN is tied to a single PINN architecture.

Adaptation under distribution shift and PEFT. When labeled target data are scarce, a key question is which source parameters should be updated. Surgical fine-tuning shows that the effective subset can depend on the type of distribution shift (Lee et al., 2022). Unsupervised test-time methods instead adapt from unlabeled target batches during inference (Wang et al., 2020). This differs from the proposed supervised few-shot setting, where labels are available for the target support scenarios. PEFT controls the optimization subspace through mechanisms such as bottleneck adapters (Houlsby et al., 2019), low-rank weight updates (Hu et al., 2021), bias-only tuning (Zaken et al., 2022), and activation scaling (Liu et al., 2022a). Architecture-aware PEFT has also been studied for state-space models (Yoshimura et al., 2025), and F-Adapter extends this line to large neural-operator models for scientific machine learning (Zhang et al., 2026). These approaches alter the parameterization of adaptation, whereas the PA instead adds elevation-conditioned structure to the prediction interface. We therefore evaluate PA and PEFT both separately and in combination across heterogeneous neural surrogates.

3 METHOD

This section details the proposed approach and its use for target adaptation. We separate how each backbone represents a flood scenario from how that representation is converted into elevation-conditioned PWL, so that this conversion can be shared across all twelve backbones. Sec. 3.1 defines the prediction problem and this separation, Sec. 3.2 describes the adapter, Sec. 3.3 explains how it is attached to each backbone, and Sec. 3.4 defines source training and the adaptation regimes.

3.1 PROBLEM FORMULATION

A prediction domain is a pair $d = (r, \lambda)$ of a geographic region r and a forcing condition λ, which we vary through SLR. For region r, let Ω_r denote the discrete prediction sites and $\Omega_r^v \subseteq \Omega_r$ the valid sites. A flood scenario has input $\mathbf{x}$ and target $\mathbf{y} = \{y_i\}_{i \in \Omega_r^v}$, where $y_i \geq 0$ is the simulated PWL. Raster backbones take a four-channel 1024×1024 tensor $\mathbf{x}_{\text{grid}} = [\mathbf{c}^{(\mathbf{s})}, \mathbf{z}, \boldsymbol{\ell}, \mathbf{v}]$ containing scenario-dependent protection status, the DEM, land cover, and a binary validity mask $v_i \in \{0, 1\}$. Graph and mesh backbones encode the same variables as node features on the valid sites, so the mask is implicit (Appendix A). In every case, the PA receives the raw DEM value z_i in physical units, rather than a normalized or embedded version recovered from backbone features.

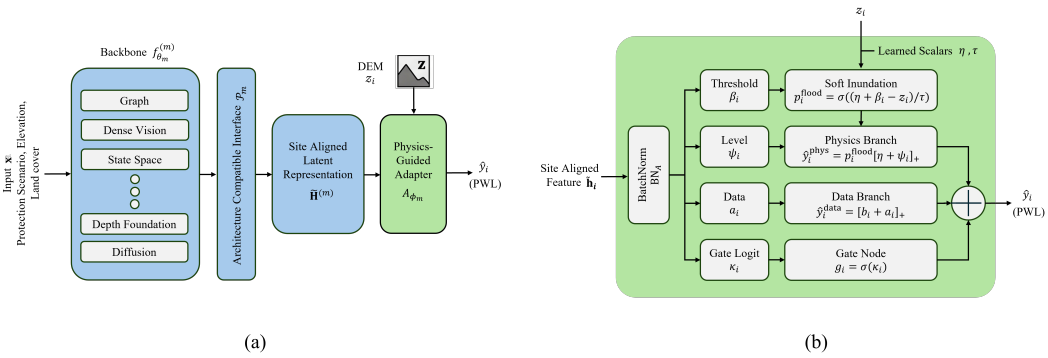


Figure 1: Backbone-agnostic Physics Adapter. (a) Each backbone exposes features aligned with the PWL prediction sites through its own interface, and the shared adapter produces the prediction. (b) Inside the adapter, raw terrain elevation enters the physics-guided branch through a soft wet/dry response, a parallel data-driven branch captures effects that terrain alone does not explain, and a learned site-wise gate combines the two. Feature extraction is architecture-specific, while terrain conditioning, the two branches, gated fusion, and target adaptation are shared.

Each of the twelve backbones $m \in \mathcal{M}$ (Sec. 4.2) defines a native representation, a site-alignment interface $\mathcal{P}_m : \mathcal{H}_m \to \mathbb{R}^{|\Omega_r| \times C_m}$, and the shared adapter computation

$$\begin{aligned} \mathbf{h}^{(m)} &= f_{\theta_m}^{(m)}(\mathbf{x}), \\ \widetilde{\mathbf{H}}^{(m)} &= \mathcal{P}_m\big(\mathbf{h}^{(m)}\big), \qquad \widetilde{\mathbf{h}}_i^{(m)} \in \mathbb{R}^{C_m}, \\ \widehat{\mathbf{y}}^{(m)} &= A_{\phi_m}\big(\widetilde{\mathbf{H}}^{(m)}, \mathbf{z}, \mathbf{v}\big). \end{aligned} \tag{1}$$

The interface $\mathcal{P}_m$ (attachment point, tensor layout, feature dimension, decoder, interpolation, and projection) differs across backbones. Its parameters are grouped with the backbone parameters θ_m, except in the diffusion case, where the post-decoder stem belongs to the adapter (Sec. 3.3). The adapter parameters are ϕ_m, so the full PA model has parameters $\Theta^{(m)} = \theta_m \cup \phi_m$. Here θ_m determines how a scenario is represented, and ϕ_m determines how that representation is converted into elevation-conditioned PWL. Figure 1 summarizes the formulation.

3.2 Physics-Guided Adapter

Adapter heads. The site-aligned features are first normalized by a BatchNorm layer, and four lightweight representation-compatible functions then produce a threshold correction (β_i), a water-level correction (ψ_i), a data-branch correction (a_i), and a gate logit (κ_i),

$$\begin{aligned} \mathbf{h}_i &= \mathrm{BN}_A^{(m)}\big(\widetilde{\mathbf{h}}_i^{(m)}\big), \\ \beta_i = f_\beta^{(m)}(\mathbf{h}_i), \quad \psi_i &= f_\psi^{(m)}(\mathbf{h}_i), \quad a_i = f_a^{(m)}(\mathbf{h}_i), \quad \kappa_i = f_g^{(m)}(\mathbf{h}_i). \end{aligned} \tag{2}$$

For dense feature maps, the heads are pointwise 1×1 convolutions, and for node-aligned backbones they are node-wise two-layer multilayer perceptrons (MLPs). The running statistics of $\mathrm{BN}_A^{(m)}$ are part of the adapter state and are recalibrated during target adaptation (Sec. 3.4). Backbone-specific settings, including affine BatchNorm parameters and head dropout are listed in Appendix B.

Elevation-conditioned inundation. A learned scalar $\eta \in \mathbb{R}$ defines a reference water level, and a second scalar ϑ parameterizes a positive transition temperature. The local inundation response is

$$p_i^{\mathrm{flood}} = \sigma\left(\frac{\eta + \beta_i - z_i}{\tau}\right), \qquad \tau = \max\big\{\exp(\vartheta),\, 10^{-3}\big\}, \tag{3}$$

where z_i is the raw DEM elevation and σ is the logistic sigmoid. The correction β_i shifts the effective inundation threshold based on the backbone features, and τ controls how sharp the wet/dry transition is. Both η and ϑ are learned, with region-specific initialization given in Appendix B.

Physics and data branches. The two prediction branches are

$$\widehat{y}_i^{\mathrm{phys}} = p_i^{\mathrm{flood}} \left[\eta + \psi_i\right]_+, \qquad \widehat{y}_i^{\mathrm{data}} = \left[b_i^{(m)} + a_i\right]_+, \tag{4}$$

where $[\cdot]_+ = \max(\cdot, 0)$. The correction ψ_i adjusts the water level separately from the threshold shift β_i in Eq. (3), so the physics-guided branch ties PWL to absolute terrain elevation even when the backbone does not preserve DEM units. For eleven backbones $b_i^{(m)} = 0$, and the data branch is a direct non-negative prediction. For the diffusion backbone, $b_i^{(m)}$ is the generated base PWL, so the data branch predicts a residual around the generated map (Sec. 3.3).

Gated fusion and masking. A site-wise gate mixes the branches, and invalid raster cells are removed from the output,

$$g_i = \sigma(\kappa_i), \qquad \widetilde{y}_i = g_i\, \widehat{y}_i^{\mathrm{phys}} + (1 - g_i)\, \widehat{y}_i^{\mathrm{data}}, \qquad \widehat{y}_i = v_i\, \widetilde{y}_i. \tag{5}$$

For graph and mesh backbones, nodes already coincide with prediction sites, so masking is implicit. The gate is initialized toward the physics-guided branch with a logit of $+3$, so that $g_i = \sigma(3) \approx 0.95$ at the start of training while the gate gradient remains large enough for the model to shift weight toward the data branch where needed. This value is a fixed design choice rather than a tuned hyperparameter (Appendix B).

Eqs. (3)–(5) thus act as an elevation-conditioned *architectural inductive bias*, *not* a hydrodynamic solver, and add no PDE-residual or conservation constraint. In the same sense, the PA is not a new backbone, since ϕ_m only converts an exposed representation into PWL. It is also distinct from parameter-efficient fine-tuning (PEFT) methods such as LoRA, BitFit and IA^3, which modify selected weights or activations inside an existing model. The two can therefore be used separately or together, and the adaptation regimes in Sec. 3.4 compare both options.

3.3 Backbone-Compatible Instantiation

The interfaces $\mathcal{P}_m$ align features with the prediction sites but do not make the architectures identical, so tensor shape, insertion depth, feature dimension, and PEFT target modules differ across backbones. Node-aligned backbones apply the adapter node-wise, dense backbones use pointwise heads on grid-aligned decoder features after a projection where needed, and the diffusion backbone passes its decoded base PWL map through an adapter-owned stem. In all cases the raw DEM bypasses the backbone and enters Eq. (3) directly, so the PA is model-agnostic in its formulation rather than in its placement inside each network (Appendix B, Table 2).

3.4 Source Training and Target Adaptation

Objective. Deterministic backbones use masked mean squared error over valid sites. PA models add a penalty that discourages the gate from collapsing onto the data-driven branch,

$$\begin{aligned} \mathcal{L}_{\mathrm{pred}} &= \frac{\sum_{n\in\mathcal{B}} \sum_{i\in\Omega_r} v_{n,i} \big(\widehat{y}_{n,i} - y_{n,i}\big)^2}{\max\big(1,\ \sum_{n\in\mathcal{B}} \sum_{i\in\Omega_r} v_{n,i}\big)}, \\ \mathcal{L} &= \mathcal{L}_{\mathrm{pred}} + \mathbb{I}_{\mathrm{PA}}\, \frac{\lambda_g}{|\Omega_A|} \sum_{i\in\Omega_A} (1 - g_i), \qquad \lambda_g = 0.1, \end{aligned} \tag{6}$$

where $\mathcal{B}$ is a mini-batch, $\mathbb{I}_{\mathrm{PA}} = 1$ only for PA models, and Ω_A is the set of sites on which the gate is defined (Appendix B). For node-based models, $\mathcal{L}_{\mathrm{pred}}$ reduces to standard MSE over valid nodes. The diffusion backbone keeps its native diffusion objective alongside the map-level objective, with the decoded base PWL map detached from the diffusion computation (Appendix B).

Source training. For each backbone, the PA source model is trained jointly on the source domain,

$$(\theta_{m,s}^{\star}, \phi_{m,s}^{\star}) = \arg\min_{\theta_m, \phi_m} \ \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}_{\mathrm{train}}^{s}} \big[\mathcal{L}(\mathbf{x}, \mathbf{y}; \theta_m, \phi_m)\big], \tag{7}$$

so PA-only adaptation starts from an adapter trained together with its source representation rather than a newly initialized module. We also train a raw source model, in which the PA is replaced by a direct prediction head with parameters χ_m, trained on $\mathcal{L}_{\mathrm{pred}}$ alone, which serves as the no-PA control (Appendix B). We write $\Theta_s^{(m)\star}$ for the selected source parameters of either model.

Target adaptation. Let $\mathcal{S}_K^t = \{(\mathbf{x}_j^t, \mathbf{y}_j^t)\}_{j=1}^K$ be the labeled target support set, disjoint from the held-out target test set. An adaptation regime a specifies a trainable subset $\Theta_a^{(m)}$ and is optimized from the corresponding source checkpoint,

$$\widehat{\Theta}_{a,K}^{(m)} = \arg\min_{\Theta_a^{(m)}} \mathcal{L}_t\big(\mathcal{S}_K^t;\, \Theta_a^{(m)},\, \Theta_{\neg a}^{(m),\mathrm{frozen}}\big), \tag{8}$$

where $\mathcal{L}_t$ is the objective of Eq. (6) evaluated on $\mathcal{S}_K^t$. For PEFT regimes, $\Theta_a^{(m)}$ also includes the injected PEFT parameters $\xi_{m,a}$. The ten regimes and their trainable sets are defined in Sec. 4.2. Frozen components are frozen in both parameters and internal state, and for every PA regime with $K > 0$, adaptation starts with a gradient-free recalibration of the adapter BatchNorm on the support inputs (Appendix C). At $K = 0$, no recalibration or gradient update is performed, so $\widehat{\Theta}_{a,0}^{(m)} = \Theta_s^{(m)\star}$ and zero-shot results directly evaluate the source checkpoint.

We consider two shifts. Geographic transfer changes the region ($r_s \neq r_t$), whereas SLR transfer keeps the region fixed and changes only the forcing ($r_s = r_t$, $\lambda_s \neq \lambda_t$). Transfer directions, support sizes, and the adaptation budget are given in Sec. 4.3.

4 Experimental Setup

4.1 Datasets and Representations

We use the publicly released coastal-flood simulations of the CASPIAN studies for AD and SF (Karapetyan et al., 2026; Hassan et al., 2026), produced with Delft3D under defined SLR, tidal forcing, and binary shoreline-protection configurations (Appendix A). Each protection scenario protects a subset of N_r operational landscape units (OLUs), with $N_{\mathrm{SF}} = 30$ and $N_{\mathrm{AD}} = 17$. The regional datasets contain 285 SF scenarios at $1.0\,\mathrm{m}$ SLR and 142 AD scenarios at $0.5\,\mathrm{m}$ SLR, and two SF SLR-transfer targets at $0.5\,\mathrm{m}$ and $1.5\,\mathrm{m}$ contain 32 scenarios each. Each retained coastal location carries a scenario-dependent protection status derived from simulated single-OLU responses, together with elevation from Copernicus DEM GLO-30 (European Space Agency, 2022) and land cover from ESA WorldCover (Zanaga et al., 2022), which we sample at every location since the original datasets do not include them. Dense backbones use the 1024×1024 tensor of Sec. 3.1, and graph backbones use a graph over the same locations whose edges follow the hydrodynamic mesh. Both are encodings of the same samples rather than separate datasets (Appendix A, Figure 4).

4.2 Benchmark Models and Comparison Methods

The twelve backbones cover clearly different model families. These are graph and mesh models (GCN (Kipf & Welling, 2017), GAT (Veličković et al., 2018), and MeshGraphNet (MGN) (Pfaff et al., 2021)), scientific attention over physical sites (Transolver++ (Luo et al., 2025)), dense vision models (CASPIAN (Karapetyan et al., 2026), ConvNeXt V2 (Woo et al., 2023), MaxViT (Tu et al., 2022), and Swin Transformer V2 (Liu et al., 2022b)), a visual state-space model (VM-UNet with a VMamba encoder–decoder (Ruan et al., 2024; Liu et al., 2024)), pretrained depth-foundation models (Depth Anything V2 (Yang et al., 2024) and Depth Pro (Bochkovskiy et al., 2025)), and a conditional diffusion model (ControlNet (Zhang et al., 2023)). The depth-foundation models are adapted to PWL through task-specific input and feature interfaces, not by reinterpreting their depth outputs. Table 2 in Appendix B lists the representation each backbone passes to the PA.

Each backbone has two matched source models. The *PA source* model trains the backbone and PA jointly (Eq. (7)), and the *raw source* model replaces the PA with a direct prediction head χ_m. We compare ten target-adaptation regimes per backbone. Full fine-tuning is run with the PA (FT+PA) and without it (FT). Partial fine-tuning without PA (NPA) trains only χ_m, and PA-only adaptation (PA) trains only ϕ_m. Three PEFT methods, namely LoRA (Hu et al., 2021) with rank $r = 8$, BitFit (Zaken et al., 2022), and IA^3 (Liu et al., 2022a), train only their injected parameters $\xi_{m,a}$, both on their own (PEFT) and combined with the PA (PEFT+PA). Table 3 in Appendix C gives the trainable and frozen parameters of each regime, along with the PEFT formulations and insertion sites. All regimes share the same scenario manifests, support sets, test sets, seeds, and metric code, while batch sizes, PEFT insertion sites, and parameter counts remain architecture-specific.

Table 1: Per-backbone results. In-domain values report the better PA or raw source model over SF and AD. Transfer results report the lowest mean RMSE over all K and all four settings, with and without full fine-tuning. Values are mean ± std over three seeds. Bold shows the best results.

		In-domain		Transfer (mean over K)			
Backbone	**Source**	**RMSE (m)**	R^2	**Best regime**	**RMSE**	**Best without full FT**	**RMSE**
VM-UNet	PA	**0.0542** ± 0.0015	**0.9660** ± 0.0008	FT+PA	0.1421	LoRA	**0.1467**
Swin V2	PA	0.0630 ± 0.0025	0.9582 ± 0.0055	FT+PA	0.1635	LoRA+PA	0.1662
MaxViT	PA	0.0639 ± 0.0033	0.9564 ± 0.0039	FT+PA	0.1663	LoRA+PA	0.1749
CASPIAN	PA	0.0686 ± 0.0036	0.9560 ± 0.0060	FT+PA	**0.1388**	LoRA+PA	0.1477
Depth Pro	PA	0.0713 ± 0.0046	0.9522 ± 0.0005	FT+PA	0.2245	LoRA	0.2423
Depth Anything V2	PA	0.0747 ± 0.0017	0.9503 ± 0.0052	FT+PA	0.2200	LoRA	0.2340
ControlNet	PA	0.0754 ± 0.0030	0.9244 ± 0.0096	FT+PA	0.1637	LoRA+PA	0.1722
ConvNeXt V2	PA	0.0893 ± 0.0028	0.9217 ± 0.0157	FT+PA	0.1648	LoRA+PA	0.1663
MGN	PA	0.2990 ± 0.0034	0.9266 ± 0.0011	FT+PA	0.6199	IA^3+PA	0.6449
GAT	Raw	0.3106 ± 0.0019	0.9235 ± 0.0007	FT+PA	0.4950	BitFit+PA	0.5760
Transolver++	Raw	0.3417 ± 0.0052	0.9215 ± 0.0079	LoRA+PA	0.7125	LoRA+PA	0.7125
GCN	PA	0.3418 ± 0.0020	0.9121 ± 0.0009	FT+PA	0.5209	BitFit+PA	0.6456

4.3 Splits and Transfer Protocols

All backbones and seeds share one fixed scenario split, stratified by protection level into approximately $60/20/20$ train, validation, and test sets, and model selection uses the validation split only. Geographic transfer is evaluated in both directions between SF and AD with $K \in \{0, 1, 3, 5, 10\}$, and SLR transfer from $\mathrm{SF}_{1.0}$ to $\mathrm{SF}_{0.5}$ and $\mathrm{SF}_{1.5}$ with $K \in \{0, 1, 3, 5, 10\}$. For each $K > 0$, eight support draws are adapted independently from the source checkpoint with a fixed budget of 50 support passes and no target validation, and all are evaluated on the same held-out target test set (Appendix D, Table 4).

4.4 Hyperparameter Optimization , Training And Evaluation Metrics

For the details on hyperparameter optimization and training, we refer the reader to Appendix E.

We report the mean absolute error (MAE), RMSE, the coefficient of determination (R^2), the drypoint accuracy (Acc_0), the relative total absolute error (RTAE), and the error exceedance rates $\delta_{0.5}$ and $\delta_{0.1}$. All metrics are computed at the same retained locations for every model, and their definitions and aggregation are given in Appendix F.

5 Results

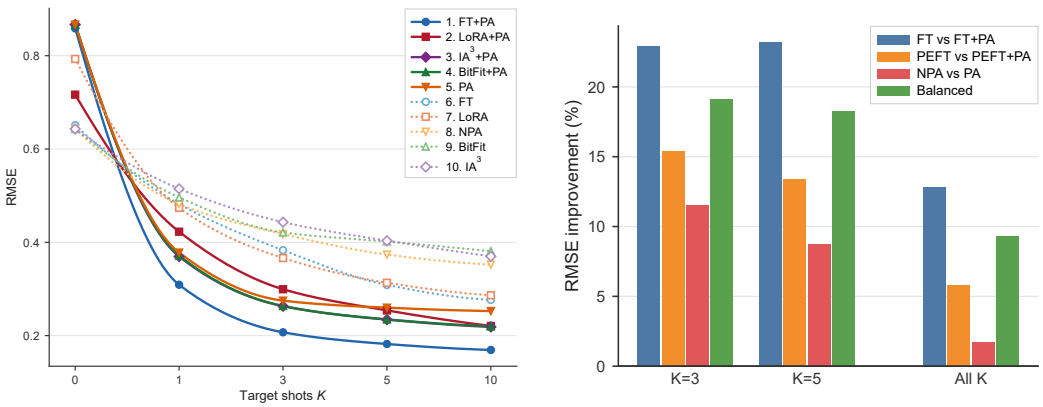
5.1 In-Domain Prediction

Table 1 reports in-domain RMSE and R^2 for each backbone, averaged over the SF and AD test sets. For every backbone we trained both a PA source model and a raw source model, and the table shows whichever of the two had the lower combined RMSE. The PA source model is the better of the two for ten of the twelve backbones. The two exceptions are GAT and Transolver++, where the raw model is slightly better in domain. The PA therefore does not cost in-domain accuracy in most cases, even though its main purpose is transfer.

VM-UNet is the most accurate backbone in both regions, with a combined RMSE of 0.054 m and $R^2 = 0.966$. Swin V2, MaxViT, and CASPIAN follow closely, and the two depth-foundation models come next. The graph and operator backbones reach a similar R^2 of about 0.92, but their absolute errors are several times larger than those of the dense backbones, so we compare RMSE mainly within each family. Across all backbones, errors are lower in SF than in AD, which is consistent with the stronger wave forcing and run-up in the AD simulations (Appendix A.1). Full results for all seven metrics, per region, are given in Appendix G.1.

5.2 Few-Shot Transfer

We first compare the ten regimes averaged over all twelve backbones and all four transfer settings (SF→AD, AD→SF, $\mathrm{SF}_{1.0}$→$\mathrm{SF}_{0.5}$, and $\mathrm{SF}_{1.0}$→$\mathrm{SF}_{1.5}$). Since the SLR targets stop at $K = 10$, this



(a) (b)

Figure 2: Few-shot transfer averaged over 12 backbones and four settings. (a) RMSE across K for the ten adaptation regimes, ranked by mean RMSE. Solid lines use PA; dotted lines do not. (b) RMSE reduction from PA at $K = 3$, $K = 5$, and across all K. "Balanced" averages the three pairings.

comparison uses $K \leq 10$. Figure 2a shows the resulting RMSE curves, with regimes ranked by their mean RMSE over K.

Regimes with the PA. The five regimes that include the PA take the top five places, and the five regimes without it take the bottom five. From $K = 1$ onward, every PA regime has a lower RMSE than every non-PA regime at every value of K. FT+PA is the best regime overall, reducing RMSE from 0.856 m at $K = 0$ to 0.169 m at $K = 10$. The more useful comparison for practice is PA-only adaptation, which updates only the adapter parameters ϕ_m. It has a lower RMSE than full fine-tuning without the PA at every $K \geq 1$, even though FT updates the whole backbone. Among the PEFT methods, adding the PA lowers the error for LoRA, BitFit, and IA^3 alike, and the three PEFT+PA regimes end close to each other at $K = 10$.

Zero-shot behavior. At $K = 0$ the ordering is reversed, and the raw source models transfer better than the PA source models (RMSE of about 0.674 m against 0.856 m). A likely reason is that the reference water level η in Eq. (3) is learned for the source region, so without any target data the terrain comparison is made against the wrong water level. A single labeled target scenario is enough to reverse this, and at $K = 1$ all PA regimes are already ahead. In practice, the PA should be used with at least one target simulation, and zero-shot use requires care.

Where the gain comes from. Figure 2b isolates the effect of the PA by comparing each regime with its matched counterpart, namely FT with FT+PA, each PEFT method with its PEFT+PA version, and NPA with PA. The PA lowers RMSE by 22.87% for full fine-tuning, 15.39% for PEFT, and 11.51% for head-only adaptation at $K = 3$, and by 12.77%, 5.78%, and 1.68% averaged over all K. The NPA against PA pairing is the cleanest test, since both regimes train only a small head on a frozen backbone and differ only in whether that head is conditioned on terrain. This gain points to the terrain conditioning as the main source of the improvement, although the two heads also differ somewhat in size (Appendix F).

Per-backbone results. The last four columns of Table 1 show the best regime for each backbone. FT+PA is the best choice for eleven of the twelve backbones, and LoRA+PA is best for Transolver++. Because full fine-tuning is the most expensive option, we also report the best regime when both full fine-tuning regimes are excluded. In this case, a regime with the PA is still best for nine of the twelve backbones. LoRA+PA is preferred by most dense backbones, while the graph backbones prefer the lighter BitFit+PA and IA^3+PA. The three exceptions, VM-UNet, Depth Pro, and Depth Anything V2, prefer LoRA without the PA, and for these the gap to LoRA+PA is 0.03 m or less. The full per-backbone curves are given in Appendix G.2.

5.3 Qualitative Results

Figure 3 compares VM-UNet PWL predictions for one held-out case per region, using the in-domain model and four $K = 3$ transfer regimes. In AD, the in-domain model matches the flooded area within 1.5%. After SF-to-AD transfer, FT and LoRA without PA overpredict flooding by 63.5% and 26.3%, with false wet patches over dry inland areas. Adding PA largely removes these errors, reducing the flooded-area error to 13.3% for FT+PA and 7.8% for PEFT+PA. This is consistent with Eq. (3), where elevated terrain remains dry unless the learned features support flooding. It also shows that lower RMSE does not always imply a better flood extent. In SF, all AD-to-SF regimes recover the flooded area within 5%, with differences mainly in predicted water levels inside flooded regions. Error maps are provided in Appendix G.3.

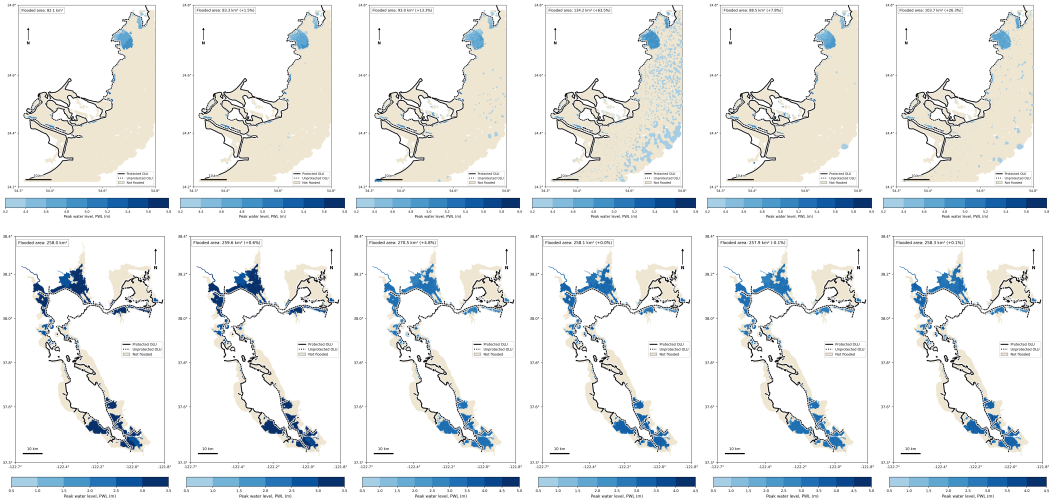


Figure 3: VM-UNet PWL predictions for held-out AD (top) and SF (bottom) scenarios. (a) Ground truth, (b) in-domain, and $K = 3$ transfer using (c) FT+PA, (d) FT, (e) best PEFT+PA, and (f) best PEFT. Transfer directions are SF→AD and AD→SF, respectively. Each panel reports flooded area and its difference from ground truth. Color scales vary across panels.

6 Conclusion

We introduced the Physics Adapter, a small module that conditions coastal-flood predictions on terrain elevation through a differentiable wet/dry response and a gated physics-guided branch. The same formulation was attached to twelve backbones from graph, mesh, operator, vision, state-space, depth-foundation, and diffusion families and tested on geographic and SLR transfer with up to 10 labeled target scenarios. With at least one target scenario, every regime that includes the PA outperforms every regime without it in our aggregate comparison. Training the adapter alone is already better than fully fine-tuning a backbone without it, and adding it on top of LoRA, BitFit, or IA^3 improves each of them. The PA also keeps or improves in-domain accuracy for ten of the twelve backbones.

Limitations. This study covers two regions, and SLR transfer is tested only within San Francisco, so broader claims need more regions and hazard types. The PA encodes a terrain comparison and not hydrodynamics, so it gives no guarantee of mass or momentum conservation. Without target data, PA source models transfer worse than raw ones, which limits zero-shot use. Finally, the interfaces, batch sizes, and PEFT insertion sites differ across architectures by design, so small differences between backbones should be read with care.

AI USE STATEMENT

In this paper, we used generative AI tools for editing and rephrasing the text to improve grammar and readability, and for drafting and editing the source code for experiments and visualization. We have not used generative AI tools to generate synthetic data sets; help develop theoretical models or conceptual frameworks; formulate mathematical claims; provide critical ingredients for proving mathematical claims; assist in the writing of proofs; propose or refine hypotheses; design or provide feedback on research methodology or experiments; implement methods; assist with translation; clean and reformat dataset; support qualitative and thematic data analysis; and interpret results. All AI-paraphrased or AI-edited text was reviewed and revised by the authors. We take full responsibility for the final content of this work, including all text and claims produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects, personal data, or privacy-sensitive information. All experiments use numerical hydrodynamic simulations and publicly available geospatial datasets, used in accordance with their respective licenses. We are not aware of any conflicts of interest or other ethical concerns associated with this work.

REPRODUCIBILITY STATEMENT

We will release the full codebase, including all scripts for data preprocessing, source training, target adaptation, evaluation, and the reproduction of every table and figure in this paper. Given the size of the codebase, which spans twelve backbones and ten adaptation configurations, we are currently consolidating and documenting it, and we will share an anonymized repository link with the reviewers during the discussion period. In the meantime, the paper provides the details needed to re-implement our method and experiments.

REFERENCES

Wanchao Bian, Jiayi Fang, Pin Wang, Qinke Sun, Jian Fang, Feng Kong, and Tangao Hu. Deep learning surrogate models for spatiotemporal prediction of coastal flooding inundations in tianjin, china. *Journal of Hydrology: Regional Studies*, 60:102593, 2025.

Aleksei Bochkovskiy, Amaël Delaunoy, Hugo Germain, Marcel Santos, Yichao Zhou, Stephan R. Richter, and Vladlen Koltun. Depth pro: Sharp monocular metric depth in less than a second. In *International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=aueXfY0Clv`.

N. Booij, R. C. Ris, and L. H. Holthuijsen. A third-generation wave model for coastal regions: 1. model description and validation. *Journal of Geophysical Research: Oceans*, 104(C4):7649–7666, 1999. doi: 10.1029/98JC02622.

Aaron C. Chow and Jian Sun. Combining sea level rise inundation impacts, tidal flooding and extreme wind events along the abu dhabi coastline. *Hydrology*, 9(8):143, 2022. doi: 10.3390/hydrology9080143.

Samuel Daramola, David F. Muñoz, and Chaopeng Shen. Toward transferable models for efficient spatiotemporal flood prediction across coastal-estuarine systems. *Cambridge Prisms: Coastal Futures*, 4:e13, 2026. doi: 10.1017/cft.2026.10037.

European Space Agency. Copernicus global digital elevation model (GLO-30). ESA Copernicus Data Space Ecosystem, 2022.

Mohammad Ahmadi Gharehtoragh and David R. Johnson. Using surrogate modeling to predict storm surge on evolving landscapes under climate change. *npj Natural Hazards*, 1:33, 2024. doi: 10.1038/s44304-024-00032-9.

Bilal Hassan, Areg Karapetyan, Aaron Chung Hin Chow, and Samer Madanat. Climate adaptation-aware flood prediction for coastal cities using deep learning. *Hydrology and Earth System Sciences*, 30(5):1333–1358, 2026.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean-Raymond Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. doi: 10.1002/qj.3803.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International conference on machine learning*, pp. 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. 2021.

Gaofeng Jia, Ruo Qian Wang, and Mark T. Stacey. Investigation of impact of shoreline alteration on coastal hydrodynamics using Dimension REduced Surrogate based Sensitivity Analysis. *Advances in Water Resources*, 126:168–175, 4 2019. ISSN 0309-1708. doi: 10.1016/J.ADVWATRES.2019.03.001.

Areg Karapetyan, Aaron CH Chow, and Samer Madanat. Deep vision-based framework for coastal flood prediction under sea level rise and shoreline protection. *Scientific Reports*, 16(1):3663, 2026.

Arnold Kazadi, James Doss-Gollin, and Arlei Lopes Da Silva. Pluvial flood emulation with hydraulics-informed message passing. In *Forty-first International Conference on Machine Learning*, 2024.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=SJU4ayYgl`.

Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. 2022.

Giles R Lesser, JA v Roelvink, JA Th M van Kester, and GS Stelling. Development and validation of a three-dimensional morphological model. *Coastal engineering*, 51(8-9):883–915, 2004.

Zongyi Li, Hongkai Zheng, Nikola Kovachki, David Jin, Haoxuan Chen, Burigede Liu, Kamyar Azizzadenesheli, and Anima Anandkumar. Physics-informed neural operator for learning partial differential equations. *ACM/IMS Journal of Data Science*, 1(3):1–27, 2024.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin Raffel. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. volume 35, pp. 1950–1965, 2022a.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. VMamba: Visual state space model. In *Advances in Neural Information Processing Systems*, volume 37, pp. 103031–103063, 2024. doi: 10.52202/079017-3273.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12009–12019, 2022b. doi: 10.1109/CVPR52688.2022.01170.

Emiliano Longo, Andrea Ficchì, Martin Verlaan, Sanne Muis, and Andrea Castelletti. A deep learning framework for extreme storm surge modeling under future climate scenarios. *Earth's Future*, 14(3):e2025EF007072, 2026. doi: 10.1029/2025EF007072.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 41432–41449. PMLR, 2025. URL https://proceedings.mlr.press/v267/luo25o.html.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=roNqYL0_XP.

Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational physics*, 378:686–707, 2019.

Julian R. Rice, Karthik Balaguru, Fadia Ticona Rollano, John Wilson, Brent Daniel, David Judi, Ning Sun, and L. Ruby Leung. Projecting U.S. coastal storm surge risks and impacts with deep learning. *Environmental Research Letters*, 20:104013, 2025. doi: 10.1088/1748-9326/adfd74.

Peter Rivera-Casillas, Sourav Dutta, Shukai Cai, Mark Loveland, Kamaljyoti Nath, Khemraj Shukla, Corey Trahan, Jonghyun Lee, Matthew Farthing, and Clint Dawson. A neural operator emulator for coastal and riverine shallow water dynamics. *arXiv preprint arXiv:2502.14782*, 2025.

Jiacheng Ruan, Jincheng Li, and Suncheng Xiang. VM-UNet: Vision mamba UNet for medical image segmentation. *arXiv preprint arXiv:2402.02491*, 2024. URL https://arxiv.org/abs/2402.02491.

Jiayun Sun, Aaron C. H. Chow, and Samer M. Madanat. Multimodal transportation system protection against sea level rise. *Transportation Research Part D: Transport and Environment*, 88: 102568, 2020.

Mehdi Taghizadeh, Zanko Zandsalimi, Mohammad Amin Nabian, Jonathan L Goodall, and Negin Alemazkoor. Floodforecaster: A domain-adaptive geometry-informed neural operator framework for rapid flood forecasting. *Journal of Hydrology*, pp. 134512, 2025a.

Mehdi Taghizadeh, Zanko Zandsalimi, Mohammad Amin Nabian, Majid Shafiee-Jood, and Negin Alemazkoor. Interpretable physics-informed graph neural networks for flood forecasting. *Computer-Aided Civil and Infrastructure Engineering*, 40(18):2629–2649, 2025b.

Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan C. Bovik, and Yinxiao Li. MaxViT: Multi-axis vision transformer. In *Computer Vision – ECCV 2022*, pp. 459–479. Springer, 2022. doi: 10.1007/978-3-031-20053-3_27.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJXMpikCZ.

Yogesh Verma, Markus Heinonen, and Vikas Garg. Climode: Climate and weather forecasting with physics-informed neural odes. In *International Conference on Learning Representations*, volume 2024, pp. 8408–8430, 2024.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. 2020.

Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. ConvNeXt V2: Co-designing and scaling convnets with masked autoencoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16133–16142, 2023. doi: 10.1109/CVPR52729.2023.01548.

Qingsong Xu, Leon Frederik De Vos, Yilei Shi, Nils Rüther, Axel Bronstert, and Xiao Xiang Zhu. Urban flood modeling and forecasting with deep neural operator and transfer learning. *Journal of Hydrology*, 661:133705, 2025.

Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems*, volume 37, pp. 21875–21911, 2024. doi: 10.52202/079017-0688.

Masakazu Yoshimura, Teruaki Hayashi, and Yota Maeda. Mambapeft: Exploring parameter-efficient fine-tuning for mamba. In *International Conference on Learning Representations*, volume 2025, pp. 94093–94117, 2025.

Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, 2022.

Daniele Zanaga, Ruben Van De Kerchove, Dirk Daems, Wanda De Keersmaecker, Carsten Brockmann, Grit Kirches, Jan Wevers, Oliver Cartus, Maurizio Santoro, Steffen Fritz, Myroslava Lesiv, Martin Herold, Nandin-Erdene Tsendbazar, Panpan Xu, Fabrizio Ramoino, and Olivier Arino. ESA WorldCover 10 m 2021 v200, 2022. URL `https://zenodo.org/records/7254221`.

Hangwei Zhang, Chun Kang, Yan Wang, and Difan Zou. F-adapter: Frequency-adaptive parameter-efficient fine-tuning in scientific machine learning. volume 38, pp. 111120–111162, 2026.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3836–3847, 2023. doi: 10.1109/ICCV51070.2023.00355.

Jinpai Zhao, Albert Cerrone, Eirik Valseth, Leendert Westerink, and Clint Dawson. Storm surge in color: Rgb-encoded physics-aware deep learning for storm surge forecasting. *Computational Geosciences*, 30(4):78, Aug 2026. ISSN 1573-1499. doi: 10.1007/s10596-026-10477-8. URL `https://doi.org/10.1007/s10596-026-10477-8`.

Kejun Zhu, Xiaoping Chen, Xin Ao, and Zhipeng He. Parameter-efficient transfer of physics-informed neural networks for buoyancy-driven enclosures via geometry-conditioned adapters. *Physics of Fluids*, 38(2), 2026.

APPENDIX

A DATA CONSTRUCTION

The datasets used in this research were built from physics-based coastal flood simulations over a common set of retained spatial locations. We filtered raw hydrodynamic outputs to the learning locations, and linked each location to its hydrodynamically derived shoreline-protection dependence, terrain elevation, land cover, and scenario-specific PWL. The resulting coordinate-level data were then encoded in two forms, a regular 1024×1024 spatial tensor for dense-grid backbones and a graph that keeps the neighborhood structure of the hydrodynamic computational grid for graph-native backbones. These are two representations of the same flood-prediction problem, not separately generated datasets. Figure 4 summarizes the pipeline.

A.1 HYDRODYNAMIC SIMULATION DATA

The ground-truth flood fields come from the hydrodynamic models described in the CASPIAN studies and their supplementary material (Hassan et al., 2026). In both regions, Delft3D (Lesser et al., 2004) was used to resolve time-varying coastal water levels over the computational domain under prescribed SLR, tidal forcing, shoreline-protection configurations, and the other regional forcings of the original setup. The simulator produces spatially resolved water-level time series, from which peak water level (PWL) is kept as the regression target.

The Abu Dhabi configuration also accounts for the wind and wave environment of the Arabian Gulf. The validated Delft3D model was forced with ERA5 winds (Hersbach et al., 2020), and its results were coupled to the SWAN spectral wave model (Booij et al., 1999) to represent wind-wave generation and nearshore wave transformation. The SWAN significant wave heights were then

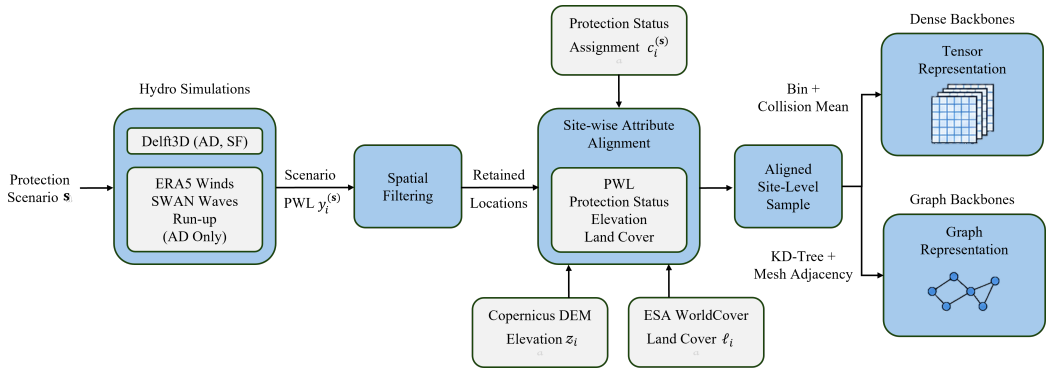


Figure 4: From hydrodynamic simulation to the two learning representations. Each retained coastal location carries scenario-dependent OLU status, elevation, and land cover, with simulated PWL as the target. Dense backbones use the rasterized tensor, and graph backbones use the mesh-topology graph built over the same locations.

combined with local shoreline slope to estimate coastal run-up under conditions typical of prolonged Shamal events (Chow & Sun, 2022). San Francisco Bay is treated differently because its shoreline lies inside a sheltered bay. The CASPIAN study did not apply SWAN there, and Delft3D alone was used to generate the SLR-driven flood fields (Sun et al., 2020).

The shoreline of region r is divided into N_r operational landscape units (OLUs), with $N_{\mathrm{AD}} = 17$ and $N_{\mathrm{SF}} = 30$. A protection configuration is a binary vector $\mathbf{s} = (s_1, \ldots, s_{N_r})$, with $s_k = 1$ when OLU k is protected and $s_k = 0$ otherwise, and it is realized in the hydrodynamic model through the corresponding shoreline-defense setup. Figures 5 and 6 show the OLUs of each region and the flooding produced when none of them is protected. For scenario $\mathbf{s}$, the simulator output used for learning is the set

$$\mathcal{R}^{(\mathbf{s})} = \left\{ \left(\mathbf{r}_i,\, y_i^{(\mathbf{s})} \right) \right\}_{i=1}^{N}, \qquad \mathbf{r}_i = (x_i, y_i^{\mathrm{coord}}), \tag{9}$$

where $\mathbf{r}_i$ is a simulator location and $y_i^{(\mathbf{s})}$ its PWL. Small negative PWL values in intermediate files are clipped to zero during preprocessing. The terrain and bathymetry used inside Delft3D belong to the hydrodynamic model and are separate from the elevation feature of Appendix A.3, which is sampled independently for the learning representation.

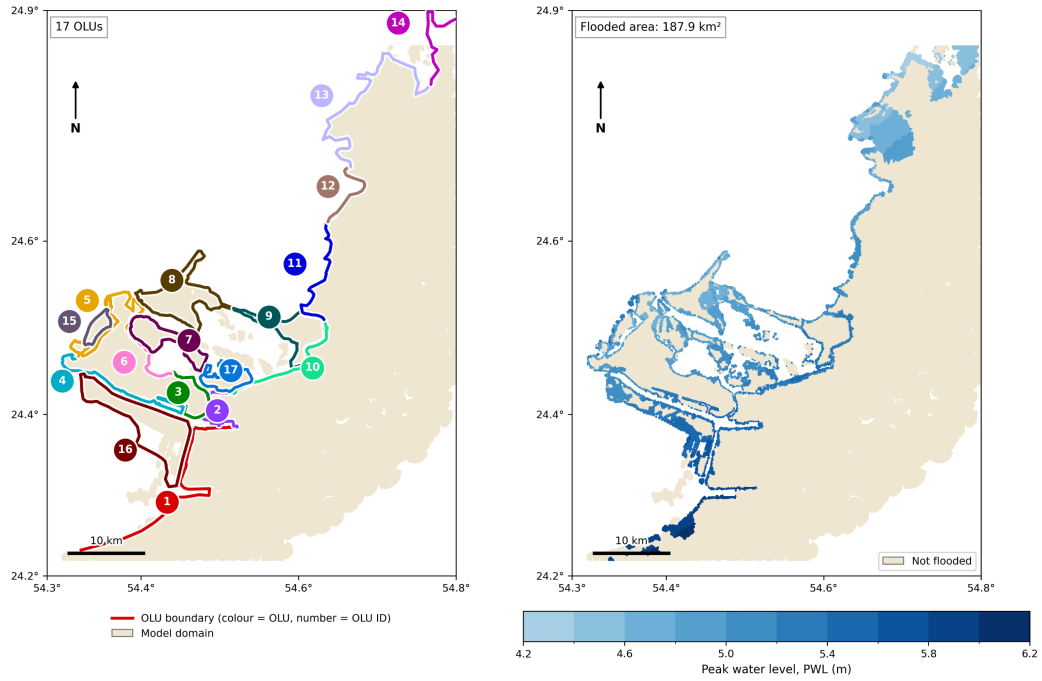
Spatial curation. The full hydrodynamic domain contains locations that are not prediction sites for the learning task. Simulator coordinates were therefore curated to retain study-relevant coastal locations and exclude offshore, open-water, and other non-target parts of the domain. The representation scripts start from the resulting region-specific master coordinate sets and match each scenario to them. PWL, elevation, land cover, and OLU dependence are aligned on these retained locations by explicit coordinate matching rather than row order, and duplicate coordinates are removed during feature extraction. Curation does not remove persistently wet locations, since the dependence construction keeps and labels locations that stay flooded even under full protection.

A.2 Hydrodynamically Derived OLU Dependence

The protection status of a location is derived from its simulated response to OLU perturbations, not from its distance to the nearest protected or unprotected shoreline segment. We compute the dependence once per retained location and then combine it with each scenario's protection vector. Figure 7 shows why proximity alone is not enough. Protecting part of the shoreline dries most of the areas behind it, but it can also raise water levels or cause new flooding elsewhere.

Let $\mathbf{0}$ denote the all-unprotected configuration, $\mathbf{1}$ the all-protected configuration, $\mathbf{e}_k$ the configuration protecting only OLU k, and $\mathbf{1} - \mathbf{e}_k$ the configuration unprotecting only OLU k. For every available single-OLU perturbation,

$$\Delta_{ik}^{+} = \left[y_i^{(\mathbf{0})} - y_i^{(\mathbf{e}_k)} \right]_+, \qquad \Delta_{ik}^{-} = \left[y_i^{(\mathbf{1}-\mathbf{e}_k)} - y_i^{(\mathbf{1})} \right]_+, \qquad \Delta_{ik} = \max\left(\Delta_{ik}^{+}, \Delta_{ik}^{-} \right). \tag{10}$$



(a) (b)

Figure 5: Abu Dhabi study region. (a) The 17 OLUs, with each shoreline segment colored and numbered by its OLU. (b) Simulated PWL at 0.5 m SLR when no OLU is protected, giving a flooded area of 187.9 km^2.

The first term measures the PWL reduction from protecting OLU k on an otherwise unprotected shoreline, and the second measures the PWL increase from removing protection at k on an otherwise fully protected shoreline. Either direction is enough to identify an influence, and a term stays zero when its perturbation simulation is unavailable. With the largest local response $\Delta_i^{\max} = \max_k \Delta_{ik}$, the location-specific guardian threshold is

$$T_i = \max\big(0.10\ \mathrm{m},\ 0.5\,\Delta_i^{\max}\big), \tag{11}$$

and OLU k is a guardian of location i when $\Delta_{ik} > T_i$. Because the threshold is relative to the strongest local response, several OLUs can guard the same location. The guardian set is stored as the integer bitmask

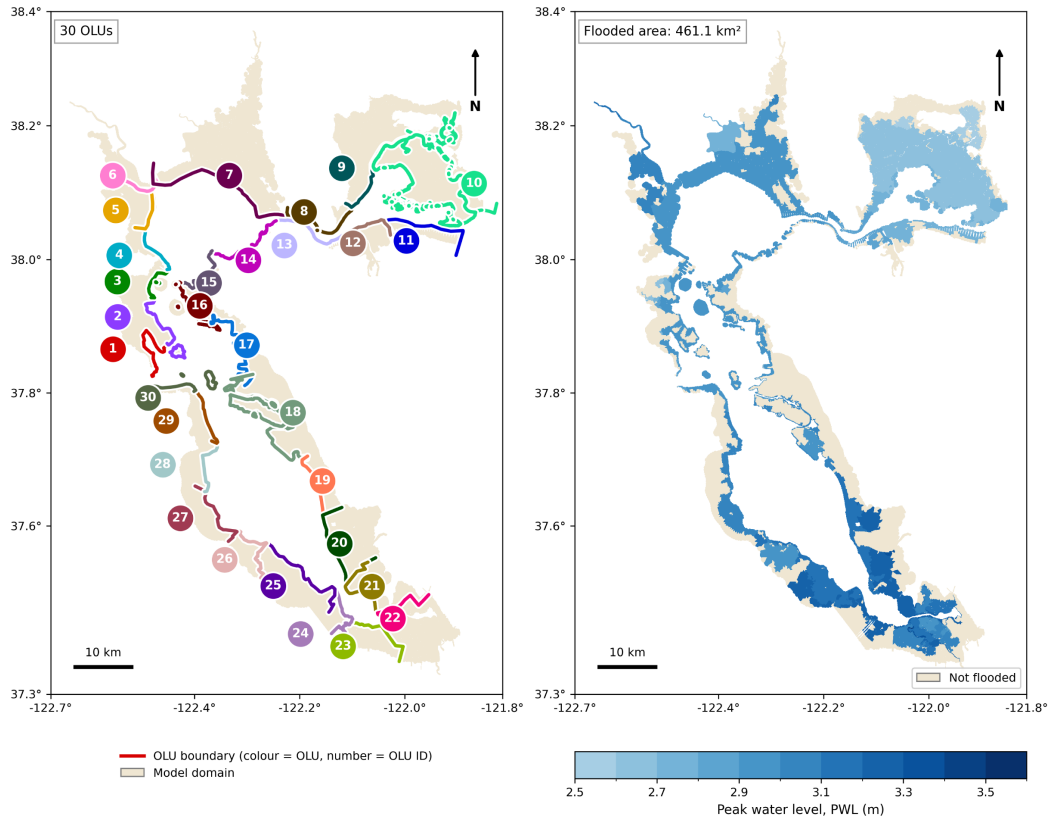
$$G_i = \sum_{k=1}^{N_r} \mathbb{I}\big[\Delta_{ik} > T_i\big]\, 2^{k-1}, \tag{12}$$

where bit $k-1$ marks dependence on OLU k. The number of guardians and the maximum-impact OLU are also recorded for diagnostics, but the bitmask is the dependence representation used downstream.

Two special cases are handled with $\epsilon_{\mathrm{dry}} = 0.05$ m before the bitmask is finalized. A point with $y_i^{(\mathbf{0})} < \epsilon_{\mathrm{dry}}$ stays dry even with no protection, so its guardian set is forced empty to avoid spurious dependence from numerical noise. A point with $y_i^{(\mathbf{1})} > \epsilon_{\mathrm{dry}}$ stays wet even under full protection. Its bitmask is also cleared, and a separate always-flooded flag keeps the distinction.

Scenario-specific OLU status. With guardian set $\mathcal{G}_i = \{k : \Delta_{ik} > T_i\}$ and unprotected OLUs $\mathcal{U}(\mathbf{s}) = \{k : s_k = 0\}$, the categorical status used by both representations is

$$c_i^{(\mathbf{s})} = \begin{cases} 2, & \text{point } i \text{ is always flooded}, \\ 0, & \mathcal{G}_i = \varnothing, \\ 2, & \mathcal{G}_i \cap \mathcal{U}(\mathbf{s}) \neq \varnothing, \\ 1, & \text{otherwise}. \end{cases} \tag{13}$$



(a) (b)

Figure 6: San Francisco Bay study region. (a) The 30 OLUs, with each shoreline segment colored and numbered by its OLU. (b) Simulated PWL at 1.0 m SLR when no OLU is protected, giving a flooded area of 461.1 km^2.

Status 0 means no active OLU dependence, status 1 an OLU-dependent location whose guardians are all protected, and status 2 an OLU-dependent location with at least one unprotected guardian, or an always-flooded location. The grid and graph generators use the same definition for both regions.

A.3 Terrain and Land-Cover Attributes

Each retained location is given an elevation and a land-cover value, both sampled independently of the simulator's internal bathymetry. Land cover ℓ_i is sampled from ESA WorldCover 10 m 2021 v200 (Zanaga et al., 2022) with its original class codes (10 tree cover, 20 shrubland, 30 grassland, 40 cropland, 50 built-up, 60 bare or sparse vegetation, 70 snow and ice, 80 permanent water bodies, 90 herbaceous wetland, 95 mangroves, and 100 moss and lichen). The grid representation keeps these codes, and the graph remaps them to contiguous categories (Appendix A.5). Elevation z_i is sampled from Copernicus DEM GLO-30 at a nominal 30 m resolution (European Space Agency, 2022). It is different from the bathymetric and terrain products used to build the Delft3D domains, and it is the elevation passed to the models and to the PA.

San Francisco coordinates are processed in WGS 84 / UTM Zone 10N (EPSG:32610), and Abu Dhabi coordinates in UTM Zone 40N (EPSG:32640). Both are transformed to WGS 84 geographic coordinates (EPSG:4326) before raster sampling, and land cover and elevation are read directly at the transformed coordinates from region-specific WorldCover tiles and GLO-30 rasters. Each location therefore has three model attributes $(c_i^{(\mathbf{s})}, z_i, \ell_i)$ and the target PWL $y_i^{(\mathbf{s})}$. The occupancy mask introduced below is a structural grid indicator, not a physical attribute. Figure 8 shows elevation, land cover, and the OLU dependence for both regions.

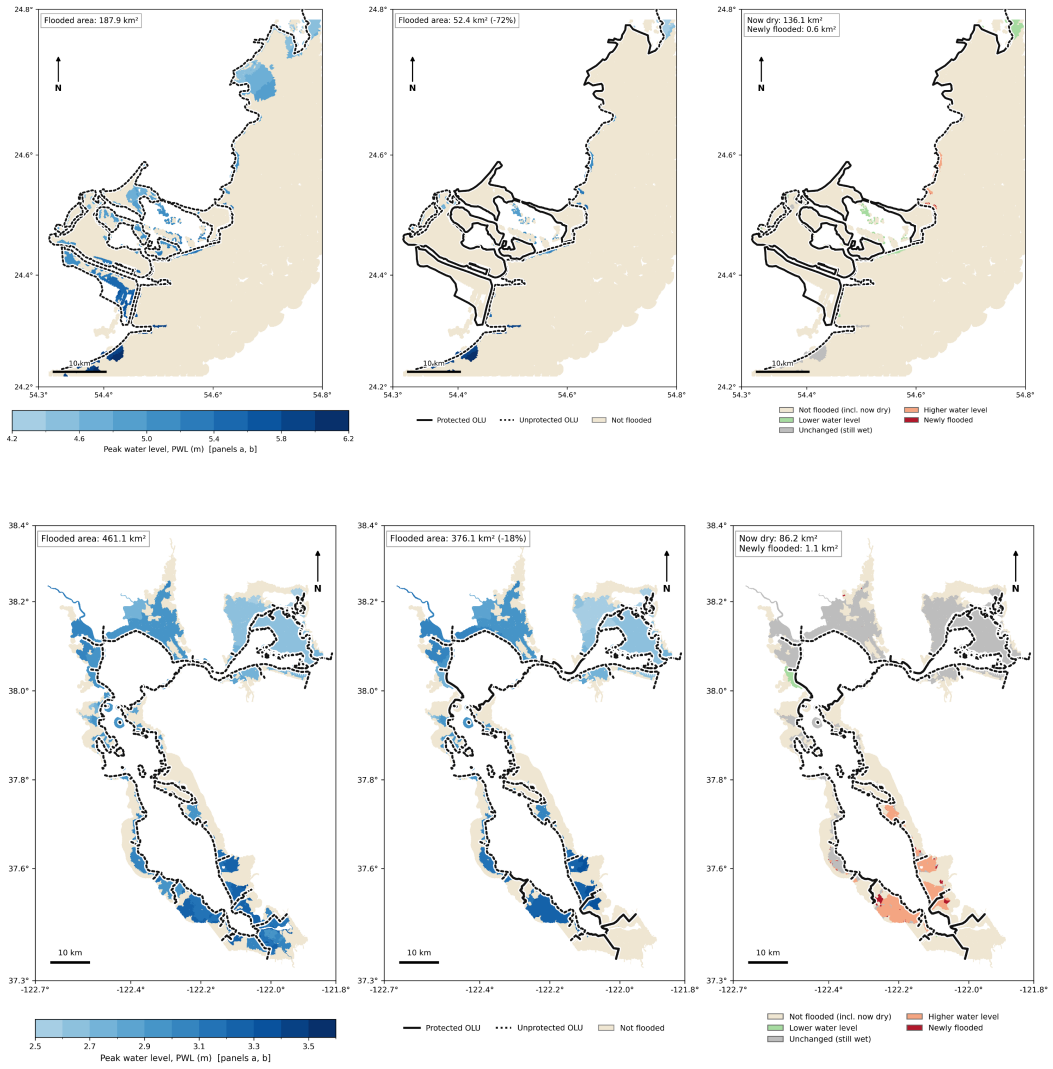


Figure 7: Effect of shoreline protection on flooding in AD (top, 0.5 m SLR) and SF (bottom, 1.0 m SLR). (a) PWL when no OLU is protected. (b) PWL for one protection scenario, with protected OLUs drawn as solid lines and unprotected OLUs as dotted lines. The flooded area falls by 72% in AD and 18% in SF. (c) Change between (a) and (b). Most of the change is locations that become dry (136.1 km^2 in AD and 86.2 km^2 in SF), but some locations see a higher water level and a small area becomes newly flooded (0.6 km^2 in AD and 1.1 km^2 in SF).

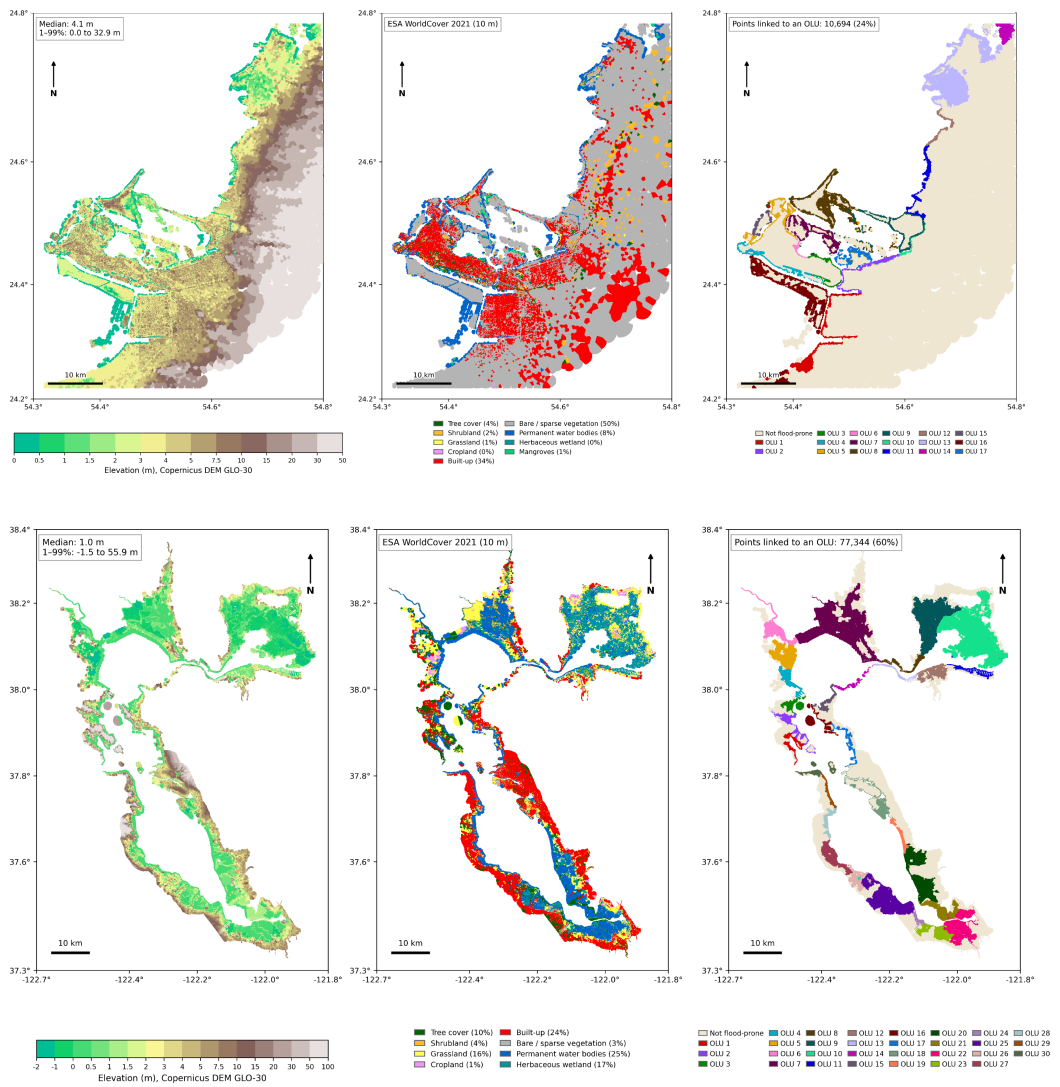
A.4 Regular-Grid Representation

Dense backbones use a 1024×1024 representation of the retained coordinates. The generator uses natural geographic bins with $N_g = 1024$ and does not move colliding points into neighboring cells. Let $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ be the extrema of the retained coordinates. A 2% margin of the coordinate range is added on each axis, giving $\widetilde{x}_{\min}, \widetilde{x}_{\max}, \widetilde{y}_{\min}, \widetilde{y}_{\max}$, and each coordinate is assigned the bin

$$u_i = \mathrm{clip}\Big(\Big\lfloor N_g \, \tfrac{x_i - \widetilde{x}_{\min}}{\widetilde{x}_{\max} - \widetilde{x}_{\min}} \Big\rfloor, 0, N_g - 1\Big), \qquad w_i = \mathrm{clip}\Big(\Big\lfloor N_g \, \tfrac{y_i^{\mathrm{coord}} - \widetilde{y}_{\min}}{\widetilde{y}_{\max} - \widetilde{y}_{\min}} \Big\rfloor, 0, N_g - 1\Big). \tag{14}$$

This mapping is deterministic and stored for every retained coordinate.

Because the hydrodynamic locations are irregularly spaced, several coordinates can fall into the same bin, and they stay there. For each occupied bin, the coordinates are sorted lexicographically



(a) (b) (c)

Figure 8: Input attributes for AD (top) and SF (bottom). (a) Copernicus GLO-30 elevation, with median 4.1 m in AD and 1.0 m in SF. (b) ESA WorldCover land cover. AD is dominated by bare or sparse vegetation (50%) and built-up land (34%), while SF has more permanent water (25%), built-up land (24%), and herbaceous wetland (17%). (c) Locations that depend on at least one OLU, colored by the OLU with the largest local impact, covering 24% of the retained locations in AD and 60% in SF. Locations with no OLU dependence are shown in beige. Panel (c) shows only the strongest OLU for display, while the models use the full guardian set of Eq. (12).

and the first (x, y) pair supplies the status, elevation, and land-cover inputs. The target instead uses all points in the bin. With $\mathcal{C}_{uw} = \{i : (u_i, w_i) = (u, w)\}$,

$$Y_{uw}^{(\mathbf{s})} = \frac{1}{|\mathcal{C}_{uw}|} \sum_{i \in \mathcal{C}_{uw}} y_i^{(\mathbf{s})}, \qquad M_{uw} = \mathbb{I}\big[|\mathcal{C}_{uw}| > 0\big], \tag{15}$$

so a single point keeps its own PWL and a shared bin receives the mean. The occupancy mask M_{uw} fills the validity channel $\mathbf{v}$ of Sec. 3.1 and separates occupied prediction sites from empty background. Validity cannot be inferred safely from a zero DEM, land-cover, or PWL value.

For scenario $\mathbf{s}$, the stored input tensor has shape $1024 \times 1024 \times 4$ with channel order OLU status, DEM, WorldCover class, and occupancy mask. The target is a 1024×1024 matrix with non-negative

PWL at occupied bins and zero elsewhere. Both arrays are built with the (u, w) indexing of Eq. (14), and their two spatial axes are transposed together before saving, so they share the same orientation. The stored geographic mapping keeps the full coordinate-to-bin map and the coordinates of every occupied bin. A reconstruction utility uses it to return grid predictions to the coordinate level, where coordinates sharing a bin take that bin's prediction.

A.5 Graph Representation

Graph-native backbones use the same retained locations and scenario definitions but keep the neighborhood structure of the hydrodynamic grid instead of rasterizing. Each scenario graph has one node per retained coordinate. Node i stores the continuous feature $[z_i]$, the categorical features $[\widetilde{\ell}_i,\, c_i^{(\mathbf{s})}]$ as integer indices, the position $[x_i, y_i^{\mathrm{coord}}]$, and the target $y_i^{(\mathbf{s})}$. WorldCover codes are remapped to contiguous categories $10 \to 0$, $20 \to 1$, $30 \to 2$, $40 \to 3$, $50 \to 4$, $60 \to 5$, $70 \to 6$, $80 \to 7$, $90 \to 8$, $95 \to 9$, and $100 \to 10$, with unknown values assigned category 11. The OLU status is the same variable as in Eq. (13). Unlike the grid, the graph contains only prediction nodes and needs no occupancy channel.

Hydrodynamic-grid connectivity. Edges follow the topology of the hydrodynamic computational grid rather than a generic k-nearest-neighbor rule. The generator reads cell centers and face-node coordinates from the region-specific grid files and links each retained coordinate to its nearest cell center with a KD-tree. The San Francisco grid is already in UTM Zone 10N. The Abu Dhabi grid is stored in EPSG:4326, so its cell centers and face nodes are transformed to EPSG:32640 before matching. Two cells are adjacent when they share a complete boundary edge, each adjacent pair is added in both directions, and the adjacency is then restricted to the cells linked to retained coordinates. For an edge $i \to j$, the edge feature is $\mathbf{e}_{ij} = [\Delta x_{ij}, \Delta y_{ij}, d_{ij}]$, with $\Delta x_{ij} = x_i - x_j$, $\Delta y_{ij} = y_i^{\mathrm{coord}} - y_j^{\mathrm{coord}}$, and Euclidean distance d_{ij}. No separate simulations are run for the graph. Scenario PWL, DEM, and land cover are aligned by coordinate before construction, and the same guardian logic gives the scenario-dependent status.

Consistency across representations. In both encodings, a sample is defined by the same retained coordinates, OLU configuration, guardian dependence, Copernicus elevation, WorldCover class, and PWL target. The grid merges points only when they share a geographic bin, while the graph keeps every point as a separate node linked by mesh topology. Any later normalization, projection, or embedding belongs to the model, not to the dataset.

B Architecture-Specific Physics Adapter Interfaces

Table 2 lists, for each backbone, the representation passed to the PA and how the interface $\mathcal{P}_m$ of Eq. (1) is realized.

Adapter normalization and heads. For raster representations, $\mathrm{BN}_A^{(m)}$ in Eq. (2) is a channel-wise $\mathrm{BatchNorm2d}$, and node-aligned implementations use $\mathrm{BatchNorm1d}$. The CASPIAN, ConvNeXt V2, GAT, GCN, MGN, and Transolver++ adapters have affine BatchNorm parameters, while the MaxViT, Swin V2, VM-UNet, Depth Anything V2, Depth Pro, and ControlNet adapters use affine-free BatchNorm. The data branch applies dropout before its final prediction, with rate 0.10 in the dense and diffusion adapters and 0.30 in the node-aligned adapters.

Initialization. The temperature is initialized to $\tau = 0.5$, and η is initialized from the source-region reference level, 3.0 for San Francisco and 5.0 for Abu Dhabi. Both η and ϑ are then learned. The $+3$ gate initialization of Eq. (5) is a fixed additive logit offset in the dense and ControlNet adapters and an initial value of the final gate-head bias in the node-aligned adapters. This changes the parameterization but not the meaning of Eq. (5). In dense implementations, the gate penalty of Eq. (6) is computed over the full gate tensor before validity masking, so Ω_A covers all grid cells. In node-aligned implementations, it is computed over the represented nodes.

State-space and depth-foundation interfaces. In VM-UNet, the representation just before the native final regression layer is projected to the adapter width before entering the adapter BatchNorm.

Family	Backbone	Representation passed to PA	Interface realization
Graph	GCN (Kipf & Welling, 2017)	Node-aligned latent embeddings	Node-wise PA heads over retained graph locations.
Graph attention	GAT (Veličković et al., 2018)	Node-aligned attention embeddings	Node-wise PA heads over retained graph locations.
Mesh graph	MGN (Pfaff et al., 2021)	Node-aligned latent embeddings	Node-wise PA heads after native mesh message passing.
Scientific attention	Transolver++ (Luo et al., 2025)	Site-aligned operator representation	Node-wise PA heads at the physical prediction sites.
Dense vision	CASPIAN (Karapetyan et al., 2026)	Dense spatial feature map	Dense features, then pointwise PA heads.
Dense vision	ConvNeXt V2 (Woo et al., 2023)	Dense and multi-scale visual features	Dense decoding and alignment, then pointwise PA heads.
Dense vision	MaxViT (Tu et al., 2022)	Dense and multi-scale visual features	Dense decoding and alignment, then pointwise PA heads.
Dense vision	Swin V2 (Liu et al., 2022b)	Dense and multi-scale transformer features	Dense decoding and alignment, then pointwise PA heads.
State space	VM-UNet (Ruan et al., 2024; Liu et al., 2024)	Decoder feature before the native final regression layer	Decoder feature projected to the PA width, then pointwise PA heads.
Depth foundation	Depth Anything V2 (Yang et al., 2024)	Pretrained dense DPT feature before depth regression	Flood-specific input stem and feature projection align the feature to the PWL grid.
Depth foundation	Depth Pro (Bochkovskiy et al., 2025)	Pretrained dense depth representation	Task-specific projection and interpolation align the feature to the PWL grid.
Diffusion	ControlNet (Zhang et al., 2023)	Generated base PWL map with local conditioning	Adapter-owned post-decoder stem, with the data branch predicting a residual around the generated map.

Table 2: Backbones and their realization of the shared PA interface. The representation column describes the site-aligned quantity passed to the PA, not the full internal architecture. Terrain conditioning, the two prediction branches, gated fusion, and the adaptation protocol are shared across all rows.

Depth Anything V2 similarly takes a dense DPT feature before the original depth regressor, using a learned flood-input stem and a feature projection. In both cases, the raw DEM bypasses this path and enters Eq. (3) directly. These operations belong to the interface, not to the shared adapter equations.

ControlNet interface and objective. The output of ControlNet is generative rather than a deterministic regression field, so its interface differs from the other backbones. The pretrained SD3.5 transformer and VAE stay frozen by design. The trainable ControlNet branch produces the conditional representation, and a clean-latent estimate is decoded to a base PWL map. A small adapter-owned convolutional stem then combines this map with protection status, DEM, and land cover before the shared adapter computation, so the stem parameters belong to ϕ_m. The data branch predicts a residual around the generated base value b_i. ControlNet keeps its diffusion training objective alongside the map objective. Writing $\mathbf{l}_0$ for a clean diffusion latent, $\widehat{\mathbf{l}}_0$ for the preconditioned clean-latent estimate, and ς for the noise level,

$$\mathcal{L}_{\text{flow}} = \mathbb{E}\big[w(\varsigma)\,\big\|\widehat{\mathbf{l}}_0 - \mathbf{l}_0\big\|_2^2\big], \qquad \mathcal{L}_{\text{ControlNet}} = \mathbb{I}_{\theta_{\text{CN}}}\mathcal{L}_{\text{flow}} + \mathbb{I}_{\text{route}}\big(\mathcal{L}_{\text{pred}} + \mathbb{I}_{\text{PA}}\mathcal{L}_g\big), \tag{16}$$

where the indicators depend on the adaptation regime. The decoded base map is detached from the diffusion computation. Map-level gradients therefore train the post-decoder PA or raw head, and diffusion gradients train the ControlNet branch when it is trainable. For ControlNet, θ_m denotes the trainable ControlNet branch and its interface.

Raw-head regularization. The raw head χ_m is regularized independently of the PA. Raster backbones use a statistics-free GroupNorm and convolutional prediction head, and graph backbones use a node-wise normalized MLP. Both use train-time feature jitter and dropout, with a fixed weight decay of 10^{-4} on the head weights. This makes the raw model a regularized no-PA control rather than a plain linear probe.

Regime	Source checkpoint	Trainable during adaptation	Frozen during adaptation
FT+PA	PA source	Trainable backbone and interface, and PA parameters ϕ_m	Only components fixed by design (e.g., the SD3.5 foundation in ControlNet).
FT	Raw source	Trainable backbone and interface, and prediction head χ_m	Only components fixed by design (e.g., the SD3.5 foundation in ControlNet).
NPA	Raw source	Prediction head χ_m	Backbone and interface.
PA	PA source	PA parameters ϕ_m	Backbone and interface.
LoRA (PEFT)	Raw source	Rank-8 LoRA parameters	Prediction head and all base parameters.
BitFit (PEFT)	Raw source	Selected bias parameters	Prediction head and all other parameters.
IA^3 (PEFT)	Raw source	IA^3 scaling parameters	Prediction head and all base parameters.
LoRA+PA (PEFT+PA)	PA source	ϕ_m and rank-8 LoRA parameters	All base backbone and interface parameters.
PA+BitFit (PEFT+PA)	PA source	ϕ_m and selected bias parameters	All other backbone and interface parameters.
PA+IA^3 (PEFT+PA)	PA source	ϕ_m and IA^3 scaling parameters	All base backbone and interface parameters.

Table 3: The ten target-adaptation regimes. FT+PA and FT denote full fine-tuning with and without the PA, and NPA denotes partial fine-tuning of the prediction head χ_m without the PA. In the PEFT regimes without the PA, the prediction head stays frozen, so only the PEFT parameters are updated. PEFT parameters are injected after the source checkpoint is loaded and attach to architecture-compatible modules, so their locations and counts differ across backbones.

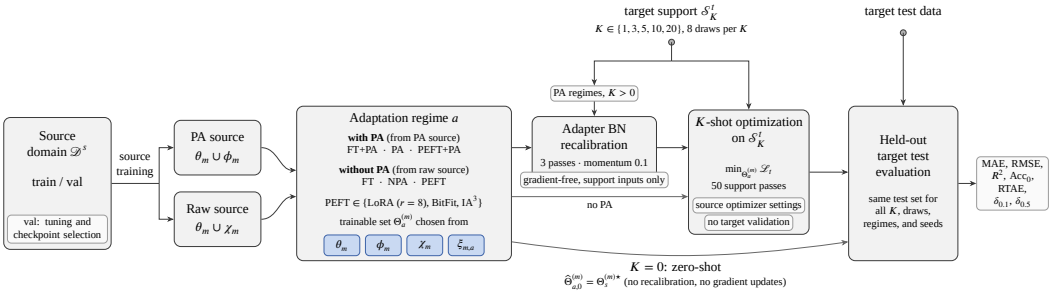


Figure 9: Two-stage training and adaptation protocol. Source checkpoints are selected with source-domain validation only. Target support examples may update only the state permitted by the selected regime, and target-test examples are held out until final evaluation; at $K = 0$, both support-statistics recalibration and gradient adaptation are skipped.

Because the PA realization differs across families in BatchNorm affine state, convolutional or MLP heads, and the ControlNet stem, its trainable parameter count is measured separately for each backbone.

C Adaptation and PEFT Details

Table 3 lists the ten target-adaptation regimes of Sec. 4.2, and Figure 9 summarizes the two-stage protocol.

PEFT formulations. LoRA (Hu et al., 2021) replaces a selected frozen linear or channel-mixing transformation $\mathbf{W}_0$ by

$$\mathbf{W}\mathbf{x} = \mathbf{W}_0\mathbf{x} + \frac{\alpha}{r}\mathbf{B}\mathbf{A}\mathbf{x}, \qquad r = 8, \quad \alpha = 16, \tag{17}$$

with $\mathbf{A}$ Kaiming-initialized and $\mathbf{B} = 0$, so the injected update starts at zero. IA^3 (Liu et al., 2022a) learns multiplicative activation scales $\mathbf{y} = \mathbf{s}_{\mathrm{IA}} \odot (\mathbf{W}_0\mathbf{x})$, initialized at $\mathbf{1}$. BitFit (Zaken et al., 2022) updates selected bias parameters and keeps all other tensors fixed.

Insertion sites. PEFT modules are placed according to each backbone's structure. They target attention and feed-forward projections in attention-based backbones, channel-mixing layers in convolutional models, and the SS2D input and output projections in VM-UNet, where the selective-scan recurrence tensors are not LoRA or IA^3 targets. In graph, operator, and ControlNet modules, they target compatible linear layers. In the graph and operator implementations, PEFT injection runs recursively over all compatible linear layers and can therefore also place PEFT parameters inside node-wise head modules. These parameters still belong to $\xi_{m,a}$, and the base parameters still follow Table 3. In the PEFT+PA regimes, ϕ_m and $\xi_{m,a}$ are optimized jointly. The trainable-parameter fraction of regime a is

$$\rho_a^{(m)} = \frac{|\Theta_a^{(m)}|}{|\Theta_{\mathrm{total}}^{(m)}|} \times 100\%, \tag{18}$$

which describes the size of the optimization problem only and says nothing about performance by itself.

Freezing. A frozen component is frozen in both its parameters and its internal state. Setting `requires_grad=False` is not enough for modules with BatchNorm, Dropout, stochastic depth, or other stateful operations. During adaptation with a frozen backbone, backbone normalization layers stay in inference mode and frozen Dropout, DropPath, and other stochastic modules are disabled. The implementation re-applies these settings after every change of training mode and checks that frozen running statistics stay unchanged. The two full fine-tuning regimes are the only exception, and there the trainable backbone updates normally.

Adapter BatchNorm recalibration. For every regime with a PA and $K > 0$, adaptation begins with a gradient-free recalibration of the adapter BatchNorm of Eq. (2). The whole model is set to inference mode, only the adapter BatchNorm is switched to training mode, and over three complete passes of the support set its running statistics are updated as

$$\widehat{\boldsymbol{\mu}}_A \leftarrow (1-\mu_{\mathrm{BN}})\,\widehat{\boldsymbol{\mu}}_A + \mu_{\mathrm{BN}}\,\boldsymbol{\mu}_{\mathcal{B}}, \qquad \widehat{\boldsymbol{\sigma}}_A^2 \leftarrow (1-\mu_{\mathrm{BN}})\,\widehat{\boldsymbol{\sigma}}_A^2 + \mu_{\mathrm{BN}}\,\boldsymbol{\sigma}_{\mathcal{B}}^2, \qquad \mu_{\mathrm{BN}} = 0.1, \tag{19}$$

with no optimizer step or gradient. The mini-batch size for this step depends on the architecture. It is four support graphs for GAT, GCN, MGN, and Transolver++, two samples for CASPIAN and ConvNeXt V2, and one sample for MaxViT, Swin V2, VM-UNet, Depth Anything V2, Depth Pro, and ControlNet. For ControlNet, the base map used here is generated from support conditioning only. Neither support labels nor target test samples are used to estimate the adapter statistics. When ϕ_m stays trainable during the supervised support optimization that follows, its BatchNorm keeps updating from the same support data, and any affine BatchNorm parameters receive gradients as part of ϕ_m. All frozen backbone statistics remain fixed. Regimes without a PA skip this step.

Fixed support budget. After recalibration, the parameters selected in Eq. (8) are optimized on $\mathcal{S}_K^t$ only. The eleven non-diffusion backbones run 50 complete passes over the support set, with one optimizer update per support mini-batch in each pass. The value 50 is therefore a fixed number of passes, which gives more than 50 updates when K exceeds the support batch size. ControlNet instead runs exactly 50 optimizer steps while cycling through its support loader with its configured gradient accumulation. The optimizer, learning rate, weight decay, and other settings are taken from source training, gradients are clipped to global norm 1, and no target validation, early stopping, or test-based checkpoint selection is used. Each combination of regime, K, seed, and support draw starts once from its source checkpoint, is adapted once, and is then evaluated on the target test set.

D Scenario Splits and Support Construction

Table 4 summarizes the scenario partitions and evaluation settings. Source training uses the training split, hyperparameter search, early stopping, and checkpoint selection use only the validation split, and the test split is used only for final evaluation. A PA source model and a raw source model are trained for every backbone, region, and seed.

Table 4: Scenario partitions and evaluation settings. For the regional datasets, the scenario column gives train/validation/test counts. For the SLR targets, it gives the support pool and held-out test, with no validation partition. For $K > 0$, eight support draws are evaluated per K, and $K = 0$ is evaluated once per seed.

Experiment	Source → target	Scenarios	K	Seeds
In-domain SF	SF → SF	285 (168/57/60)	–	3
In-domain AD	AD → AD	142 (82/28/32)	–	3
Cross-region	SF → AD	142 (82/28/32)	0,1,3,5,10	3
Cross-region	AD → SF	285 (168/57/60)	0,1,3,5,10	3
Cross-SLR	$\mathrm{SF}_{1.0} \to \mathrm{SF}_{0.5}$	32 (26 pool / 6 test)	0,1,3,5,10	3
Cross-SLR	$\mathrm{SF}_{1.0} \to \mathrm{SF}_{1.5}$	32 (26 pool / 6 test)	0,1,3,5,10	3

Protection buckets and regional splits. Each scenario is identified by its OLU configuration $\mathbf{s}$, with protection level $n_{\mathrm{prot}}(\mathbf{s}) = \sum_k s_k$. The all-unprotected and all-protected configurations form their own `none` and `all` buckets. The remaining scenarios are split by the quartiles of n_{prot} into `very_low`, `low`, `mid`, and `high`, with boundaries $Q_{25} = 7$, $Q_{50} = 8$, $Q_{75} = 9$ for SF and $Q_{25} = 4$, $Q_{50} = 8$, $Q_{75} = 11$ for AD. Scenarios with $n_{\mathrm{prot}} \le Q_{25}$, $Q_{25} < n_{\mathrm{prot}} \le Q_{50}$, and $Q_{50} < n_{\mathrm{prot}} \le Q_{75}$ fall into the first three buckets, and the rest into `high`. The buckets therefore follow the observed scenario distribution rather than equal protection intervals.

For a protection bucket b with n_b scenarios, the number of test scenarios is

$$n_{\mathrm{test}}(b) = \begin{cases} 1, & n_b = 1, \\ \min\{n_b - 1,\ \max(2, \lceil 0.2\, n_b \rceil)\}, & n_b > 1, \end{cases} \tag{20}$$

which keeps at least one non-test scenario in any bucket with more than one sample. The single-scenario `none` and `all` buckets both go to test. This gives 60 SF and 32 AD test scenarios. With the test sets fixed, the remaining scenarios are split again by the same buckets, aiming for a validation share of about 20% of the full regional dataset. The result is 168/57/60 train/validation/test scenarios for SF and 82/28/32 for AD, or roughly $60/20/20$ after rounding within buckets. The regional manifest is generated once with seed 42 and shared by all twelve backbones and all training seeds, with no architecture-specific resampling, so differences between models cannot come from different partitions.

Cross-region support draws. Cross-region support sets are drawn from the target non-test pool, which is the union of the target train and validation partitions, so that $\mathcal{S}_K^t \subseteq \mathcal{D}_{\mathrm{train}}^t \cup \mathcal{D}_{\mathrm{val}}^t$ and $\mathcal{S}_0^t = \varnothing$. Eight reproducible draws are made for each $K \in \{0, 1, 3, 5, 10, 20\}$. Draws are stratified by protection bucket, with bucket shares roughly matching the non-test pool, and within each K, scenarios used less often in earlier draws are preferred to limit repetition. The target test set is the same for all K, so differences across K reflect the amount of support rather than the test scenarios. In both transfer directions, the source model is trained on the source training split and selected on the source validation split.

SLR target manifests. The SLR targets use their own fixed construction instead of the 285-scenario SF split. The $0.5\,\mathrm{m}$ and $1.5\,\mathrm{m}$ SF targets each contain 32 configurations, namely the all-unprotected and all-protected anchors and the 30 single-OLU configurations. The two anchors are never used for testing. Six single-OLU configurations, with OLU indices spread evenly over the index range, are held out as the test set (seed 42), and the remaining 24 single-OLU configurations and the two anchors form a 26-scenario support pool. No validation partition is defined. The pool is stored under the `train` field of the transfer manifest for loader compatibility, but it serves only as the support pool. SLR adaptation uses $K \in \{0, 1, 3, 5, 10\}$ with eight draws per K. All $K = 0$ draws are empty. At $K = 1$, two draws use the all-unprotected anchor, two use the all-protected anchor, and four use distinct single-OLU configurations. For $K \ge 3$, both anchors are always included and the remaining $K - 2$ positions are distinct single-OLU configurations. All K are evaluated on the same six test scenarios, and no test scenario appears in any support set.

E TRAINING AND HPO DETAILS

Hyperparameter optimization (HPO) searches only optimization settings, namely a log-scaled learning rate in $[10^{-5}, 5 \times 10^{-3}]$, the optimizer (Adam, AdamW, or RMSprop), and a feasible batch size. Architectural settings are never searched.

HPO runs 25 Optuna trials of at most 80 epochs per study and minimizes masked validation MSE. Batch-size candidates are set separately for each architecture according to what fits in memory, so no batch-size set is reused across backbones. The selected hyperparameters, optimizer settings, and batch sizes are kept fixed after HPO for all subsequent training and adaptation runs.

Source training uses `ReduceLROnPlateau` on validation loss with factor 0.5, patience of 5 epochs, and minimum learning rate 10^{-6}. Gradients are clipped to global norm 1 in both source training and target adaptation. Weight decay is applied through the optimizer and is not part of Eq. (6). Final source training runs for at most 400 epochs for all backbones, with early stopping after 20 epochs without improvement in validation MSE. Early stopping usually ends training well before this limit, and the checkpoint with the lowest validation MSE is kept for evaluation and transfer.

The random state used to generate the manifest is separate from the training seeds $\{0, 1, 2\}$. Dense-grid models use seed-controlled spatial flips as training augmentation, with validation and test samples left unaugmented. Augmentation for the other families is architecture-specific.

F EVALUATION AND COMPLEXITY DETAILS

Metric definitions. All models are scored at the same retained physical locations. Grid predictions are first mapped back to these locations through the stored geographic mapping, while graph models already predict on them. For a held-out scenario with N retained locations, true PWL y_i, predicted PWL $\widehat{y}_i$, and mean true PWL $\bar{y}$, the metrics are

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \widehat{y}_i\right|, \tag{21}$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(y_i - \widehat{y}_i\right)^2}, \tag{22}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(y_i - \widehat{y}_i\right)^2}{\sum_{i=1}^{N}\left(y_i - \bar{y}\right)^2}, \tag{23}$$

$$\mathrm{RTAE} = 100\,\frac{\sum_{i=1}^{N}\left|y_i - \widehat{y}_i\right|}{\sum_{i=1}^{N}\left|y_i\right|}, \tag{24}$$

$$\delta_\epsilon = \frac{100}{N}\sum_{i=1}^{N}\mathbb{1}\left[\left|y_i - \widehat{y}_i\right| > \epsilon\right], \qquad \epsilon \in \{0.1, 0.5\}\,\mathrm{m}, \tag{25}$$

$$\mathrm{Acc}_0 = 100\,\frac{\sum_{i=1}^{N}\mathbb{1}\left[y_i = 0 \wedge \widehat{y}_i = 0\right]}{\sum_{i=1}^{N}\mathbb{1}\left[y_i = 0\right]}. \tag{26}$$

Lower values are better for MAE, RMSE, RTAE, $\delta_{0.1}$, and $\delta_{0.5}$, and higher values are better for R^2 and Acc_0. These metrics are used for reporting only, and the only quantity used for model selection is validation MSE.

Aggregation. Metrics are computed separately for each held-out scenario. For in-domain evaluation, scenario metrics are averaged within each seed, and we report the mean and sample standard deviation of the three seed-level means. For few-shot transfer, scenario metrics are computed separately for every support draw, averaged over all scenario and draw pairs within a seed so that each draw and scenario has equal weight, and then summarized per K as the mean and sample standard deviation over the three seeds. SLR transfer uses the same procedure. At $K = 0$, the single empty-

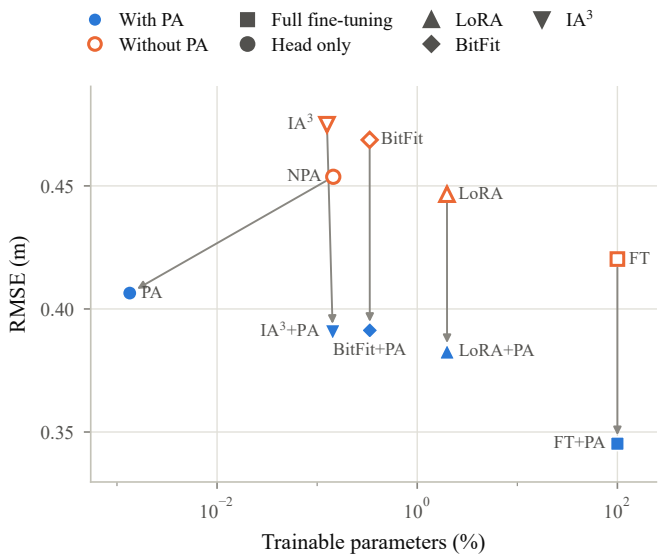


Figure 10: Mean transfer RMSE against the trainable parameter fraction of each adaptation regime. RMSE is averaged over the twelve backbones, the four transfer settings, and $K \in \{0, 1, 3, 5, 10\}$, as in Table 8, and each marker sits at the median trainable fraction over the backbones. Filled blue markers include the PA and open orange markers do not, while the marker shape gives the regime type. Each arrow joins a regime to its matched version with the PA. The horizontal axis is logarithmic.

support evaluation per seed replaces the repeated draws. No confidence intervals are reported, and no target test data are used for selection.

Complexity profiling. For every backbone and regime, we record the total and trainable parameter counts, the frozen count as their difference, and the trainable fraction of Eq. (18). These counts are exact and are read from the instantiated models. We do not compare FLOPs, MACs, or wall-clock times, since they could not be traced reliably for every architecture, and the runs used different GPUs (H100 or H200) and software environments. Figure 10 plots the mean transfer RMSE of each regime against its trainable fraction. Adding the PA lowers the RMSE of every matched regime, by 17.9% for full fine-tuning, 17.8% for IA^3, 16.5% for BitFit, 14.3% for LoRA, and 10.4% for head-only adaptation. How many parameters this costs depends on the size of the backbone. In the dense-grid backbones the PA has 294 or 438 parameters, which is at most 0.08% of the model and 50 to 160 times fewer than the raw head trained by NPA. The graph and operator backbones are much smaller, with about 18,000 to 131,000 parameters, so their PA of 4,358 parameters makes up 3.3% to 23.8% of the model. Across all backbones, adding the PA to a PEFT method raises the median trainable fraction by only 0.0033 percentage points, so the arrows in Figure 10 are almost vertical. PA-only adaptation, which trains a median of 0.0012% of the parameters, also reaches a lower mean RMSE than full fine-tuning without the PA (0.406 m against 0.420 m).

G Additional Results

G.1 In-Domain Results

Tables 5–7 give all seven metrics of Appendix F for in-domain prediction, averaged over both regions and for each region separately. Values are the mean and standard deviation over three seeds. For each backbone, the *Source* column shows which source model (PA or raw) has the lower combined RMSE, and the same source model is reported in all three tables. Bold marks the best value in each column. VM-UNet is best on every metric in both regions. The dense backbones have lower MAE, RMSE, and error exceedance rates than the graph and operator backbones throughout,

Table 5: In-domain results averaged over SF and AD.

Backbone	Source	MAE (m)	RMSE (m)	R^2	Acc_0 (%)	RTAE (%)	$\delta_{0.5}$ (%)	$\delta_{0.1}$ (%)
VM-UNet	PA	**0.0026** ± 0.0001	**0.0542** ± 0.0015	**0.9660** ± 0.0008	**99.9364** ± 0.0046	**3.5310** ± 0.0806	**0.0648** ± 0.0035	**0.3044** ± 0.0230
Swin V2	PA	0.0037 ± 0.0002	0.0630 ± 0.0025	0.9582 ± 0.0055	99.8055 ± 0.0267	4.8003 ± 0.2261	0.1000 ± 0.0115	0.6650 ± 0.0454
MaxViT	PA	0.0038 ± 0.0003	0.0639 ± 0.0033	0.9564 ± 0.0039	99.7953 ± 0.0427	4.8321 ± 0.6176	0.0980 ± 0.0113	0.6697 ± 0.0680
CASPIAN	PA	0.0041 ± 0.0003	0.0686 ± 0.0036	0.9560 ± 0.0060	99.8178 ± 0.0077	5.5597 ± 0.4478	0.1266 ± 0.0161	0.7107 ± 0.0376
Depth Pro	PA	0.0046 ± 0.0007	0.0713 ± 0.0046	0.9522 ± 0.0005	99.7172 ± 0.0215	6.2101 ± 1.0633	0.1425 ± 0.0363	0.7708 ± 0.1173
Depth Anything V2	PA	0.0049 ± 0.0001	0.0747 ± 0.0017	0.9503 ± 0.0052	99.6402 ± 0.0806	6.7128 ± 0.1161	0.1561 ± 0.0092	0.8686 ± 0.0269
ControlNet	PA	0.0054 ± 0.0004	0.0754 ± 0.0030	0.9244 ± 0.0096	98.0455 ± 0.1461	15.4634 ± 7.0646	0.1618 ± 0.0124	1.0396 ± 0.1821
ConvNeXt V2	PA	0.0072 ± 0.0006	0.0893 ± 0.0028	0.9217 ± 0.0157	99.4124 ± 0.2542	26.5863 ± 13.7362	0.2307 ± 0.0139	1.4426 ± 0.3020
MGN	PA	0.0600 ± 0.0016	0.2990 ± 0.0034	0.9266 ± 0.0011	94.5578 ± 0.4711	9.0683 ± 0.3178	2.0276 ± 0.1528	8.3739 ± 0.2469
GAT	Raw	0.0617 ± 0.0021	0.3106 ± 0.0019	0.9235 ± 0.0007	96.7846 ± 0.6652	8.7774 ± 0.5934	1.9891 ± 0.0708	8.8130 ± 0.5266
Transolver++	Raw	0.0649 ± 0.0078	0.3417 ± 0.0052	0.9215 ± 0.0079	96.5758 ± 4.0528	9.4857 ± 1.4286	1.7225 ± 0.3828	8.8934 ± 2.8365
GCN	PA	0.0737 ± 0.0025	0.3418 ± 0.0020	0.9121 ± 0.0009	93.5166 ± 0.7720	11.8118 ± 0.6456	1.6124 ± 0.0473	13.2854 ± 1.1726

Table 6: In-domain results for Abu Dhabi.

Backbone	Source	MAE (m)	RMSE (m)	R^2	Acc_0 (%)	RTAE (%)	$\delta_{0.5}$ (%)	$\delta_{0.1}$ (%)
VM-UNet	PA	**0.0039** ± 0.0002	**0.0866** ± 0.0030	**0.9469** ± 0.0015	**99.9012** ± 0.0098	**4.4593** ± 0.1712	**0.1058** ± 0.0095	**0.4604** ± 0.0384
Swin V2	PA	0.0044 ± 0.0002	0.0902 ± 0.0014	0.9450 ± 0.0013	99.7979 ± 0.0659	5.5141 ± 0.0702	0.1175 ± 0.0084	0.7135 ± 0.0074
MaxViT	PA	0.0048 ± 0.0004	0.0943 ± 0.0045	0.9431 ± 0.0034	99.7823 ± 0.0865	5.8221 ± 0.6174	0.1261 ± 0.0105	0.7930 ± 0.0977
CASPIAN	PA	0.0055 ± 0.0007	0.0984 ± 0.0061	0.9412 ± 0.0034	99.7413 ± 0.0271	7.0318 ± 0.9230	0.1648 ± 0.0330	0.9159 ± 0.1249
Depth Pro	PA	0.0064 ± 0.0015	0.1099 ± 0.0106	0.9328 ± 0.0103	99.7026 ± 0.0619	8.4506 ± 2.3253	0.2215 ± 0.0785	0.9217 ± 0.2590
Depth Anything V2	PA	0.0069 ± 0.0005	0.1143 ± 0.0055	0.9293 ± 0.0036	99.5482 ± 0.1792	9.3712 ± 0.1613	0.2330 ± 0.0247	1.0656 ± 0.0793
ControlNet	PA	0.0061 ± 0.0007	0.1025 ± 0.0059	0.9148 ± 0.0150	98.0762 ± 0.2425	14.4761 ± 11.2414	0.1907 ± 0.0175	1.0372 ± 0.1651
ConvNeXt V2	PA	0.0076 ± 0.0010	0.1140 ± 0.0074	0.9155 ± 0.0271	99.4878 ± 0.4062	23.5902 ± 22.2227	0.2613 ± 0.0251	1.3143 ± 0.2378
MGN	PA	0.0664 ± 0.0023	0.3734 ± 0.0068	0.9134 ± 0.0019	95.0695 ± 0.3636	10.2942 ± 0.3939	2.2620 ± 0.0900	8.1504 ± 0.3404
GAT	Raw	0.0704 ± 0.0021	0.3881 ± 0.0029	0.9106 ± 0.0013	96.4951 ± 0.8523	10.3491 ± 0.8927	2.5371 ± 0.0782	8.5326 ± 0.7478
Transolver++	Raw	0.0834 ± 0.0163	0.4557 ± 0.0072	0.8960 ± 0.0141	93.9065 ± 8.0915	12.4061 ± 2.5507	2.4486 ± 0.7725	9.9514 ± 5.7027
GCN	PA	0.0802 ± 0.0008	0.4283 ± 0.0027	0.8959 ± 0.0011	94.1315 ± 0.5815	12.9014 ± 0.3223	2.0481 ± 0.0692	11.5205 ± 0.4503

but ControlNet and ConvNeXt V2 have a high and variable RTAE. This is likely because RTAE becomes unstable for scenarios with little flooding, where its denominator is small.

G.2 Per-Backbone Transfer Results

Figures 11 and 12 show, for each backbone, the RMSE curve of its best regime over all four transfer settings, first over all ten regimes and then with the two full fine-tuning regimes excluded. The best regime is the one with the lowest RMSE averaged over $K \in \{0, 1, 3, 5, 10\}$, with equal weight for each K. In Figure 11, FT+PA is best for every backbone except Transolver++, where LoRA+PA is best. The dense and depth-foundation backbones reach RMSE close to 0.1 m by $K = 10$, while the graph and operator backbones start from a much higher zero-shot error and level off near 0.3 m. When full fine-tuning is excluded (Figure 12), nine of the twelve backbones still select a PA regime. Table 8 breaks the regime comparison down by transfer setting.

G.3 Qualitative Error Maps

Figure 13 shows the VM-UNet error maps for the scenario and regimes of Figure 3, computed as prediction minus ground truth at each location, so red marks overestimated PWL and blue marks underestimated PWL. Locations that are dry in both the prediction and the ground truth are shown in beige. In AD, the errors of FT and LoRA are spread over a large inland area, which matches the false flooding seen in Figure 3. The PA regimes keep the errors close to the coast, where the true flooding occurs. In SF, the in-domain errors stay within ± 0.1 m. After transfer, the largest errors appear in the northern basin, where FT and LoRA underestimate PWL, and in a few small areas along the southern shoreline.

Table 7: In-domain results for San Francisco.

Backbone	Source	MAE (m)	RMSE (m)	R^2	Acc_0 (%)	RTAE (%)	$\delta_{0.5}$ (%)	$\delta_{0.1}$ (%)
VM-UNet	PA	**0.0013** ± 0.0000	**0.0217** ± 0.0008	**0.9851** ± 0.0002	**99.9715** ± 0.0028	**2.6026** ± 0.0221	**0.0239** ± 0.0024	**0.1484** ± 0.0133
Swin V2	PA	0.0030 ± 0.0005	0.0357 ± 0.0050	0.9714 ± 0.0106	99.8130 ± 0.0193	4.0864 ± 0.3839	0.0824 ± 0.0268	0.6164 ± 0.0879
MaxViT	PA	0.0027 ± 0.0002	0.0335 ± 0.0021	0.9697 ± 0.0044	99.8083 ± 0.0133	3.8421 ± 0.6210	0.0698 ± 0.0126	0.5464 ± 0.0425
CASPIAN	PA	0.0028 ± 0.0001	0.0387 ± 0.0012	0.9708 ± 0.0091	99.8943 ± 0.0297	4.0875 ± 0.0611	0.0885 ± 0.0010	0.5055 ± 0.0569
Depth Pro	PA	0.0028 ± 0.0002	0.0327 ± 0.0021	0.9716 ± 0.0101	99.7319 ± 0.0192	3.9696 ± 0.2843	0.0636 ± 0.0088	0.6199 ± 0.0340
Depth Anything V2	PA	0.0030 ± 0.0003	0.0352 ± 0.0022	0.9712 ± 0.0097	99.7323 ± 0.0281	4.0544 ± 0.2756	0.0793 ± 0.0114	0.6716 ± 0.1031
ControlNet	PA	0.0047 ± 0.0006	0.0483 ± 0.0025	0.9340 ± 0.0079	98.0148 ± 0.0794	16.4508 ± 5.3722	0.1328 ± 0.0077	1.0420 ± 0.2952
ConvNeXt V2	PA	0.0069 ± 0.0010	0.0646 ± 0.0029	0.9279 ± 0.0116	99.3370 ± 0.1479	29.5824 ± 10.2942	0.2001 ± 0.0029	1.5708 ± 0.5574
MGN	PA	0.0535 ± 0.0010	0.2247 ± 0.0015	0.9398 ± 0.0008	94.0462 ± 0.5802	7.8424 ± 0.4243	1.7932 ± 0.2360	8.5974 ± 0.1784
GAT	Raw	0.0530 ± 0.0022	0.2330 ± 0.0009	0.9364 ± 0.0009	97.0741 ± 0.6491	7.2056 ± 0.6281	1.4410 ± 0.1047	9.0934 ± 0.3633
Transolver++	Raw	0.0463 ± 0.0022	0.2277 ± 0.0036	0.9469 ± 0.0030	99.2451 ± 0.0141	6.5653 ± 0.3107	0.9964 ± 0.0152	7.8354 ± 0.4394
GCN	PA	0.0671 ± 0.0042	0.2552 ± 0.0012	0.9283 ± 0.0007	92.9016 ± 0.9625	10.7222 ± 0.9688	1.1767 ± 0.0254	15.0503 ± 1.8948

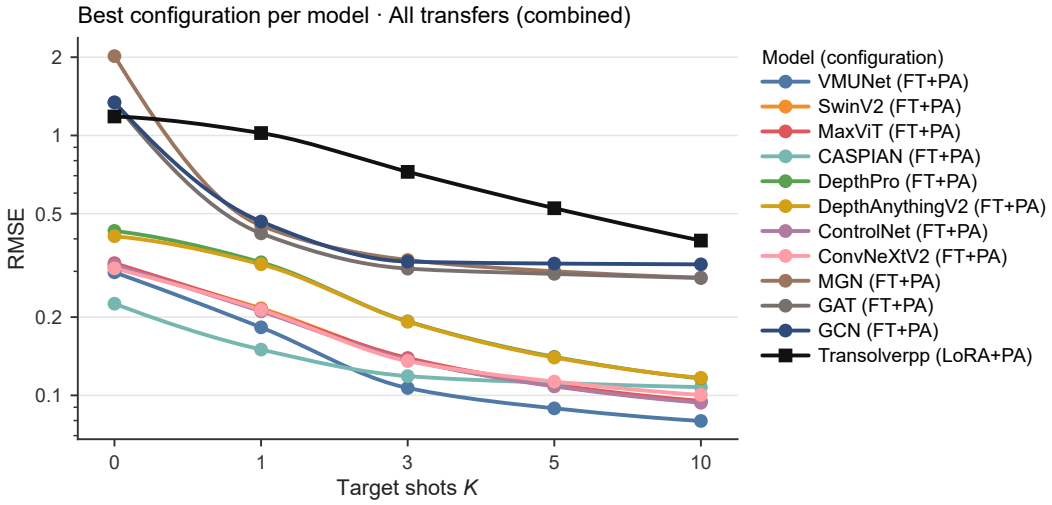


Figure 11: Best regime per backbone over all transfer settings, selected by mean RMSE over K. The vertical axis is logarithmic.

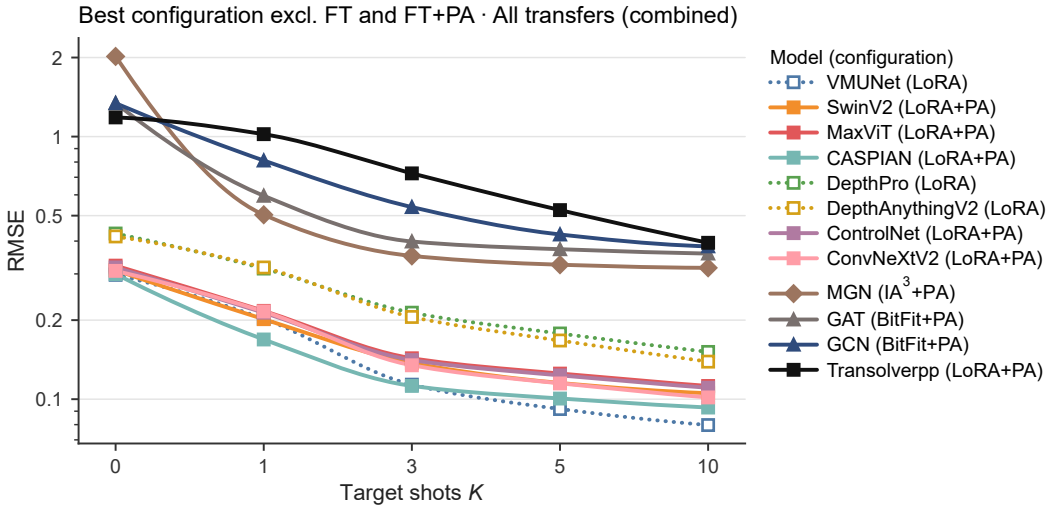
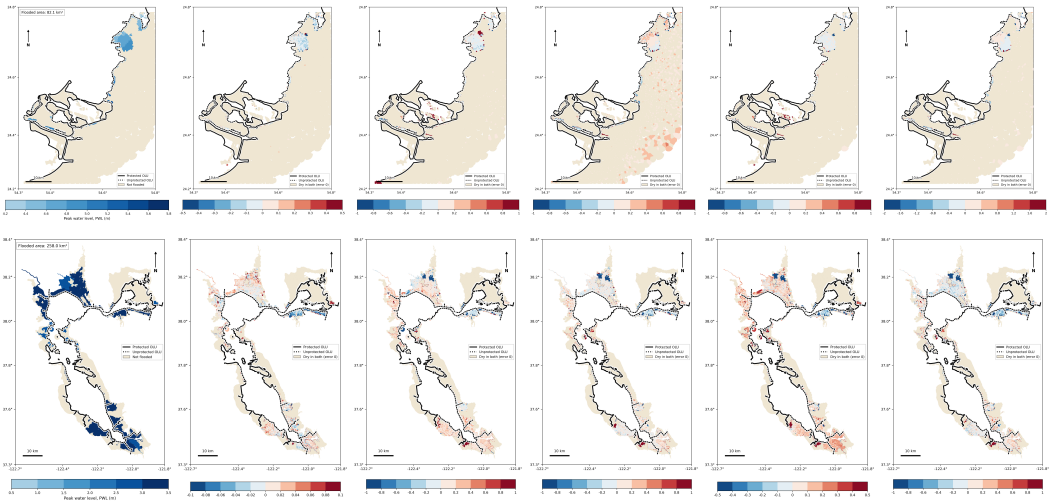


Figure 12: Best regime per backbone when FT and FT+PA are excluded. Solid lines are regimes with the PA and dotted lines are regimes without it. The vertical axis is logarithmic.

Table 8: RMSE (m) of each regime averaged over all twelve backbones and over $K \in \{0, 1, 3, 5, 10\}$, for each transfer setting. PEFT rows give the mean of LoRA, IA^3 and BitFit, with the individual methods listed beneath. Bold marks the best regime in each column.

Regime	**SF→AD**	**AD→SF**	**SF$_{1.0}$ →SF$_{0.5}$**	**SF$_{1.0}$ →SF$_{1.5}$**
FT+PA	**0.4338**	**0.5401**	**0.1668**	**0.2403**
PEFT+PA (mean)	*0.4989*	*0.5992*	*0.1819*	*0.2727*
LoRA+PA	0.5233	0.5653	0.1793	0.2629
IA^3+PA	0.4847	0.6170	0.1818	0.2785
BitFit+PA	0.4887	0.6152	0.1847	0.2766
PA	0.5093	0.6198	0.1969	0.2998
FT	0.6106	0.6233	0.1786	0.2686
PEFT (mean)	*0.6800*	*0.7061*	*0.1859*	*0.2817*
LoRA	0.6254	0.7122	0.1781	0.2708
IA^3	0.6891	0.7286	0.1913	0.2905
BitFit	0.7256	0.6774	0.1882	0.2836
NPA	0.6902	0.6243	0.1973	0.3030



(a) (b) (c) (d) (e) (f)

Figure 13: PWL error maps with VM-UNet (prediction minus ground truth, in m) for the scenario and regimes of Figure 3, AD in the top row and SF in the bottom row. (a) Ground-truth PWL for reference. (b) In-domain error. (c)–(f) Errors after transfer with $K = 3$ using (c) FT+PA, (d) FT, (e) best PEFT+PA, and (f) best PEFT. Beige marks locations that are dry in both the prediction and the ground truth. Color scales differ between panels.