



Cheap and accurate higher-order ice flow emulator for mountain glaciers

Sebastian H. R. Rosier¹, Thomas Gregov², Guillaume Jouvét², and Andreas Vieli¹

¹Department of Geography, Universität Zürich, Zürich, Switzerland

²Institute of Earth Surface Dynamics, Université de Lausanne, Lausanne, Switzerland

Correspondence: Sebastian Rosier (sebastian.rosier@geo.uzh.ch)

Abstract. Numerical projections of mountain-glacier evolution increasingly rely on large ensembles that sample uncertain climate forcing, geometry, and ice-flow parameters, yet explicitly resolving higher-order ice dynamics within such ensembles remains computationally expensive. Here we take an alternative approach by training a neural network once, offline, to reproduce the velocity field of a higher-order ice-flow model, and then use the trained network in place of the ice-flow solver during transient simulations. In doing so, we retain the higher-order model's physical basis while avoiding the repeated cost of its velocity solve. The network predicts depth-dependent horizontal ice-flow velocities for land-terminating mountain glaciers directly from gridded glacier geometry and physical parameters, and its weights are then held fixed, so that no retraining is required when it is applied to a new glacier. Leveraging the computational efficiency of GPU operations together with the Instructed Glacier Model (IGM), we generate a large synthetic training set of transient glacier states across diverse mountain topography and climate histories, and at fixed intervals we compute reference velocities from the Blatter-Pattyn ice-flow formulation. The resulting dataset contains over 300,000 higher-order velocity solutions, orders of magnitude more than the catalogues used to train earlier ice-flow emulators. We train on the misfit between the network outputs and target velocities, together with a physics term based on the same discretised ice-flow energy. Our preferred architecture, trained with this hybrid objective, accurately reproduces velocity on examples unseen during training, with a median surface-speed error of 0.71 m yr^{-1} and a median flux-divergence error of 0.10 m yr^{-1} . In transient simulations of two real glacier systems that lie outside the training set, the emulator tracks reference simulation ice volume through a 500-year advance-retreat cycle with maximum errors below 6%. Crucially, we find that errors do not accumulate over time. Instead, velocity biases induce geometric changes that tend to counteract them, stabilising the coupled emulator–mass-conservation system. Taken together, our results show that the expensive part of higher-order ice flow can be paid for once during training and reused indefinitely: velocity is predicted in a single forward pass rather than solved for iteratively at every time step. Higher-order ice dynamics are therefore no longer the limiting cost in mountain-glacier modelling, and become practical in large regional and global projection ensembles.

1 Introduction

Mountain glaciers are important indicators and drivers of environmental change, contributing to sea-level rise, altering seasonal water availability, and affecting natural hazards in high-mountain regions (Zekollari et al., 2019; Rounce et al., 2023).



25 Numerical projections of glacier evolution therefore require models that can be applied across many glaciers, climate regimes and scenarios, and uncertain parameter choices. In this setting, computational cost matters. Regional and global glacier studies often need large ensembles, but explicitly resolving higher-order ice dynamics remains expensive, especially when model resolution is high enough to represent mountain topography and narrow valley glaciers (MauSSION et al., 2019; Zekollari et al., 2022).

30 Many large-scale glacier models rely on simplified ice-flow approximations or parameterised dynamics. These approaches are efficient and appropriate where flow is dominated by local driving stress, for example in global-scale mass-balance-driven projections of small glaciers, but they can struggle where ice redistribution, valley geometry, basal motion, or other complex dynamic processes strongly affect glacier evolution. Higher-order models provide a more complete representation of longitudinal and lateral stress coupling than shallow-ice-type approximations (e.g. Blatter, 1995; Pattyn, 2003), but this comes at
35 a substantial cost in transient simulations because velocities require solving a three-dimensional nonlinear problem at every time step, rather than using a local algebraic relation. Thus, regional and global mountain-glacier modellers are faced with a difficult trade-off: use computationally efficient flowline or local shallow-ice approximations, or restrict higher-order dynamics to a much smaller number of glaciers, scenarios, and parameter combinations. This motivates computationally efficient alternatives that retain the physics of a higher-order model while being cheap enough for large ensemble glacier modelling at high
40 resolution.

The Instructed Glacier Model (IGM) provides one route towards this goal by formulating glacier evolution on raster grids and performing calculations on a GPU wherever possible, making operations both highly efficient and differentiable (Jouvet et al., 2022, 2026). In IGM, higher-order ice flow is expressed through a variational Blatter-Pattyn energy, and the horizontal velocity field can be represented by a neural mapping. Jouvet et al. (2022) pioneered the use of neural networks as offline emulators of
45 glacier-flow models, training a convolutional network on velocity fields generated by external higher-order model simulations and then using it as a drop-in ice-flow module. That emulator was trained on a comparatively small catalogue of simulations, limited by the cost of generating targets with CPU-based solvers, and was therefore typically retrained on data representative of the target setting. Subsequent work replaced supervised training with direct minimisation of the discretised ice-flow energy during the simulation itself, in an ‘online’ mode where network weights are repeatedly optimised against the current glacier
50 state (Jouvet and Cordonnier, 2023). A companion study develops this online approach further by adding deterministic physical feature fields to the neural trial space, improving the efficiency and robustness of finite-budget variational solves (Gregov et al., 2026). That study also explores how compact such networks can be made when they are re-optimised for a specific glacier, which is the complementary regime to the fixed general-purpose mapping considered here.

Here, we take the complementary approach of paying the bulk of the cost of solving the higher-order ice-flow equations once,
55 during offline training, rather than repeatedly during each glacier simulation. We generate a synthetic dataset of mountain-glacier states with IGM and compute reference velocities using its Blatter-Pattyn ice-flow formulation. The dataset contains over 300,000 higher-order velocity solutions, spanning tens of thousands of independent synthetic glaciers, and is therefore several orders of magnitude larger than the catalogues used to train earlier ice-flow emulators. The transient trajectories that supply these states are generated with the cheaper online neural solver, but every training target is recomputed from its corresponding



60 glacier state with the classical higher-order solver. We then train an emulator to predict the depth-dependent horizontal velocity field directly from gridded glacier geometry and physical parameters. The network uses the physics-enriched input representation developed in the companion study (Gregov et al., 2026), but in a different regime: instead of being re-optimised online for each glacier state, it is trained once using both supervised reference velocities and a variational penalty based on the same discretised ice-flow energy. To our knowledge this is the first time an ice-flow emulator has been trained against supervised
 65 velocity targets and the variational ice-flow energy simultaneously, with the relative weight of the two terms set adaptively during training. We show that this hybrid objective improves accuracy, with the clearest gains in the upper tail of the error distribution and in flux divergence, the quantity that matters most when the emulator is coupled back to mass conservation.

The central question addressed here is whether the cost of repeatedly solving the higher-order ice-flow problem can be reduced across many glacier simulations. Although glaciers differ in geometry, spatial resolution, and a limited set of physical
 70 parameters, much of the associated computation is redundant. A single network that learns the mapping from glacier state to velocity across a sufficiently broad range of geometries and flow regimes could reuse this information across simulations. This shifts the computational cost to generation of a training set. We show that this cost is now tractable, and describe a synthetic generation procedure that produces the necessary scale and diversity of glacier states.

Here, we evaluate whether this fixed, ‘offline’ approach can provide a cheap, accurate, and stable approximation to higher-
 75 order glacier dynamics with no retraining. We first test its ability to reproduce reference velocities on held-out synthetic glacier states, and then use it for transient simulations of real mountain-glacier geometries. This allows us to assess not only instantaneous velocity error, but also whether emulator errors accumulate, cancel, or remain dynamically controlled when the predicted velocities are coupled back to the mass-conservation equation. The remainder of the paper describes the IGM formulation, synthetic training set, emulator architecture, and hybrid training objective. We then evaluate the emulator in terms
 80 of both validation-set and transient performance, and discuss the computational implications and limits of using a pretrained emulator as an ice-flow solver in IGM.

2 Methods

We simulate glacier evolution with IGM, a modular, differentiable, and GPU-accelerated model formulated on regular raster grids (Jouvet et al., 2022, 2026). We use IGM here as the forward model that generates transient synthetic glacier states, as the
 85 high-fidelity solver used to compute reference velocities, and as the physical framework in which the trained emulator is later evaluated. The full derivation of the governing equations, numerical discretisation, boundary conditions, quadrature choices, and software implementation of the ice-flow model is given by Jouvet et al. (2026); we therefore limit our description to the most relevant information needed for the emulator experiments.

Ice thickness evolves according to mass conservation,

$$90 \quad \frac{\partial h}{\partial t} + \nabla \cdot (h \bar{\mathbf{u}}) = \dot{a}_s, \quad (1)$$



where h is ice thickness, $\bar{\mathbf{u}}$ is the depth-averaged horizontal velocity (the overbar denotes depth-averaging throughout), and $\dot{a}_s$ is the surface mass-balance rate. Basal melt is zero in all simulations presented here. Glacier geometry evolves through Eq. (1) under a prescribed surface mass-balance forcing.

2.1 Ice-flow formulation and notation

95 The ice-flow component is the Blatter-Pattyn higher-order approximation to the Stokes equations (Blatter, 1995; Pattyn, 2003). IGM obtains the horizontal velocity field by minimising a discrete energy functional $\mathcal{J}_H$, whose terms represent internal viscous dissipation, basal friction, gravitational driving work, and, where relevant, marine boundary forcing. In this study all synthetic glaciers are land-terminating, so the calving-front term is inactive. Viscous ice deformation is modelled using the Glen-Steinemann law (Glen, 1955; Steinemann, 1958) and basal resistance is represented by a classical Weertman sliding law
 100 (Weertman, 1957),

$$\boldsymbol{\tau}_b = -\tau_{\text{ref}} (|\mathbf{u}_b|/u_{\text{ref}})^m \mathbf{u}_b/|\mathbf{u}_b|, \quad (2)$$

where τ_{ref} is the basal shear-stress magnitude at the reference sliding speed $u_{\text{ref}} = 100 \text{ m yr}^{-1}$, and we use exponent $m = 3$ throughout.

The prescribed state for one diagnostic ice-flow solve is

$$105 \quad \mathbf{I}_H = [h_H, s_H, A_H, \tau_{\text{ref},H}, \Delta x], \quad (3)$$

where s_H is the surface elevation, A_H is the Arrhenius factor and Δx is the uniform horizontal grid spacing supplied as a spatially constant input field. We use subscript H throughout to denote a field discretised on the uniform raster grid. The bed is implicit in the geometry, $b_H = s_H - h_H$.

IGM represents the horizontal velocity degrees of freedom using a differentiable mapping of one of two types,

$$110 \quad \mathbf{U}_H = \mathcal{M}(\boldsymbol{\theta}; \mathbf{Z}_H), \quad (4)$$

where $\mathbf{U}_H = (\mathbf{u}_H, \mathbf{v}_H)$, $\boldsymbol{\theta}$ denotes the optimised variables of the mapping, and $\mathbf{Z}_H$ is the tensor supplied to it. In the neural case this is an image-to-image mapping: a stack of input rasters describing the glacier state is transformed into a stack of output rasters representing the velocity degrees of freedom, at the same grid resolution. For either mapping type, the optimal variables are obtained by minimising the discretised ice-flow energy,

$$115 \quad \boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\text{argmin}} \mathcal{J}_H[\mathcal{M}(\boldsymbol{\theta}; \mathbf{Z}_H)]. \quad (5)$$

The two mappings we use are instances of Eq. (4). For the identity mapping $\mathcal{M}^{\text{id}}$, $\boldsymbol{\theta}_{\text{id}}$ is the velocity field itself, so minimising $\mathcal{J}_H$ corresponds to a direct solve in the full discrete velocity space. We denote the resulting high-fidelity velocity by $\mathbf{U}_H^{\text{ref}} = \mathcal{M}^{\text{id}}(\boldsymbol{\theta}_{\text{id}}^*; \mathbf{Z}_H)$. For the network mapping $\mathcal{M}^{\text{net}}$, $\boldsymbol{\theta}_{\text{net}}$ contains the network weights, which are optimised against the same discrete Blatter-Pattyn energy. This is the Deep Ritz formulation (E and Yu, 2018) as adapted to ice flow by Jouvett and

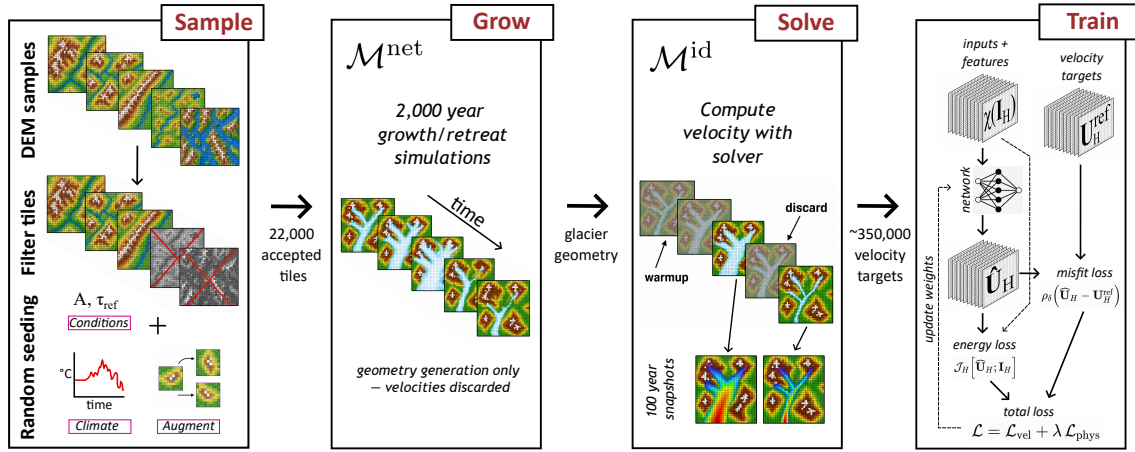


Figure 1. Workflow for generating the synthetic training set and training the emulator. **Sample:** DEM tiles are filtered and assigned a grid spacing, ice rheology A , basal shear-stress scale τ_{ref} , and a randomised climate history. **Grow:** each tile is integrated forward to produce a trajectory of glacier geometries, using the online neural solver $\mathcal{M}^{\text{net}}$. Its velocities are discarded: this stage determines only which geometries are sampled, never what the emulator is trained to reproduce. **Solve:** geometry is frozen at 100-yr intervals and the velocity recomputed with the identity mapping $\mathcal{M}^{\text{id}}$. All training targets originate here. **Train:** the network maps the glacier state and physics features $\mathbf{Z}_H$ to a velocity $\hat{\mathbf{U}}_H$, scored against the targets $\mathbf{U}_H^{\text{ref}}$ by a Huber misfit and against the discretised Blatter-Pattyn energy $\mathcal{J}_H$ by a variational penalty, combined with an adaptive weight λ (Sect. 2.4).

- 120 Cordonnier (2023), and as implemented in IGM (Jouvet et al., 2026; Gregov et al., 2026). Note that the network here maps discretised raster fields to velocity fields, rather than spatial coordinates to a solution as in the original Deep Ritz formulation. Because $\mathcal{M}^{\text{id}}$ and $\mathcal{M}^{\text{net}}$ use the same glacier state, physical parameters, discretised energy, and boundary conditions, differences between their output velocities primarily measure the approximation introduced by the neural parametrisation and the optimisation of its trainable parameters.
- 125 The two mappings differ in the dimension of the optimisation. For the identity mapping, the number of free variables equals the number of velocity degrees of freedom on the grid, and grows with domain size. For the network mapping, it is the number of network weights, which is fixed and much smaller. This reduction is what makes the neural formulation cheap to optimise. In both cases the horizontal velocity is expanded in a terrain-following vertical coordinate using the two-function MOLHO basis (Dias dos Santos et al., 2022), whose degrees of freedom represent basal motion and an SIA-derived vertical-shear contribution, with bilinear horizontal basis functions. Both mappings in Eq. (4) therefore represent the same $2N_z = 4$ velocity
- 130 degrees of freedom per grid cell. In what follows we use *emulator* exclusively for the network mapping $\mathcal{M}^{\text{net}}$ evaluated at fixed, offline-trained weights. Where the same architecture is optimised during a simulation, we refer to it as the online neural solver.



2.2 Synthetic training-data generation

135 The purpose of the training set is to expose the emulator to as wide a range of glacier shapes, sizes, and flow regimes as we can generate, so that it does not have to extrapolate when applied to a glacier it has never seen. This is why the data are synthetic: real glaciers do not provide the required diversity of geometry, resolution, and physical parameters in the quantities needed for training. We therefore require input-target pairs spanning the mountain-glacier states on which the emulator will be evaluated. For each retained glacier state k , the stored example consists of the glacier state $\mathbf{I}_{H,k}$ and the reference velocity field $\mathbf{U}_{H,k}^{\text{ref}}$ obtained with the identity mapping $\mathcal{M}^{\text{id}}$. We construct these examples in two stages. First, ensembles of synthetic transient glacier simulations use the more efficient online neural solver $\mathcal{M}^{\text{net}}$ to sample dynamically plausible geometries across a broad range of terrain, grid spacing, basal resistance, ice rheology, and climate histories. Second, each accepted glacier state is frozen and its velocity field is recomputed with $\mathcal{M}^{\text{id}}$. The velocities generated by $\mathcal{M}^{\text{net}}$ are therefore not used as supervised labels. Figure 1 gives an overview of the four stages: sampling terrain and physical conditions, growing synthetic glaciers, 145 recomputing reference velocities, and training the emulator against them.

2.2.1 Terrain sampling

We sample terrain that is currently ice-free, or very nearly so, and grow synthetic glaciers on it from scratch. For each synthetic experiment we define a fixed topography, derived from the Copernicus DEM GLO-90 product (European Space Agency, 2023). We define sampling domains in the Alps, Pyrenees, northern Andes, Mackenzie Mountains and Caucasus. These regions were 150 chosen to provide diverse mountain topography while limiting dependence on uncertain reconstructions of present-day glacier beds, since subtracting a regional glacier-thickness product from a surface DEM can transfer thickness errors directly into the bed geometry on which the synthetic glaciers are grown, and hence into the training targets. We also deliberately select formerly glaciated regions to ensure consistent morphometric properties of glacially shaped terrain.

For the Pyrenees, northern Andes, Mackenzie Mountains and Caucasus, candidate windows are sampled directly from the 155 Copernicus DEM and filtered using Randolph Glacier Inventory version 6.0 outlines (RGI Consortium, 2017) to reject windows with substantial existing glacier cover ($>5\%$). The Alps are treated separately because Alpine thickness observations and derived thickness products are comparatively better constrained. There, we first construct an approximately ice-free bed DEM by subtracting the glacier-thickness estimates of Cook et al. (2023), which are well constrained by radio echo-sounding measurements, over the present day surface DEM and sample from this deglaciated topography without applying the same 160 RGI-based glacier-cover rejection.

For each draw, a target horizontal grid spacing (Δx) is sampled from discrete values between 50 and 250 m. A window with the corresponding extent is extracted at the native DEM resolution and bilinearly resampled to a fixed (180×180) grid. All model domains therefore contain the same number of grid cells, which allows examples to be stacked into fixed-size mini-batches during training, while spanning side lengths from 9 to 45 km, representing a variety of physical and topographic scales. 165 The resulting tiles are screened to reject substantial low-elevation or sea-level areas, insufficient relief, low mean slope, and excessive flat terrain.

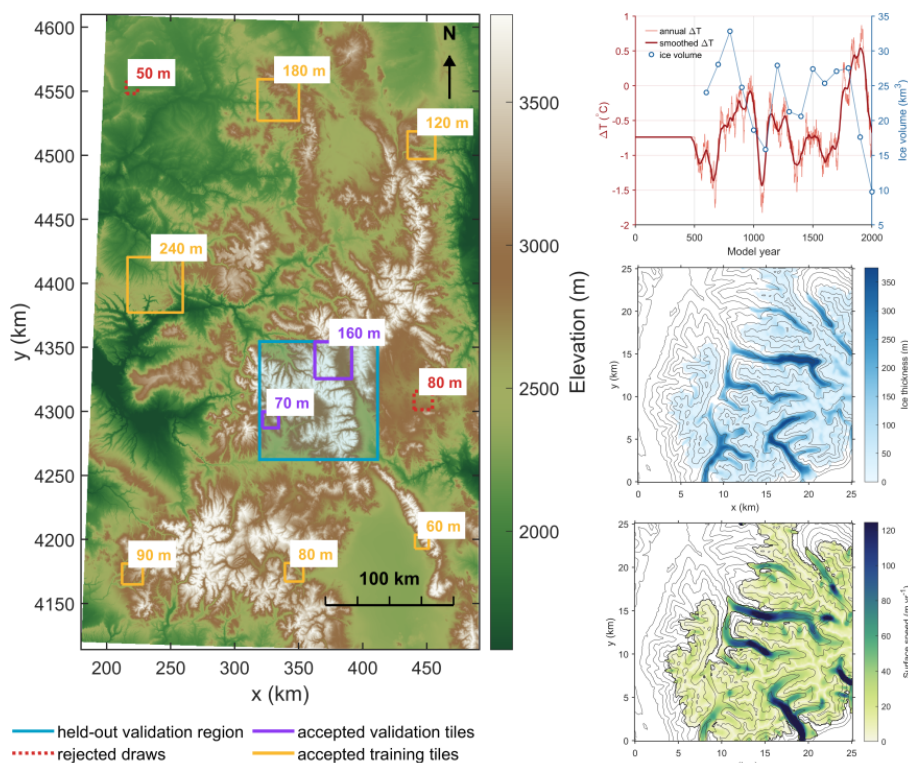


Figure 2. Terrain sampling and an example synthetic glacier realisation. Panel a shows the Rocky Mountains domain, from which we randomly sample tiles at different resolutions. Tiles are sampled at resolutions from 50-250 m. Tiles are rejected in regions that do not have sufficient mountain-like relief or where >5% of the tile is ice covered (red boxes). The region outlined in blue is our validation hold out, and accepted tiles sampled within form the validation set (purple boxes). Panel b shows an example realization of our random climate forcing (red line) and the corresponding simulated ice volume. Panels c and d show ice thickness and surface speed for an example validation tile simulation.

For validation, a contiguous block is withheld within each regional DEM. Training windows are constrained not to intersect this block, and validation windows are sampled entirely within it, producing a spatially independent split. Figure 2a shows a random subset of rejected, training, and validation tiles with different grid spacings in one of the selected mountain regions.

170 We draw a total of 100,000 candidate tiles, and after applying the acceptance criterion, end up with 21,795 training tiles and 1942 validation tiles, split across sampling regions according to their respective total areas (note these tiles provide variety in geometry and resolution, the full training set is an order of magnitude larger because each tile has multiple realisations in terms of basal resistance, rheology and climate forcing).



2.2.2 Physical parameters, augmentation, and climate forcing

175 Each terrain tile that passes our acceptance criteria is reused multiple times in the final training set to form independent glacier realizations that share the same topography but differ in basal resistance, ice rheology, stochastic climate history, and geometric augmentation. Basal resistance (τ_{ref}) is constant in space but is drawn uniformly with a random seed, equivalent to a basal shear-stress magnitude of 40-300 kPa in the Weertman law (Eq. 2). Ice rheology is varied independently through a spatially uniform Arrhenius factor A describing ice softness, drawn over the range 5-140 MPa⁻³a⁻¹. These ranges are not
 180 calibrated regional priors; their purpose is to expose the emulator to a broad span of dynamically plausible mountain-glacier flow regimes. Geometric augmentation is applied randomly to tiles, whereby they are rotated and reflected before initialization, changing the orientation of valleys relative to the raster grid while preserving the training/validation designation of the original tile. Within a single training-set simulation, the bed elevation, ice rheology, basal resistance, and grid spacing are held constant, while ice thickness and surface elevation evolve in response to the climate forcing. Each realisation therefore traces a trajectory
 185 through glacier states at fixed physical parameters. Unlike earlier ice-flow emulators, which were trained on small patches extracted from larger domains, we train on complete model domains. Each example is a full 180 × 180 tile, cropped by 10 cells at the boundary. This ensures that the receptive field of the network is never artificially truncated relative to the domain it will be applied to, and removes patch size as a tuning parameter. Separate deterministic random-numbers drive climate forcing, basal resistance, ice rheology, and augmentation, keeping nominally independent sources of variability reproducible
 190 and uncorrelated.

Surface mass balance provides the forcing through which initially ice-free domains develop into transient glacier geometries. For each realization, a reference equilibrium-line altitude $z_{\text{ELA},0}$ is selected from the upper part of the ice-free terrain hypsometry, and a temperature anomaly $\Delta T(t)$ shifts it according to $z_{\text{ELA}}(t) = z_{\text{ELA},0} + \gamma_T \Delta T(t)$. Let $\Delta z(\mathbf{x}, t) = s_H(\mathbf{x}, t) - z_{\text{ELA}}(t)$, then the surface mass-balance rate is

$$195 \quad \dot{a}_s(\mathbf{x}, t) = \begin{cases} \max(\beta_{\text{abl}} \Delta z, -8), & \Delta z < 0, \\ \min(\beta_{\text{acc}} \Delta z, 2), & \Delta z \geq 0, \end{cases} \quad (6)$$

where $\beta_{\text{abl}} = 0.007$ and $\beta_{\text{acc}} = 0.0025$ are the ablation and accumulation zone gradients. This deliberately simple scheme preserves the first-order coupling between surface elevation, equilibrium-line altitude, and mass balance while avoiding the assumptions required by a regional climate model, and is consistent with the widespread use of temperature-index and ELA-gradient approaches in glacier modelling (Hock, 2003; Zemp et al., 2007).

200 After a 500 year spin-up with constant temperature, the temperature anomaly varies randomly about a mean value chosen for each realisation, but with memory: each year is correlated with the years before it, over a timescale of about 150 years, and the anomaly has a standard deviation of 0.6 °C. This produces sustained warm and cold periods rather than independent year-to-year fluctuations, which matters because glacier geometry responds to the integrated history of mass balance over timescales comparable to the glacier response time (?). Formally, this is a stationary first-order autoregressive process, a standard representation of persistent internal climate variability that has been used to force glacier models in the same way (Hasselmann,
 205



1976; Roe and Baker, 2016). An example of a randomly selected resulting timeseries, along with the corresponding ice volume response over time, can be seen in Figure 2b.

2.2.3 Transient states and reference targets

Each unique terrain/parameter combination is run forwards in time for 2000 years from an initially ice-free state. At every model step, the evolving fields $(h_H(t), s_H(t))$ and fixed parameters $(A_H, \tau_{\text{ref}, H}, \Delta x)$ define the ice-flow input state $\mathbf{I}_H(t)$ in Eq. (3). During ensemble generation, ice flow is evaluated with the online neural solver $\mathcal{M}^{\text{net}}$, and the resulting velocity enters the mass-conservation update in Eq. (1). This makes it computationally feasible to generate many transient trajectories spanning topographic and physical variability and a range of climatic disequilibrium states.

Candidate glacier states are extracted every 100 years, after an initial 500-year spin-up period that ensures glaciers have had a chance to grow under the constant climatic forcing. At each sampling interval, the geometry and physical parameters are frozen and the horizontal velocity is recomputed with the identity mapping $\mathcal{M}^{\text{id}}$, which optimises the velocity field directly and therefore provides the most accurate solution available within this discretisation. The final dataset is therefore

$$\mathcal{D} = \{(\mathbf{I}_{H,k}, \mathbf{U}_{H,k}^{\text{ref}})\}_{k=1}^N, \quad (7)$$

where $\mathbf{U}_{H,k}^{\text{ref}} = \mathcal{M}^{\text{id}}(\boldsymbol{\theta}_{\text{id},k}^*; \mathbf{Z}_{H,k})$ denotes the depth-varying horizontal velocity obtained by optimising the velocity field directly, without any neural parametrisation. The online neural solver $\mathcal{M}^{\text{net}}$ is used only to generate glacier trajectories spanning states of advance and retreat; the stored target $\mathbf{U}_{H,k}^{\text{ref}}$ is always computed with $\mathcal{M}^{\text{id}}$ from the frozen state $\mathbf{I}_{H,k}$. Velocities from the online neural solver are therefore not included in $\mathcal{D}$, although they influence the sampled geometries through the prior evolution of h_H and s_H . This construction concentrates the expensive $\mathcal{M}^{\text{id}}$ solves on the states retained for training and validation while preserving the diversity of the much larger transient ensemble.

We use a Neumann/stress-free boundary condition along the domain boundary, and to further ensure that our training set is not compromised by boundary effects, we crop all inputs and outputs by 10 pixels around the boundary. We also apply filters to retain only useful examples. The most important of these reject examples where the 99th-percentile surface speed over ice exceeds 1500 m yr^{-1} or falls below 10 m yr^{-1} , and examples where ice (thickness $> 5 \text{ m}$) covers less than 10% of the total domain. The upper limit removes rare states in which an extreme combination of sampled rheology, basal resistance, and geometry produces flow speeds far outside the range of the land-terminating mountain glaciers the emulator is intended for. The lower limit removes near-stagnant states, which carry little information about the mapping from geometry to velocity. Finally, after completing the training set simulations and filtering, we end up with $N = 347,751$ targets, of which $\sim 10\%$ are retained for validation. An example input-target pair is shown in Fig. 2c-d.

2.3 Emulator architecture

The offline emulator uses the network mapping with weights learned across the complete training dataset to approximate the depth-varying horizontal velocity degrees of freedom,

$$\hat{\mathbf{U}}_H = \mathcal{M}^{\text{net}}(\hat{\boldsymbol{\theta}}; \mathbf{Z}_H), \quad (8)$$



where $\hat{\theta}$ denotes the final offline-trained weights. Once training is complete, these weights are held fixed, so no further optimisation is required when the emulator is applied to a new glacier state.

240 We compare two emulator designs. The first is the previous default within IGM: a convolutional neural network (CNN) that receives only the standardised glacier state as input, $\mathbf{Z}_H = \tilde{\mathbf{I}}_H$, where standardised means each input field is rescaled to zero mean and unit variance using training-set statistics. The second architecture is DahuNet, following the physics-enriched parametrisation introduced in the companion study (Gregov et al., 2026). In addition to the standardised glacier state, DahuNet receives five deterministic feature fields,

$$245 \quad \mathbf{Z}_H = [\tilde{\mathbf{I}}_H, \Psi_5(\mathbf{I}_H)], \quad (9)$$

where Ψ_5 is calculated from the unstandardised ice thickness, surface elevation, and grid spacing. The five features describe the two components and magnitude of the surface slope, together with two simple geometric velocity-scale proxies proportional to $|\nabla s_H|^3 h_H^4$ and $|\nabla s_H|^3 h_H^3$. They therefore provide the network with the direction of the local driving slope and with leading thickness slope scalings associated with ice deformation and basal sliding. Because these fields are calculated directly from the
 250 current glacier state, they introduce no additional trainable parameters.

The CNN and DahuNet configurations have similar convolutional structures and closely comparable numbers of trainable parameters. This allows us to test whether providing physically meaningful input features improves emulator accuracy without substantially changing model capacity. Full details of the output representation, input preprocessing, network layers, feature normalisation, and parameter counts are given in Appendix A.

255 2.4 Hybrid data physics pretraining

The emulator is trained using two complementary sources of information. First, its predicted velocities are compared with the reference velocities $\mathbf{U}_{H,k}^{\text{ref}}$ computed with the identity mapping $\mathcal{M}^{\text{id}}$. Second, the predicted velocity fields are evaluated using the same discretised Blatter-Pattyn energy $\mathcal{J}_H$ used to generate those reference solutions. The combined training objective is

$$\mathcal{L}_t = \mathcal{L}_{\text{vel}} + \lambda_t \mathcal{L}_{\text{phys}}, \quad (10)$$

260 where $\mathcal{L}_{\text{vel}}$ measures the difference between predicted and reference velocities, $\mathcal{L}_{\text{phys}}$ measures the physical consistency of the predicted velocity field, and λ_t controls their relative influence during training.

The velocity term uses a Huber loss. This behaves quadratically for typical velocity errors while limiting the influence of rare, very large errors in fast-flowing or geometrically unusual states. The physics term inserts the predicted velocity field directly into $\mathcal{J}_H$ and therefore introduces no second physical model. Instead, it penalises predictions that may reproduce
 265 the reference velocities locally but remain energetically inconsistent with the same Blatter-Pattyn problem from which those reference velocities were generated.

Because the data and physics terms have different units, magnitudes, and sensitivities to glacier state, their relative weight cannot be prescribed by directly comparing their numerical values. Training therefore begins with a supervised warm-up in which only $\mathcal{L}_{\text{vel}}$ is applied. Once the emulator has learned an approximate mapping from glacier state to velocity, the physics



Table 1. Median (p50) and 90th-percentile (p90) emulator errors (RMSE) over the validation set for the four trained emulators, evaluated at every ice-covered cell across all N validation members. Lower is better; the lowest median in each row is highlighted.

Metric (units)	Dahu $_{\lambda}$		Dahu $_0$		CNN $_{\lambda}$		CNN $_0$	
	p50	p90	p50	p90	p50	p90	p50	p90
Surface-speed error (m yr^{-1})	0.71	3.3	0.90	3.6	1.6	7.2	1.6	6.5
Basal-speed error (m yr^{-1})	0.64	2.9	0.83	3.3	1.4	6.1	1.5	5.6
Flow-direction error (deg)	1.1	3.4	1.3	4.2	2.3	7.3	2.4	7.4
Flux-divergence error $\nabla \cdot (h\bar{u})$ (m yr^{-1})	0.10	0.86	0.15	1.3	0.27	2.3	0.29	2.6

term is introduced with a deliberately low weight. This weight is then adjusted during the remainder of training using the relative spread of the gradients produced by the two loss terms. The complete loss definitions, adaptive weighting scheme, training schedule, and optimisation settings are provided in Appendix A.

Our primary comparison spans four emulators that isolate the effects of the input representation and the hybrid training objective. The data-only controls use $\lambda_t = 0$ throughout, giving CNN $_0$ and Dahu $_0$; their hybrid counterparts, trained with the adaptive physics weight, are CNN $_{\lambda}$ and Dahu $_{\lambda}$. Comparing the CNN and DahuNet variants tests the effect of the physics-enriched input representation, while comparing the subscripts 0 and λ tests the contribution of the variational physics term. All four models use otherwise identical training settings.

To assess whether emulator accuracy is limited by network capacity, we additionally train smaller and larger variants of Dahu $_{\lambda}$. These experiments and the corresponding architectural modifications are described in Appendix B.

3 Results

3.1 Offline emulator accuracy on the validation set

We first evaluate the four primary emulators: CNN $_0$, CNN $_{\lambda}$, Dahu $_0$, and Dahu $_{\lambda}$ against the held-out validation targets $\mathbf{U}_{H,k}^{\text{ref}}$ computed with the identity mapping $\mathcal{M}^{\text{id}}$. All metrics are evaluated per raster cell over ice-covered terrain ($h_H > 1$ m), with the slowest-flowing cells ($< 10 \text{ m yr}^{-1}$) excluded from the flow-direction error, and are reported as median (p50) and 90th-percentile (p90) values across all members of the validation set (Table 1).

We measure the skill of our trained emulators with four complementary error measures. The surface-speed and basal-speed errors quantify the discrepancy between predicted and reference horizontal speed at the top and bottom of the ice column, and therefore expose both the SIA-derived vertical-shear contribution and the basal-motion contribution to the error. The flow-direction error measures the angular discrepancy between predicted and reference horizontal velocity vectors, and is insensitive to errors in speed alone. The flux-divergence error targets $\nabla \cdot (h\bar{u})$, the depth-integrated horizontal flux divergence that drives the thickness evolution in Eq. (1). This last measure is a more demanding and arguably more relevant diagnostic than pointwise

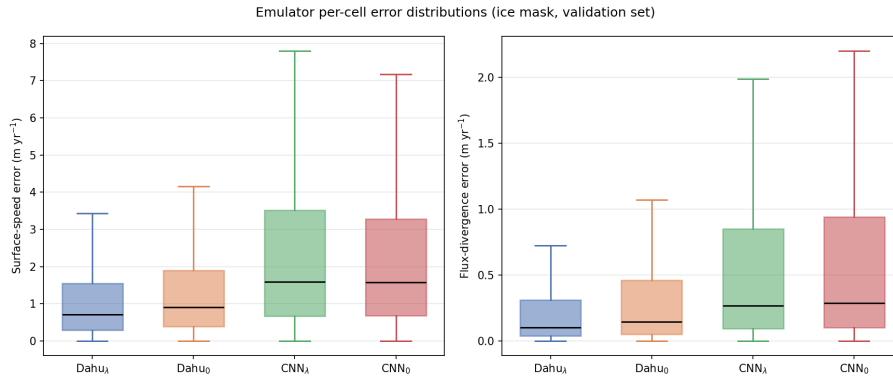


Figure 3. Distribution of emulator (a) Surface-speed error and (b) flux-divergence error, evaluated over the validation set for the four trained emulators. Boxes span the interquartile range with the median marked; whiskers extend to the most extreme value within $1.5 \times \text{IQR}$ of the quartiles.

speed because it depends on spatial gradients of the velocity field rather than its local magnitude. An emulator could reproduce speeds accurately and yet still misrepresent the convergence and divergence patterns that govern mass redistribution. Errors in $\nabla \cdot (h\bar{u})$ therefore indicate how faithfully an emulator would drive transient geometry change when coupled back into the mass-conservation update, which is the setting in which it is ultimately used.

Table 1 shows a clear separation between the four emulators. DahuNet beats the CNN by a wide margin, and within each architecture the physics penalty gives a further, smaller gain. The data-only Dahu₀ already roughly halves the median surface-speed error of either CNN variant, with comparable gains at the base, in flow direction, and in flux divergence (Table 1). We attribute this to the design of DahuNet: the deterministic feature fields Ψ_5 supply the driving-slope direction and the geometric velocity-scale proxies directly to the network, rather than requiring the convolutional layers to learn these strongly nonlinear dependencies from the raw state alone.

The variational penalty improves both architectures, but its effect is smaller and largely confined to DahuNet, for which Dahu_λ attains the lowest error on every metric. For the CNN the two variants are nearly indistinguishable at the median. This suggests that the physics penalty is only useful once the architecture is expressive enough to exploit the energetic-consistency signal; for the weaker CNN backbone, supervised velocity error dominates and the penalty adds little.

The benefit of both factors is most pronounced in the upper tail of the error distributions, and strongest of all for flux divergence (Fig. 3). Relative to the CNN variants, Dahu_λ reduces the p90 surface-speed error by roughly a factor of two, and the p90 flux-divergence error by a factor of three to four. The DahuNet flux-divergence distributions are also markedly narrower than their CNN counterparts, confirming that the improvement holds across the full distribution and not only at its centre. Taken together, these results establish Dahu_λ as the most accurate of the four emulators, with the architecture contributing the larger share of the improvement and the variational penalty adding a further, tail-weighted refinement. In the sections that follow, therefore, when talking about emulator performance we refer specifically to the Dahu_λ emulator, unless stated otherwise.

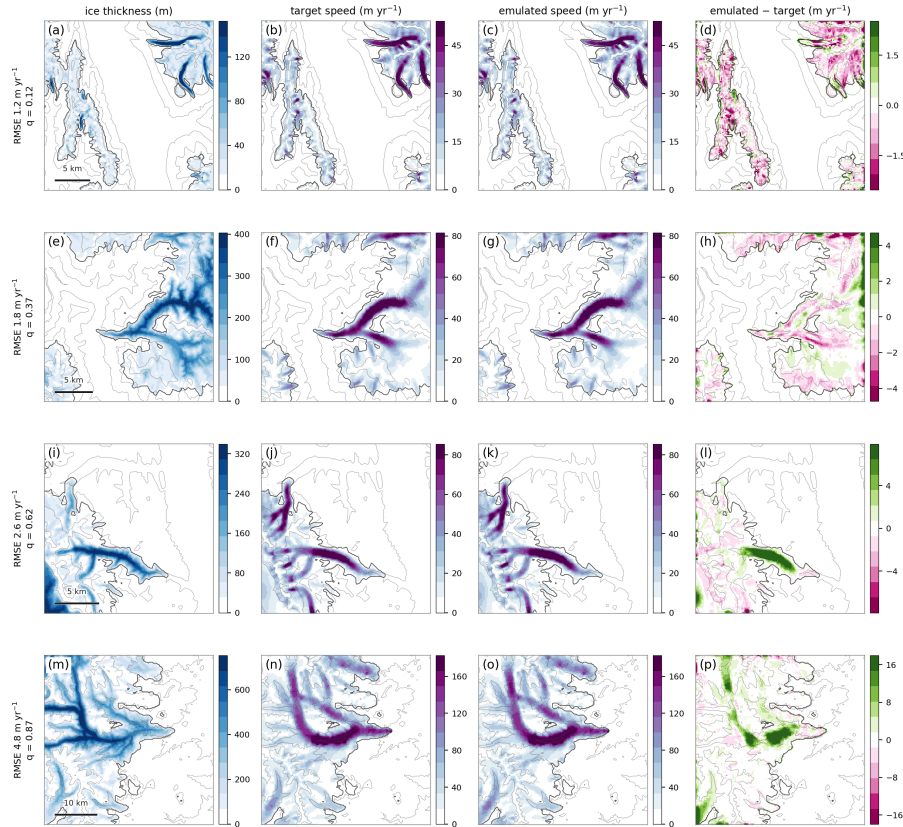


Figure 4. Qualitative comparison of emulated and target ice surface speeds for the Dahu λ emulator. Each row shows one validation member, with members selected at evenly spaced quantiles of the per-member surface-speed RMSE distribution to span performance from best (top) to worst (bottom). Columns show, from left to right: ice thickness (m), the target surface speed from the numerical ice-flow solver, the emulated surface speed, and their difference (emulated - target). Light gray contours show bed topography, black lines delineate ice extent.

As an additional visual diagnostic, we show representative validation examples in Fig. 4. These maps confirm that the error metrics described above are not dominated by a few favourable cases: across examples spanning the per-member RMSE distribution, the emulator reproduces speed patterns well. The largest residuals occur mainly in fast-flowing regions and near sharp spatial gradients, consistent with flux divergence being the more demanding diagnostic for transient use. The residual error is unlikely to be limited by network capacity: tripling the parameter count produces little further improvement (Appendix B). The largest residuals instead occur where the velocity field varies on scales approaching the grid spacing, which points to the local, fixed-receptive-field nature of the convolutional architecture (its receptive field is the spatial extent of input that can influence any single output cell), and to the relative scarcity of such states in the training distribution, as the more likely limitations.



3.2 Transient simulations

Validation-set accuracy measures how closely $\hat{\mathbf{U}}_H$ reproduces $\mathbf{U}_H^{\text{ref}}$ for a prescribed glacier state $\mathbf{I}_H$, but it does not establish fidelity when the emulator is used within a transient simulation. In this setting, $\hat{\mathbf{U}}_H$ determines ice-flux divergence and hence the subsequent evolution of h_H and s_H through Eq. (1). Persistent or spatially structured velocity errors therefore alter the states presented to the emulator at later time steps, potentially causing the emulator-driven trajectory to diverge from the reference trajectory even when individual predictions are relatively accurate. The distinction between offline accuracy and prognostic performance is well recognised in machine-learning weather and climate modelling, where models with good one-step skill can drift, encounter states outside their training distribution, or become unstable under repeated integration (Rasp et al., 2020; Wang et al., 2022; Parthipan et al., 2024). Glacier evolution is slower and less chaotic than atmospheric dynamics, but the same requirement remains: an emulator intended for century-scale simulations must reproduce both the instantaneous velocity field and the coupled evolution of glacier geometry. We therefore assess transient fidelity by comparing emulator-driven and reference trajectories through extended periods of glacier growth and retreat.

We evaluate transient fidelity for two contrasting glacier systems: the Columbia Icefield, a large temperate icefield in the Canadian Rockies, and Hofsjökull, an Icelandic subpolar ice cap. Note that neither of these domains form part of either the training or validation sets. For each domain, we conduct both a reference simulation and an emulator-driven simulation that differ only in the mapping used to evaluate the velocity field. The reference simulation uses the identity mapping $\mathcal{M}^{\text{id}}$, following the same procedure used to generate the targets in Eq. (7). The emulator-driven simulation instead uses the network mapping $\mathcal{M}^{\text{net}}$ with the fixed, offline-trained weights $\hat{\theta}$ from Eq. (8). All other model components and numerical settings are identical. Each simulation is run for 500 years through a prescribed advance-retreat cycle driven by a lowering and subsequent raising of the equilibrium-line altitude. Any divergence between the two trajectories can therefore be attributed to the replacement of the identity-mapping velocity solution with a fixed pretrained emulator, consisting of both instantaneous velocity errors and the accumulated effects of those errors on the evolving glacier geometry. More details on these simulations can be found in Appendix D.

Figure 5 (left column) compares the emulator-driven and reference trajectories over the full advance-retreat cycle for both domains. The top panel plots integrated ice volume over time for the two domains, showing the growth and retreat cycle of both experiments. The emulator reproduces this trajectory closely: the curves remain nearly indistinguishable as each glacier grows and retreats, and the maximum relative volume error remains below 6% in both cases. The errors in speed and thickness (lower panels) also remain bounded relative to the corresponding reference fields. Surface-speed RMSE remains a few m yr^{-1} , less than 10% of the mean reference speed at all times, while thickness RMSE reaches about 10 m for mean ice thicknesses of roughly 100-200 m. None of these metrics shows long-term growth over the 500-year integrations; the errors vary with the dynamical regime but do not accumulate into systematic drift. The emulator therefore reproduces both the instantaneous velocity field and the coupled evolution of glacier geometry through the transition from growth to retreat.

Recent versions of IGM are built on the principle of physics informed neural networks, as described in Jouvét and Cordonnier (2023). In this approach, the emulator is pretrained so that the network has already learnt a useful representation of ice-flow, but

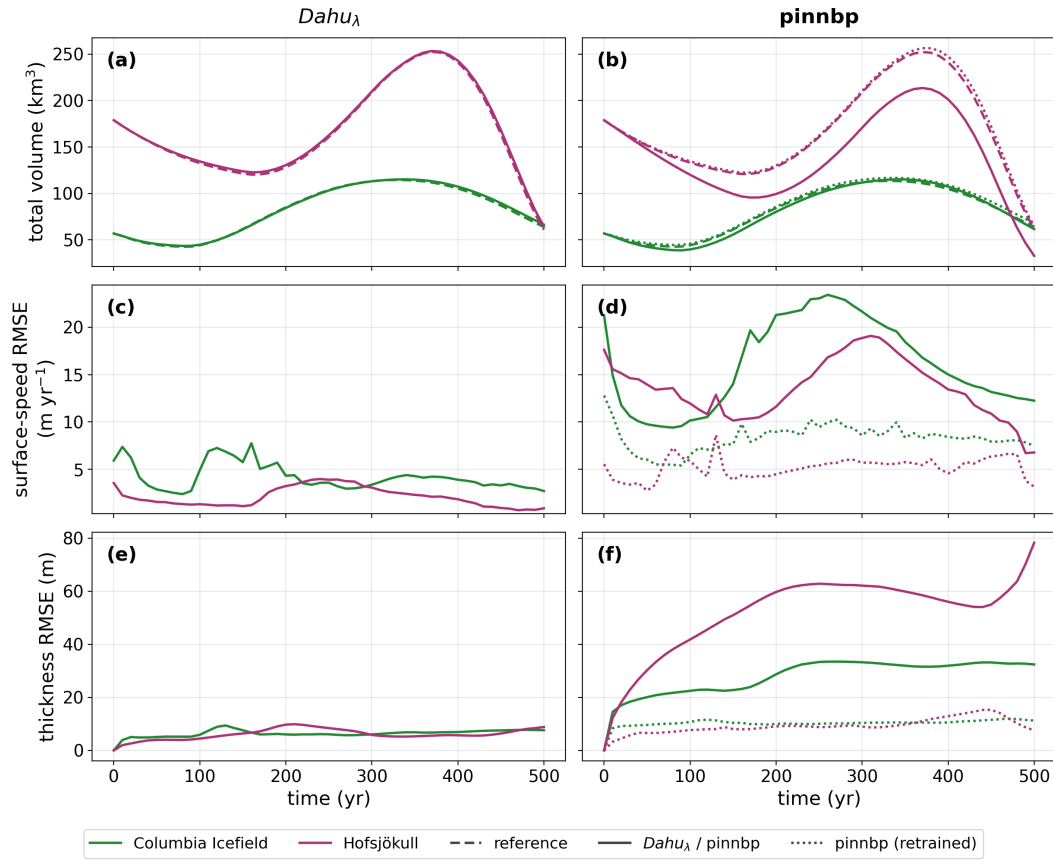


Figure 5. Accuracy of two ice-flow emulators over a 500-yr advance-retreat experiment, for Columbia Icefield (green) and Hofsjökull (magenta). Left column: $Dahula$ emulator; right column: $pinnbp$ emulator (Jouvet and Cordonnier, 2023). Each emulator, using the network mapping $\mathcal{M}^{net}$ with fixed offline-trained weights $\hat{\theta}$, is compared against its own reference simulation obtained with the identity mapping $\mathcal{M}^{id}$ at the emulator’s native discretisation ($Dahula$: $N_z = 2$, MOLHO; $pinnbp$: $N_z = 10$, Lagrange). The two references are not directly comparable to each other; within each column the only difference between the two trajectories is the use of $\mathcal{M}^{net}$ versus $\mathcal{M}^{id}$. (a, b) Total ice volume: emulator ($\mathcal{M}^{net}$, solid) vs. reference ($\mathcal{M}^{id}$, dashed). (c, d) Surface-speed RMSE and (e, f) ice-thickness RMSE between emulator and reference, evaluated over the union of their ice masks. Dotted lines in the right column show $pinnbp$ with online retraining enabled

355 it must be continually retrained on current glacier state during a transient simulation to ensure the solution remains accurate throughout. The key advancement in the work presented here is that we can achieve high accuracy without any retraining. To make this contrast between old and new pretrained emulators explicit, we conduct the same transient simulations for Columbia Icefield and Hofsjökull with the previous emulator, which we refer to as $pinnbp$ hereafter. Since the $pinnbp$ and $Dahula$ emulators differ in several respects beyond their learned weights, most notably vertical discretization scheme, we
 360 generate a separate reference solution for $pinnbp$ so that each emulator is compared to the equivalent ‘true’ solution that they emulate.



Figure 5 (right column) shows the results of these transient simulations for the *pinnbp* emulator. For both the Columbia Icefield and Hofsjökull, Dahu_λ tracks its reference volume trajectory through the full advance-retreat cycle, whereas the *pinnbp* emulator departs visibly from its own reference, with the divergence most pronounced near the volume maximum (Fig. 5b). Dahu_λ keeps surface-speed RMSE to a few m yr⁻¹ and thickness RMSE near 10m throughout (Fig. 5c, e), while the *pinnbp* emulator exhibits substantially larger surface-speed RMSE that persists across the integration and thickness RMSE that grows through the advance phase to several tens of metres (Fig. 5d, f). We emphasise again that applying the *pinnbp* emulator without retraining is not its intended use case, and to make this point clear we also show results with greatly reduced misfit when retraining is enabled (dotted lines in Fig. 5). In this setting, we can see that thickness and speed errors are similar to Dahu_λ and the most important distinction between the two simulations becomes computational cost. Retraining carries a substantial computational cost in both run time and memory. For the 500-year Columbia Icefield simulation, Dahu_λ is more than 18× faster than *pinnbp* with retraining enabled and uses ~37% less peak GPU memory, while still achieving greater accuracy (6.39 m vs. 10.26 m thickness RMSE, each measured against its own reference solution). This speed advantage holds despite *pinnbp* being a much lighter network than Dahu_λ (~140 K vs. 464 K parameters); without retraining, *pinnbp* is in fact nearly twice as fast as Dahu_λ. The cost is therefore attributable to retraining itself: under our retraining schedule, enabling it makes *pinnbp* ~33× slower and increases its peak GPU memory by ~2.7×.

3.3 Decomposition of the transient error

A well bounded transient error is not guaranteed even if the emulator is accurate. A finite error in $\hat{\mathbf{U}}_H$ perturbs the evolving geometry, which in turn changes the state supplied to the emulator at later time steps. The fact that Dahu_λ emulator error stays bounded in Figure 5 therefore motivates a decomposition of the instantaneous emulator error and the geometry-induced contribution associated with the divergence of the emulator-driven state. Let $\mathbf{I}_H^R(t)$ and $\mathbf{I}_H^E(t)$ denote the glacier states along the reference and emulator-driven trajectories. For a given state we reduce the depth-averaged horizontal velocity field to a single scalar diagnostic, $V(\mathbf{I}_H) = \langle |\bar{\mathbf{u}}_H| \rangle_{\mathcal{A}}$, where $|\bar{\mathbf{u}}_H|$ is the depth-averaged speed and $\langle \cdot \rangle_{\mathcal{A}}$ denotes the spatial mean over the ice mask $\mathcal{A}$: the cells with positive ice thickness in both the reference and emulator-driven states. We write $V^{\text{id}}(\mathbf{I}_H)$ and $V^{\text{net}}(\mathbf{I}_H)$ for this diagnostic evaluated with $\mathcal{M}^{\text{id}}$ and with $\mathcal{M}^{\text{net}}$ at the fixed weights $\hat{\theta}$, respectively. In this notation the superscript identifies which mapping evaluates the diagnostic, and the argument identifies which trajectory supplied the glacier state. The total free-running error in this diagnostic can then be decomposed as

$$\underbrace{V^{\text{net}}(\mathbf{I}_H^E) - V^{\text{id}}(\mathbf{I}_H^R)}_{e_{\text{tot}}} = \underbrace{V^{\text{net}}(\mathbf{I}_H^R) - V^{\text{id}}(\mathbf{I}_H^R)}_{e_{\text{inst}}} + \underbrace{V^{\text{net}}(\mathbf{I}_H^E) - V^{\text{net}}(\mathbf{I}_H^R)}_{e_{\text{geom}}}, \quad (11)$$

where e_{inst} is the instantaneous emulator error on the reference state and e_{geom} is the geometry-induced term: the change in the emulator diagnostic caused by evaluating it on the emulator-driven rather than the reference geometry. This signed diagnostic

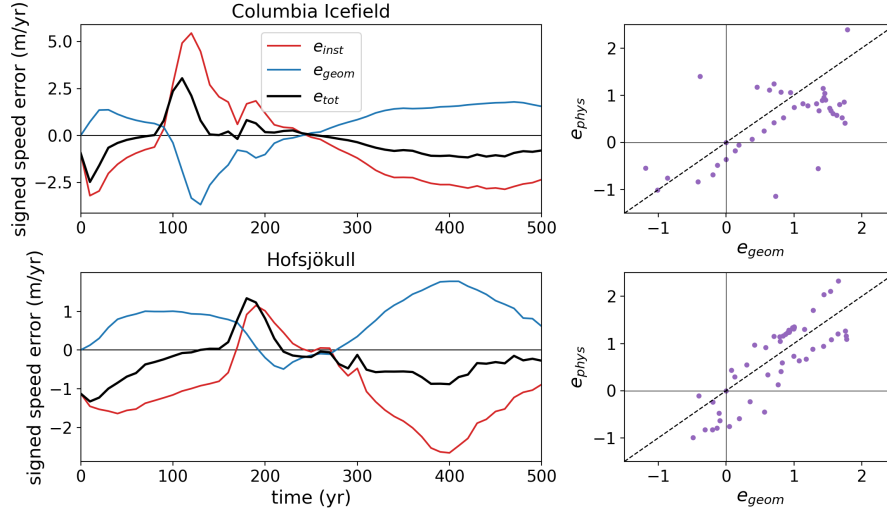


Figure 6. Decomposition of the free-running transient error in the signed mean-speed diagnostic V (Eq. 11), for Columbia Icefield (top) and Hofsjökull (bottom). Left column: time series over the 500-yr advance-retreat experiment of the total error $e_{\text{tot}} = V^{\text{net}}(I_H^E) - V^{\text{id}}(I_H^R)$ (black), the instantaneous emulator error $e_{\text{inst}} = V^{\text{net}}(I_H^R) - V^{\text{id}}(I_H^R)$ (red), and the geometry-induced term $e_{\text{geom}} = V^{\text{net}}(I_H^E) - V^{\text{net}}(I_H^R)$ (blue), where V^{net} and V^{id} denote V evaluated with the network mapping $\mathcal{M}^{\text{net}}$ at fixed weights $\hat{\theta}$ and with the identity mapping $\mathcal{M}^{\text{id}}$, respectively, and I_H^R and I_H^E are the reference and emulator-driven states. Right column: the geometry-induced term e_{geom} plotted against the identity-mapping response to the same state difference, $e_{\text{phys}} = V^{\text{id}}(I_H^E) - V^{\text{id}}(I_H^R)$.

differs in purpose from the surface-speed RMSE reported in Figure 5: the RMSE measures the magnitude of pointwise field errors, whereas V isolates the net spatial bias and its partial cancellation.

Evaluating Eq. (11) requires two diagnostic ‘replay’ simulations in addition to the free-running reference and emulator-driven simulations. In the *emulator replay*, $\mathcal{M}^{\text{net}}$ with fixed weights $\hat{\theta}$ is evaluated at each saved time on the prescribed reference state $\mathbf{I}_H^R(t)$, yielding $V^{\text{net}}(\mathbf{I}_H^R)$ and hence e_{inst} . In the *solver replay*, $\mathcal{M}^{\text{id}}$ is applied to the prescribed emulator-evolved state $\mathbf{I}_H^E(t)$, yielding $V^{\text{id}}(\mathbf{I}_H^E)$. This defines the further diagnostic $e_{\text{phys}} = V^{\text{id}}(\mathbf{I}_H^E) - V^{\text{id}}(\mathbf{I}_H^R)$, which is the response of the identity-mapping solution to the same state difference. Comparing e_{geom} with e_{phys} tests whether the emulator reproduces the geometry sensitivity of the identity-mapping solution.

Figure 6 presents this decomposition. In both domains, the instantaneous error and geometry-induced term generally have opposite signs: a persistent negative emulator bias changes the geometry and driving stress in a direction that partly offsets the original bias. The terms therefore largely cancel, leaving e_{tot} smaller than either contribution separately. The time-mean absolute magnitudes of the signed-speed diagnostics are $|e_{\text{inst}}| \approx 1.2$, $|e_{\text{geom}}| \approx 0.8$, and $|e_{\text{tot}}| \approx 0.5 \text{ m yr}^{-1}$ for Hofsjökull, and $|e_{\text{inst}}| \approx 1.9$, $|e_{\text{geom}}| \approx 1.2$, and $|e_{\text{tot}}| \approx 0.8 \text{ m yr}^{-1}$ for the Columbia Icefield. The geometry-induced term also tracks the reference-solver response: e_{geom} plotted against e_{phys} lies close to the 1:1 line. The correspondence is stronger for Hofsjökull (correlation ≈ 0.8) than for the Columbia Icefield (correlation ≈ 0.5), but is evident in both cases. For these two domains,



the cancellation is consistent with a stabilising geometry-velocity feedback, and the comparison with e_{phys} indicates that the emulator captures the relevant geometry sensitivity closely enough to keep its trajectory near the reference.

4 Discussion

The central result of this study is that the iterative velocity solve at the heart of higher-order glacier modelling can be replaced by a single direct evaluation of a fixed pretrained mapping, on previously unseen mountain-glacier geometries and without glacier-specific training. The emulator accurately reproduces $\nabla \cdot (h\bar{u})$, the flux divergence through which ice flow controls thickness evolution, and remains close to the reference solution throughout the 500-year transient experiments. For the preferred model, Dahu_λ , median surface- and basal-speed errors are below 1 m yr^{-1} , median flow-direction error is close to one degree, and the flux-divergence error distribution is substantially narrower than for the plain CNN architecture.

This study builds directly on the ice-flow emulator concept introduced by Jouvét et al. (2022) and the subsequent variational neural-solver and physics-informed ice-flow emulation work of Jouvét and Cordonnier (2023), but the emphasis here is different. Earlier IGM applications demonstrated that neural mappings can greatly accelerate ice-flow calculations, but the emulator was treated as a component to be retrained on the target problem to ensure an accurate solution. Iteration was therefore reduced rather than removed: the optimisation moved from the velocity field to the network weights, and still had to be repeated for each new problem. Here, the objective is a more general pretrained mapping for land-terminating mountain glaciers, in which the optimisation is performed once during training and never repeated. Applying the emulator to a new glacier requires no solve of either kind. In this setting, Dahu_λ performs well: applied without retraining it follows its own reference solution much more closely over the Columbia Icefield and Hofsjökull advance-retreat experiments than *pinnbp* does, though this is not *pinnbp*'s intended operating mode and the two emulators differ in several respects. The fairer comparison is with *pinnbp* in its intended configuration, with retraining enabled. There the two reach comparable accuracy, and the distinction becomes cost: Dahu_λ is more than $18\times$ faster and uses 37% less GPU memory, because it has no retraining loop and therefore no optimiser state to carry alongside the forward model. The result shows that the pretrained Dahu_λ mapping is well suited to the role considered here: a fixed emulator that can be deployed on unseen mountain-glacier and ice-cap geometries without additional training.

4.1 Role of the synthetic training set

A major methodological contribution of this study is the scale and diversity of the synthetic training catalogue. Earlier ice-flow emulators were necessarily trained on relatively small collections of model solutions, limiting their ability to generalise without glacier-specific training. Here, more than 300,000 reference velocity solutions span a wide range of mountain topographies, glacier geometries, grid spacings, basal resistances, ice rheologies, and stages of advance and retreat. The emulator therefore encounters much of the variability relevant to land-terminating mountain glaciers during training, before it is applied to a real glacier.

This catalogue is made computationally feasible by separating the generation of glacier geometries from the calculation of training targets. The inexpensive online neural solver produces transient trajectories containing advancing, retreating, and near-



equilibrium states, but its velocities are not used as reference data. Instead, the velocity for every retained state is recomputed with the higher-order reference solver. The online solver therefore influences which geometries are sampled, but not the velocity fields that the fixed emulator is trained to reproduce.

The use of synthetic glacier growth also avoids a common difficulty in mountain-glacier modelling: the bed geometry beneath most glaciers is poorly constrained. Global and regional thickness products have become available in recent years and are now indispensable for large-scale modelling (Farinotti et al., 2019; Millan et al., 2022), but they remain model-based estimates rather than direct measurements for most glaciers. Growing synthetic glaciers on mostly ice-free topography therefore reduces the risk that errors in present-day thickness products are transferred directly into the bed geometries used for training. This choice is especially appropriate for an emulator whose purpose is to learn the mapping from geometry and physical parameters to velocity, since existing glacier thickness products would likely introduce a systematic bias into the emulator predictions.

4.2 Architecture, hybrid training, and transient stability

The four-emulator comparison indicates that the physically enriched input representation is more important than the variational penalty by itself. The deterministic feature fields Ψ_5 give DahuNet direct access to surface-slope direction, slope magnitude, and simple geometric velocity-scale proxies. These are among the dominant controls on mountain glacier flow, so it is unsurprising that exposing them explicitly reduces the challenge for the convolutional layers to learn an accurate physical mapping. The data-only DahuNet already outperforms both CNN variants, while the hybrid objective provides a smaller but systematic improvement, most clearly in the upper tail of the error distributions and in flux divergence. This is an important distinction. Our results show the accuracy of Dahu $_{\lambda}$ stems from a combination of our broad training distribution, a physically useful state representation, and a variational regularisation term that discourages physically inconsistent velocity fields.

The transient experiments provide the strongest evidence that the emulator can serve as an ice-flow component rather than merely reproduce isolated velocity fields. In both Columbia Icefield and Hofsjökull, Dahu $_{\lambda}$ follows the reference volume trajectory throughout the 500-year advance-retreat cycle, with no long-term growth in velocity or thickness error. The decomposition in Eq. (11) identifies a stabilising negative feedback: an emulator speed bias changes ice thickness and surface slope, which modifies the driving stress in a direction that often opposes the initial bias. A positive speed bias, for example, tends to thin the glacier and reduce its driving stress, producing a subsequent negative speed correction. The coupled emulator mass-conservation system is therefore more robust than its instantaneous velocity error alone might suggest. This compensation mechanism is not a guarantee, as it still requires the emulator to have learnt the correct sensitivity between thickness and velocity, as demonstrated by the *pinnbp* experiment, but it explains why the relatively small errors of Dahu $_{\lambda}$ remain bounded rather than accumulating over time.

As an additional local sanity check of the emulator, Appendix C perturbs a patch in each emulator input field in the main trunk of Aletsch glacier (Fig. C1). This experiment explores whether the learned mapping responds smoothly and with the expected sign to controlled input changes. The response is spatially coherent: softer ice increases local speed, stronger basal resistance reduces it, and a surface-elevation perturbation produces an along-flow speed-up/slow-down dipole through its effect on surface slope. This supports the interpretation that the emulator has learned a locally physical input-velocity relationship.



4.3 Computational implications

The fixed emulator developed here and the online neural solver studied in the companion paper address the same computational bottleneck in different ways. The online approach adjusts the network to each evolving glacier state, providing an opportunity for state-specific improvement but requiring choices about optimisation frequency, learning rate, warm starts, and stopping criteria (Gregov et al., 2026). The emulator presented here instead pays its main cost beforehand through the construction of a large training catalogue and a single training procedure. Once trained, evaluating it requires no further optimisation and introduces no additional solver-tuning choices for the user. It cannot specialise itself to a new glacier state, but within its training regime the same fixed mapping can be reused as an inexpensive default ice-flow component.

For mountain-glacier modelling, the practical gain is not limited to accelerating an individual simulation. A reusable emulator makes it feasible to include two-dimensional higher-order ice dynamics in experiments that require many repeated model evaluations: regional simulations at valley-glacier resolution, ensembles sampling uncertain climate, basal resistance and ice rheology, and systematic comparisons of alternative initial geometries or mass-balance forcings. Many such experiments currently rely on flowline or other simplified flow approximations because applying a higher-order solver to every glacier, parameter set and time step would be prohibitively expensive. Within its training regime, the emulator removes much of this computational barrier without requiring glacier-specific training or expert choices about numerical optimisation.

The same properties create opportunities beyond forward projection. Inversions and data-assimilation experiments require repeated model evaluations and, in gradient-based approaches, differentiation through the ice-flow calculation. The emulator provides both at low cost because it is a fixed differentiable mapping from glacier geometry and physical parameters to velocity. The smooth and physically coherent perturbation responses in Appendix C provide an initial indication that it may be suitable for such applications. The results presented here open a route towards higher-order inversions of quantities such as basal resistance, ice rheology, and poorly constrained glacier geometry at scales where repeated numerical ice-flow solves are currently difficult to afford.

4.4 Limitations and future work

The limits of this emulator stem largely from the construction of the training set. The synthetic states contain land-terminating mountain glaciers on fixed-size raster domains with horizontal grid spacings between 50 and 250 m. They use the two-layer MOLHO vertical representation, spatially uniform Arrhenius factors and basal resistance parameters, a simple ELA-gradient surface mass balance, and no explicit thermomechanical coupling, evolving basal hydrology, calving, floating ice, or marine boundary forcing. The emulator should therefore be interpreted as a pretrained mapping over this sampled mountain-glacier state distribution, not as a general ice-flow solver capable of handling ice sheets or tidewater glaciers.

Future work should first expand the training distribution towards more heterogeneous and dynamically complex regimes, including spatially variable basal properties, stronger sliding contrasts, thermomechanical coupling, and eventually marine-terminating or floating ice. A second priority is to reduce the dependence on a fixed raster size and fixed resolution range, either through resolution-independent architectures or through training procedures that expose the network to a wider range



505 of grid geometries. Finally, the offline and online approaches described here and in the companion study should be combined explicitly, testing whether pretrained weights or pretrained physics-enriched representations can reduce the cost of subsequent online variational refinement.

5 Conclusions

Higher-order ice flow is normally the expensive component of transient mountain-glacier modelling because the velocity
510 problem must be solved iteratively whenever glacier geometry changes. We show that this repeated calculation can be replaced by a direct evaluation of a pretrained nonlinear mapping. The emulator is trained on more than 300,000 higher-order velocity solutions spanning diverse mountain topographies, geometries, climate histories, resolutions, and physical parameters. Once trained, it returns the depth-dependent velocity field for a new glacier state directly, without optimising either the velocity field or the network weights.

515 The emulator accurately reproduces velocity and ice-flux divergence on glacier states outside its training set. In 500-year advance-retreat experiments for two real glacier systems, it follows the reference ice-volume trajectories with maximum errors below 6 % and without progressive drift. Velocity biases induce changes in geometry and driving stress that tend to oppose the initial error, providing a stabilising feedback within the coupled emulator mass-conservation system.

Within the land-terminating regime represented by its training data, the same fixed emulator can be applied directly to new
520 glaciers without additional training or iterative solver optimisation. This largely removes the higher-order velocity calculation as a computational constraint, making it practical to resolve individual valley glaciers at high spatial resolution across a region, to run large ensembles to better constrain uncertainty, and to extend the number or duration of transient simulations. Modellers can therefore choose higher-order dynamics because they are appropriate to the scientific question, rather than being forced towards simplified approximations primarily by computational cost. The emulator does not remove uncertainty in glacier
525 geometry, climate forcing, or basal conditions, and its application remains limited to the regime represented during training. The main achievement is therefore not only a faster approximation, but a change in how higher-order ice flow can be used: from a repeatedly solved nonlinear problem to a direct and transferable model evaluation.

Code and data availability. The IGM source code is open access and available from the GitHub repository at <https://github.com/instructed-glacier-model/igm.git> (last access: 11 Sept 2026). The Dahunet emulator and its variants, described in this paper, as well as the pretraining
530 module that was used to train the emulators, are included in the IGM repository. IGM's documentation is available from its official website at <https://igm-model.org> (last access: 11 Sept 2026). The full training set is not archived due to its size (>100 GB) but is available from the authors on request.



Appendix A: Emulator architecture and training details

This appendix provides the implementation and optimisation details for the emulator architectures and hybrid training procedure introduced in Sects. 2.3 and 2.4.

A1 Network outputs and input preprocessing

Each emulator returns the four velocity degrees of freedom per grid cell defined in Sect. 2.1: two MOLHO coefficients for each horizontal velocity component. All raw input channels are standardised using statistics estimated from the training data.

The deterministic DahuNet feature fields Ψ_5 in Eq. (9) are calculated from the unstandardised ice thickness, surface elevation, and grid spacing. They comprise the two surface-slope components, the slope magnitude, and two log-scaled geometric velocity proxies proportional to $|\nabla s_H|^3 h_H^4$ and $|\nabla s_H|^3 h_H^3$. The latter are normalised using $\kappa_0 = 0.1$ and $h_0 = 200$ m. These features represent the local driving-slope direction and the leading thickness slope scalings associated with shallow-ice deformation and local sliding for Glen exponent $n = 3$. Their complete formulation is also described by Gregov et al. (2026).

A2 Network architectures

The CNN consists of eight same-padded two-dimensional convolutional layers, each with 48 filters and a 5×5 kernel. It uses leaky-rectified-linear activations, residual additions after every second convolution, a network-level 1×1 skip projection from the input tensor to the hidden channel dimension, and a final linear 1×1 convolution mapping the hidden representation to the $2N_z$ velocity channels.

DahuNet uses eight 5×5 convolutional layers with 48 channels and the same residual cadence. It uses a pointwise dense lift and GELU activations in place of the CNN's LeakyReLU activations. Before this convolutional processing, the five deterministic feature fields Ψ_5 are appended to the standardised glacier-state fields as shown in Eq. (9).

The two architectures have comparable capacity, with approximately 4.1×10^5 trainable parameters for the CNN and 4.6×10^5 for DahuNet. Their key scientific distinction is therefore the physics-enriched input representation used by DahuNet, rather than a large difference in the total number of trainable parameters.

A3 Data and physics loss terms

For a mini-batch $\mathcal{B}$ containing $N_{\mathcal{B}}$ examples, the physics term in Eq. (10) is

$$\mathcal{L}_{\text{phys}} = \sum_{k \in \mathcal{B}} \mathcal{J}_H \left[\hat{\mathbf{U}}_{H,k}; \mathbf{I}_{H,k} \right]. \quad (\text{A1})$$

The predicted velocity field is therefore evaluated using the discretised Blatter-Pattyn energy for its corresponding glacier state.



The supervised velocity term is a Huber loss (Huber, 1964) averaged over the mini-batch, the N_z vertical levels, and the N_c raster cells, and summed over the two horizontal velocity components:

$$\mathcal{L}_{\text{vel}} = \frac{1}{N_B N_z N_c} \sum_{k \in \mathcal{B}} \sum_{\ell=1}^{N_z} \sum_{\mathbf{x} \in \Omega} \left[\rho_\delta(\hat{\mathbf{u}}_{H,k}(\mathbf{x}, \ell) - \mathbf{u}_{H,k}^{\text{ref}}(\mathbf{x}, \ell)) + \rho_\delta(\hat{\mathbf{v}}_{H,k}(\mathbf{x}, \ell) - \mathbf{v}_{H,k}^{\text{ref}}(\mathbf{x}, \ell)) \right], \quad (\text{A2})$$

where ρ_δ is the Huber penalty with transition scale $\delta = 50 \text{ m yr}^{-1}$:

$$\rho_\delta(r) = \begin{cases} \frac{1}{2} r^2, & |r| \leq \delta, \\ \delta (|r| - \frac{1}{2} \delta), & |r| > \delta. \end{cases} \quad (\text{A3})$$

The data-only controls use the same velocity loss, architectures, training data, and optimiser settings as their hybrid counterparts, but set $\lambda_t = 0$ throughout training. The variational physics penalty is therefore the only difference between the data-only and hybrid versions of each architecture.

A4 Adaptive weighting of the physics loss

The two terms in Eq. (10) have different units and numerical scales. We initially attempted to balance the magnitudes of their gradients, but this produced frequent training instability, particularly for the CNN architecture. Instead, we balance the spread of the per-parameter gradient distributions, following the inverse-Dirichlet approach of Maddu et al. (2022), which sets each loss weight from the standard deviation of that term's gradients so that no term dominates the update, and is more robust to the heavy-tailed gradients produced by the variational energy.

Every K_λ optimiser updates, we calculate the gradient of each loss term with respect to the network weights. A target value for the physics weight is then obtained from the ratio of the standard deviations of the two gradient distributions:

$$\hat{\lambda} = \rho \frac{\sigma(\nabla_{\theta} \mathcal{L}_{\text{vel}})}{\sigma(\nabla_{\theta} \mathcal{L}_{\text{phys}}) + \epsilon}, \quad (\text{A4})$$

where $\sigma(\cdot)$ denotes the standard deviation across the network parameters. This scaling holds the spread of the weighted physics gradient at a fraction ρ of the data-gradient spread. We use $\rho = 0.2$, so the physics term remains subordinate to the supervised velocity signal.

To prevent abrupt changes in the training objective, the target value is clamped to the multiplicative interval

$$[\lambda_t / r_{\text{max}}, \lambda_t r_{\text{max}}], \quad (\text{A5})$$

with $r_{\text{max}} = 3$, and then exponentially smoothed according to

$$\lambda_{t+1} = \alpha \lambda_t + (1 - \alpha) \hat{\lambda}, \quad (\text{A6})$$

with $\alpha = 0.9$.



A5 Training schedule and optimisation

585 Training is divided into two phases. During an initial supervised warm-up of 10^5 optimiser updates, λ_t is held at zero and only $\mathcal{L}_{\text{vel}}$ is applied. This allows the network to learn an approximate velocity mapping before the variational penalty is introduced. After the warm-up, the physics term is activated and its weight is re-estimated using Eq. (A4) every $K_\lambda = 200$ updates. The weight remains fixed between successive re-estimations.

590 All four primary models are trained for 800 epochs of 1000 optimiser updates each, using Adam (Kingma and Ba, 2015) with a fixed learning rate of 10^{-4} and a batch size of $N_B = 32$. We did not conduct a hyperparameter search over learning rate, batch size, or network depth. All four models instead use the default IGM settings so that the comparison in Sect. 3.1 is not affected by unequal model-specific tuning.

A6 Sensitivity to training-set size

To assess how emulator accuracy depends on the amount of distinct training data, we trained seven instances of Dahu $_\lambda$ using
 595 subsets ranging from 0.5% to 100% of the available training pool. All runs used the same architecture, learning-rate schedule, supervised warm-up, and total budget of 500 000 gradient updates. The smaller datasets were repeated and shuffled as necessary, so the experiment changes the number of distinct training examples while holding the optimisation effort fixed. All models were evaluated on the same validation set of 27 745 examples, and we report both the mean velocity RMSE and, as a measure of the error tail, the fraction of validation tiles whose per-tile RMSE exceeds 10 m yr^{-1} .

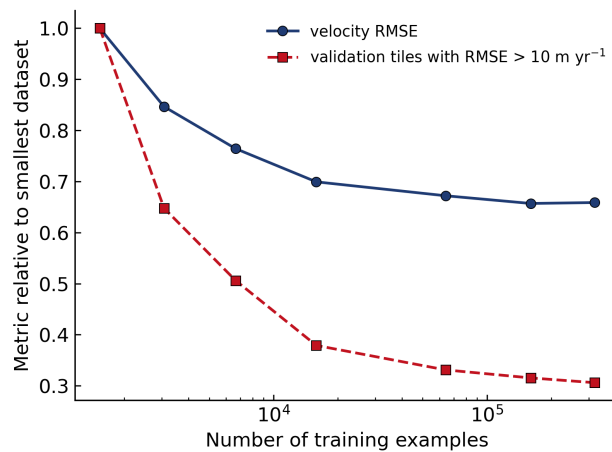


Figure A1. Validation error of the DahuNet ice-flow emulator as a function of training-set size. Each point is a separate model trained on a distinct pool of examples (0.5-100% of the full training set), with all other settings and the total number of optimizer steps held fixed. Both the velocity RMSE and the fraction of validation tiles with per-tile surface-velocity RMSE above 10 m yr^{-1} are shown relative to their value at the smallest (0.5%) training set.



600 Validation velocity RMSE decreases rapidly as the training pool grows from a few thousand to approximately 3×10^4 distinct examples (Fig. A1, blue). Beyond this point the mean error saturates, improving only slightly out to the full pool of 320006 examples, which nonetheless gives the lowest error. The error tail, however, continues to contract. The fraction of validation tiles with per-tile RMSE above 10 m yr^{-1} falls from $\sim 13\%$ at the smallest dataset to $\sim 4\%$ for the full pool (Fig. A1, red), and keeps decreasing in the regime where the mean RMSE has already plateaued. Thus, while average accuracy shows diminishing
 605 returns beyond $\sim 3 \times 10^4$ examples, additional data continues to reduce the frequency of poorly-emulated states within the sampled glacier-state distribution and under the fixed optimisation budget used here.

Appendix B: DahuNet capacity sensitivity

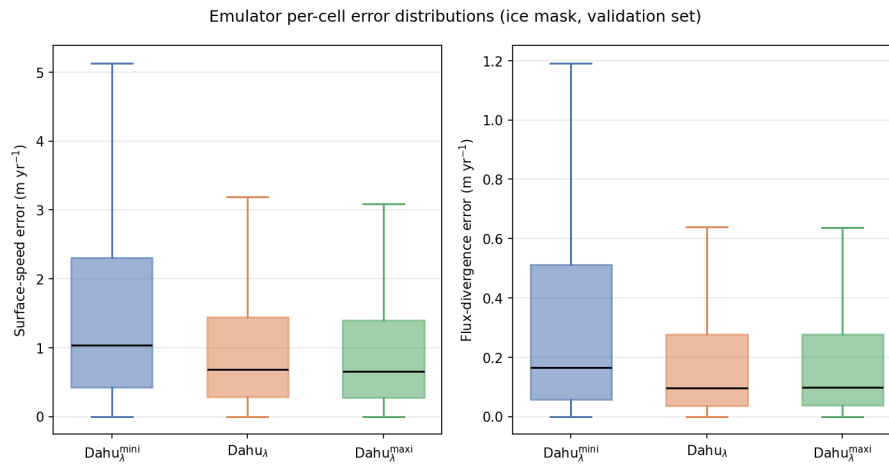


Figure B1. Distribution of emulator (a) Surface-speed error and (b) flux-divergence error, evaluated over the validation set for the three DahuNet variants. Boxes span the interquartile range with the median marked; whiskers extend to the most extreme value within $1.5 \times \text{IQR}$ of the quartiles.

To assess how emulator accuracy depends on network size, we train two additional DahuNet variants: a smaller Dahu_λ^{mini} model and a larger Dahu_λ^{maxi} model. These variants use the same DahuNet input representation and general convolutional
 610 design as the default Dahu_λ emulator, but change the number of convolutional layers and filters per layer (Table B1). This provides a simple sensitivity test of whether increasing model capacity substantially improves accuracy, and whether a much smaller network can retain acceptable performance.

We evaluate the two size variants using the same validation-set diagnostics as in Sect. 3.1. The results are shown in Fig. B1. Increasing the parameter count by a factor of ~ 3 from Dahu_λ to Dahu_λ^{maxi} produces little or no improvement in validation
 615 accuracy. In contrast, the smaller Dahu_λ^{mini} model shows clearly degraded performance relative to the default emulator, particularly in the upper tails of the error distributions. These results suggest that the default Dahu_λ model is a good middle ground



Table B1. Architecture size and inference cost of the three pretrained DahuNet emulators. All three use the same DahuNet feature representation and convolutional design, with 5×5 residual convolutions and $N_z = 2$ MOLHO/Q1 velocity bases, but differ in depth and width. GPU memory is the peak allocation for a single forward pass on a 256×256 grid, excluding fixed runtime overhead.

Model	Depth (conv layers)	Filters (per layer)	Parameters	GPU mem. (inference)
Dahu $_{\lambda}^{\text{maxi}}$	12	64	1,234,703	125 MiB
Dahu $_{\lambda}$	8	48	464,271	93 MiB
Dahu $_{\lambda}^{\text{mini}}$	6	32	155,343	73 MiB

between accuracy and inference cost. They also indicate that further improvements are unlikely to come simply from increasing network size, and may instead require changes to the training data, training objective, or network architecture.

Appendix C: Local perturbation response

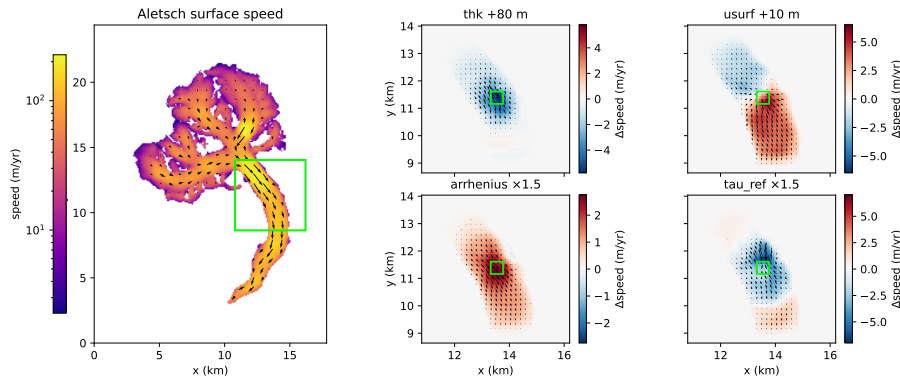


Figure C1. (a) Baseline modelled surface speed of Aletsch glacier with flow-direction quivers; the green box marks the region shown in (b). (b) Change in surface speed (red = speed-up, blue = slow-down) at the trunk pixel (green square) for an independent perturbation of each input field: thickness (+80 m), surface elevation (+10 m), Arrhenius factor ($\times 1.5$), and basal-traction scale τ_{ref} ($\times 1.5$). Quivers show the induced velocity-change vectors.

620 The validation and transient experiments assess how accurately the emulator reproduces velocity fields, but not whether it responds correctly to changes in its inputs. This distinction matters for two reasons. First, the bounded transient behaviour reported in Sect. 3.2 depends on the emulator reproducing the sensitivity of the solver to geometry, not only its output. Second, IGM's inversion and data-assimilation workflows differentiate through the ice-flow module, so an emulator with an accurate forward map but a poorly behaved derivative would be unsuitable for those applications. As a qualitative sanity check, we



625 therefore perturb a small patch in the main trunk of Aletsch glacier and evaluate the resulting change in modelled surface speed
 with all other inputs held fixed (Fig. C1).

A thickness increase of +80 m and an increase in basal traction scale $\tau_{\text{ref}} \times 1.5$ both reduce the local surface speed, while
 a softer ice rheology, represented by an Arrhenius factor increase of $\times 1.5$, increases speed. The surface-elevation perturbation
 of +10 m produces a speed-up/slow-down dipole aligned with the local flow direction, as expected for an input that affects the
 630 emulator mainly through the surface-slope field.

The responses are smooth, spatially coherent, and decay away from the perturbed patch, with no obvious checkerboard
 or discontinuous artefacts. This suggests that the emulator has learned a locally well-behaved mapping. The experiment is
 qualitative, however, and should be interpreted as a sanity check rather than a complete sensitivity analysis.

Appendix D: Setup of the transient validation domains

635 D1 Domains and input data

The transient experiments of Sect. 3.2 use two real-glacier systems that were deliberately chosen to contrast in size, geometry
 and climatic regime, and neither of which appears in the synthetic training set: the Columbia Icefield, a large temperate plateau
 icefield in the Canadian Rockies, and Hofsjökull, a smaller Icelandic subpolar ice cap. Ice thickness is the Farinotti consensus
 estimate (Farinotti et al., 2019), surface elevation is the Copernicus GLO-30 DEM (European Space Agency, 2023), and the
 640 glacier outlines that define the merge are the same RGI v6.2 polygons (RGI Consortium, 2017). The fields are mapped to a
 grid with 100 m horizontal resolution. Bed elevation is obtained internally as $b_H = s_H - h_H$ from the supplied surface and
 thickness.

Before the 500-year experiment, each input state is relaxed for 10 years under a fixed steady-state climate, using the identity
 mapping $\mathcal{M}^{\text{id}}$ with the velocity degrees of freedom optimised by L-BFGS, i.e. exactly the configuration of the reference
 645 run. The relaxation climate is the $t = 0$ surface-mass-balance state of the subsequent experiment (simple ELA-gradient SMB
 with ELA = 2500 m for Columbia and 1300 m for Hofsjökull). The single relaxed geometry is then used to initialise both the
 reference ($\mathcal{M}^{\text{id}}$) and the emulator-driven ($\mathcal{M}^{\text{net}}$) trajectories, so the two runs differ only in the velocity mapping. The advance-
 retreat cycle is imposed purely through lowering and then raising the ELA, with the same balance-gradient parameters as those
 used elsewhere in the study.

650 States are saved every 10 years. The surface-speed and thickness RMSEs in Fig. 5 are evaluated over the union of the
 reference and emulator ice masks at each saved time, whereas the signed mean-speed diagnostic V in the error decomposition
 (Fig. 6) is averaged over their intersection (cells with $h_H > 1$ m in both states). Integrated ice volume is computed for each
 trajectory over its own ice mask, summed across the full model domain. Finally, because the *pinnbp* emulator differs from
 Dahu $_{\lambda}$ in a number of important respects, it is compared only against a separate $\mathcal{M}^{\text{id}}$ reference integrated at its own native
 655 discretisation.



Author contributions. S.H.R.R. conceived the study, designed the synthetic data-generation strategy and the hybrid training objective, ran all experiments and wrote the original draft. T.G. and G.J. developed the physics-enriched DahuNet architecture in the companion study. T.G. and G.J. contributed to the development of the methodology through regular discussion throughout the project. G.J. developed the original version of IGM. G.J. and A.V. secured funding. All authors reviewed and edited the manuscript.

660 *Competing interests.* The authors declare that they have no conflict of interest.

Acknowledgements. This research was supported by the Swiss National Science Foundation (SNSF) through the project 'PINNACLE: Developing the next generation of ice flow models to predict the future evolution of glaciers' under grant number 10002401. We are extremely grateful to Ludovic Räss for his tireless work maintaining the cluster that made this work possible. This work also benefited from discussions with Brandon Finley, Andreas Henz and Tancrède Leger.



665 References

- Blatter, H.: Velocity and stress fields in grounded glaciers: a simple algorithm for including deviatoric stress gradients, *Journal of Glaciology*, 41, 333–344, <https://doi.org/10.3189/S002214300001621X>, 1995.
- Cook, S. J., Juvet, G., Millan, R., Rabatel, A., Zekollari, H., and Dussaillant, I.: Committed Ice Loss in the European Alps Until 2050 Using a Deep-Learning-Aided 3D Ice-Flow Model With Data Assimilation, *Geophysical Research Letters*, 50, e2023GL105 029, <https://doi.org/https://doi.org/10.1029/2023GL105029>, e2023GL105029 2023GL105029, 2023.
- 670 Dias dos Santos, T., Morlighem, M., and Brinkerhoff, D.: A new vertically integrated MOno-Layer Higher-Order (MOLHO) ice flow model, *The Cryosphere*, 16, 179–195, <https://doi.org/10.5194/tc-16-179-2022>, 2022.
- E, W. and Yu, B.: The Deep Ritz Method: A Deep Learning-Based Numerical Algorithm for Solving Variational Problems, *Communications in Mathematics and Statistics*, 6, 1–12, <https://doi.org/10.1007/s40304-018-0127-z>, 2018.
- 675 European Space Agency: Copernicus Digital Elevation Model, <https://doi.org/10.5270/ESA-c5d3d65>, copernicus DEM GLO-90 product, 2023.
- Farinotti, D., Huss, M., Fürst, J. J., Landmann, J., Machguth, H., Maussion, F., and Pandit, A.: A consensus estimate for the ice thickness distribution of all glaciers on Earth, *Nature Geoscience*, 12, 168–173, <https://doi.org/10.1038/s41561-019-0300-3>, 2019.
- Glen, J. W.: The creep of polycrystalline ice, *Proceedings of the Royal Society of London. Series A*, 228, 519–538, <https://doi.org/10.1098/rspa.1955.0066>, 1955.
- 680 Gregov, T., Rosier, S., Finley, B., Vieli, A., and Juvet, G.: Physics-enriched neural solvers for transient ice-flow simulation, <https://doi.org/https://doi.org/10.48550/arXiv.2609.12900>, 2026.
- Hasselmann, K.: Stochastic climate models. Part I. Theory, *Tellus*, 28, 473–485, <https://doi.org/10.3402/tellusa.v28i6.11316>, 1976.
- Hock, R.: Temperature index melt modelling in mountain areas, *Journal of Hydrology*, 282, 104–115, [https://doi.org/10.1016/S0022-](https://doi.org/10.1016/S0022-1694(03)00257-9)
- 685 1694(03)00257-9, 2003.
- Huber, P. J.: Robust Estimation of a Location Parameter, *The Annals of Mathematical Statistics*, 35, 73–101, <https://doi.org/10.1214/aoms/1177703732>, 1964.
- Juvet, G. and Cordonnier, G.: Ice-flow model emulator based on physics-informed deep learning, *Journal of Glaciology*, 69, 1941–1955, <https://doi.org/10.1017/jog.2023.73>, 2023.
- 690 Juvet, G., Cordonnier, G., Kim, B., Lüthi, M., Vieli, A., and Aschwanden, A.: Deep learning speeds up ice flow modelling by several orders of magnitude, *Journal of Glaciology*, 68, 651–664, <https://doi.org/10.1017/jog.2021.120>, 2022.
- Juvet, G., Finley, B., Gregov, T., et al.: IGM: an accessible, modular, differentiable, and GPU-accelerated high-order ice flow model, <https://eartharxiv.org/repository/view/13712/>, preprint, 2026.
- Kingma, D. P. and Ba, J.: Adam: A Method for Stochastic Optimization, in: *International Conference on Learning Representations*, <https://arxiv.org/abs/1412.6980>, 2015.
- 695 //arxiv.org/abs/1412.6980, 2015.
- Maddu, S., Sturm, D., Müller, C. L., and Sbalzarini, I. F.: Inverse Dirichlet weighting enables reliable training of physics informed neural networks, *Machine Learning: Science and Technology*, 3, 015 026, <https://doi.org/10.1088/2632-2153/ac3712>, 2022.
- Maussion, F., Butenko, A., Champollion, N., Dusch, M., Eis, J., Fourteau, K., Gregor, P., Jarosch, A. H., Landmann, J., Oesterle, F., Recinos, B., Rothenpieler, T., Vlug, A., Wild, C. T., and Marzeion, B.: The Open Global Glacier Model (OGGM) v1.1, *Geoscientific Model Development*, 12, 909–931, <https://doi.org/10.5194/gmd-12-909-2019>, 2019.
- 700



- Millan, R., Mouginot, J., Rabatel, A., and Morlighem, M.: Ice velocity and thickness of the world's glaciers, *Nature Geoscience*, 15, 124–129, <https://doi.org/10.1038/s41561-021-00885-z>, 2022.
- Parthipan, R., Anand, M., Christensen, H. M., Hosking, J. S., and Wischik, D. J.: Defining error accumulation in ML atmospheric simulators, *arXiv preprint arXiv:2405.14714*, <https://doi.org/10.48550/arXiv.2405.14714>, 2024.
- 705 Pattyn, F.: A new three-dimensional higher-order thermomechanical ice sheet model: basic sensitivity, ice stream development, and ice flow across subglacial lakes, *Journal of Geophysical Research: Solid Earth*, 108, 2382, <https://doi.org/10.1029/2002JB002329>, 2003.
- Rasp, S., Düben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thürey, N.: WeatherBench: A Benchmark Data Set for Data-Driven Weather Forecasting, *Journal of Advances in Modeling Earth Systems*, 12, e2020MS002203, <https://doi.org/10.1029/2020MS002203>, 2020.
- RGI Consortium: Randolph Glacier Inventory (RGI) – A Dataset of Global Glacier Outlines: Version 6.0, Technical report, Global Land Ice
- 710 Measurements from Space, Boulder, Colorado, USA, <https://doi.org/10.7265/N5-RGI-60>, 2017.
- Roe, G. H. and Baker, M. B.: The response of glaciers to climatic persistence, *Journal of Glaciology*, 62, 240–254, <https://doi.org/10.1017/jog.2016.4>, 2016.
- Rounce, D. R., Hock, R., Maussion, F., Hugonnet, R., Kochtitzky, W., Huss, M., Berthier, E., Brinkerhoff, D. J., Compagno, L., Copland, L., Farinotti, D., Menounos, B., and McNabb, R. W.: Global glacier change in the 21st century: Every increase in temperature matters,
- 715 *Science*, 379, 78–83, <https://doi.org/10.1126/science.abo1324>, 2023.
- Steinemann, S.: Experimentelle Untersuchungen zur Plastizität von Eis, Ph.D. thesis, ETH Zürich, Zürich, Switzerland, <https://doi.org/10.3929/ethz-a-000096707>, *beiträge zur Geologie der Schweiz, Geotechnische Serie, Hydrologie, Lieferung 10*, 1–72; Diss. ETH Nr. 2772, 1958.
- Wang, X., Han, Y., Xue, W., Yang, G., and Zhang, G. J.: Stable climate simulations using a realistic general circulation model with neural
- 720 network parameterizations for atmospheric moist physics and radiation processes, *Geoscientific Model Development*, 15, 3923–3940, <https://doi.org/10.5194/gmd-15-3923-2022>, 2022.
- Weertman, J.: On the Sliding of Glaciers, *Journal of Glaciology*, 3, 33–38, <https://doi.org/10.3189/S0022143000024709>, 1957.
- Zekollari, H., Huss, M., and Farinotti, D.: Modelling the future evolution of glaciers in the European Alps under the EURO-CORDEX RCM ensemble, *The Cryosphere*, 13, 1125–1146, <https://doi.org/10.5194/tc-13-1125-2019>, 2019.
- 725 Zekollari, H., Huss, M., Farinotti, D., and Lhermitte, S.: Ice-Dynamical Glacier Evolution Modeling—A Review, *Reviews of Geophysics*, 60, e2021RG000754, <https://doi.org/https://doi.org/10.1029/2021RG000754>, e2021RG000754 2021RG000754, 2022.
- Zemp, M., Hoelzle, M., and Haeberli, W.: Distributed modelling of the regional climatic equilibrium line altitude of glaciers in the European Alps, *Global and Planetary Change*, 56, 83–100, <https://doi.org/10.1016/j.gloplacha.2006.07.002>, 2007.