ETH zürich

Final Project

Course: AI in the Sciences and Engineering

Author: Maja Gwóźdź

Date: February 1, Autumn Semester

Contents

Task 1.1

We sampled $a_{ij} \sim \mathcal{N}(0,1)$ and evaluated the closed-form series with $r = 0.5$ on an $N = 64$ grid (with $x, y \in [0,1]$). For each $K \in \{1,4,8,16\}$, we created $S = 100$ samples (with seed=0), and we stored f, u, with the coefficients a_{ij} in `.npz` files. In Figure 1, we show three example pairs (f,u) per K.

As regards implementation, we work with the sine bases $\sin(\pi ix)$ and $\sin(\pi jy)$. We use the spectral weights: $(i^2+j^2)^r$ and $(i^2+j^2)^{r-1}$. We then create f and u by applying an `einsum` contraction over i, j. We apply constants for scaling: $s_f = \pi/K^2$ and $s_u = 1/(\pi K^2)$. We then obtain `.npz` arrays that include `x` and `y` (length 64), `f` and `u` (shape $S \times 64 \times 64$, float32), `a` (shape $S \times K \times K$), and scalars `r` and `K`. For the 100 samples, we measured RMS magnitudes (that is, over all samples and grid points) of $f_{\text{rms}} \approx \{2.11, 1.49, 1.39, 1.32\}$ and $u_{\text{rms}} \approx \{1.07\text{e}{-1}, 1.33\text{e}{-2}, 4.02\text{e}{-3}, 1.18\text{e}{-3}\}$ for $K = \{1,4,8,16\}$. This clearly shows the $1/K^2$ scaling and also for larger K, the higher-frequency details.

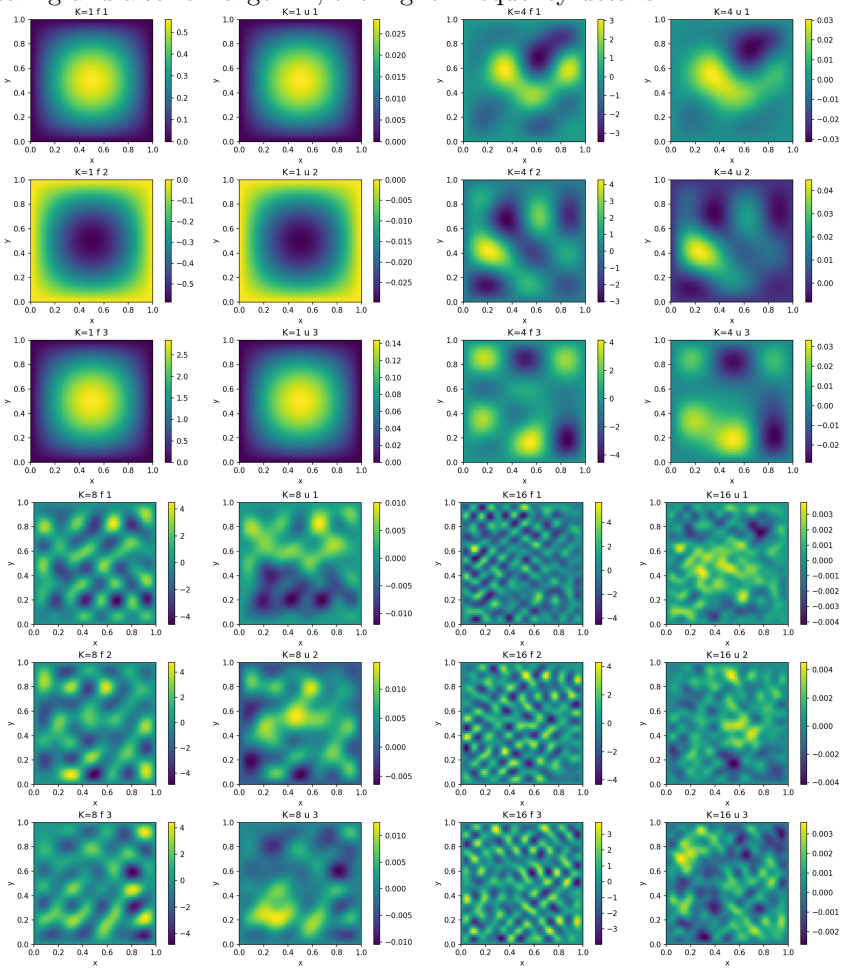

Figure 1: Task 1 examples: three samples per K (rows) with f (left) and u (right). Top: $K = 1, 4$. Bottom: $K = 8, 16$.

Task 1.2

We trained a Tanh MLP (4 layers, width 128, Xavier initialisation) on a fixed sample (sample_idx=0) for $K \in \{1,4,16\}$. For the loss, we apply MSE over the N^2 grid points. More precisely, the PINN loss has the interior residual $-\Delta u - f$ and a boundary loss, so that we have $u = 0$ on ∂D. We use Adam for optimisation (5000 steps, lr=10^{-3}), and then apply L-BFGS (500 steps, strong_wolfe). We report relative L_2 errors (for the full grid). In Table 1, we show final errors. In Figure 2, we present the predictions and loss curves.

Method	$K = 1$	$K = 4$	$K = 16$
Data-driven	0.1202	0.5803	0.9406
PINN	0.01524	0.1531	11.01

Table 1: Task 2: final relative L_2 error $\|\hat{u} - u\|_2 / \|u\|_2$.

Metric	$K = 1$	$K = 4$	$K = 16$
Data loss (MSE)	2.801e-6	4.473e-5	1.337e-6
PINN L_f (interior)	1.586e-6	9.561e-5	3.163e-2
PINN L_{BC} (boundary)	1.490e-7	1.268e-5	6.869e-4

Table 2: Task 2: Final training loss details.

We note that the grid has $N^2 = 4096$ points (interior 3844, boundary 252). We analyse the PINN residual on the interior mask, and calculate the boundary loss over ∂D. We compute errors on the full grid. We do not apply any subsampling (so we use all boundary/interior points in the loss evaluations). To be consistent, we always use the same sampling index (0)

for all K. The MLP has 4 hidden layers with width 128 and a scalar output. In total, this makes 50,049 parameters. We use the same architecture and initialisation for all models. Note that only the loss definition (that is, data or PINN) and the field f are different. From Tables 1–2, we observe that the PINN interior residual increases with K.

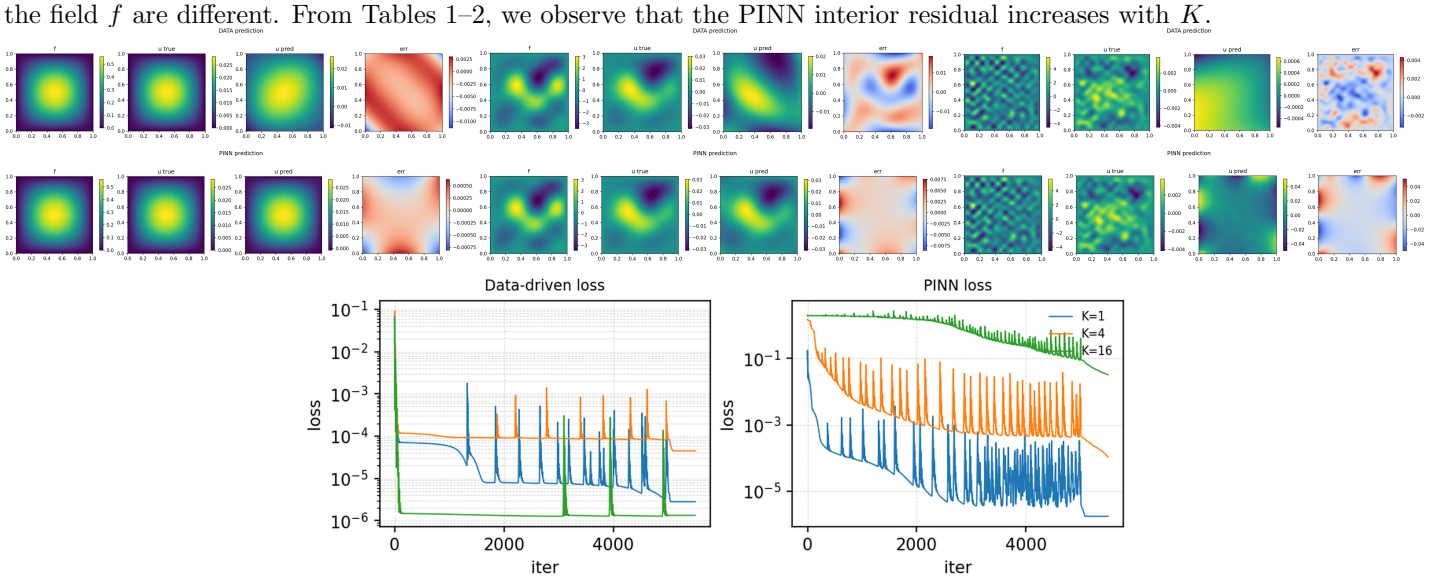

Figure 2: Task 2 predictions (top) and training loss curves (bottom), $K = 1, 4, 16$. Each prediction shows f, u_{true}, u_{pred}, and the error.

Task 3

We create snapshots for parameters at initialisation and also every 50 (Adam) steps. We also create a final snapshot after the application of L-BFGS. We then extract the top 2 directions from the PCA (in weight space), and evaluate the loss on a 41×41 grid with $\alpha, \beta \in [-1, 1]$. We apply contours (log scale) and normalisation of directions for each layer and filter (as in the paper reference for this task). In Figure 3, we present the landscapes for both models (data and PINN).

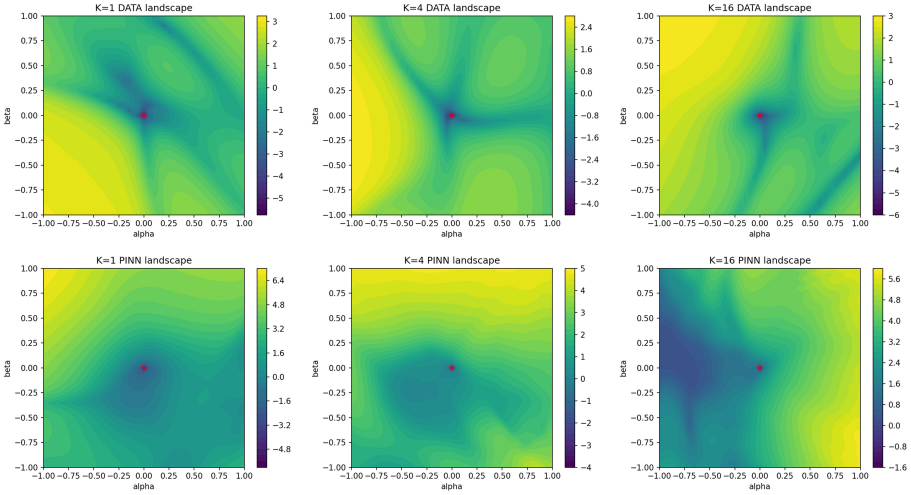

Figure 3: Task 3: log-scaled loss contours for data-driven (top) and PINN (bottom), $K = 1, 4, 16$.

We evaluate the losses on a 41×41 grid with $\alpha, \beta \in [-1, 1]$ (around the weights that converged). As before, we use the two top PCA components of the training trajectory (102 snapshots with 50,049 parameters) as the directions. We apply the filter to normalise them. For this report, we plot the losses in $\log_{10}$ scale. We use the full-grid MSE for the model that is data driven, and for the PINN, we use the same (interior) residual and the boundary term as in training. We have 3844 points for the interior and 252 points for the boundary. As above, we do not apply subsampling.

For the 41×41 grids, the data-driven landscapes are in the range of approximately 10^{-6}–10^{3} for the loss, and the PINN values lie in 10^{-6}–10^{7} (K=1) and 3×10^{-2}–8×10^{5} (K=16). This implies that the PINN case is more dynamic for larger K. In particular, as K gets larger, the data-driven model has smoother basins, and the PINN case is more irregular and "sharper". This is also in line with the results from the Krishnapriyan et al. (2021) paper. More precisely, we observe spectral bias for lower frequencies, stiffness, worse conditioning, and sharper loss shapes.

Task 2.1

We kept the snapshots for $t = 0$ and $t = 1$. We trained a 1D FNO with input channels $[u_0, x]$ (in_channels=2), modes=16, width=64, depth=4, and GELU activations. An rFFT spectral convolution (that is truncated to 16 modes) is applied in each of the 4 spectral blocks. The blocks also apply a 1×1 conv residual. We trained with Adam (lr=10^{-3}, weight decay 0), batch size of 32, 500 epochs, and seed of 0. We provide the test average relative L_2 error in Table 1 (the metric is the mean $\|\hat{u} - u\|_2 / \|u\|_2$ over 128 trajectories).

Metric	Value
Test relative L_2 (t=1.0)	0.001846

Table 1: Task 2.1: test error on `data_test_128.npy` at $t = 1.0$.

We use train/val/test sizes of 1024/32/128 trajectories. We provide here test metrics from the checkpoint that achieved the lowest validation relative L_2 (best val = 0.002402). We measure test MSE at $t = 1.0$ as 6.686×10^{-7}. The best validation relative L_2 for Task 1 is at epoch 445. We use this checkpoint for the test metrics. For all results, we compute the relative L_2 per trajectory as $\|\hat{u} - u\|_2 / \|u\|_2$ over the grid. We then average over the test trajectories. We calculate MSE as the mean squared error over all points, and average across trajectories for the split. The convolution uses the channel dimension (width 64) and complex weights that have shape $(64, 64, 16)$. For the forward pass, we apply rFFT→truncate→linear mixing→iFFT. Finally, we map the 2 input channels to width 64 by a linear layer. We then use 2 linear layers to map width 64 to the final scalar output.

Task 2.2

We use the model from the Task 1 and evaluate on `data_test_{s}.npy` with $s \in \{32, 64, 96, 128\}$ at $t = 1.0$. We use the same metric definition and no interpolation.

Resolution	32	64	96	128
Test relative L_2	0.08434	0.02665	0.008639	0.001846

Table 2: Task 2.2: resolution generalisation at $t = 1.0$.

Resolution	32	64	96	128
Test MSE	2.170e-3	1.516e-4	2.061e-5	6.686e-7

Table 3: Task 2.2: test MSE at $t = 1.0$.

We observe that the error increases as the resolution decreases, from which we infer that the 128-trained model generalises best to the resolution. However, for coarser grids, it begins to degrade in a smooth way. We obtain the following test MSEs at $t = 1.0$: 2.170×10^{-3} (32), 1.516×10^{-4} (64), 2.061×10^{-5} (96), and 6.686×10^{-7} (128). We create, for each resolution, the input coordinate channel x as a uniform space on $[0, 1]$ at that resolution. We then evaluate the model on the grid points.

Task 2.3

We use 5 snapshots (10 pairs per trajectory and 10240 training pairs) to create all ($t_{in} < t_{out}$) pairs. For the model input, we have $[u_{in}, x, t_{out}]$ (in_channels=3) with time normalisation (InstanceNorm + FiLM from t_{out}). In the time signal, we apply t_{out} (not Δt) along the grid. We reuse the training settings from Task 1. For evaluation, we fix $t_{in} = 0$ and predict $u(t_{out})$ for $t_{out} \in \{0.25, 0.5, 0.75, 1.0\}$. The test relative L_2 errors are as follows:
We measure test MSEs of 3.658×10^{-5} (t=0.25), 1.252×10^{-4} (t=0.5), 1.573×10^{-4} (t=0.75), and 1.731×10^{-4} (t=1.0). We note that the best validation relative L_2 during all2all training was 0.03951. We ran Tasks 1 and 3 for 500 epochs, used 300 epochs for fine-tuning, and 500 epochs for the scratch subtask (see below). The best validation relative L_2 for the all2all model was at epoch 492. We also use this checkpoint for all reported Task 3 test metrics. The all2all pairing gives $5 \cdot 4/2 = 10$ pairs per trajectory, with 5 time snapshots. We apply the same pairing method on 32 validation trajectories (320 pairs in total). For evaluation, we fix $t_{in} = 0$ and predict all t_{out} values from the initial condition. Finally, we note that the error increases with the time. The $t = 1.0$ error (0.02669) is greater than Task 1 (0.001846). This shows that the all2all setup and the conditioning on time are harder.

Time t	0.25	0.5	0.75	1.0
Test relative L_2	0.01247	0.02003	0.02324	0.02669

Table 4: Task 2.3: test errors on `data_test_128.npy`.

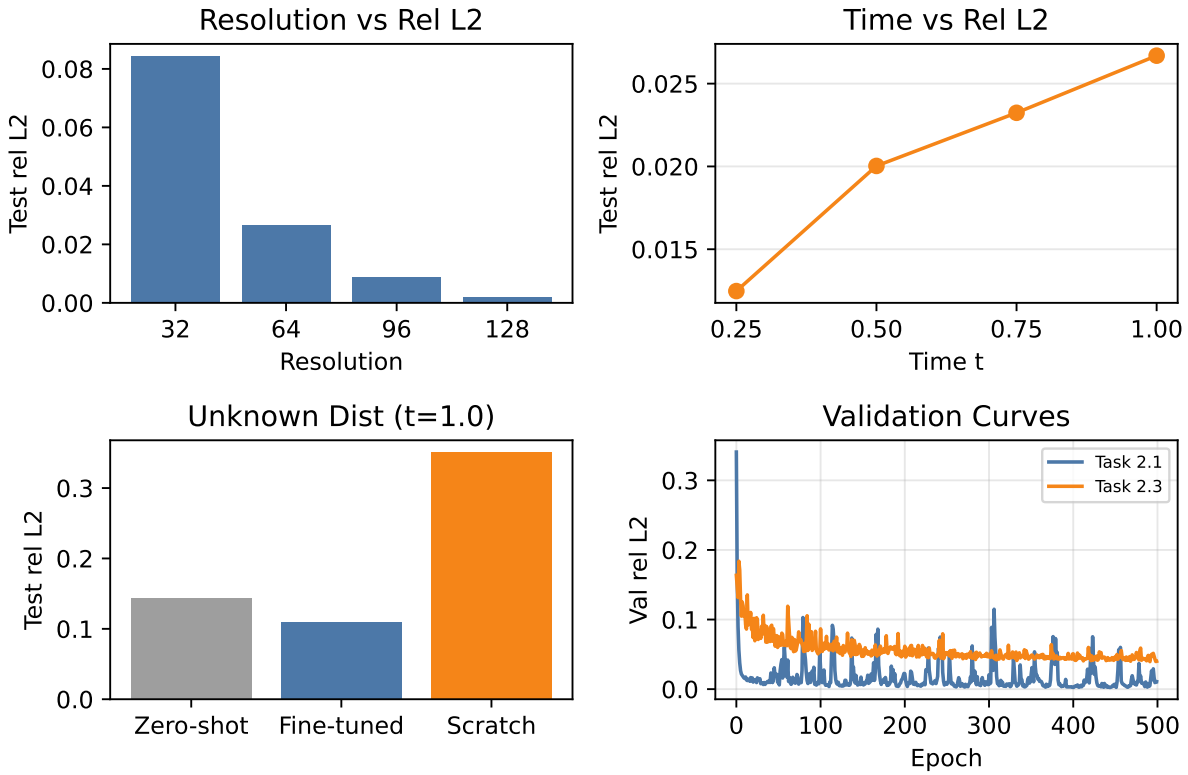

Figure 1: Top-left: Task 2.2 relative L_2 error *vs* resolution. Top-right: Task 2.3 relative L_2 error *vs* time. Bottom-left: Task 2.4 relative L_2 error at $t = 1.0$. Bottom-right: validation curves for Tasks 2.1 and 2.3.

Task 2.4

We use zero-shot evaluation (`data_test_unknown_128.npy`) for the model from Task 3. We perform fine-tuning on 32 unknown trajectories (all2all pairing, which gives 320 pairs in total). We do validation on 8 trajectories (80 pairs). To solve the bonus task, we train the same architecture from scratch for 500 epochs on the 32 unknown trajectories.

Setting	Zero-shot	Fine-tuned	Scratch (bonus)
Test relative L_2 at $t = 1.0$	0.1432	0.1097	0.3504

Table 5: Task 4: results on unknown distribution at $t = 1.0$.

We observe that fine-tuning improves over the zero-shot approach and, as expected, is significantly better than training from scratch on the (small) unknown dataset. This implies that transfer learning succeeds. We note that the zero-shot error at $t = 1.0$ (0.1432) is also much larger than Task 1 on the known distribution (0.001846). This result confirms the distribution shift. 0.1272 was the best validation relative L_2 during fine-tuning. On the other hand, training from scratch gives 0.2658. The test MSE at $t = 1.0$ on the unknown distribution is 2.033×10^{-3} (zero-shot), 1.276×10^{-3} (fine-tuned), and 9.314×10^{-3} (scratch).

We use the all2all model weights to initialise the fine-tuning run. We also keep the same architecture, optimiser setup, and batch size. The only detail we change is the training data to the unknown distribution. To establish the scratch baseline, we use random initialisation and identical hyperparameters. The fine-tune model has the best validation relative L_2 at epoch 264. The scratch model reaches best validation relative L_2 occurs at epoch 14. From this trend, we infer that applying early stopping could be better for the small unknown dataset.

For the one2one case, we use in-channels=2, modes/width/depth = 16/64/4, batch 32, and 500 epochs. For all2all (500 epochs), fine-tuning (300 epochs), and scratch (500 epochs), we use: in-channels=3 with the same modes/width/depth. We also apply 10 pairs per trajectory for all2all training (10240 train pairs, 320 val pairs). For fine-tuning, we have 320 train pairs and 80 val pairs (32 and 8 trajectories, respectively). Finally, we have: 1024 train, 32 val, 128 test trajectories for the known distribution, and 32 train, 8 val, and 128 test trajectories for the unknown case.

Task 3.1

We use Elasticity (train/val/test sizes 1024/128/256) with structured stencil grid tokenisation for training the default GAOT. As regards the details, we have: latent_tokens_size 64×64, patch size 2 (that is, 1024 patch tokens processed by the transformer), magno radius 0.033, hidden size 64, transformer hidden size 256, AdamW (lr 8×10^{-4} with a scheduler mix, 1000 epochs). We read off the relative L_1 and training time from the final CSV row ("relative error (direct)" / "training time").
We run the baseline (Task 3.1) with the original GAOT implementation (in the code, we call it "GAOT-upstream"). We run Tasks 3.2–3.3 with the modified code ("GAOT-main"), where we add random tokenisation, dynamic radius, and the PE/cross-attention components. For the mix schedule, we apply a maximum LR of 10^{-3}, and the minimum LR of 10^{-4}. The final LR is 5×10^{-5} with evaluation every 2 epochs. Finally, we apply weight decay 10^{-5} and batch size 64. We compute model sizes (bytes) from parameter counts and tensor element sizes in the training logs (this applies to all subtasks).

Task 3.2

We sampled random latent tokens from the point cloud with latent_tokens_size 16×16 (256 tokens), switched on dynamic radius ($k = 8$, $\alpha = 1.5$), and precomputed graphs for neighbour search (we applied search with chunking and the seed of 42 for tokenisation). In random tokenisation, the points are drawn from the dataset coordinate cloud. For this method, the transformer uses 256 latent tokens. Note that there is no patch aggregation in the token count.

Task 3.3

We added absolute positional encoding for the coordinate-MLP (with hidden size 64), continuous distance bias (hidden size 32), and cross-attention with 128 seed tokens.

Task	Rel. L_1	Train time (s)	Train time (h)	Tokens
3.1 Baseline	0.01527	13159	3.655	1024 (patches)
3.2 Random + dyn. radius	0.1918	11140	3.094	256 (latent)
3.3 + PE + cross-attn	0.2404	11540	3.206	256 latent / 128 seed

Table 1: GAOT test relative L_1 errors, training times, and token counts. For Task 3.1, the transformer processes 1024 patch tokens (from 64×64 latent grid with patch size 2). For Tasks 3.2–3.3, the tokens refer to random latent tokens. Task 3.3 also uses 128 seed tokens for cross-attention.

Task	Parameters	Model size (bytes)	Latent / transformer tokens
3.1 Baseline	3,396,033	13,584,132	4096 / 1024
3.2 Random + dyn. radius	6,707,713	26,830,852	256 / 256
3.3 + PE + cross-attn	6,725,346	26,901,384	256 / 128

Table 2: Model size and tokenisation details.

In Task 3.1, we use grid tokenisation with latent_tokens_size 64×64 and patch size 2, magno radius is 0.033 with hidden size 64, and the transformer hidden size is 256. In Task 3.2, we use random latent tokens instead (16×16). We also enable dynamic radius ($k = 8$, $\alpha = 1.5$) and precompute graphs. In Task 3.3, we keep the random tokens (16×16) but we add the absolute coordinate MLP PE. We also use the relative bias (distance based), and 128 seed tokens (patch size 1) for the cross-attention. In all configurations, we apply coord_dim=2, magno hidden size 64 with 3 MLP layers, and lifting channels 64. This corresponds to the default GAOT settings.
Relative L_1 errors are 0.01527 (Task 3.1), 0.1918 (Task 3.2), and 0.2404 (Task 3.3). For training times (all experiments have been run on A100 via Google Colab), we have: 13158.99s (3.655h), 11139.94s (3.094h), and 11540.10s (3.206h), respectively. Parameter counts are 3,396,033 (Task 3.1), 6,707,713 (Task 3.2), and 6,725,346 (Task 3.3). The model sizes are 13,584,132 bytes (Task 3.1), 26,830,852 bytes (Task 3.2), and 26,901,384 bytes (Task 3.3). Those are related to the parameter counts and the method details we provided in Table 2.
The baseline, with structured grid, reaches the best accuracy. If we apply random tokenisation with dynamic radius, we increase the error. Also, the PE + cross-attention method decreases accuracy. For all the runs, we used the same dataset sizes (train/val/test 1024/128/256) and the optimiser settings (1000 epochs with AdamW).

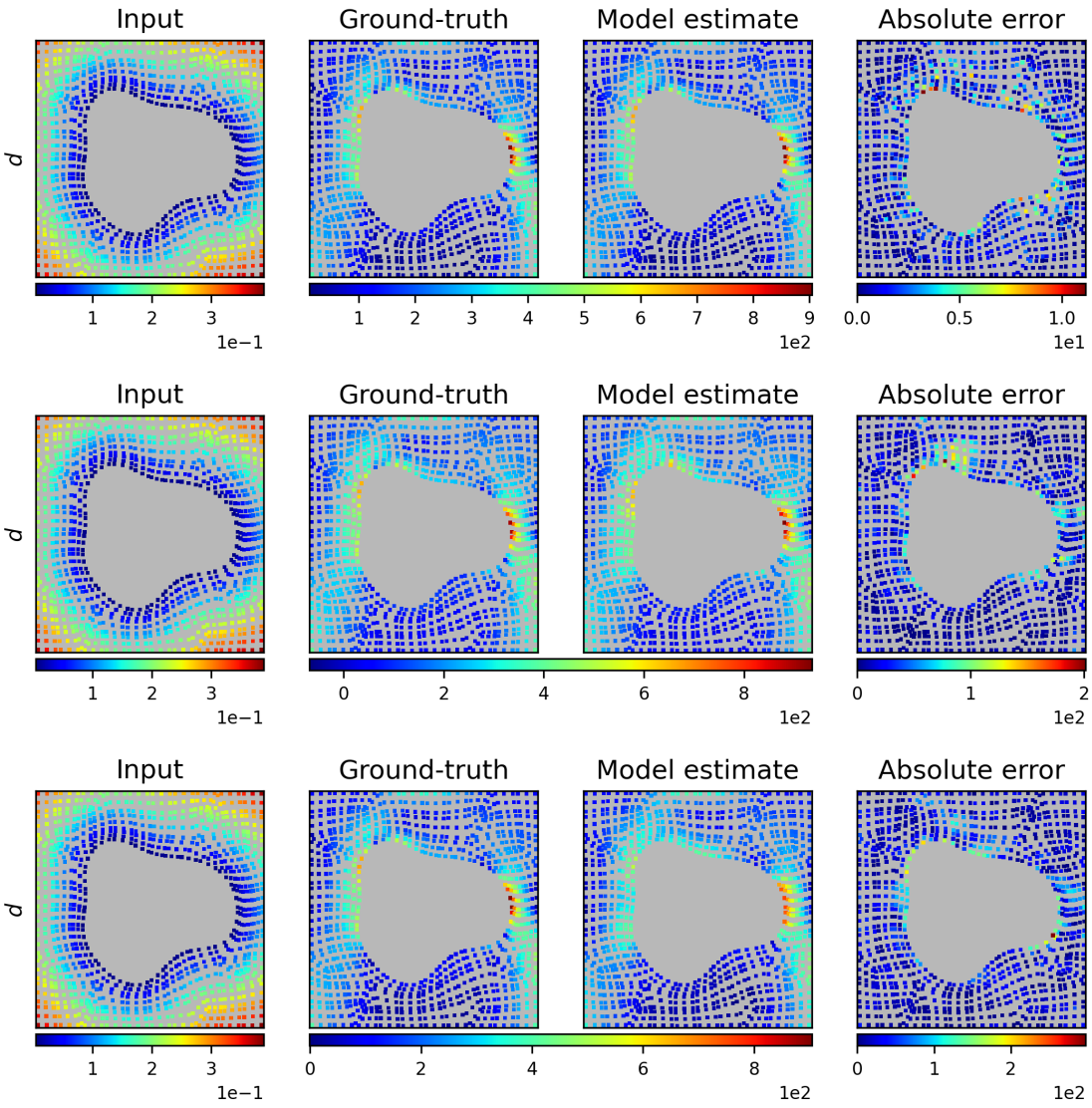

Figure 1: Elasticity results. The rows are for Tasks 3.1, 3.2, and 3.3. Each row shows the input, the absolute error, the ground truth, and the model estimate.