MESH-NATIVE TOKENISATION: RETROFITTING FOUNDATION MODELS TO UNSTRUCTURED DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Physics foundation models are an increasingly popular approach for building versatile emulators for spatiotemporal systems. They require tokenisation layers that lift physical field data into a latent representation suitable for dynamic processing. A typical choice for this layer is a convolutional encoder, which lifts field information in a locality-aware way and can be trained very efficiently. However, this choice cannot be applied to a large class of scientific computing problems where data are represented on unstructured meshes, for example simulations on domains with curved boundaries. Such data typically require graph neural network architectures, which are more expensive to train and, in the form currently used, encode information differently from their CNN counterparts, thus limiting the portability of physics emulators between these data regimes. In this work we introduce Mesh-Native Convolution (MNC), a locality-aware tokenisation layer that encodes unstructured mesh-based data in a way that is analogous to CNNs. This approach leads to models that can be trained once on mesh data and then deployed to arbitrary unstructured meshes, even if these are irregular, with minimal retraining. We show that this tokenisation approach can be used to retrofit grid-pretrained foundation models efficiently on unstructured meshes and provide theoretical results to confirm the consistency of this approach with classical CNN encoders. We demonstrate the favourable performance of this retrofitting approach, which in many cases reaches acceptable accuracy even in zero-shot deployment, and outperforms existing baselines after only a few epochs of fine-tuning, thus providing an efficient bridge between the two data regimes.

1 INTRODUCTION

Machine-learning surrogates are increasingly used to predict the evolution of complex physical systems across a wide range of scientific domains Wyder et al. (2026); Yermakov et al. (2026). Many recent large pretrained models for continuum physics adopt an encoder-processor-decoder architecture, in which discretised fields are encoded into latent tokens, evolved by a large processor, and decoded back to physical space (McCabe et al., 2024; Herde et al., 2024; McCabe et al., 2026; Mukhopadhyay et al., 2026). Many of the most advanced models build on processors that originate in computer vision, thus relying on tokenisation layers, which lift physical information, often using strided convolutions (McCabe et al., 2024; Mukhopadhyay et al., 2026). This has proven to be a particularly efficient choice, as dense convolutions on regular grid data are highly optimised on modern accelerators (Chetlur et al., 2014), but it restricts the admissible data to structured grids (cf. Fig. 1a). However, in many scientific applications field information is not available on regular grids due to the presence of irregular or curved boundaries (engineering parts, wave modelling in coastal regions, weather forecasting) or the requirement to represent information at finer resolution in regions of the computational domain that show finer features (e.g. boundary layers in fluids, or turbulent structures behind jet engines). In such cases data are often represented on unstructured meshes (cf. Fig. 1b).

While graph neural networks (GNNs) are a natural choice for processing such data, they are more expensive to train and, in the form currently used, encode information differently from their CNN counterparts, thus limiting the portability of physics emulators between these data regimes (Pfaff et al., 2021). A convolutional token has an explicit physical footprint, which mirrors the dynamics it summarises: the differential operators of continuum models are local and information travels a bounded physical distance per time step, so a token should describe a controlled physical

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

neighbourhood. Standard mesh-edge message passing instead defines neighbourhoods by graph hops, whose physical extent varies with the local edge length (Gilmer et al., 2017; Pfaff et al., 2021), and supplying coordinates as features does not remove this dependence directly. To overcome this, we introduce Mesh-Native Convolution (MNC), a locality-aware tokenisation layer that encodes unstructured mesh-based data analogously to CNNs. MNC specifies the window geometrically and aggregates information through a continuous analogue of a convolutional kernel (Fig. 2). This leads to a highly versatile tokenisation layer that can be used to retrofit grid-pretrained foundation models efficiently on unstructured meshes. We show that this approach is consistent with classical CNN encoders, thus permitting the direct transfer of weights from grid-pretrained models, and demonstrate the mesh-agnostic performance of this approach in zero-shot deployment. In particular, **our contributions are as follows:**

1. **A mesh-native convolutional tokeniser.** We introduce mesh-native convolution (MNC), a tokenisation layer for unstructured mesh-based data. This encoder is able to lift continuous physical information from unstructured data points, removing the restriction to grid-based data.
2. **Exact consistency with CNN encoders.** We prove that MNC is consistent with classical CNN encoders, thus permitting the direct loading of weights from grid-pretrained models.
3. **Retrofitting pretrained foundation models to meshes.** We show through extensive numerical examples that retrofitting state-of-the-art physics emulators and foundation models with MNC reaches acceptable accuracy zero-shot (cf. Table 1) and outperforms existing baselines after only a few epochs of fine-tuning, thus providing an efficient bridge between the two data regimes.

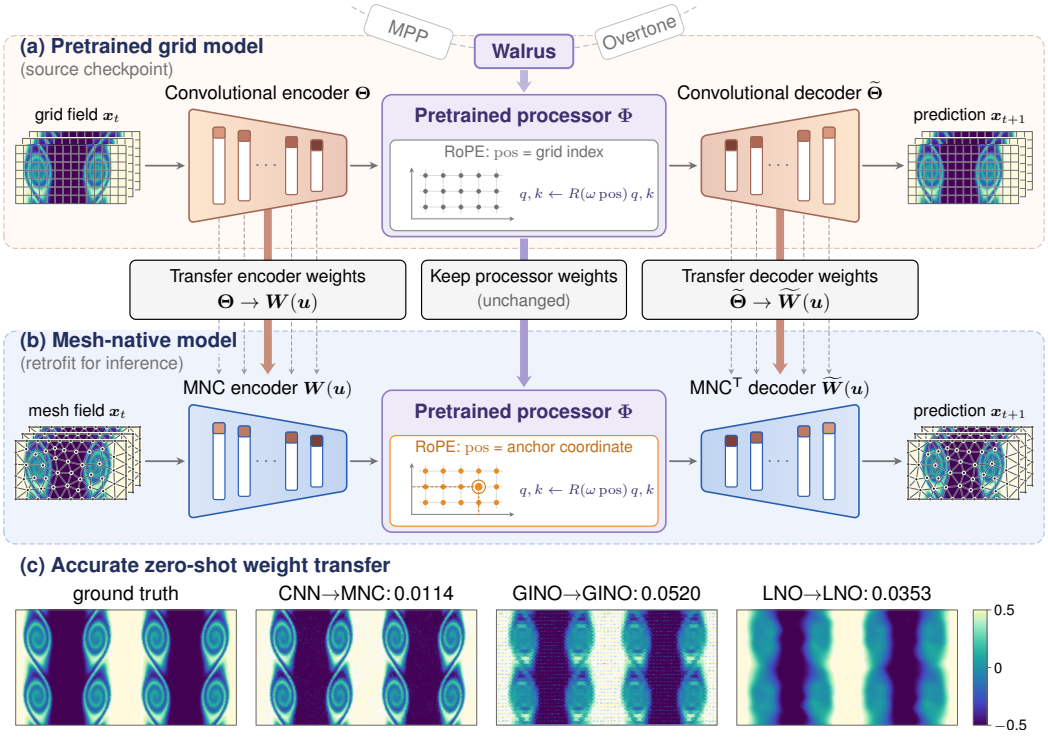


Figure 1: **(a)** MNC loads grid-pretrained weights directly onto a mesh-compatible GNN architecture, for consistent retrofitting of pretrained physics emulators. **(b)** Zero-shot shear-flow single-step prediction after grid pretraining, including MNC (ours), GINO (Li et al., 2023c) and LNO (Wang & Wang, 2024). Values are normalised MSE for the displayed trajectories.

1.1 Background and Related Work

Physics foundation models and retrofitting. The most powerful physics foundation models, such as MPP, Poseidon and Walrus, pretrain transformer surrogates across physical systems, all with convolutional patch embeddings on uniform grids (McCabe et al., 2024; Herde et al., 2024; McCabe et al., 2026). Their architectures often build on Vision Transformers (ViTs) (Dosovitskiy et al.,

2021). The cost of retraining is prohibitive, and recent work by Diaconu et al. (2026) has provided initial steps towards retrofitting pretrained learned simulators post hoc with probabilistic predictions. Our approach is to retrofit the model to a new, more versatile data regime.

Physics models on unstructured meshes. The need to simulate physical systems on unstructured meshes is ubiquitous in scientific computing (Brenner & Scott, 2008; Rowbottom et al., 2025). MeshGraphNets (Pfaff et al., 2021) and related GNN simulators operate natively on meshes but encode information differently from ViT-based foundation models. Operator-learning approaches handle general geometries via coordinate transforms (Li et al., 2023b), attention over point sets (Li et al., 2023a; Wu et al., 2024; Luo et al., 2025), neural fields (Serrano et al., 2023), continuous convolutions (Hagnberger et al., 2025) or latent point-cloud encoders (Alkin et al., 2024), as do GINO (Li et al., 2023c) and LNO (Wang & Wang, 2024). Our approach is fundamentally different: we seek to encode information similarly to classical CNN architectures, exploiting physical locality as a guiding principle and thus allowing for a simple transition between grid and unstructured data regimes.

Continuous convolutions and discretisation invariance. Several ingredients underlying MNC have appeared separately in prior work. SplineCNN parametrises geometric convolutions using continuous B-spline kernels (Fey et al., 2018), while discretisation-invariant networks show how pretrained discrete convolutional filters can be interpolated to continuous kernels and recovered exactly on their native grid (Wang & Golland, 2023). MNC combines these ideas into a mesh-native, multilevel encoder–decoder architecture designed specifically to retrofit pretrained grid-based physics surrogates, including their positional interface. Its operator interpretation follows the numerical-discretisation viewpoint of Boullé & Townsend (2024); Kovachki et al. (2023): convolution is treated as the discretisation of a local integral operator, with MNC's occupancy normalisation correcting for the non-uniform sampling induced by irregular meshes. Through this, our work provides the central insight that connects this encoder layer with physical consistency.

2 Mesh-native tokenisation and weight transfer

We begin by introducing mesh-native convolution, the central ingredient of our retrofitting approach.

Setup. We assume the physical field is sampled at positions $\mathbf{p}_j \in \Omega \subset \mathbb{R}^2$, with features $\mathbf{x}_j \in \mathbb{R}^{F_{\text{in}}}$. On a grid of spacing (h_x, h_y), a standard convolutional layer with kernel size $k_h \times k_w$ has offsets

$$\mathcal{O} = \{-L_x, -L_x + h_x, \ldots, L_x\} \times \{-L_y, -L_y + h_y, \ldots, L_y\}, \qquad L_x = \tfrac{k_w - 1}{2} h_x,\ L_y = \tfrac{k_h - 1}{2} h_y.$$

Writing $\mathbf{p}_{\hat{v}}$ for the window centre, a strided convolution can be written as

$$\mathbf{x}'_{\hat{v}} = \sum_{\mathbf{u} \in \mathcal{O}} \Theta[\mathbf{u}]\, \mathbf{x}_{j(\mathbf{p}_{\hat{v}} - \mathbf{u})} + \mathbf{b}, \tag{1}$$

where $\Theta[\mathbf{u}] \in \mathbb{R}^{F_{\text{out}} \times F_{\text{in}}}$ $j(\mathbf{p})$ is the node at $\mathbf{p}$, and $P = k_h k_w$. MNC follows this form closely.

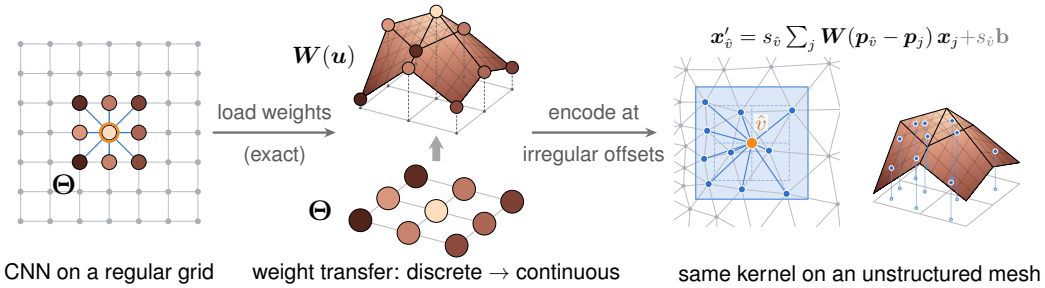


Figure 2: Schematic overview of the MNC layer. MNC loads a pretrained CNN's taps into a continuous spline and evaluates the same kernel at physical offsets on an unstructured mesh.

Continuous spline kernel. Let $\phi(t) = (1 - |t|)_+$. Then we can define a continuous map

$$\mathbf{W}(\mathbf{u}) = \sum_{\mathbf{u}' \in \mathcal{O}^+} \mathbf{K}[\mathbf{u}']\, \phi\left(\frac{u_x - u'_x}{h_x}\right) \phi\left(\frac{u_y - u'_y}{h_y}\right), \tag{2}$$

which provides a continuous extension of the discrete kernel $\mathbf{K}$ (cf. Fig. 2). This continuous kernel forms the basis of MNC's gather and scatter operations, which are used to encode and decode information on unstructured meshes. In particular, the kernel weights $\mathbf{K}$ will become learnable parameters of the MNC layer, which can also be initialised from pretrained CNN weights. To preserve physical weighting we normalise the MNC accumulation using the occupancy profile

$$\omega(\mathbf{u}) = \sum_{\mathbf{u}' \in \mathcal{O}} \phi\left(\frac{u_x - u'_x}{h_x}\right) \phi\left(\frac{u_y - u'_y}{h_y}\right), \tag{3}$$

which takes values in $[0, 1]$ and measures whether a node falls within the (graded) support of $\mathbf{W}$.

Mesh hierarchy. Much like traditional convolutional layers, we introduce the action of a single MNC layer as a "gathering" of information in the sense that it aggregates information from neighbouring physical locations weighted by distance, enforcing locality by virtue of the finite support of the kernel $\mathbf{K}$. For this we need to assign a physical footprint to each gathering operation. This can be seen as the analogue of the position of a single convolutional window on a grid, and results in a natural definition of a mesh hierarchy of virtual anchors (cf. Fig. 3). Stacking several such interconnected meshes forms the encoder block. Analogously to transposed convolution, the action of the decoder is to distribute information across nodes and, as such, the decoder traverses the levels in reverse.

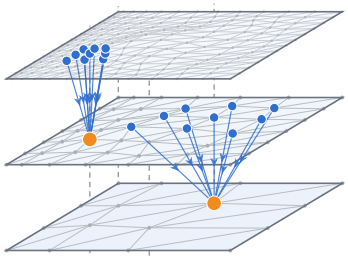


Figure 3: The mesh hierarchy, finest level at the top. Encoder steps gather onto virtual anchors; decoder steps reuse the same edges in reverse.

The MNC layer and its adjoint. We are now in a position to define the functional form of a single MNC layer, which joins each anchor $\hat{v}$ to all nodes in its geometric support (cf. Fig. 3) and computes

$$\mathbf{x}'_{\hat{v}} = s_{\hat{v}} \left[\sum_{j \in \mathcal{N}(\hat{v})} \mathbf{W}(\mathbf{p}_{\hat{v}} - \mathbf{p}_j)\mathbf{x}_j + \mathbf{b} \right], \qquad s_{\hat{v}} = \frac{P}{\max\{\sum_{j \in \mathcal{N}(\hat{v})} \omega(\mathbf{p}_{\hat{v}} - \mathbf{p}_j), 1\}}, \tag{4}$$

where $P = k_h k_w$. The scaling factor $s_{\hat{v}}$ ensures that the total occupancy of the window is correctly normalised. The decoder reverses the same edges and uses its own kernel $\widetilde{\mathbf{W}}$:

$$\mathbf{x}'_j = t_j \left[\sum_{\hat{s}: j \in \mathcal{N}(\hat{s})} \widetilde{\mathbf{W}}(\mathbf{p}_j - \mathbf{p}_{\hat{s}})\mathbf{x}_{\hat{s}} + \widetilde{\mathbf{b}} \right], \quad t_j = \frac{1}{\max\{\sum_{\hat{s}: j \in \mathcal{N}(\hat{s})} \omega(\mathbf{p}_j - \mathbf{p}_{\hat{s}}), 1\}}. \tag{5}$$

This construction is a continuous analogue of a convolutional layer, and we now show that it is consistent with the discrete convolutional layer (1). The proof is provided in Appendix A.1.

Proposition 1 (Exact grid reduction)**.** *Suppose the input nodes $\{\mathbf{p}_j\}$ are exactly the points of the source grid of spacing (h_x, h_y), the tap spacings equal the grid spacings, and the anchors and padding reproduce the output windows of a unit-dilation* `Conv2d`*. Load $\mathbf{K}[\mathbf{u}'] = \Theta[\mathbf{u}']$ for $\mathbf{u}' \in \mathcal{O}$ and realise padding as zero-valued nodes with their occupancy included. Then* (4) *equals* (1) *at every anchor node. Similarly, if $\widetilde{\mathbf{K}}$ is loaded from the kernel weights of a* `ConvTranspose2d` *with unit dilation, stride equal to its kernel size and zero output padding, then* (5) *equals that* `ConvTranspose2d` *at every grid node.*

Preserving the processor interface. For transformer-based processors, the tokenisation layer must also provide a positional embedding. In MNC, the source processor encodes position with axial rotary position embedding (RoPE) (Su et al.): on an $n_x \times n_y$ token grid, every query and key channel pair r is rotated by the angle $\omega_{a,r}\,\mathrm{pos}_a$, where $\omega_{a,r}$ are fixed per-axis frequencies and $\mathrm{pos}_a = \mathrm{linspace}(-1, 1, n_a)$ is the token's normalised index along axis a. The MNC encoder returns a flat list of anchors rather than a grid, so we evaluate the same embedding at the anchor coordinates,

$$\mathrm{pos}_a(\mathbf{p}) = \frac{2\,(p_a - p_a^{\min})}{p_a^{\max} - p_a^{\min}} - 1, \qquad R_{a,r}(\mathbf{p}) = \begin{pmatrix} \cos(\omega_{a,r}\mathrm{pos}_a) & -\sin(\omega_{a,r}\mathrm{pos}_a) \\ \sin(\omega_{a,r}\mathrm{pos}_a) & \cos(\omega_{a,r}\mathrm{pos}_a) \end{pmatrix}, \tag{6}$$

with $p_a^{\min}, p_a^{\max}$ the outermost anchor coordinates and the source frequencies, channel pairing and temporal embedding unchanged. On a regular grid this construction reduces to standard RoPE, and

the processor sees the same angles for the same physical locations. As a result, this spatial encoding lends itself naturally to the retrofitting of pretrained models to unstructured meshes, and we will demonstrate this favourable property in §4. Figure 8 in Appendix A.1 illustrates the coordinate form, why the stock axial embedding cannot be applied to a flat list of anchors, and where the change enters the processor.

3 A FINITE-WINDOW OPERATOR INTERPRETATION FOR MNC

A key feature of MNC is its ability to handle unstructured data. It turns out that, beyond mere compatibility with pretrained CNN kernels, MNC is also invariant under the choice of discretisation of a given field, at least in the limit of fine sampling. This provides additional justification for the use of MNC as a physics encoder architecture, as it is consistent with the underlying continuous operator that the CNN kernel approximates. In this section we formalise this property.

Setup. Let the domain, Ω, be bounded. We fix the spline support $S = [-R_x, R_x] \times [-R_y, R_y]$ and refine the kernel spacing so that $h_x, h_y \to 0$. Let $a_h = h_x h_y$, $P_h = k_h k_w$, and choose

$$\mathbf{K}_h[\mathbf{u}'] = a_h \kappa(\mathbf{u}'), \qquad (\mathcal{K}u)(\mathbf{p}) = \int_\Omega \kappa(\mathbf{p} - y) u(y)\, dy, \tag{7}$$

where $\kappa \in W^{1,\infty}(\mathbb{R}^2; \mathbb{R}^{F_{\text{out}} \times F_{\text{in}}})$ is a Lipschitz kernel with $\operatorname{supp} \kappa \subseteq S$ and $u \in C(\overline{\Omega}; \mathbb{R}^{F_{\text{in}}})$.

Proposition 2 (Continuum limit and discretisation invariance)**.** *Suppose the mesh nodes satisfy $c_n \sum_j \delta_{\mathbf{p}_j^{(n)}} \rightharpoonup \rho(y)\, dy$, with $c_n \downarrow 0$ and $\rho \in L^1(\Omega)$ positive almost everywhere. For each fixed anchor whose full window lies inside Ω, MNC applied to $\mathbf{x}_j = u(\mathbf{p}_j^{(n)})$ has the joint limit*

$$\mathrm{MNC}_{h,n}\, u(\mathbf{p}) \longrightarrow |S| \, \frac{\int_\Omega \kappa(\mathbf{p} - y) u(y) \rho(y)\, dy}{\int_{\mathbf{p}-S} \rho(y)\, dy}. \tag{8}$$

Thus two equidistributed refinements of the same field, with the same limiting kernel, have asymptotically identical outputs at common anchors (Fig. 4a).

The proof of this result is included in Appendix A.2. Essentially, this means that, for sufficiently fine meshes, the output of MNC is independent of the specific mesh used, as long as the mesh is equidistributed and the kernel is appropriately defined. This means that MNC truly provides a physically consistent representation of the underlying continuous process. This is a specific property of our MNC construction and does not hold for arbitrary GNN architectures: in particular, for Graph Convolutional Networks (GCNs) (Kipf & Welling, 2017) and their variants, one can easily show that the output of the GCN depends on the local connectivity of the mesh, and thus on the specific triangulation used. This is a key difference between MNC and standard GNNs, and highlights the importance of our construction for retrofitting pretrained models to unstructured meshes.

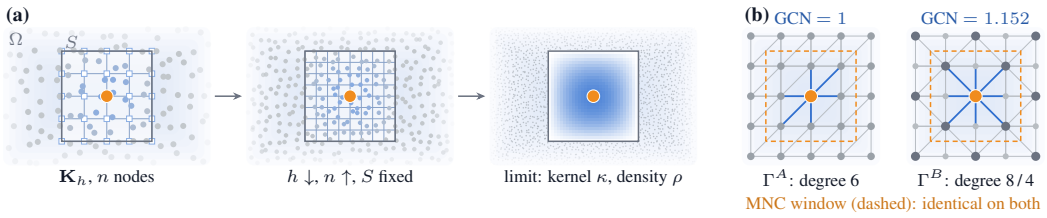


Figure 4: Discretisation invariance of MNC. **(a)** Refining kernel spacing and mesh together (Proposition 2). In the limit MNC corresponds to the continuous kernel κ and the mesh enters only through its sampling density ρ. **(b)** Two triangulations of the same vertices. The GCN neighbourhood of the origin is shown in blue. On the constant field the degree-normalised GCN returns 1 and 1.152 at the origin (9), whereas the MNC window (dashed) yields identical outputs.

A mesh-edge GNN need not share this property. Consider the GCN (Kipf & Welling, 2017) with symmetric degree normalisation, $G_\Gamma(\mathbf{x}) = \mathrm{ReLU}(\widetilde{D}^{-1/2}(A + I)\widetilde{D}^{-1/2}\mathbf{x})$, with scalar weight one.

There are two shape-regular triangulations $\Gamma_\eta^A, \Gamma_\eta^B$ of the *same* square-grid vertices of mesh width η for which the constant field $u \equiv 1$ produces, at their common origin,

$$G_{\Gamma_\eta^A}(\mathbf{1})_0 = 1, \qquad G_{\Gamma_\eta^B}(\mathbf{1})_0 = \frac{5}{9} + \frac{4}{3\sqrt{5}} = 1.15184\ldots \tag{9}$$

for every $\eta = 1/n$ with $n \geq 3$: local degrees are six in one triangulation and eight/four in the other (see Appendix A.3). MNC agrees exactly on this pair (Fig. 4b). Symmetric normalisation does not preserve constants on irregular graphs. Mean aggregation $\widetilde{D}^{-1}(A + I)$ would return 1 on both meshes, but any fixed number of hops has a footprint of size $O(\eta)$, so such a layer tends to the pointwise value $u(\mathbf{p})$ rather than to a finite-window operator.

4 Experiments

We now test MNC in retrofitting experiments of physics emulators for dynamical problems against four mesh-native architectures that span the state of the art in encoder families for unstructured data.

Setup. For this evaluation we use three datasets from The Well (Ohana et al., 2024), the two-dimensional turbulent radiative layer (TRL2D, 128×384), shear flow (256×512) and active matter, and the MeshGraphNets cylinder-flow benchmark (Pfaff et al., 2021), whose trajectories are simulated on per-trajectory unstructured meshes of about 1.9k nodes around an obstacle and have no grid representation. For the Well datasets the grid is the pretraining domain and the target is an irregular Delaunay triangulation of the same domain at matched resolution. Each dataset comes as a sequence of time-trajectories and trained models are evaluated on held-out test trajectories. Full details of the datasets and meshes used are provided in Appendix B.

The models are asked to predict the next step of the physical field given six input frames, and are evaluated on their ability to roll out predictions over ten steps. We report variance-normalised RMSE (VRMSE) on the physical fields, node-area weighted and averaged over non-constant channels, for the next step and averaged over a ten-step autoregressive rollout; mean $\pm$ standard deviation is over three seeds. Full details of the training and evaluation protocol are provided in Appendix D.

MNC retrofit. We consider the MNC retrofit of two types of physics emulators: (i) in Experiments 1 and 2 (§4.1–§4.2) we retrofit a grid-pretrained Overtone-family model (Mukhopadhyay et al., 2026) and test on an irregular triangulation of its own dataset. (ii) in Experiment 3 (§4.3) we retrofit a released multi-physics foundation model (McCabe et al., 2026) and apply it to the MeshGraphNets cylinder-flow benchmark. Full model specifications can be found in Appendix C. In the following we note that for the MNC retrofitted models *any pretraining is always done on grid-based data with a standard CNN encoder/decoder pair* whose weights are then directly loaded onto the MNC encoder/decoder pair, and the processor is left unchanged. *Only fine-tuning is performed using the MNC encoder/decoder pair* on the target mesh data.

Baselines. We compare against four state-of-the-art baseline physics emulators that are able to treat unstructured data. The important difference is that our MNC layer allows for retrofitting and grid pretraining while the baselines inherently only treat mesh data, gaining no benefit from a grid-based data format. Our baselines are the neural operators GINO (Li et al., 2023c) and LNO (Wang & Wang, 2024), whose encoders act on point sets, the point-cloud transformer Transolver++ (Luo et al., 2025), and the mesh-edge graph network MeshGraphNets (Pfaff et al., 2021). Where available, we also performed a processor-free comparison of the tokenisers alone, which is included in Appendix F and shows the favourable transfer properties of the MNC layer even in the absence of a processor.

4.1 Experiment 1: zero-shot transfer onto an irregular mesh

Our first experiment evaluates the zero-shot transfer of grid-pretrained models to irregular mesh representations. The results are shown in Table 1, which evaluates each grid-pretrained model on an irregular mesh representation of the unseen test dataset with similar physical properties (cf. Appendix B). The Overtone+MNC model was pretrained using a CNN encoder/decoder pair on the grid representation of the dataset (using the model (Mukhopadhyay et al., 2026)). These pretrained weights were loaded onto the MNC encoder/decoder pair for evaluation on the irregular mesh. The baselines were trained from scratch on the grid mesh representation of the dataset. The results show that the MNC retrofit retains the most pretrained skill on all three datasets and under both metrics,

outperforming all baselines in zero-shot transfer. An example of the zero-shot transfer for shear flow is shown in Fig. 1b, and the error fields in next-step prediction for TRL2D are shown in Fig. 5. Here the fainter error field of the Overtone+MNC model means that it retains more of the pretrained skill.

Table 1: Zero-shot transfer onto the irregular mesh (Experiment 1): next-step and ten-step rollout VRMSE, in units of 10^{-2}. Mean $\pm$ standard deviation over three pretraining seeds.

	Overtone+MNC	GINO	LNO	Transolver++	MeshGraphNets
Next-step VRMSE					
TRL2D	$\mathbf{26.25 \pm 0.45}$	29.55 ± 0.10	48.78 ± 3.67	40.46 ± 0.28	45.07 ± 0.99
Active matter	$\mathbf{4.88 \pm 0.12}$	11.54 ± 0.20	8.00 ± 0.45	12.44 ± 0.07	19.58 ± 0.18
Shear flow	1.42 ± 0.03	1.82 ± 0.00	3.85 ± 0.04	$\mathbf{1.04 \pm 0.22}$	2.55 ± 0.02
10-step rollout VRMSE					
TRL2D	$\mathbf{52.27 \pm 0.98}$	63.64 ± 0.85	85.03 ± 7.48	79.93 ± 1.90	85.04 ± 2.62
Active matter	$\mathbf{26.05 \pm 0.61}$	73.59 ± 4.04	54.04 ± 1.73	84.74 ± 0.94	113.84 ± 10.00
Shear flow	$\mathbf{8.19 \pm 0.53}$	9.80 ± 0.39	16.48 ± 0.35	11.51 ± 1.14	19.88 ± 1.48

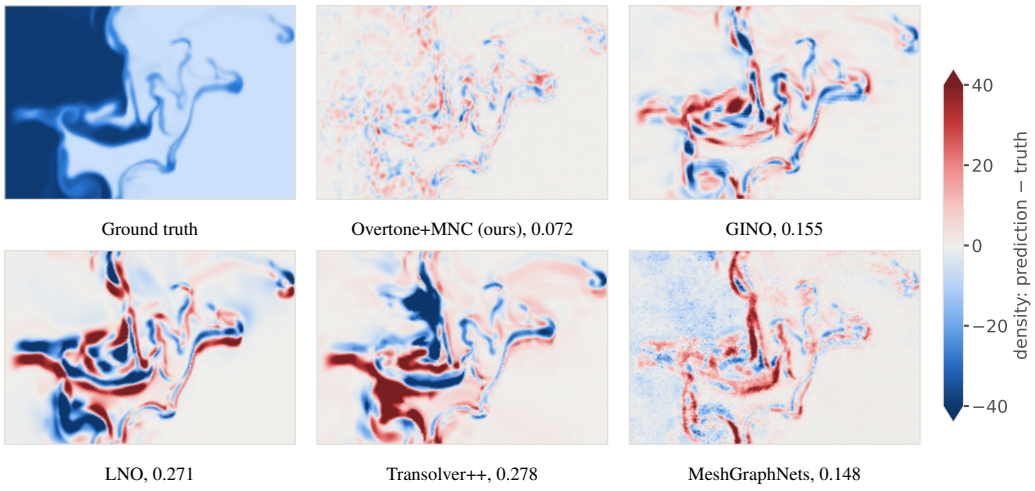


Figure 5: Zero-shot one-step density error on the irregular TRL2D mesh. Numbers are the window's density VRMSE; predicting no change (persistence) scores 0.308.

4.2 Experiment 2: Fine-tuning on irregular meshes

While zero-shot performance provides a useful baseline for judging the effectiveness of weight transfer onto the MNC tokeniser, it turns out that the mesh-native tokenisation also provides further advantages for fine-tuning models on unstructured meshes. In this experiment we compare the performance of the MNC retrofit against the baselines when fine-tuned on the irregular mesh, for 200 epochs (Appendix D) on each of the three Well datasets. The results for next-step VRMSE on the irregular TRL2D are shown in Fig. 6, with full results for each dataset provided in Table 2. The MNC retrofit leads to extremely fast fine-tuning: it reaches its plateau within ten epochs, while the same MNC-tokenised transformer trained from scratch needs about a hundred epochs to arrive at the same level. The grid-pretrained baselines start below their scratch counterparts but improve little (MeshGraphNets the most, levelling off well above the retrofit), indicating that their pretraining insights do not transfer as easily onto unstructured meshes. The MNC-tokenised transformer ends below every baseline whether or not it was pretrained, showing that the retrofit inherits a strong architecture as well as a strong initialisation.

Table 2 shows that this observation holds across all three datasets. After fine-tuning, the MNC retrofit achieves the lowest next-step and rollout VRMSE on TRL2D, active matter and shear flow, with errors a factor of two to three below the best baseline on active matter and shear flow. The benefit of grid pretraining is most pronounced on these two datasets, where it reduces the next-step error of the MNC model by about a quarter and a third, respectively, compared to training from scratch, while on TRL2D the pretrained and scratch models end up at similar levels.

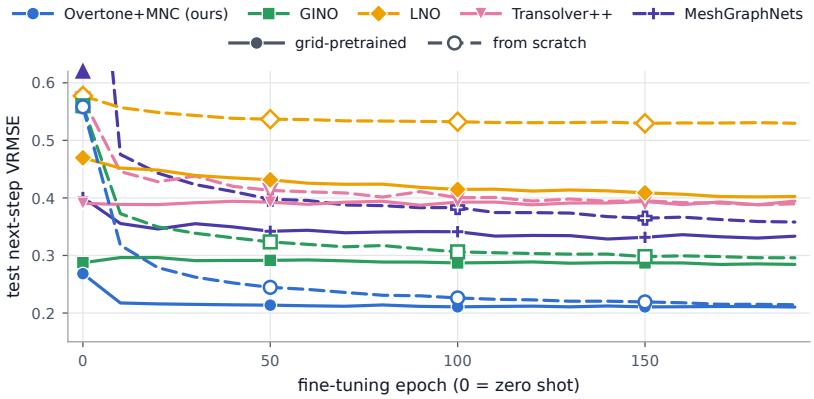


Figure 6: Test next-step VRMSE (mean) on the irregular TRL2D mesh against fine-tuning epoch. Grid-pretrained models are marked in solid lines and randomly initialised ones in dashed lines.

Table 2: Fine-tuning on the irregular mesh: next-step and ten-step rollout VRMSE, in units of 10^{-2}.

		Overtone+MNC	GINO	LNO	Transolver++	MeshGraphNets
Next-step VRMSE						
TRL2D	pretrained	$\mathbf{22.10 \pm 0.04}$	29.40 ± 0.08	40.89 ± 0.12	40.76 ± 0.07	33.91 ± 0.43
	scratch	22.77 ± 0.20	29.78 ± 0.27	53.39 ± 0.11	39.78 ± 0.13	36.84 ± 0.46
Active matter	pretrained	$\mathbf{3.30 \pm 0.06}$	10.71 ± 0.04	8.12 ± 0.05	12.81 ± 0.04	11.20 ± 0.26
	scratch	4.36 ± 0.08	12.84 ± 0.20	15.86 ± 1.27	14.09 ± 0.08	13.35 ± 0.03
Shear flow	pretrained	$\mathbf{0.43 \pm 0.01}$	1.73 ± 0.02	3.56 ± 0.12	0.98 ± 0.05	1.78 ± 0.05
	scratch	0.66 ± 0.02	2.12 ± 0.01	3.89 ± 0.00	1.34 ± 0.05	1.84 ± 0.08
10-step rollout VRMSE						
TRL2D	pretrained	$\mathbf{49.04 \pm 1.61}$	59.23 ± 2.24	68.29 ± 1.38	71.38 ± 2.63	75.11 ± 2.39
	scratch	50.60 ± 4.03	64.14 ± 1.62	79.32 ± 1.41	69.63 ± 1.08	75.09 ± 5.43
Active matter	pretrained	$\mathbf{22.79 \pm 1.83}$	63.71 ± 1.81	56.51 ± 1.67	91.08 ± 2.33	107.97 ± 9.89
	scratch	29.73 ± 1.62	85.80 ± 15.30	81.59 ± 2.52	91.12 ± 2.15	102.29 ± 11.67
Shear flow	pretrained	$\mathbf{2.65 \pm 0.16}$	8.86 ± 0.52	17.32 ± 3.18	10.16 ± 0.97	30.27 ± 9.40
	scratch	4.04 ± 0.45	12.05 ± 2.36	16.72 ± 0.00	13.07 ± 1.21	22.40 ± 5.22

4.3 Experiment 3: Zero-shot retrofitting of a large-scale foundation model

Our final experiment confirms that this retrofitting approach is not limited to small-scale models and datasets. Indeed, the MNC retrofit can provide an efficient bridge between large-scale grid-pretrained foundation models and unstructured mesh problems. In this experiment we retrofit Walrus (McCabe et al., 2026), a large-scale multi-physics foundation model pretrained on a multi-physics corpus, to the MeshGraphNets cylinder-flow benchmark. A particularly noteworthy aspect of this experiment is that the cylinder-flow dataset is not included in Walrus's pretraining corpus, so the model has no prior knowledge of this flow configuration. The MNC retrofit allows us to leverage the pretrained knowledge of Walrus and apply it to a new problem domain without requiring any additional training on the cylinder-flow dataset. In Table 3 we report the next-step and ten-step rollout VRMSE for the retrofitted Walrus model, both in zero-shot and after fine-tuning on the cylinder-flow training split. We also include the performance of the baselines trained from scratch on the cylinder-flow dataset for comparison. In each of the fine-tuning/training configurations we allowed the same compute budget of 7.5h on a single NVIDIA GH200 GPU. The results show that (i) zero-shot, the retrofitted Walrus model outperforms all trained baselines except Transolver++, meaning that grid-pretrained knowledge is effectively transferred to the unstructured mesh problem even for unseen datasets, and (ii) the retrofitted Walrus model achieves the lowest next-step error after fine-tuning, outperforming all baselines by a significant margin.

Table 3: Cylinder flow dataset: next-step and ten-step rollout VRMSE, in units of 10^{-2}. All training and fine-tuning runs were given the same compute budget of 7.5h on a single NVIDIA GH200 GPU.

	Next-step VRMSE	10-step rollout VRMSE
Walrus+MNC, pre-trained, fine-tuned	$\mathbf{0.15 \pm 0.01}$	$\mathbf{1.02 \pm 0.00}$
Walrus+MNC, pre-trained, zero-shot	0.41	3.30
Walrus+MNC, random init, fine-tuned	0.44 ± 0.01	2.47 ± 0.04
MeshGraphNets, trained on cylinder flow	0.80 ± 0.04	8.26 ± 0.03
Transolver++, trained on cylinder flow	0.18 ± 0.01	2.05 ± 0.16
GINO, trained on cylinder flow	0.70 ± 0.01	3.80 ± 0.21
LNO, trained on cylinder flow	2.26 ± 0.00	12.10 ± 0.00
Persistence	2.34	12.50

Figure 7 shows a one-step prediction of the x-velocity on a cylinder-flow test trajectory. Without any training on cylinder flow, the retrofitted Walrus predicts the vortex street with an error below all baseline models. Walrus was only pretrained on grid data and weights from its CNN tokeniser were loaded onto MNC. By contrast, each of the baselines was trained on the dataset with 7.5h of compute budget, demonstrating the advantage of transferring grid-based models onto unstructured datasets.

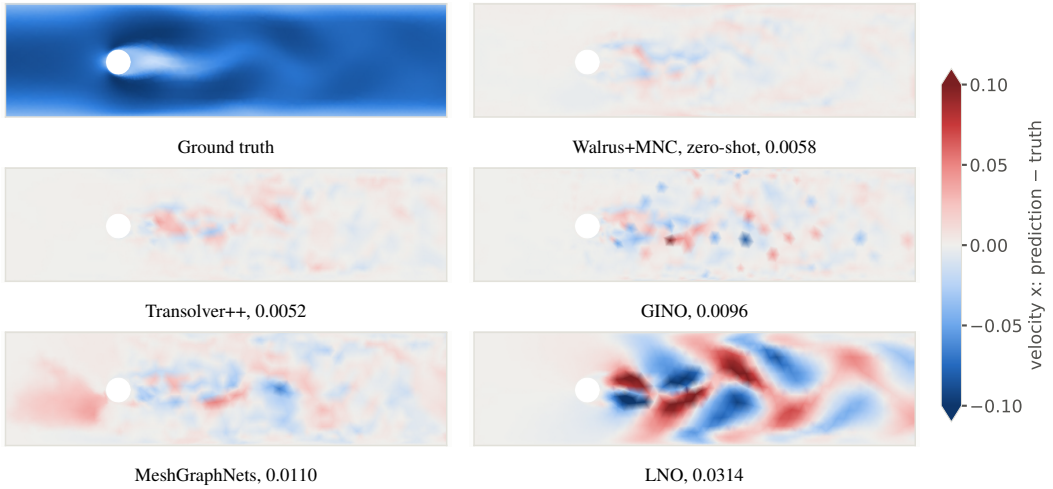


Figure 7: Error in next-step x-velocity prediction on cylinder flow. Numbers are the window's x-velocity VRMSE; predicting no change (persistence) scores 0.0314.

5 Discussion and Conclusion

In this work we introduced mesh-native tokenisation as a means for retrofitting grid-pretrained physics emulators to unstructured meshes. The MNC layer is a simple locality-aware encoder that allows the transfer of pretrained weights from a CNN encoder/decoder pair to a mesh-native tokeniser, while retaining the source model's processor. We demonstrated that this approach allows for zero-shot transfer of grid-pretrained models onto irregular meshes, and that fine-tuning on the target mesh can be done efficiently and effectively. We also showed that this approach can be applied to large-scale foundation models, allowing for the transfer of knowledge from grid-based pretraining to unstructured mesh problems. This allows for significant computational savings through both repurposing of pretrained weights and reduced training time of processors by leveraging fast CNN-based pretraining (Chetlur et al., 2014).

Limitations and future work. MNC retrofitting mainly improves sample efficiency, leading to quicker fine-tuning runs rather than overall improved accuracy after sufficient training. This suggests that the benefits of MNC are most pronounced in scenarios with limited training data or computational resources, and future work can focus on improving the performance of the retrofitted models in these scenarios, for example by introducing geometric quadrature weights for consistency under varying sampling density (Proposition 2).

AI USE STATEMENT

In this work, we used generative AI tools to help formulate mathematical claims, to provide ingredients for and assist in writing the proofs of these claims, to provide feedback on the design of the experiments, to implement methods, and to clean and reformat datasets (in particular the conversion of grid and mesh data into a common format). We have not used generative AI tools to generate synthetic data sets, develop theoretical models or conceptual frameworks, propose or refine hypotheses, or interpret results, and translation and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools to create and edit software code, create and modify schematic figures, suggest experimental parameters, and draft parts of the paper. We have reviewed all AI-assisted work: all mathematical claims and proofs were checked line by line by the authors; AI-assisted code was reviewed and tested by the authors, including exactness checks of the weight transfer against the source models; all AI-drafted text was reviewed and edited by the authors, with every reported number checked against the underlying results; and all figures were checked against the underlying data. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only publicly available simulation data of physical systems (The Well and the MeshGraphNets cylinder-flow benchmark) and a publicly released pretrained model. No human subjects, personal data or other sensitive information were involved at any stage. All authors have read and adhere to the ICLR Code of Ethics, and we are not aware of any ethical concerns or potential for harm specific to this work beyond those of general-purpose scientific machine learning.

REPRODUCIBILITY STATEMENT

All data and pretrained weights used in this work are publicly available. The TRL2D, shear-flow and active-matter datasets are part of The Well (Ohana et al., 2024), and the cylinder-flow benchmark is taken from the official MeshGraphNets (Pfaff et al., 2021) release. Appendix B provides details on the dataset splits, the construction of the irregular meshes and the resampling of the grid fields. The Walrus checkpoint retrofitted in Experiment 3 is the publicly released one (McCabe et al., 2026), used without modification of its weights. The MNC layer is specified in §2, and complete proofs of Propositions 1 and 2 and of the GCN counterexample are given in Appendix A. Appendix C lists every architecture with its reference implementation, configuration and parameter count, together with the training settings, and Appendix D gives the training and evaluation protocol, including the metrics, seeds and rollout procedure. After the review process we will release the complete code on GitHub, together with the trained model weights and the irregular meshes on Hugging Face.

REFERENCES

Benedikt Alkin, Andreas Fürst, Simon Lucas Schmid, Lukas Gruber, Markus Holzleitner, and Johannes Brandstetter. Universal physics transformers: A framework for efficiently scaling neural operators. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Nicolas Boullé and Alex Townsend. A mathematical guide to operator learning. In Siddhartha Mishra and Alex Townsend (eds.), *Numerical Analysis Meets Machine Learning*, volume 25 of *Handbook of Numerical Analysis*, pp. 83–125. Elsevier, 2024.

Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Texts in Applied Mathematics. Springer, 3 edition, 2008.

Sharan Chetlur, Cliff Woolley, Philippe Vandermersch, Jonathan Cohen, John Tran, Bryan Catanzaro, and Evan Shelhamer. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint arXiv:1410.0759*, 2014.

Cristiana Diaconu, Miles Cranmer, Richard E. Turner, Tanya Marwah, and Payel Mukhopadhyay. Probabilistic Retrofitting of Learned Simulators. In *Forty-third International Conference on Machine Learning*, 2026.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*, 2021.

Matthias Fey, Jan Eric Lenssen, Frank Weichert, and Heinrich Müller. SplineCNN: Fast geometric deep learning with continuous B-spline kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1263–1272. PMLR, 06–11 Aug 2017.

Jan Hagnberger, Daniel Musekamp, and Mathias Niepert. CALM-PDE: Continuous and Adaptive Convolutions for Latent Space Modeling of Time-dependent PDEs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Maximilian Herde, Bogdan Raonic, Tobias Rohner, Roger Käppeli, Roberto Molinaro, Emmanuel de Bezenac, and Siddhartha Mishra. Poseidon: Efficient foundation models for PDEs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023.

Zijie Li, Kazem Meidani, and Amir Barati Farimani. Transformer for partial differential equations' operator learning. *Transactions on Machine Learning Research*, 2023a. ISSN 2835-8856.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier Neural Operator with Learned Deformations for PDEs on General Geometries. *Journal of Machine Learning Research*, 24(388):1–26, 2023b.

Zongyi Li, Nikola Borislavov Kovachki, Chris Choy, Boyi Li, Jean Kossaifi, Shourya Prakash Otta, Mohammad Amin Nabian, Maximilian Stadler, Christian Hundt, Kamyar Azizzadenesheli, and Anima Anandkumar. Geometry-Informed Neural Operator for Large-Scale 3D PDEs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023c.

Huakun Luo, Haixu Wu, Hang Zhou, Lanxiang Xing, Yichen Di, Jianmin Wang, and Mingsheng Long. Transolver++: An accurate neural solver for PDEs on million-scale geometries. In *Forty-second International Conference on Machine Learning*, 2025.

Michael McCabe, Bruno Régaldo-Saint Blancard, Liam Parker, Ruben Ohana, Miles Cranmer, Alberto Bietti, Michael Eickenberg, Siavash Golkar, Geraud Krawezik, Francois Lanusse, Mariel Pettee, Tiberiu Tesileanu, Kyunghyun Cho, and Shirley Ho. Multiple Physics Pretraining for Spatiotemporal Surrogate Models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 119301–119335, 2024. doi: 10.52202/079017-3791.

Michael McCabe, Payel Mukhopadhyay, Tanya Marwah, Bruno Régaldo-Saint Blancard, François Rozet, Cristiana Diaconu, Lucas Thibaut Meyer, Kaze W. K. Wong, Mohammad-Hadi Sotoudeh, Alberto Bietti, Irina Espejo Morales, Rio Alexa Fear, Siavash Golkar, Tom Hehir, Keiya Hirashima, Geraud Krawezik, Francois Lanusse, Rudy Morel, Ruben Ohana, Liam Holden Parker, Mariel Pettee, Jeff Shen, Kyunghyun Cho, Miles Cranmer, and Shirley Ho. Walrus: A Cross-domain Foundation Model for Continuum Dynamics. In *Forty-third International Conference on Machine Learning*, 2026.

Payel Mukhopadhyay, Michael McCabe, Ruben Ohana, and Miles Cranmer. Overtone: Cyclic Patch Modulation for Clean, Efficient, and Flexible Physics Emulators. In *The Fourteenth International Conference on Learning Representations*, 2026.

Ruben Ohana, Michael McCabe, Lucas Meyer, Rudy Morel, Fruzsina Agocs, Miguel Beneitez, Marsha Berger, Blakesly Burkhart, Stuart Dalziel, Drummond Fielding, et al. The well: a large-scale collection of diverse physics simulations for machine learning. *Advances in Neural Information Processing Systems*, 37:44989–45037, 2024.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2021.

James Rowbottom, Georg Maierhofer, Teo Deveney, Eike Hermann Müller, Alberto Paganini, Katharina Schratz, Pietro Lio, Carola-Bibiane Schönlieb, and Chris Budd. G-Adaptivity: optimised graph-based mesh relocation for finite element methods. In *Forty-second International Conference on Machine Learning*, 2025.

Louis Serrano, Lise Le Boudec, Armand Kassaï Koupaï, Thomas X Wang, Yuan Yin, Jean-Noël Vittaut, and patrick gallinari. Operator Learning with Neural Fields: Tackling PDEs on General Geometries. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*.

Clinton Wang and Polina Golland. Discretization Invariant Networks for Learning Maps between Neural Fields. *Transactions on Machine Learning Research*, 2023. ISSN 2835–8856.

Tian Wang and Chuang Wang. Latent Neural Operator for Solving Forward and Inverse PDE Problems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Haixu Wu, Huakun Luo, Haowen Wang, Jianmin Wang, and Mingsheng Long. Transolver: A fast transformer solver for PDEs on general geometries. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53681–53705. PMLR, 21–27 Jul 2024.

Philippe Martin Wyder, Judah A Goldfeder, Alexey Yermakov, Yue Zhao, Stefano Riva, Jan P. Williams, David Zoro, Amy Sara Rude, Matteo Tomasetto, Joe Germany, Joseph Bakarji, Georg Maierhofer, Miles Cranmer, and J. Nathan Kutz. Common Task Framework For a Critical Evaluation of Scientific Machine Learning Algorithms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026.

Alexey Yermakov, Yue Zhao, Marine Denolle, Yiyu Ni, Philippe Martin Wyder, Judah Goldfeder, Stefano Riva, Jan P. Williams, David Zoro, Amy Sara Rude, Matteo Tomasetto, Joe Germany, Joseph Bakarji, Georg Maierhofer, Miles Cranmer, and J. Nathan Kutz. The Seismic Wavefield Common Task Framework. In *The Fourteenth International Conference on Learning Representations*, 2026.

A TRANSFER IDENTITIES AND CONSISTENCY PROOFS

A.1 PROOF OF PROPOSITION 1

The hats in (2) satisfy $\phi(m) = \mathbf{1}_{\{m=0\}}$ for integer m. At each stencil offset $\mathbf{u} \in \mathcal{O}$, therefore, $\mathbf{W}(\mathbf{u}) = \mathbf{K}[\mathbf{u}]$ and $\omega(\mathbf{u}) = 1$. Every other grid offset has zero kernel and occupancy. A complete encoder window has $P = k_h k_w$ such offsets, so $s_{\hat{v}} = 1$ and (4) is exactly (1). Missing boundary samples must be represented by the same padding values as the source; for zero padding their features are zero but their occupancy still counts. Omitting those occupancy contributions would rescale boundary windows and destroy the identity.

For clarity, let $\Theta_{o,i,r,c}$ be a tensor stored by `Conv2d`, with $0 \le r < k_h$ and $0 \le c < k_w$. The offset loaded into the encoder is

$$\mathbf{u}_{r,c} = \big(((k_w - 1)/2 - c)h_x,\ ((k_h - 1)/2 - r)h_y\big), \qquad \mathbf{K}[\mathbf{u}_{r,c}]_{o,i} = \Theta_{o,i,r,c}. \tag{10}$$

Indeed, $\mathbf{p}_{\hat{v}} - \mathbf{u}_{r,c}$ is the input location read by the stored cross-correlation kernel. Even kernel sizes simply give half-integer offsets relative to the window centre; the differences between offsets remain integer multiples of h_x, h_y.

For `ConvTranspose2d`, a stored tensor $\Psi_{i,o,r,c}$ scatters from an anchor to the offset

$$\mathbf{v}_{r,c} = \big((c - (k_w - 1)/2)h_x,\ (r - (k_h - 1)/2)h_y\big), \qquad \widetilde{\mathbf{K}}[\mathbf{v}_{r,c}]_{o,i} = \Psi_{i,o,r,c}. \tag{11}$$

Here $\mathbf{v}$ denotes a spatial offset. With unit dilation, stride equal to kernel size and zero output padding, every retained grid output belongs to exactly one discrete footprint. Other anchors can meet it only at zero spline values. Hence its occupancy is one, $t_j = 1$, and (5) recovers the transposed convolution, including its bias. Output cropping and padding must match the source convention. This argument does not assert exact reduction of the normalised decoder for arbitrary overlapping strides.

For a hierarchy, apply these identities at each level with the corresponding coarse-grid spacing. Preserve the pointwise projections, activations, feature normalisation and token order. In (6), the source token centres $p_a = p_a^{\min} + mH_a$, $m = 0, \dots, n_a - 1$, with H_a the source token spacing, give $\mathrm{pos}_a = 2m/(n_a - 1) - 1$, the $\mathrm{linspace}(-1, 1, n_a)$ values of the source implementation, and hence exactly the source rotations. The deterministic processor therefore receives and returns the same tensors. This proves equality of the complete grid models. Induction gives equality of autoregressive rollouts when the history update and normalisation are also identical. These are algebraic identities; finite-precision scatter reductions can produce small numerical discrepancies.

A.2 PROOF OF PROPOSITION 2

Let Ω be bounded and $u \in C(\overline{\Omega}; \mathbb{R}^{F_{\mathrm{in}}})$. Extend κ as a globally Lipschitz, matrix-valued function supported in $S = [-R_x, R_x] \times [-R_y, R_y]$; in particular it vanishes on ∂S. All vector norms are Euclidean and matrix norms are the corresponding induced norms. To fix the *entire* support as the number of learned taps grows, take

$$h_x = \frac{2R_x}{k_w + 1}, \quad h_y = \frac{2R_y}{k_h + 1}, \quad L_x = R_x - h_x, \quad L_y = R_y - h_y. \tag{12}$$

The zero ring then lies on ∂S and $P_h a_h = k_h k_w h_x h_y \to |S|$.

Put $\kappa_h = \mathbf{W}_h / a_h$, where the taps are chosen as in (7). This is the bilinear interpolant of κ, extended by zero. Convexity of the interpolation weights gives

$$\|\kappa_h - \kappa\|_\infty \le L_\kappa d_h, \qquad d_h = (h_x^2 + h_y^2)^{1/2}, \quad L_\kappa = \mathrm{Lip}(\kappa), \tag{13}$$

since every point in a tap cell is at most d_h from its corners. Also

$$0 \le \omega_h \le 1, \qquad \int_{\mathbb{R}^2} \omega_h(z)\, dz = P_h a_h, \qquad \mathbf{1}_{S^{-\varepsilon}} \le \omega_h \le \mathbf{1}_S \tag{14}$$

once $\max(h_x, h_y) < \varepsilon$, where $S^{-\varepsilon}$ is S eroded by ε on each side. Each tensor-product hat integrates to a_h, proving the middle identity.

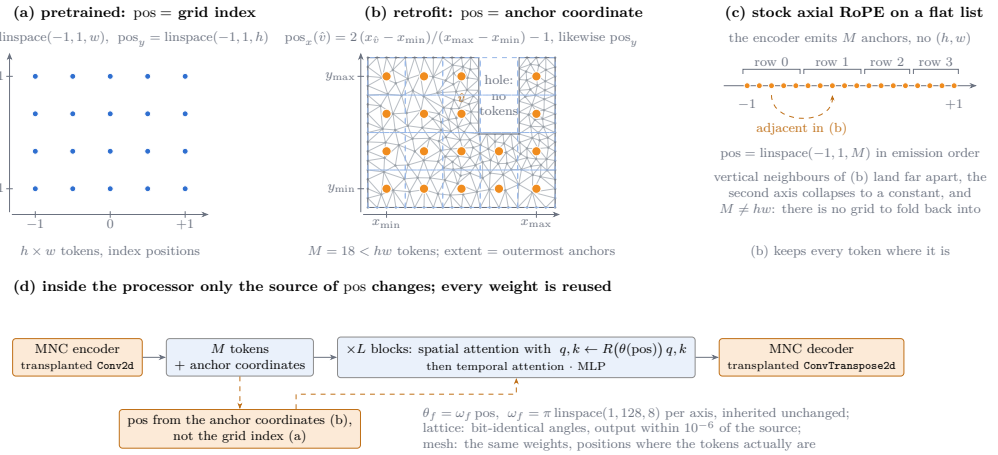


Figure 8: Coordinate rotary position embedding for mesh tokens, and where it enters the pretrained model. **(a)** The source processor rotates query/key channel pairs by $\theta_f = \omega_f \,\text{pos}$, with per-axis positions $\text{pos} = \text{linspace}(-1, 1, \cdot)$ read off the $h \times w$ token grid. **(b)** After the transplant the tokens are the window anchors of the mesh hierarchy; each carries its own coordinate, rescaled to $[-1, 1]$ by the outermost anchors as in (6). Windows emptied by a hole produce no token ($M < hw$). On a lattice the equally spaced anchors are snapped to the same linspace evaluation as in (a), giving bit-identical angles; an interior hole, as shown, leaves the extent unchanged, whereas an empty *outer* column would move $x_{\max}$ inward. Row 0 is $y_{\min}$, following the ascending emission order of the hierarchy builder. **(c)** The stock axial code cannot be reused as is: the mesh encoder emits a flat list of M anchors, which it would string along one axis in emission order, separating vertical neighbours and collapsing the second axis to a constant. **(d)** Inside the processor only the source of pos changes; the frequencies ω_f, the channel pairing and all weights are inherited. The third positional axis of the axial embedding, unused in two dimensions, carries the constant -1 and contributes a common rotation that cancels in $q \cdot k$. On the source lattice the retrofitted model reproduces the source output to a relative error of order 10^{-6}.

Proof. Write $\mu_n = c_n \sum_j \delta_{\mathbf{p}_j^{(n)}}$, with weak convergence understood against every function in $C(\overline{\Omega})$. In particular, $\mu_n(\Omega)$ is bounded. Uniform convergence in (13) and weak convergence give, jointly as $h_x, h_y \to 0$ and $n \to \infty$,

$$\int \kappa_h(\mathbf{p} - y) u(y)\, d\mu_n(y) \longrightarrow \int_\Omega \kappa(\mathbf{p} - y) u(y) \rho(y)\, dy. \tag{15}$$

For the denominator, squeeze $\omega_h(\mathbf{p} - \cdot)$ between the indicators of the translated rectangles in (14). Their boundaries have zero $\rho\, dy$ measure, so weak convergence applies to these indicators. Letting $\varepsilon \downarrow 0$ yields

$$\int \omega_h(\mathbf{p} - y)\, d\mu_n(y) \longrightarrow \int_{\mathbf{p} - S} \rho(y)\, dy > 0.$$

Since $c_n \to 0$, the unscaled occupancy sum tends to infinity, and the lower clipping in (4) is eventually inactive. The bias-free output can then be written as

$$P_h a_h \, \frac{\int \kappa_h(\mathbf{p} - y) u(y)\, d\mu_n(y)}{\int \omega_h(\mathbf{p} - y)\, d\mu_n(y)}.$$

Together with $P_h a_h \to |S|$, these limits prove (8). For constant sampling density on the window, the density and window-area factors cancel. Applying this result to two sequences with the same field and kernel gives the stated asymptotic agreement. □

There is no hidden condition that the input mesh resolve every tap cell: the proof first controls the kernel uniformly, then passes to the limit in the sampling measure. For a *fixed* checkpoint, weak convergence applies directly to the continuous kernel and occupancy. Under constant density on a

full window, $\int \omega_h = P_h a_h$ gives the limiting kernel $\mathbf{W}_h/a_h$. Conversely, arbitrary growing tap arrays need not converge: scalar taps all equal to one, $u \equiv 1$, and sufficiently many input samples give output $P_h \to \infty$. The result is a consistency statement for a specified kernel sequence, not a convergence guarantee for learning.

A.3 A CONNECTIVITY COUNTEREXAMPLE FOR A NORMALISED GCN

Consider the scalar GCN layer of Kipf & Welling (2017),

$$G_\Gamma(\mathbf{x}) = \mathrm{ReLU}\Big(\widetilde{D}^{-1/2}(A+I)\widetilde{D}^{-1/2}\mathbf{x}\Big), \qquad \widetilde{D}_{ii} = 1 + \deg_\Gamma(i), \tag{16}$$

with weight one and no bias. Let the vertices be $\eta(i,j)$, $-n \le i,j \le n$, $\eta = 1/n$, and $n \ge 3$.

In mesh A, split every square along its southwest–northeast diagonal. Every vertex in the origin's one-ring, including the origin, has degree six and self-loop degree seven. For $u \equiv 1$, the seven contributions at the origin each equal $1/7$, giving $G_{\Gamma_\eta^A}(\mathbf{1})_0 = 1$.

In mesh B, choose each square's diagonal to join its two vertices with even $i+j$. Even vertices have degree eight and odd vertices degree four. The origin has four axial odd neighbours and four diagonal even neighbours. Its self-loop and its four even neighbours each contribute $1/9$, while its four odd neighbours each contribute $1/\sqrt{9\cdot 5}$. Therefore

$$G_{\Gamma_\eta^B}(\mathbf{1})_0 = \frac{5}{9} + \frac{4}{\sqrt{45}} = \frac{5}{9} + \frac{4}{3\sqrt{5}} = 1.1518403495\ldots.$$

Both values are positive, so ReLU leaves them unchanged. Both triangulations comprise right isosceles triangles of diameter $\sqrt{2}\eta$, and both represent the same constant field exactly. The discrepancy also persists for any positive scalar weight, which simply multiplies both outputs.

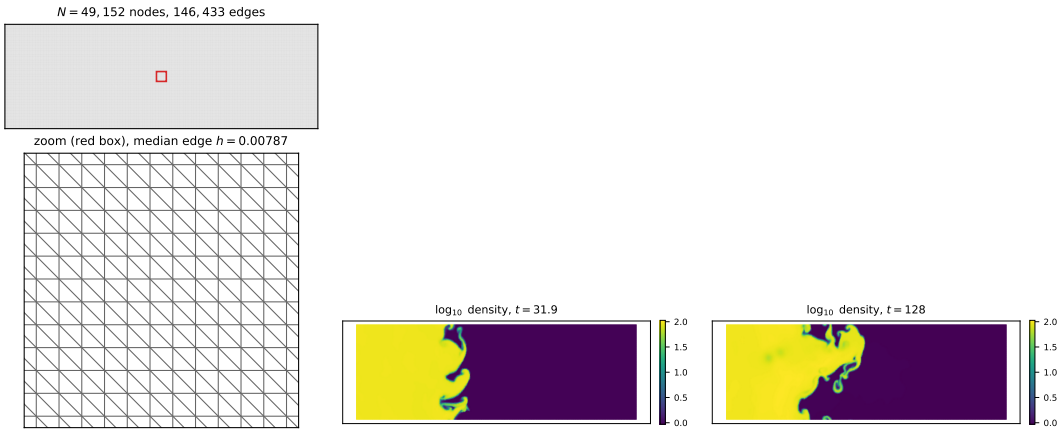
With common anchors, tap spacings and coefficients, MNC has the same geometric neighbours and occupancies on these two meshes and hence exactly the same output. This counterexample isolates connectivity: the sample locations themselves do not change. It does not rule out consistent graph architectures, for example those designed as quadrature discretisations of integral operators (Kovachki et al., 2023).

B DATASETS

All datasets are stored in one "grid + mesh" HDF5 convention: fields on the nodes of a triangulated mesh (`mesh_coords`, `edge_index`) together with, where a grid exists, the original Cartesian arrays. For TRL2D and shear flow from The Well (Ohana et al., 2024) the mesh twin is a Gmsh transfinite triangulation whose nodes coincide with the grid points, so grid and mesh tokenisers see identical field values on identical sample points and differ only in how they access them; for active matter the on-grid target is the lattice of grid points itself. The forecasting experiments of §4 are evaluated on a separate irregular Delaunay mesh of each Well dataset. Every sample is a window of six consecutive context snapshots and the following target snapshot. Figures 9–15 show each dataset's meshes and representative fields; Table 4 lists the sizes. The forecasting experiments use the official TRL2D and active-matter splits and a trajectory-level resplit of the shear-flow training files (Appendix D); the reconstruction experiments use the official splits throughout (Appendix G).

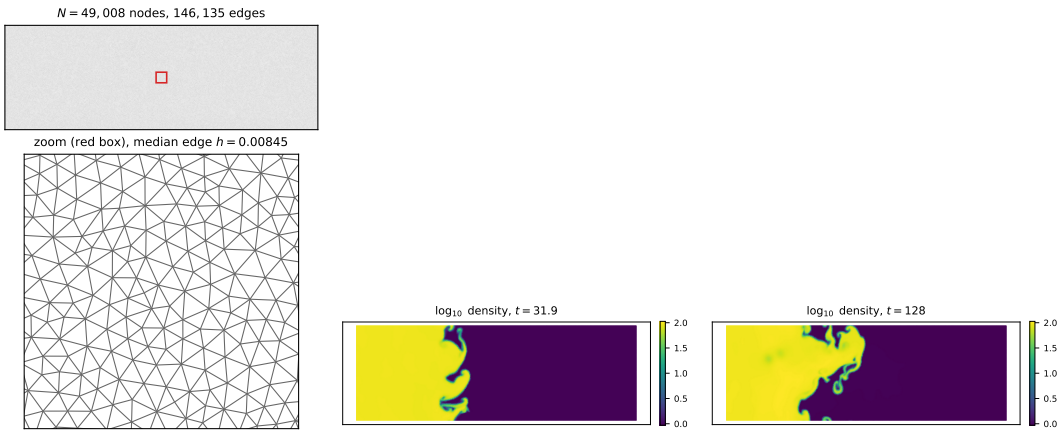
Turbulent radiative layer (TRL2D). A two-dimensional mixing layer between a cold dense and a hot dilute gas with radiative cooling, on a 128×384 grid ($x \in [-0.5, 0.5]$ periodic, $y \in [-1, 2]$ open). Fields: density, pressure and two velocity components; density spans two orders of magnitude and is displayed in $\log_{10}$. Nine values of the cooling time $t_{\mathrm{cool}} \in \{0.03, \ldots, 3.16\}$ with 8/1/1 trajectories each for train/validation/test, 101 snapshots per trajectory ($\Delta t = 1.6$). The mesh twin has 49,152 nodes and 146,433 edges. The pressure field is nearly spatially uniform (spatial standard deviation $\approx 2\%$ of its magnitude), which makes its VRMSE ill-conditioned; the reconstruction experiments therefore report normalised MSE.

Shear flow. Two-dimensional incompressible shear layers with a passive tracer on a doubly periodic 256×512 grid over $[0,1]^2$, for 28 combinations of Reynolds number (10^4 to $5\cdot10^5$) and Schmidt number (0.1 to 10) with 40 initial conditions each, 200 snapshots per trajectory ($\Delta t = 0.1$). Fields:



(a) mesh and zoom (b) $\log_{10}\rho$, $t = 31.9$ (c) $\log_{10}\rho$, $t = 128$

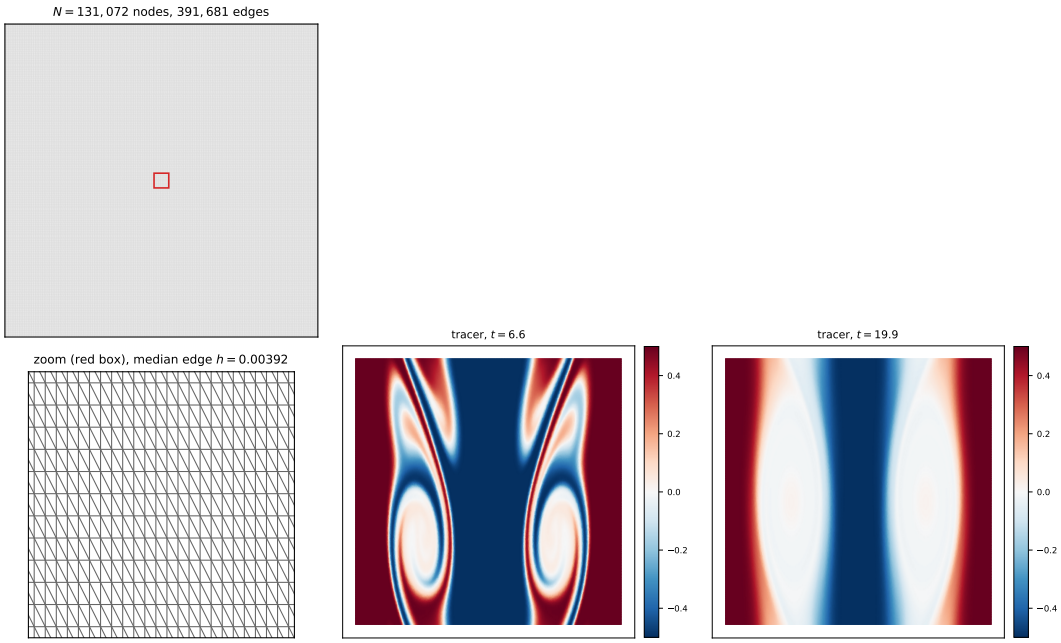
Figure 9: TRL2D: the mesh twin with the zoom region marked in red and the zoomed mesh below it (left), and $\log_{10}$ density at two times along a $t_{\text{cool}} = 0.18$ trajectory. The Well mesh twins are triangulations of the original grids, so grid and mesh tokenisers see exactly the same samples.



(a) mesh and zoom (b) $\log_{10}\rho$, $t = 31.9$ (c) $\log_{10}\rho$, $t = 128$

Figure 10: TRL2D on the Experiment 1 irregular mesh: the Gmsh Delaunay triangulation (49,008 nodes, 146,135 edges, node count matched to the grid) with the zoom region marked in red (left), and the same $t_{\text{cool}} = 0.18$ density snapshots as Figure 9, bilinearly resampled onto the mesh nodes — exactly the samples the zero-shot-evaluated models see.

tracer, pressure and velocity. Depending on the initial interface thickness a trajectory either rolls up into Kelvin–Helmholtz billows (Figure 11) or stays close to its laminar profile, so the test sets mix both regimes. The mesh twin has 131,072 nodes and 391,681 edges. The reconstruction transfer experiments additionally triangulate the domain at matched resolution as a uniform triangular lattice and as an irregular Delaunay mesh (Figure 13); the forecasting experiments use the irregular mesh of Figure 12.

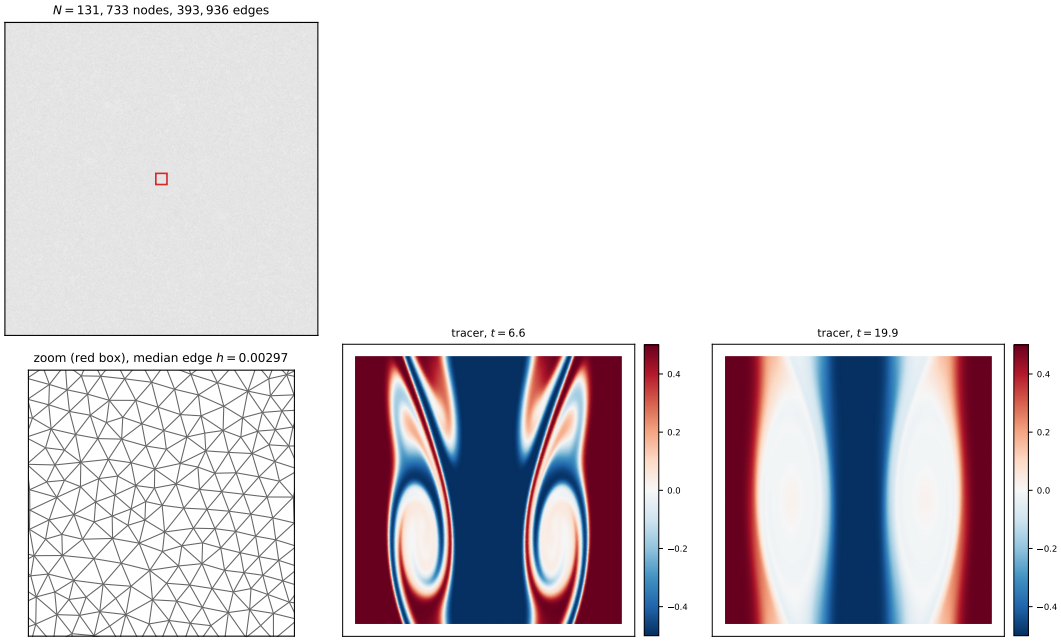


(a) mesh and zoom (b) tracer, $t = 6.6$ (c) tracer, $t = 19.9$

Figure 11: Shear flow: the mesh twin with zoom (left), and the tracer at two times along a $Re = 10^4$, $Sc = 1$ trajectory that develops Kelvin–Helmholtz billows.

Active matter. A two-dimensional continuum model of active rod-like particles in a Stokes fluid, on a doubly periodic 256×256 grid over $[0, 10]^2$. Fields: the concentration, the two velocity components, and the orientation tensor D and strain-rate tensor E with four components each, eleven channels in total; all of them, including the tensor fields, are inputs and targets. The sweep covers the active dipole strength $\alpha \in \{-1, \dots, -5\}$ and the alignment parameter $\zeta \in \{1, 3, \dots, 17\}$ at fixed $\beta = 0.8$, 45 settings with five trajectories each that differ in the plane-wave perturbation of the initially isotropic orientation tensor; each trajectory has 81 snapshots over $t \in [0, 20]$ ($\Delta t = 0.25$). We use the official split (175/24/26 trajectories), and the source is trained at the native resolution, giving $16 \times 16 = 256$ tokens. The on-grid target is the lattice of the 65,536 grid points. The irregular mesh of Experiment 1 is a Gmsh Delaunay triangulation of the square at a single characteristic length, with the node count matched to the grid: 65,632 nodes and 130,374 triangles, with edge lengths between $0.61h$ and $1.76h$ (median $1.07h$) in units of the grid spacing h. Periodicity is not built into the mesh; the grid fields are resampled onto its nodes. Unlike TRL2D and shear flow, no lattice twin is stored for active matter, so Figure 14 shows the irregular mesh only.

Cylinder flow. The MeshGraphNets incompressible flow past a cylinder (Pfaff et al., 2021) (COMSOL), 1,000/100/100 trajectories for train/validation/test, each on *its own* unstructured triangulated mesh (1,732–2,059 nodes, median 1,885) of the 1.6×0.41 channel, refined towards the cylinder and the walls. Fields: pressure and velocity; 600 snapshots per trajectory ($\Delta t = 0.01$). Node types mark inflow, outflow and wall nodes. This is the only dataset without a grid representation; the MNC hierarchy is built per trajectory. It is used in Experiment 3, on refined meshes (Appendix B.1), and in the reconstruction experiments on the native meshes (Appendix G).



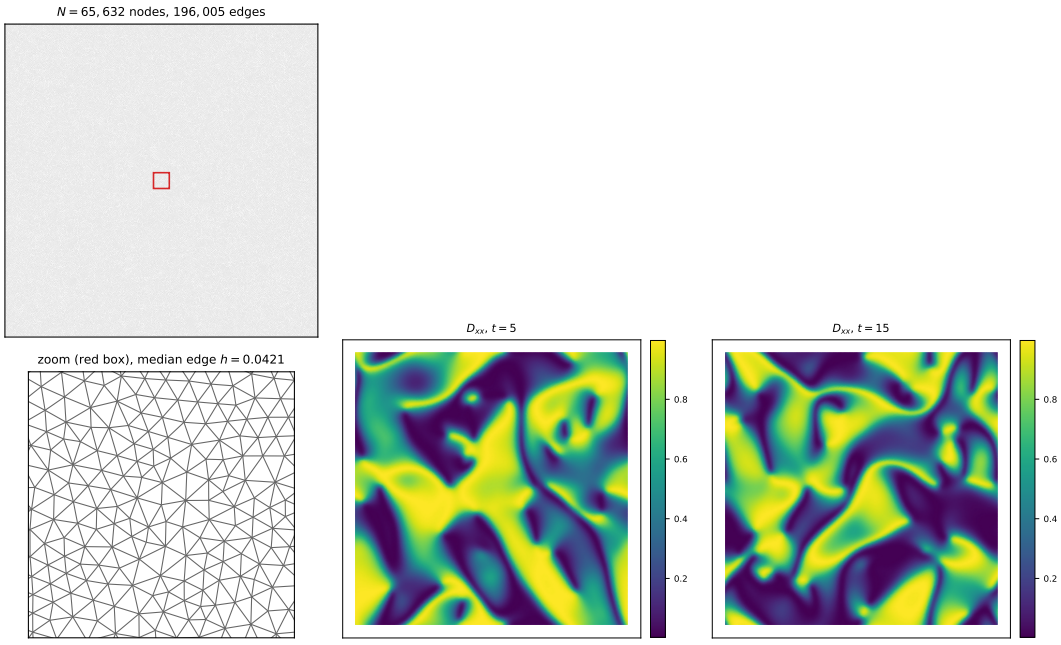
(a) mesh and zoom (b) tracer, $t = 6.6$ (c) tracer, $t = 19.9$

Figure 12: Shear flow on the Experiment 1 irregular mesh (131,733 nodes, 393,936 edges, node count matched to the grid): the same $Re = 10^4$, $Sc = 1$ tracer snapshots as Figure 11, bilinearly resampled onto the mesh nodes (trajectory from the official validation file; Experiment 1 itself uses the trajectory-level resplit of the training files).



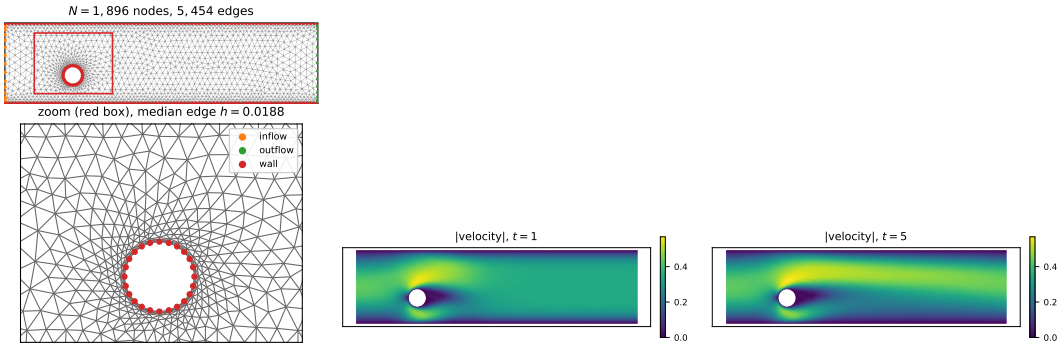
(a) Irregular target. (b) Uniform triangular target.

Figure 13: The shear-flow target meshes used for tokeniser transfer. Source-grid fields are resampled onto their nodes; the targets are not independent simulations.



(a) mesh and zoom (b) D_{xx}, $t = 5$ (c) D_{xx}, $t = 15$

Figure 14: Active matter on the Experiment 1 irregular mesh (65,632 nodes, 196,005 edges, node count matched to the grid): the mesh with the zoom region marked in red (left), and the orientation-tensor component D_{xx} at two times along a $\zeta = 9$, $\alpha = -3$ test trajectory, bilinearly resampled onto the mesh nodes. The concentration is nearly uniform ($1 \pm 5 \cdot 10^{-4}$), so a tensor channel is shown.



(a) mesh and zoom (b) $|\mathbf{v}|$, $t = 1$ (c) $|\mathbf{v}|$, $t = 5$

Figure 15: Cylinder flow: one of the per-trajectory meshes, with inflow, outflow and wall nodes marked in the zoom (left), and velocity magnitude at two times.

B.1 MESH REFINEMENT FOR CYLINDER FLOW

The native cylinder-flow meshes are far coarser than the grids on which Walrus was pretrained, so that a pretrained token window would contain only a handful of nodes. For Experiment 3 each mesh is therefore refined uniformly three times. Every refinement splits each triangle into four at its edge midpoints, and the value at a new midpoint is the mean of the values at the two end points of its edge, so the piecewise-linear field of the original mesh is reproduced exactly; no new information is added. New boundary midpoints inherit the inflow, outflow or wall type of their edge, and the cylinder remains the polygon of the original mesh. The median node count of the training meshes grows from 1,885 to 7,306, 28,756 and 114,088 over the three levels (104,184–125,224 nodes at level three). Splits are unchanged and normalisation statistics are recomputed per level. The first nodes of every refined mesh are the original nodes, in their original order and with their original values, so all predictions are scored on the original mesh nodes (Appendix D).

Table 4: Datasets summary. Trajectories are given as train/validation/test of the official splits; nodes and edges refer to the mesh representation (undirected edges): the lattice twin for TRL2D and shear flow, and the Experiment 1 irregular mesh for active matter, which has no lattice twin.

Dataset	Domain / resolution	Fields	Trajectories	Steps	Nodes	Edges
TRL2D	128×384 grid, $[-0.5, 0.5] \times [-1, 2]$	ρ, p, v_x, v_y	72/9/9	101	49,152	146,433
Shear flow	256×512 grid, $[0, 1]^2$ periodic	tracer, p, v_x, v_y	896/112/112	200	131,072	391,681
Active matter	256×256 grid, $[0, 10]^2$ periodic	c, v_x, v_y, D, E (11 ch.)	175/24/26	81	65,632	196,005
Cylinder flow	unstructured 1.6×0.41 channel	p, v_x, v_y	1000/100/100	600	1,732–2,059	$\approx 5{,}400$

C MODEL DETAILS

Every model in §4 is a complete autoregressive surrogate with its own processor. The baselines are run from their official or reference implementations, vendored with at most an import change, and are configured as in their publications wherever the data allow; the departures are listed below. Table 5 summarises the architectures and Table 6 the training settings. Parameter counts are for the complete model and were obtained by instantiating each configuration; they differ between datasets only through the number of input and output channels.

Table 5: Architectures of Experiments 1 and 2. Parameter counts are for the four-channel datasets (TRL2D, shear flow) and, in brackets, for the eleven-channel active-matter dataset. MNC and the grid source have identical counts, since the transplant replaces operations, not parameters.

Model	Implementation	Configuration	Parameters
MNC retrofit (Mukhopadhyay et al., 2026)	Overtone-family patch modulator	12 space–time blocks, width 384, 6 heads; patch 16 (two 4×4, stride-4 stages)	26,301,955 (same)
GINO (Li et al., 2023c)	`neuraloperator` 2.0.0, `GINO_Poisson2d` reference	FNO width 64, 4 layers, 20×20 modes; latent lattice of 64 cells on the short axis; GNO radius 1.5 cells	5,099,036 (5,122,381)
LNO (Wang & Wang, 2024)	official code	width 640, 8 blocks, 512 latent modes, 8 heads	29,901,316 (29,932,683)
Transolver++ (Luo et al., 2025)	official code	8 layers, width 256, 8 heads, 32 slices, MLP ratio 1	2,303,308 (2,326,611)
MeshGraphNets (Pfaff et al., 2021)	PyTorch re-implementation	15 message-passing steps, latent 128; 3-hidden-layer MLPs with LayerNorm	2,879,748 (2,886,027)

MNC retrofit (Experiments 1 and 2). The source is a patch-modulated forecasting transformer of the Overtone family (Mukhopadhyay et al., 2026): a convolutional encoder of two 4×4, stride-four

stages (channel widths $96 \to 384$, GroupNorm with 12 groups) lifts each frame to tokens of an overall patch size of 16, and a processor of 12 space–time blocks, each a full spatial attention with axial rotary position embedding followed by a causal axial attention in time, with query–key LayerNorm and a SwiGLU feed-forward of ratio four, evolves them; a mirrored transposed-convolution decoder returns field increments. Input and output use a fixed vocabulary of field embeddings, so the parameter count does not depend on the dataset. This gives 192 tokens on TRL2D, 256 on active matter and 512 on shear flow. The retrofit loads every tensor of the grid source by name. The four tokeniser kernels are converted into MNC taps (Proposition 1): the two encoder stages by the convolution indexing (10) and the two decoder stages, with the bias of the last one, by the transposed indexing (11). The conversion aborts if any tensor is neither copied nor converted. The mesh hierarchy inherits the source's tap spacings and window origin and must reproduce the source's token count. The rotary embedding of the spatial attention is evaluated at the anchor coordinates (6). Before any retrofit is used, a check on the source grid's own points confirms that the transplanted model reproduces the source to a relative error below 10^{-3}.

GINO. We use `neuralop.models.GINO` from `neuraloperator` 2.0.0 in the library's `GINO_Poisson2d` reference configuration: input GNO MLP $[256, 512, 256]$, output GNO MLP $[512, 1024, 512]$, a 16-channel positional embedding, linear GNO transforms with `half_cos` weighting, and an FNO with lifting and projection ratio four and group normalisation. Three reference keys not accepted by the 2.0.0 constructor are dropped. The latent lattice has 64 square cells along the short axis of the domain (64×192 on TRL2D, 64×64 on active matter), and the GNO radius is set to 1.5 latent cells instead of the reference value, which would produce an impractical number of edges on grids of $5 \cdot 10^4$–$1.3 \cdot 10^5$ nodes.

LNO. We use the official Latent Neural Operator code with vanilla attention, GELU activations and no time input, sized to the same parameter class as the MNC source (width 640, 8 blocks, 512 latent modes, 8 heads) rather than the smaller configurations of Wang & Wang (2024). Positions are normalised by the domain's x-extent.

Transolver++. We use the official Transolver++ code without modification except that its multi-GPU `all_reduce` calls are removed for single-GPU runs. The official repository releases only the million-scale model, so the size follows the configuration of the paper's standard benchmarks (8 layers, width 256, 8 heads, 32 slices, MLP ratio 1). As in the official code, the Gumbel noise on the slice weights is also drawn at evaluation, and node coordinates are concatenated to the input features.

MeshGraphNets. We use an established PyTorch re-implementation with the published processor size (15 message-passing steps, latent width 128). Its MLPs have one hidden layer more than DeepMind's (2.9M instead of 2.3M parameters) and use Xavier initialisation; the decoder is widened to the dataset's channel count. Edge features are the relative position and its length, standardised once per mesh. On the grid the model runs on a triangulated lattice with a fixed diagonal. To keep the comparison with the other models controlled, it receives the same per-sample-normalised stacked frames as they do, without node-type one-hot encoding, running normalisers or training noise.

Common interface of the baselines. All four baselines read the six context frames stacked as input channels, with the node coordinates, and predict the increment of the last frame; the MNC model is trained on the increments at all six causal positions. All models share the per-sample, area-weighted normalisation of fields and increments and the area-weighted MSE loss (Appendix D).

Large baselines are trained with gradient accumulation over equal micro-batches, so the effective batch is unchanged. By design, one learning rate is used for every architecture and stage; we did not tune it per architecture.

Walrus retrofit (Experiment 3). We use the released Walrus checkpoint (McCabe et al., 2026) (Hugging Face `polymathic-ai/walrus`, 1.3B parameters) with the unmodified upstream code. Walrus is an isotropic space–time transformer: 40 blocks of width 1408, each a full spatial attention with 16 heads and axial rotary embedding followed by an axial time attention, and a two-stage adaptive-stride convolutional encoder and decoder whose first stage selects the embedding columns of the fields present. Cylinder flow is presented in the format of the FlowBench flow-past-object data in Walrus's pretraining set: pressure and velocity (padded with a zero third component) on a 512×128 canvas over the 1.6×0.41 channel, with the cylinder as a constant mask field, open inflow and outflow boundaries, and walls at the top and bottom. MNC replaces only the outer-

Table 6: Training settings of Experiments 1 and 2, shared by all architectures. One "epoch" in Figure 6 is 100 mesh-training updates.

Setting	Value
Optimiser	AdamW, $\beta = (0.9, 0.95)$, weight decay 10^{-4}, gradient-norm clip 5
Learning rate	10^{-4} for every architecture and stage
Grid pretraining	50,000 updates at batch 16; 500 warm-up updates, cosine decay to 5%
Mesh training (retrofit and scratch)	20,000 updates at batch 8; 100 warm-up updates, constant rate
Seeds	three per mesh-training arm; retrofits start from the seed-0 source
Context / target	six frames / field increment
Regularisation	drop path 0.1 (MNC only); no dropout
Precision / hardware	fp32 (TF32 disabled); one NVIDIA GH200 (Isambard-AI) per run

most convolution on each side: the first encoder stage gathers directly from the mesh nodes, with lumped P1 mass weights, and the last decoder stage scatters back to them; the inner stages, normalisations and processor are unchanged, and the processor keeps Walrus's axial rotary embedding on the token grid. The MNC taps are Walrus's own kernels, so the retrofit adds no parameters. Normalisation is Walrus's samplewise RMS, area-weighted over the channel, and the model predicts the next increment from six frames. The meshes are uniformly refined three times, reproducing the piecewise-linear field exactly (median $1.1 \cdot 10^5$ nodes), and all scores are computed on the original mesh nodes (Appendix B.1).

Fine-tuning trains all parameters with AdamW (weight decay 10^{-4}), learning rate 10^{-5} after 20 warm-up updates, gradient-norm clip 10 and batch eight (one window per micro-batch, with activation recomputation), in fp32 with TF32 matrix products on one GH200, for a 7.5-hour budget (about 1.4k updates). The random-initialisation arm uses the same architecture and recipe from Walrus's own initialisation. The cylinder baselines use the implementations and sizes of Table 5, trained from scratch at mesh level three on the same windows, normalisation, optimiser and time budget, with learning rates 10^{-3} and 10^{-4} tried and the better one reported (10^{-3} for MeshGraphNets, Transolver++ and GINO; 10^{-4} for LNO). Being non-causal, they are trained on the last increment only.

D Forecasting protocol

This appendix records how the autoregressive experiments of §4 are trained and evaluated. The architectures and optimiser settings are given in Appendix C; the tokeniser reconstruction comparison in Appendix F uses a different protocol, detailed there. All experiments were performed on NVIDIA GH200 GPUs with 96 GB of memory, one GPU per run.

D.1 Splits and windows

TRL2D and active matter use the official trajectory splits of The Well (72/9/9 and 175/24/26 trajectories). For shear flow we resplit the official training files at the trajectory level, stratified by Reynolds–Schmidt pair: with seed zero, 26/3/3 trajectories of each pair go to training, validation and test, giving 727/84/84 trajectories. Before splitting, trajectories are compared on all fields at frames zero and ten, and the one exact duplicate found is removed. The official shear-flow validation and test files are not used. All sources are trained at the native grid resolution, and every stage uses the full training split.

A sample is a window of six consecutive context frames and the following target frame. Every start time with stride one defines a window, giving 95, 75 and 194 windows per trajectory for TRL2D, active matter and shear flow. Training passes over all windows in a fresh random order each pass, seeded by the run seed. The test windows are drawn once with a fixed seed that does not depend on the run, so every model of a dataset is evaluated on the same windows. Intermediate evaluations use a fixed subset of 128 test windows. The final evaluation uses all 855 TRL2D test windows and 1,024 windows each for active matter and shear flow; these contain the 128.

D.2 Normalisation and training objective

Each window is normalised with its own statistics. Per channel, the mean and standard deviation of the six context frames and, separately, of their five consecutive increments are computed with spatial area weights and averaged over time. The model receives the normalised context and predicts normalised increments, which are mapped back to physical fields by adding the denormalised increment to the last context frame. The loss is the area-weighted mean-squared error of the normalised increments. For the MNC model it is taken at all six causal positions; the baselines, which are not causal in time, predict only the increment from the last context frame to the target.

The spatial weights are the node areas: uniform cell areas on the grid and, on the irregular mesh, lumped P1 masses (one third of the area of each adjacent triangle). They enter the statistics, the loss and the metrics, so that the density of nodes in a region does not change its weight. They are not inserted into the MNC aggregation, whose normalisation remains (4); this distinction matters for interpreting Proposition 2.

D.3 Meshes and field resampling

The irregular mesh of each dataset is a Gmsh Delaunay triangulation of the grid's bounding rectangle at one characteristic length, without physics-based grading, with boundary nodes on the rectangle's edges and the node count matched to the grid (within 0.5% for all three datasets; Appendix B). Field values at its nodes are bilinear interpolants of the simulation arrays, computed from cached weights. The targets are therefore samples of the same grid data, not solutions recomputed on a different discretisation.

The physical window origin and tap spacings of the MNC hierarchy are inherited from the source grid and held fixed, and source and target hierarchies have the same token counts: 192 for TRL2D, 256 for active matter and 512 for shear flow. Windows are not wrapped across periodic boundaries: at the edge of the domain they contain only the samples inside it.Neighbour lists, relative offsets, spline weights and occupancies depend only on the geometry and are cached. Only the node locations, not the Delaunay connectivity, determine the MNC messages; the connectivity is used by MeshGraphNets and for the area weights.

D.4 Metrics

All metrics are computed on denormalised physical fields. Let w_j be the node area weights defined above, normalised to sum to one. For a predicted frame $\hat{u}$ with truth u and a channel c, the variance-normalised RMSE (Ohana et al., 2024) is

$$\mathrm{VRMSE}_c(\hat{u}, u) = \Big(\frac{\sum_j w_j \, (\hat{u}_{j,c} - u_{j,c})^2}{\sum_j w_j \, (u_{j,c} - \bar{u}_c)^2 + 10^{-6}} \Big)^{1/2}, \qquad \bar{u}_c = \sum_j w_j \, u_{j,c}, \tag{17}$$

the area-weighted RMSE divided by the area-weighted spatial standard deviation of the truth in that frame. Scores are averaged over all C channels of a dataset, including the tensor fields of active matter: $\mathrm{VRMSE}(\hat{u}, u) = \frac{1}{C} \sum_{c=1}^{C} \mathrm{VRMSE}_c(\hat{u}, u)$.

Next-step VRMSE. For a test window w with context ending at frame t_w, the model predicts $\hat{u}_{t_w+1}$ from the true context, and the window score is $e_w = \mathrm{VRMSE}(\hat{u}_{t_w+1}, u_{t_w+1})$.

Rollout VRMSE. A rollout starts from a true six-frame context ending at frame t_s and predicts $K = 10$ frames autoregressively: each prediction $\hat{u}^{(s)}_{t_s+k}$ is appended to the context, the oldest frame is dropped and the normalisation statistics are recomputed before the next call, so no ground truth enters after the initial context. The rollout score is the channel-mean VRMSE averaged over the ten steps,

$$r_s = \frac{1}{K} \sum_{k=1}^{K} \mathrm{VRMSE}\big(\hat{u}^{(s)}_{t_s+k}, \, u_{t_s+k}\big). \tag{18}$$

Rollouts start at five evenly spaced times in every test trajectory.

Aggregation. For the tables, window and rollout scores are first averaged within each test trajectory τ, giving $\bar{e}_\tau$ over its test windows and $\bar{r}_\tau$ over its five starts, and the reported next-step and rollout

VRMSE are the medians of $\bar{e}_\tau$ and $\bar{r}_\tau$ over test trajectories. The training curves of Figure 6 use the same aggregation, restricted to the 128 fixed test windows at each evaluation. Transolver++ draws Gumbel noise in its slice weights at evaluation, so its scores are single draws.

D.5 Evaluation schedule and reported checkpoints

Sources are evaluated every 5,000 of their 50,000 updates and mesh-training runs every 1,000 of their 20,000 updates (21 points); rollouts are run at 0, 25k and 50k source updates and at 0, 5k, 10k and 20k mesh updates. All reported numbers are from the final checkpoint of a run; the validation split is monitored but not used for model or checkpoint selection. The zero-shot scores of Experiment 1 are the evaluation at step zero of the retrofit run, that is, the grid-pretrained source transplanted onto the irregular mesh before any mesh update, evaluated on the full set of test windows and rollouts. For MNC the transplant must first pass the exactness check on the source grid (Appendix C); the baselines load their grid checkpoints directly, since their encoders act on point sets. The spread in Table 1 is over grid sources pretrained with three seeds. The mesh-training runs of Experiment 2 all start from the seed-0 source, and their spread is over three mesh-training seeds.

D.6 Experiment 3: cylinder flow

Experiment 3 is scored on the original MeshGraphNets meshes, although the models run on meshes refined three times: the first nodes of each refined mesh are the original ones, in order and with their original values, so all levels are compared on the same nodes. Next-step VRMSE uses 16 random start times in each of the first ten test trajectories (160 windows), and ten-step rollouts use 16 other start times per trajectory. Here VRMSE is not area-weighted: per channel (pressure and the two velocity components) it is the RMSE over the original nodes divided by the spatial standard deviation of the truth, averaged over channels, with the median over windows reported.

E Additional numerical results

Figure 16 shows the one-step density error of every model on a single TRL2D test window on the irregular mesh (test trajectory 7, frame 79 predicted from frames 73–78), for the four starting points of Experiments 1 and 2: the grid-pretrained model zero-shot and after fine-tuning on the mesh, and the same architecture trained on the mesh from scratch, at initialisation and after training. The retrofitted Overtone+MNC model already resolves the mixing layer zero-shot, with a fainter error field than any baseline at any stage, and fine-tuning reduces its error further; trained from scratch, the same architecture ends slightly above the fine-tuned retrofit. Predicting no change (persistence) scores 0.308 on this window.

F Tokeniser and autoencoding validation

In this appendix we isolate the encoder-decoder through snapshot reconstruction, comparing MNC with the tokenisers of GINO (Li et al., 2023c) and LNO (Wang & Wang, 2024), without their latent processors. CNN and MNC have about 17k parameters, GINO about 22k, and LNO about 21k with 32 latent modes. CNN, MNC and GINO have matched bottleneck sizes; LNO's smaller bottleneck is examined separately in Appendix G.2. All models reconstruct the latest frame of a six-frame context, using samplewise RMS normalisation, MSE and Adam at 10^{-3}, for 100 epochs of 200 windows. These normalised reconstruction errors are distinct from the physical-field forecasting metric of Appendix D.4.

Table 7 compares training from random initialisation on two Well datasets (Ohana et al., 2024) and native cylinder-flow meshes (Pfaff et al., 2021). On the Well grid twins all models see identical samples: MNC matches the CNN, while the other tokenisers have higher errors under this protocol. On cylinder flow MNC has the lowest reconstruction error, without requiring a grid representation. Architecture and timing details are in Appendix G.

MNC starts near 10^{-2} normalised MSE and reaches about 1.5×10^{-3} after one mesh-training epoch, close to its scratch-trained error after 100 epochs (Figure 17). GINO and LNO also benefit

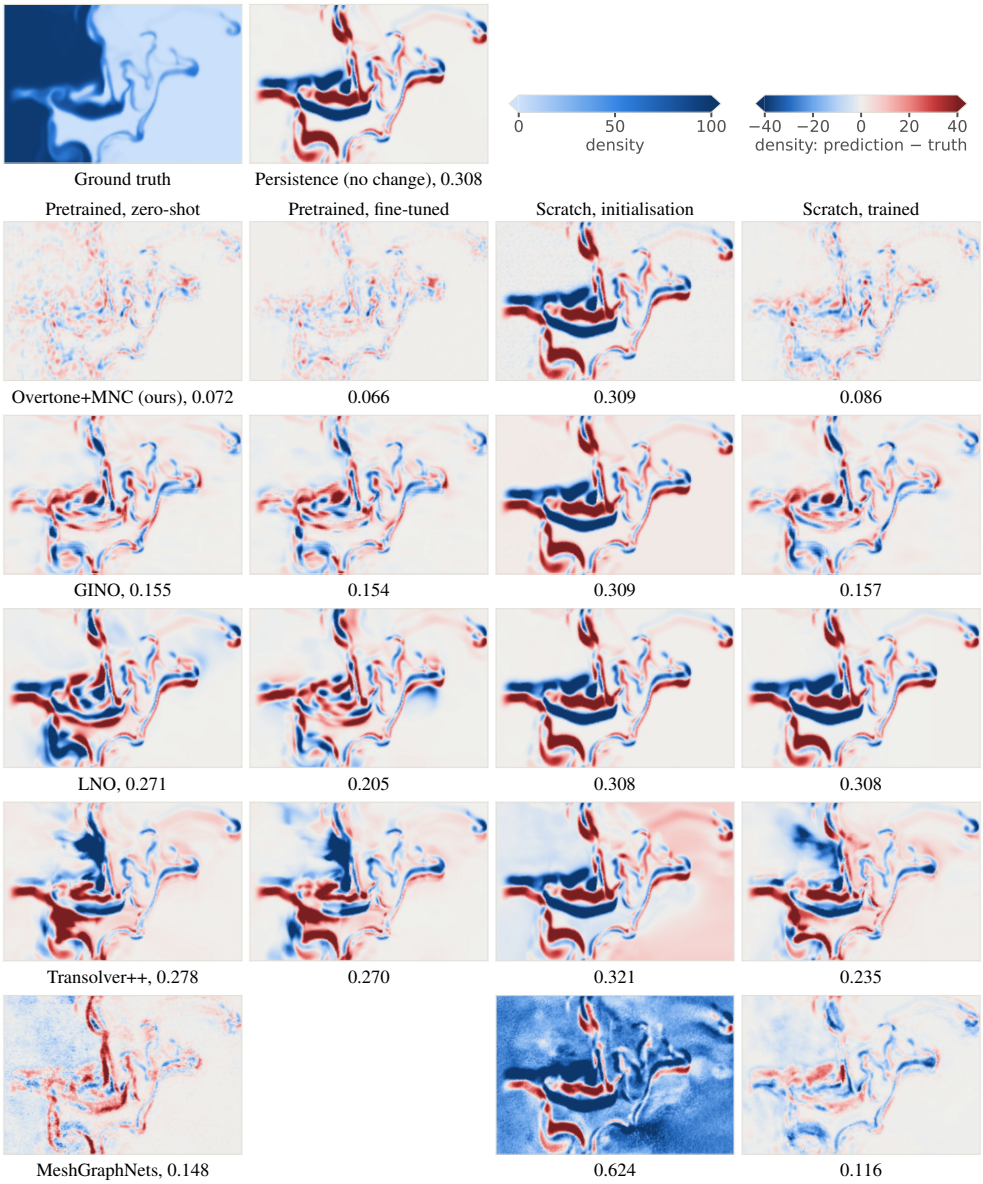


Figure 16: One-step density error (prediction − truth) on the irregular TRL2D mesh, test trajectory 7, frame 79: grid-pretrained models zero-shot and after fine-tuning, and the same architectures trained from scratch, at initialisation and after 20k updates. Numbers are the density VRMSE of this window. The fine-tuned MeshGraphNets panel was not exported for this window.

Table 7: Tokeniser reconstruction from scratch after 100 epochs: test normalised MSE, mean ± standard deviation over three seeds. The Well grid twins use identical samples; cylinder flow uses native meshes. LNO uses 32 modes; latent-parity results are in Appendix G.2.

Tokeniser	TRL2D	Shear flow	Cylinder flow
CNN (grid)	$\mathbf{(1.62 \pm 0.16) \times 10^{-3}}$	$(1.18 \pm 0.03) \times 10^{-3}$	—
MNC (ours)	$(1.68 \pm 0.06) \times 10^{-3}$	$\mathbf{(1.15 \pm 0.08) \times 10^{-3}}$	$\mathbf{(2.58 \pm 0.01) \times 10^{-2}}$
GINO	$(1.79 \pm 0.01) \times 10^{-2}$	$(5.25 \pm 0.01) \times 10^{-2}$	$(2.29 \pm 0.10) \times 10^{-1}$
LNO (32 modes)	$(9.93 \pm 0.02) \times 10^{-2}$	$(7.37 \pm 0.74) \times 10^{-2}$	$(9.12 \pm 0.64) \times 10^{-2}$

Grid-to-mesh transfer. We pretrain CNN, GINO and LNO on the 256×512 shear-flow grid for 100 epochs. CNN weights are loaded into MNC; GINO and LNO load their own pretrained weights, retaining the target geometry. Each is then fine-tuned on the full training split, bilinearly resampled onto an irregular triangulation at matched resolution. Dashed curves use the same target architectures, data and mesh-training budgets, but start from random weights.

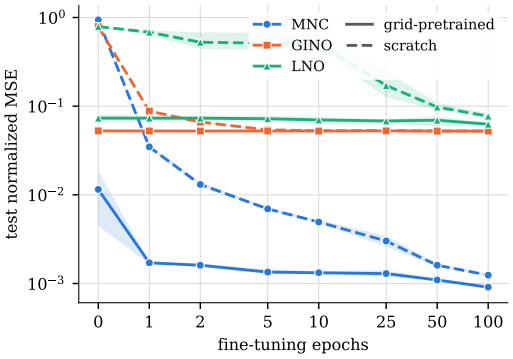


Figure 17: Shear-flow autoencoding after a change of discretisation. Solid curves: CNN→MNC, GINO→GINO and LNO→LNO transfer. Dashed curves: the corresponding models trained from scratch. Bands show variation across three seeds. Epoch zero is before any mesh training; source pretraining is additional to the horizontal-axis budget.

from pretraining at initialisation, but remain at substantially higher errors and approach their scratch-trained levels as training proceeds. Thus the result is not that other tokenisers cannot transfer: the CNN→MNC construction preserves a more accurate field representation and makes it available after far less mesh training. Figure 1c shows the corresponding zero-shot reconstructions. This compares the usefulness of the trained tokenisers, not remeshing penalties at matched source accuracy. Lattice and cross-domain controls are in Appendix G.4.

G TOKENISER RECONSTRUCTION AND TRANSFER: DETAILS AND CONTROLS

This appendix describes the processor-free experiments in Appendix F. They compare both reconstruction accuracy and grid-to-mesh transfer across tokeniser families. The complete GINO and LNO models of the forecasting experiments are much larger and have their own processors (Appendix C); their results should not be conflated with those reported here.

G.1 DATA, ARCHITECTURE AND TRAINING

CNN and MNC. A pointwise projection maps the input to 16 channels, followed by a 4×4/stride-four layer and a 2×2/stride-two layer to 64 channels, the mirrored decoder, and a pointwise output projection. GroupNorm and GELU are applied between stages. There are no skip connections, and only the output projection has a bias. Parameter counts are 16,805 for the five-channel interface (the four Well fields plus a zero channel) and 16,739 for the three-channel cylinder example. The overall reduction factor is eight, rather than 16 in forecasting.

Protocol. Only the latest frame is reconstructed; six context frames supply samplewise, per-channel RMS normalisation. Training uses MSE in these normalised variables, Adam at 10^{-3}, and 100 epochs of 200 sampled windows at batch size one, without warm-up or a learning-rate schedule. All three tokeniser families use this protocol for source pretraining and mesh fine-tuning. Reconstruction MSE is not the physical-field VRMSE of §4.

Grid twins and native meshes. The Well controls use identical field samples for CNN, MNC and the baselines. TRL2D has 128×384 samples; shear flow has 256×512. The shear reconstruction experiments retain the original dataset split, not the forecasting resplit. The MeshGraphNets cylinder-flow dataset (Pfaff et al., 2021) has a separate native mesh per trajectory, approximately 1.9k nodes, and three fields: pressure and two velocities (Figure 15, Appendix B). We use its 1,000/100/100 training/validation/test split and evaluate 500 sampled test windows. MNC builds a hierarchy for each geometry; in this dataset its two levels typically contain about 132 and 33 anchors. This tests reconstruction on a domain containing an obstacle, not grid-pretrained cylinder forecasting.

G.2 GINO AND LNO TOKENISER CONFIGURATIONS

GINO. We use the `neuraloperator` implementation with its FNO processor removed (`fno_n_layers=0`): input GNO, pointwise lifting, output GNO and projection. Kernel MLPs act on relative coordinates with a 16-channel sinusoidal embedding; the latent width is 64. There are 21,674 parameters with five fields and 21,350 with three. The regular latent grid matches the CNN/MNC bottleneck: 16×48 for TRL2D, 32×64 for shear flow and 3×11 for cylinder flow, giving 49,152, 131,072 and 2,112 latent scalars. The input radius is five mesh cells and the output radius one latent cell. These are broader, overlapping windows, rather than exactly the CNN's effective footprint. Kernel weights do not depend on the latent-grid size, so the same model can be evaluated with target geometry after pretraining.

LNO. We use the cross-attention encoder and decoder with the latent processor removed (`n_block=0`). Position and field MLPs produce 48-channel embeddings. Position-dependent scores are normalised over input nodes to encode each latent mode and over modes to decode at each query point. The main comparison and all transfer experiments use 32 modes, or 1,536 latent scalars, with approximately 21k parameters. Unlike GINO and MNC, this is a smaller bottleneck rather than a latent-size-matched control. Increasing the mode count also changes the parameters and cost; Table 8 reports the available matched-bottleneck ablation.

Table 8: LNO mode-count ablation: test normalised MSE after 100 epochs of direct training (run means). The 32-mode and MNC rows repeat Table 7 for reference.

Tokeniser	TRL2D	Shear flow
LNO (32 modes)	9.93×10^{-2}	7.37×10^{-2}
LNO (latent parity)	9.84×10^{-2}	4.17×10^{-2}
MNC (ours)	$\mathbf{1.68 \times 10^{-3}}$	$\mathbf{1.15 \times 10^{-3}}$

The latent-parity variants use 1,024 modes on TRL2D and 2,730 on shear flow (49,152 and 131,040 latent scalars, respectively). They have 69,669 and 153,263 parameters. The larger bottleneck improves shear reconstruction, but does not close the gap to MNC under the reported protocol. This is a direct-training control, not an additional latent-parity transfer experiment.

G.3 EXECUTION COST

Table 9: Mean training time per epoch in seconds for the reconstruction study (each model's summed per-step time on the same single NVIDIA GH200; mean $\pm$ standard deviation over three seeds).

Tokeniser	TRL2D	Shear flow	Cylinder flow
CNN (grid)	$\mathbf{0.55 \pm 0.04}$	$\mathbf{0.61 \pm 0.03}$	—
MNC (ours)	3.28 ± 0.19	3.46 ± 0.10	5.80 ± 0.18
GINO	1.67 ± 0.02	3.89 ± 0.09	1.32 ± 0.05
LNO (32 modes)	1.85 ± 0.00	4.45 ± 0.01	$\mathbf{0.81 \pm 0.02}$

The timings in Table 9 count training steps on one NVIDIA GH200 and exclude one-off geometry construction. MNC is approximately six times slower than dense CNN execution on these grid twins, and it is not uniformly faster than the other mesh tokenisers. The argument for transfer is the preservation of a useful representation and a reduction in target-mesh training, not intrinsically cheaper sparse layers. These measurements are not timings of the larger forecasting models.

G.4 GRID-TO-MESH TRANSFER AND COMPANION LATTICE CONTROL

For Figure 17, CNN, GINO and LNO are each pretrained for 100 epochs on the shear-flow grid. MNC loads the CNN's projections, convolutional and transposed-convolutional weights, GroupNorm affine parameters and output bias. No MNC parameters are newly initialised in this within-dataset transplant. The source's tap spacings and window origin are retained. GINO and LNO load their own weights and replace only the geometry buffers with the target mesh geometry. Every

model is fine-tuned on the full mesh training split; scratch controls use the same target architecture and target-training protocol.

The irregular target and the uniform triangular lattice in Figure 13 have matched resolution, but their points do not coincide with the source-grid samples. Field values are obtained by bilinear resampling. Figure 18 repeats the comparison on the uniform lattice: the relative accuracy and fast initial recovery of transferred MNC persist. Neither comparison is an exactness test on identical inputs, because the sample positions have changed.

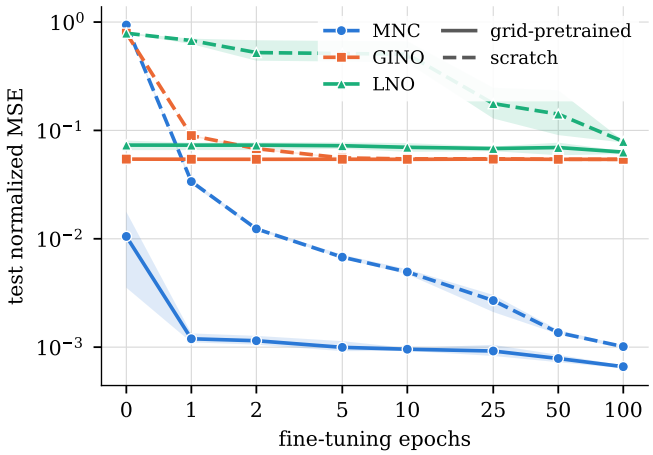


Figure 18: Companion to Figure 17 on the uniform triangular lattice. Solid curves use grid-pretrained weights; dashed curves start from random initialisation. Bands show variation across three seeds. Pretraining is not included in the horizontal-axis mesh-training budget.

Figure 1c metadata. The displayed tracer reconstruction uses window 213 and source seed zero, pretrained for 100 epochs, with no mesh fine-tuning. The normalised MSE values are 0.0113849 for CNN→MNC, 0.0520004 for GINO→GINO and 0.0353429 for LNO→LNO. They are specific to this channel and window and must not be read as the three-seed dataset means in Figure 17. The same example has scratch-initialisation errors 0.8612, 0.7469 and 0.7509, respectively: pretraining helps all three models at initialisation, not MNC alone.

G.5 CROSS-DOMAIN TRANSFER TO NATIVE CYLINDER MESHES

The original reconstruction study also transfers from TRL2D, shear flow and their mixture to cylinder flow. Channels are matched by name, and the convolutional stencil is re-instantiated at the target resolution using a characteristic mesh spacing. This differs from within-domain transfer, where the source's physical window is retained. Fine-tuning uses all 1,000 training trajectories, with no warm-up.

Table 10: Cross-domain *reconstruction* transfer to native cylinder-flow meshes: normalised MSE after e mesh-training epochs (mean $\pm$ standard deviation, three seeds). $e = 0$ is zero-shot; "scratch" has no pretraining. The source is CNN for MNC, and each baseline's own tokeniser for GINO and LNO. These are not forecasting results.

Tokeniser	e	scratch	TRL2D	Shear flow	TRL2D+Shear
MNC (ours)	0	1.116 ± 0.083	1.086 ± 0.161	0.999 ± 0.413	0.556 ± 0.034
	25	0.042 ± 0.001	0.039 ± 0.000	0.039 ± 0.002	0.037 ± 0.002
	50	0.035 ± 0.001	0.031 ± 0.001	0.031 ± 0.001	0.030 ± 0.001
	100	0.026 ± 0.000	0.024 ± 0.001	0.025 ± 0.001	0.023 ± 0.000
GINO	0	0.993 ± 0.034	0.576 ± 0.013	0.528 ± 0.008	0.547 ± 0.008
	25	0.373 ± 0.024	0.312 ± 0.014	0.362 ± 0.050	0.318 ± 0.017
	50	0.281 ± 0.017	0.261 ± 0.008	0.263 ± 0.010	0.270 ± 0.017
	100	0.228 ± 0.011	0.233 ± 0.006	0.228 ± 0.003	0.237 ± 0.013
LNO	0	0.969 ± 0.038	1.237 ± 0.734	0.674 ± 0.017	0.661 ± 0.035
	25	0.565 ± 0.002	0.327 ± 0.193	0.109 ± 0.007	0.103 ± 0.006
	50	0.408 ± 0.182	0.109 ± 0.006	0.088 ± 0.010	0.090 ± 0.008
	100	0.091 ± 0.006	0.080 ± 0.007	0.074 ± 0.012	0.071 ± 0.003

These controls limit the claim that MNC transfers best in every sense. GINO has lower zero-shot errors from either individual source; mixed-source pretraining substantially improves MNC's initialisation. After 100 target epochs, MNC has the lowest error, but the benefit of its pretraining over its own scratch run is modest (0.023–0.025 versus 0.026). LNO also benefits from pretraining here. The strong within-physics, changed-discretisation result in Figure 17 should therefore not be generalised to arbitrary cross-physics initialisation or to forecasting.