Resolving Extreme Data Scarcity by Explicit Physics Integration: An Application to Groundwater Heat Transport

Julia Pelzer
Institute for Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
julia.pelzer@ipvs.uni-stuttgart.de

Corné Verburg
Delft Institute of Applied Mathematics
Delft University of Technology (TU Delft)
Delft, the Netherlands
c.verburg@tudelft.nl

Alexander Heinlein
Delft Institute of Applied Mathematics
Delft University of Technology (TU Delft)
Delft, the Netherlands
a.heinlein@tudelft.nl

Miriam Schulte
Institute for Parallel and Distributed Systems
University of Stuttgart
Stuttgart, Germany
miriam.schulte@ipvs.uni-stuttgart.de

Abstract

Real-world flow applications in complex scientific and engineering domains, such as geosciences, challenge classical simulation methods due to large spatial domains, high spatio-temporal resolution requirements, and potentially strong material heterogeneities that lead to ill-conditioning and long runtimes. While machine learning-based surrogate models can reduce computational cost, they typically rely on large training datasets that are often unavailable in practice. To address data-scarce settings, we revisit the structure of advection–diffusion problems and decompose them into multiscale processes of locally and globally dominated components, separating spatially localized interactions and long-range effects. We propose a Local-Global Convolutional Neural Network (LGCNN) that combines a lightweight numerical model for global transport with two convolutional neural networks addressing processes of a more local nature. We demonstrate the performance of our method on city-scale geothermal heat pump interaction modeling and show that, even when trained on fewer than five simulations, LGCNN generalizes to arbitrarily larger domains, and can be successfully transferred to real subsurface parameter maps from the Munich region, Germany.

1 Introduction

Many learning problems in scientific and engineering domains (Sharma et al., 2021; Sarker, 2021; Angra & Ahuja, 2017; Jhaveri et al., 2022) are governed by underlying physical processes. Yet they challenge classical numerical simulations due to large spatial domains, high resolution requirements, and potential ill-conditioning. In addition, many applications require repeated evaluations, for example for uncertainty quantification, design optimization, and real-time control. From a machine learning (ML) perspective, such problems are attractive targets for surrogate modeling, as trained models can dramatically reduce inference cost, even if offline training cost is substantial.

However, in real-world advection-diffusion problems, standard ML approaches face two central difficulties (Dietrich & Schilders, 2025). First, training data are often scarce: measurements are indirect or sparse, and generating labeled data via high-fidelity simulations is computationally

expensive. Second, the underlying dynamics exhibit multiscale behavior with both short-range local interactions and long-range global dependencies, making it difficult for data-driven models to generalize in data-scarce regimes. Convolutional architectures such as UNets (Ronneberger et al., 2015a) effectively model local spatial patterns (Thuerey et al., 2020; Jhaveri et al., 2022), but representing non-local interactions requires deep receptive fields and large training datasets. Operator learning approaches, such as Fourier Neural Operators (FNOs) (Li et al., 2020), address long-range dependencies more explicitly in the frequency domain (Choi et al., 2024), but inherently introduce additional assumptions, including periodic boundary conditions, homogeneous coefficients, and fixed domain sizes, which limit their applicability in many real-world settings.

Physics-informed ML aims to mitigate data scarcity by embedding physical constraints or priors into the learning process. Physics-Informed NNs (PINNs) (Dissanayake & Phan-Thien, 1994; Lagaris et al., 1998; Raissi et al., 2019), for instance, replace or augment data losses with squared differential equation residuals, enabling learning with limited labeled data. Despite notable progress (Cuomo et al., 2022; Rao et al., 2020; Sun et al., 2020; Grimm et al., 2025; Wen et al., 2022; Cai et al., 2021), such methods remain at an early stage of methodological maturity and often struggle in complex, high-dimensional scenarios, for example by converging to trivial solutions (Krishnapriyan et al., 2021).

In our approach, rather than increasing architectural complexity, we build on a simple, well-suited base architecture and exploit the physical structure, i.e., of the underlying advection–diffusion system, explicitly. Such systems naturally decompose into processes governed by predominantly local interactions, such as diffusion acting at neighborhood scales and inferable from local information, as well as global transport effects, such as long-range advection, that require propagating information across large parts of the domain. Building on this observation, we propose the **Local–Global CNN (LGCNN)**, a modular architecture that separates local learning from explicit global transport modeling. LGCNN combines CNNs that are well-suited for locally dominant, diffusion-driven processes with a lightweight numerical surrogate that captures long-range transport effects. By inserting this inexpensive, physically motivated component at a critical point in the inference pipeline, LGCNN captures long-range dependencies without relying on deep architectures, large receptive fields, or extensive training data. In contrast to physics-informed approaches that softly impose physical constraints through loss terms, such as PINNs, LGCNN handles global transport at inference time via a dedicated, non-learnable module, leading to improved data efficiency and robustness.

We evaluate LGCNN on the task of predicting city-scale subsurface temperature fields induced by one hundred interacting groundwater heat pumps in the Munich region, Germany (Zosseder et al., 2022). This problem provides a challenging testbed for data-efficient learning, as the temperature field depends on heterogeneous subsurface parameters, arbitrary source configurations, and long-range advective transport over several kilometers and decades. We show that, even when trained on fewer than five simulated samples, LGCNN generalizes to much larger spatial domains without retraining. Our approach also transfers from synthetic inputs to real subsurface parameter maps derived from borehole measurements.

Formally, the learning task is to predict a temperature field T over arbitrarily large domains from heterogeneous permeability fields k, hydraulic pressure gradients ∇p, and heat pump locations i. Training labels are generated offline using expensive numerical simulations for only a small number of configurations. As illustrated in Figure 1, LGCNN addresses this setting via a three-stage pipeline: (1) a CNN predicts the velocity field $\vec{v}$ from local relations between p, k, and i; (2) a cheap numerical surrogate computes streamlines $\vec{s}$ to model global transport; and (3) a second CNN predicts the temperature field T conditioned on inputs and intermediate representations. This explicit separation of local learning and global transport is key to the scalability of the approach.

In groundwater heat pump (GWHP) modeling, existing work (Davis et al., 2023; Pelzer & Schulte, 2024; Scheurer, 2021) focuses on isolated or pairwise pump interactions, often using UNet-based architectures with or without physics-based regularization. These methods typically require substantial training data and have not been demonstrated to scale to city-wide domains with many interacting sources and heterogeneous aquifers.

Contributions We introduce a modular, physics-inspired architecture that captures complex, multiscale dynamics by combining local convolutional feature learning with explicit long-range transport modeling. To avoid deep architectures and large datasets for modeling nonlocal interactions, LGCNN

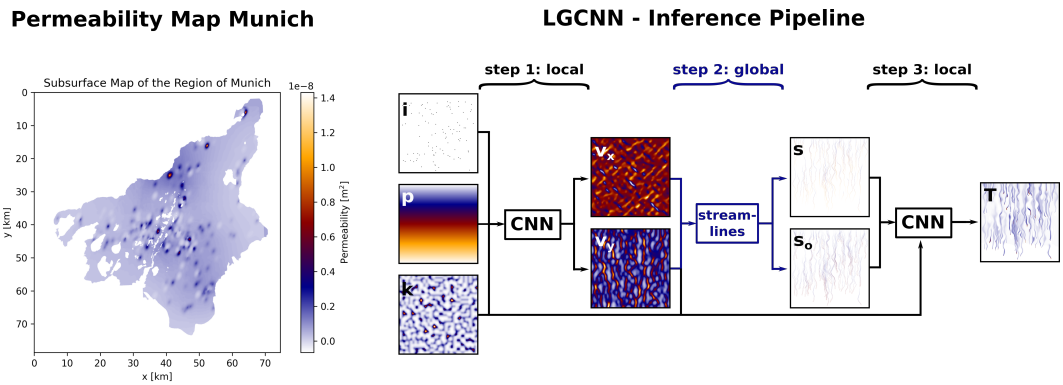

Figure 1: Left: Map of k of whole region of Munich. Right: Schematics of our Local-Global CNN-based approach (LGCNN) with 3 physics-inspired steps: CNN ($pki \rightarrow \vec{v}$), simplified solver ($i\vec{v} \rightarrow \vec{s}$), CNN ($pki\vec{v}\vec{s} \rightarrow T$).

enforces global coupling through inexpensive numerical transport modules. This structural inductive bias enables inherent generalization and scalable inference across domains and problem sizes from very limited data, i.e., less than five simulations. Compared to classical high-fidelity simulations, our approach achieves an inference-time speedup of approximately 2,000 times.

Limitations The current implementation targets a two-dimensional steady-state setting and assumes weak or spatially localized temperature-induced flow perturbations. While this work is restricted to this regime, the proposed framework is easily extendable to three-dimensional and transient problems. Empirical validation of these extensions is left to future work due to the lack of suitable training data.

2 Datasets and Metrics

Datasets Inputs for the NNs consist of a heterogeneous permeability field k, an initial hydraulic pressure field p, and a one-hot-encoded field of heat pump positions i of a constant injection rate. The (interim) labels of velocity $\vec{v}$ and temperature T fields are simulated with Pflotran (Lichtner et al., 2015a) until a quasi-steady state is reached after ≈27.5 years simulated time (Umweltministerium Baden-Württemberg, 2009).

We generate two types of datasets, one is based on synthetic, the other on real permeability fields k. Both cover a 12.8×12.8 km^2 domain with $2\,560 \times 2\,560$ cells. The **baseline dataset** uses Perlin noise (Perlin, 1985) to generate random, heterogeneous k fields. Three simulations (*3dp*) with different fields for i and k are run, generating one datapoint each for training, validation, and testing, plus a 4× larger simulation to assess scalability. For training with larger datasets as a reference in Section 3, we generate an additional dataset of 101 datapoints (*101dp*), split into 73–18–10 for train, val, test data. Simulation times are ≈ 27 hours per datapoint and 123 hours for the larger one.

The more **realistic dataset** builds on k fields cut from maps of borehole measurements in the Munich region, Germany (Bayerisches Landesamt für Umwelt, 2015). This dataset consists of four simulations (*4dp*) (three for training and one for validation) and one larger simulation for testing scalability. Due to constraints of availability of subsurface measurements, see Figure 1, the large-scale simulation is on a rectangular domain of only twice the length of the training data. Runtimes range from 38 to 91 hours (average 58 hours), and 134 hours for the scaling-test domain. Variation within the dataset for training stems from different heat pump placements in i and from the location where k is extract from the Munich region in 1.

Information on simulations and hydro-geological parameters are in Appendix A.2; hardware specifications in Appendix A.5.

Performance Metrics We evaluate model accuracy separately for each output field (directional velocities v_x, v_y, and temperature T) using standard ML metrics of Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Huber loss; the structure-focused metric of Structural Similarity Index Measure (SSIM) (Wang et al., 2004), and application-driven metrics of Measure of Connectivity

(MoC) and visual assessment. MoC expresses how physically reasonable the temperature fields are by measuring the percentage of predicted heat plume cells that are not connected via 4-connectivity[1] to a heat pump based on a temperature threshold of 0.1 °C, corresponding to measurement precision (UKB System Technology, 2024). All metrics other than SSIM are applied after re-normalization to the original data ranges to obtain physically meaningful results in [°C] or [meters/year].

3 Application of Established ML Approaches

To motivate a structure-specific architecture, we present the performance of six established ML approaches for predicting T from p, k, i: a UNet, a domain decomposition-based UNet (DDUNet) (Verburg et al., 2025), a dilated UNet, UNet++ (Zhou et al., 2018), an FNO and a PINN (results in Table 1 and Figure 2, parameters in Appendix A.3.3).

PINNs and FNOs PINNs, while promising for low-data fluid dynamics tasks (Rao et al., 2020; Sun et al., 2020; Cai et al., 2021; Takamoto et al., 2022), struggle in preliminary tests with complex scenarios (Krishnapriyan et al., 2021), such as heterogeneous media and discontinuous source terms. Physics-informed CNNs (PI-CNNs) have been evaluated in simplified single-plume aquifer settings (Hirche, 2023; Pelzer & Piller, 2024) and consistently failed to outperform standard CNNs, even when the physics-based loss was used only as an auxiliary regularization term alongside a data loss. The largest errors occur near injection locations, likely due to discontinuous source term fields. In our *3dp* problem, overfitting a PI-CNN to a single datapoint using only energy-based loss terms (cf. Appendix A.3.2) derived from the governing equations of the simulations fails to model T but instead predicts a field that resembles v_y (Figure 2). This behavior is consistent with the model capturing only a preliminary stage of the physical process and indicates a representational limitation of the learning setup regarding transport modeling.

While FNOs are in principle capable of modeling both short- and long-range interactions, we found them impractical for the target problem due to infeasible memory requirements at realistic domain sizes and their sensitivity to non-periodic boundary conditions. Additionally, they performed poorly in the presence of multiple sources, consistent with Liu-Schiaffini et al. (2024). On our example, we had to reduce the domain size to 480×480 cells to fit it on a graphics processing unit (GPU) (Appendix A.5) during training and still only managed to predict low frequency aspects, i.e., the background temperature, in our overfitting test to one datapoint.

UNet-based Methods Convolutional architectures such as UNet primarily capture local spatial dependencies and can, in principle, generalize from a single large-domain sample when interactions are local. Modeling global effects, however, requires enlarged receptive fields or architectural adaptations, such as dilated convolutions. UNet++ fuses multiscale features through nested skip connections, while still relying on convolutional locality. DDUNet addresses large domains under GPU memory constraints by decomposing the domain into, here 2×2, subdomains with learned coarse-scale feature exchange, enabling effective global context at reduced memory cost. The hyperparameters of all models are optimized according to Appendix A.3.3.

UNet and dilated UNet are trained on *3dp*; UNet, UNet++ and DDUNet are trained on *101dp* directly (Appendix A.4). All models are evaluated on the *3dp* test data for direct comparability to LGCNN, some on the scaling datapoint. Training on *3dp* leads to short, disconnected plumes (Figure 2) with high MoC. When trained on *101dp*, UNet and DDUNet show robust predictions and reasonable scaling, with DDUNet occasionally hallucinating plumes but offering lower memory usage and shorter runtimes (see Table 12). UNet++ fails due to memory constraints in the scaling test.

Overall, well-chosen established approaches can perform reasonably (best is UNet_{101dp}: MAE = 0.05°C, MoC = 0.09) and some can scale to larger domains, but none performs reasonably well in the data-scarce setting we are interested in (best is UNet_{3dp}: MAE = 0.19°C, MoC = 10.85). Their limits are clear: they are biased toward local patterns, require large datasets to capture long-range transport, and quickly run into compute and memory constraints at realistic scales. In low-data regimes they yield short, disconnected plumes or hallucinated structures, and scalability can break down (e.g., FNO and UNet++ fail under memory constraints).

[1]This means axis-aligned neighbors only, excluding diagonal connections.

Table 1: **Metrics of all experiments.** Results for standard models and LGCNN (end-to-end and Steps 1 and 3) on synthetic and real permeability fields k, including ablation variants. Models are trained on *3dp*, *101dp*, or *4dp*. Metrics are reported in physical units ([°C] for Step 3 and end-to-end, [m/y] for Step 1), except MoC ([%]) and SSIM (unitless). Velocity experiments are shaded due to differing scales; experiment details are referenced in column Sec. Only contains representative runs, statistical errors are reported in the appendix.

	Architecture	Training Data	Apply to	Step	MAE	RMSE	Huber	SSIM	MoC	Sec.
Standard	UNet	synth. 101dp	test	end2end	0.0473	0.1001	0.0048	0.8292	0.09	3
	UNet++	synth. 101dp	test	end2end	0.0918	0.1435	0.0102	0.5331	4.06	3
	DDUNet	synth. 101dp	test	end2end	0.0548	0.1134	0.0063	0.8120	0.59	3
	UNet	**synth. 3dp**	test	end2end	0.1908	0.6190	0.0736	0.6671	**10.85**	3
	Dilated UNet	synth. 3dp	test	end2end	0.1200	0.2236	0.0237	0.5004	10.95	3
	PI-CNN (overfit)	synth. 1dp	train	end2end	0.7532	0.8593	0.3454	0.1793	-	3
Ours	**LGCNN**	**synth. 3dp**	test	end2end	0.0916	0.1738	0.0146	0.6841	**0.09**	5.1
	UNet	synth. 3dp	test	3	0.0417	0.0762	0.0029	0.8540	0.07	5.1
	UNet	synth. 3dp	test	1 (vx)	22.3178	31.1860	21.8237	0.9739	-	5.1
	UNet	synth. 3dp	test	1 (vy)	32.7444	45.0703	32.2488	0.9733	-	5.1
Experiments	UNet, in sequence	synth. 3dp	test	3	0.0919	0.1695	0.0139	0.6714	1.30	5.2
	UNet, w/ 0-padding	synth. 3dp	test	3	0.0487	0.0795	0.0031	0.7784	0.57	5.2
	UNet, w/o partitions	synth. 3dp	test	3	0.0429	0.0793	0.0031	0.8349	0.29	5.2
	UNet	synth. 101dp	test	3	0.0416	0.0816	0.0032	0.8016	0.47	5.2
	DDUNet	synth. 101dp	test	3	0.0376	0.0764	0.0029	0.8843	0.26	5.2
	UNet, vary inputs	synth. 3dp	test	3	cf. A.4	cf. A.4	cf. A.4	cf. A.4	cf. A.4	A.4
	UNet, w/o partitions	synth. 3dp	test	1 (vx)	23.8584	33.8297	23.3646	0.9765	-	5.2
	UNet, w/o partitions	synth. 3dp	test	1 (vy)	35.9249	49.9040	35.4289	0.9721	-	5.2
	UNet	synth. 101dp	test	1 (vx)	12.7036	29.9379	12.2161	0.9437	-	5.2
	UNet	synth. 101dp	test	1 (vy)	324.7751	328.4992	324.2751	0.9572	-	5.2
	DDUNet	synth. 101dp	test	1 (vx)	13.6736	36.9852	13.1829	0.9468	-	5.2
	DDUNet	synth. 101dp	test	1 (vy)	322.0676	325.6697	321.5676	0.9741	-	5.2
Scaling	DDUNet (vanilla)	synth. 101dp	scaling	end2end	0.0235	0.0723	0.0025	0.8874	**1.18**	3
	UNet (vanilla)	synth. 101dp	scaling	end2end	0.0202	0.0578	0.0016	0.8914	**0.54**	3
	LGCNN	synth. 3dp	scaling	end2end	0.0413	0.1189	0.0065	0.6511	**0.06**	5.1
	UNet	synth. 3dp	scaling	3	0.0168	0.0373	0.0007	0.9405	0.08	5.1
	UNet	synth. 3dp	scaling	1 (vx)	24.9261	34.7004	24.4314	0.9869	-	5.1
	UNet	synth. 3dp	scaling	1 (vy)	26.2795	38.2599	25.7847	0.9932	-	5.1
Domain transfer	LGCNN	real 4dp	val	end2end	0.0841	0.1659	0.0137	0.6511	1.65	5.3
	UNet	real 4dp	val	3	0.0175	0.0319	0.0005	0.9405	3.30	5.3
	UNet	real 4dp	val	1 (vx)	13.3316	16.5833	12.8383	0.9869	-	5.3
	UNet	real 4dp	val	1 (vy)	9.0224	10.5612	8.5327	0.9932	-	5.3
	LGCNN	real 4dp	scaling	end2end	0.0394	0.0665	0.0022	0.6755	3.50	5.3
	UNet	real 4dp	scaling	3	0.0189	0.0287	0.0004	0.8942	1.77	5.3
	UNet	real 4dp	scaling	1 (vx)	110.0078	116.4903	109.5079	0.9436	-	5.3
	UNet	real 4dp	scaling	1 (vy)	17.6051	24.8298	17.1118	0.9905	-	5.3

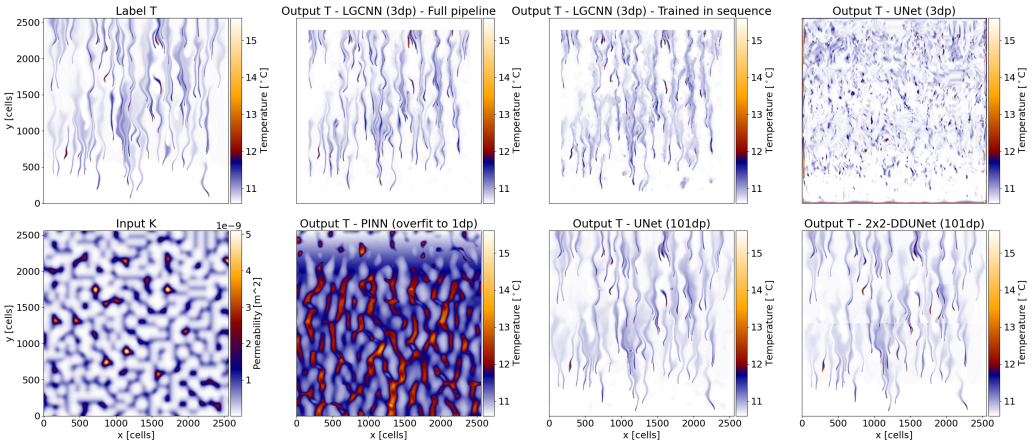

Figure 2: End-to-end: $pki \rightarrow T$ (baseline dataset (*3dp*): test). 1st column: Label T and input k. 2nd-4th column: LGCNN$_{3dp}$ (full pipeline and trained in sequence, see Section 5), standard UNet$_{3dp}$, PI-CNN$_{overfit}$, UNet$_{101dp}$, and DDUNet$_{101dp}$ (see Section 3).

4 METHODOLOGY

When generic ML methods fail in data-scarce settings, performance can be improved by adopting a more task-specific approach whose structure reflects the underlying physics (Fernández-Godino et al., 2024; Thodi et al., 2024; Zhang et al., 2020; Yousefzadeh et al., 2025; Bertels & Willems, 2023). Accordingly, this section outlines the relevant physical processes that motivate the proposed LGCNN method and discusses key aspects of its components.

Physics of Groundwater Flow with Heat Transport Heat transport induced by heat injections of GWHPs in the subsurface is an advection-diffusion process. Since the Péclet number (Pe), i.e., the advection-diffusion ratio, is high in our application (Pe$\gg$1, more in Appendix A.2), the system is advection-dominated. In the subsurface, advection is largely governed by the global hydraulic pressure gradient, driving flow from spatially higher to lower regions. Locally, the flow paths of heat plumes are influenced by the spatial distribution of permeability k, cf. Figure 2. As a result, small changes in k can lead to significant differences in flow paths further downstream, demonstrating high input sensitivity.

One-Way Coupled Approach: Local-Global CNN (LGCNN) In a fully coupled simulation, flow field computation and heat transport along streamlines (str, 2011) starting at heat pumps (advection) and their widening (diffusion) influence each other. Therefore, convergence requires that two-way coupling conditions are satisfied, whether using a monolithic or a segregated approach.

By working with steady-state simulated $\vec{v}$ as interim outputs during training (cf. Appendix A.2 *Reasoning for Decoupling Assumptions*), we can split the processes into local (diffusion) and global (advection) dominated ones and hence extract three subsequent steps that are coupled in only one direction. Our simplified physical pipeline consists of three steps: (1) compute a steady-state flow field $\vec{v}$ from initial subsurface parameters p and k, with i encoding mass influx around GWHPs; (2) transport injected heat along streamlines governed by $\vec{v}$ until quasi-steady state (Umweltministerium Baden-Württemberg, 2009); and (3) apply plume widening to these heat paths, informed by soil diffusivity and $\vec{v}$, to approximate diffusion effects. The resulting one-way coupled application of LGCNN (see Figure 1) can formally be described as:

Step 1: Velocities (local) We employ a CNN to predict the velocity field $\vec{v} = (v_x, v_y)$ from p, k, i:

$$\text{CNN}(p, k, i) = \vec{v}. \tag{1}$$

Step 2: Streamlines (global) Based on the predicted velocities $\vec{v}$, we compute streamlines $\vec{s}$ originating from all pumps in i with an initial value problem (IVP) solver:

$$\text{IVP}(i, \vec{v}) = \vec{s}. \tag{2}$$

Step 3: Temperature (local) A second CNN predicts the temperature field T:

$$\text{CNN}(k, i, \vec{v}, \vec{s}) = T. \tag{3}$$

The model outputs a steady-state temperature field T using the same inputs as a simulation, hence serving as its direct surrogate.

CNN Models in the Local Steps The local Steps 1 and 3 are approximated by a UNet, same as in other groundwater applications (Davis et al., 2023; Pelzer & Schulte, 2024), see Appendix A.3.1 for details of our architecture. It is beneficial to omit zero-padding to enforce shift invariance and reliance on purely local information (Islam et al., 2020). Due to the input sensitivity of the entire problem, the CNN in Step 3 is trained on simulated velocities $\vec{v}_{\text{sim}}$ and streamlines based on these fields. Only during inference, all steps are applied sequentially, i.e., Step 3 uses the outputs of Steps 1 and 2, e.g., by including the predicted $\vec{v}_{\text{pred}}$ as inputs.

The training is accelerated by partitioning the data into overlapping patches, which increases the effective number of datapoints while reducing their spatial size. Combined with the streamlines' embedding and localizing of global flow patterns for the fully convolutional (locally) acting CNN, this training data enrichment allows the model to train effectively on very little data, partially as little as a single simulation run, while generalizing to unseen and even to larger domains. Validation and testing are conducted on independent simulation runs, with the full domain being processed as a whole instead of patches during inference.

The architecture, input choices, and training hyperparameters, including patch size and overlap in the datapoint partitioning, were optimized; see Appendix A.3.3 for details.

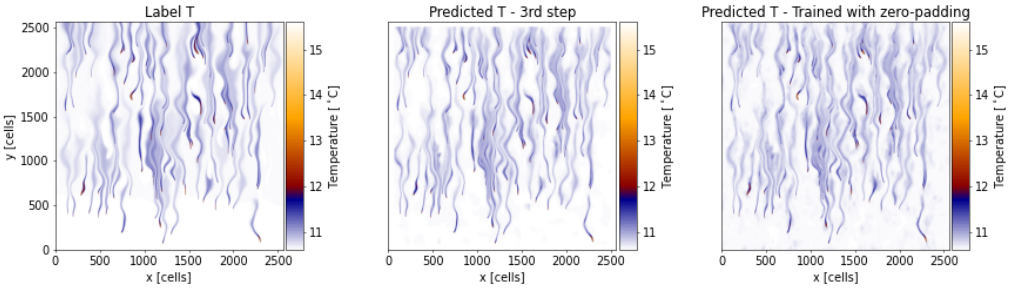

Figure 3: Step 3: $pki\vec{s}\vec{v}_{\text{sim}} \rightarrow T$ (baseline dataset (*3dp*): test). 1st column: Label T. 2nd-3rd column: LGCNN Step 3, trained without and with zero-padding.

Streamline Calculation and 2D-Embedding in the Global Step Streamlines are calculated by solving the initial value problem (IVP)

$$\frac{dy}{dt} = \vec{v}(y), \text{ with } y(t_0) = y_0, \tag{4}$$

with a lightweight numerical solver for each heat pump, where y_0 represents the location of a heat pump in i. We employ an implicit 5th-order Runge–Kutta method (sci) with 10,000 one-day time steps, corresponding to the physical timescale of heat plume formation.

The computed streamlines y represent sequences of 2D positions traversed by a particle injected at y_0 under the velocity field $\vec{v}$. These paths are embedded into 2D fields along the traversed grid cells by assigning values that decay linearly from 1 to 0, reflecting the time required to reach each cell and yielding a time-weighted 2D-embedding.

To address the sensitivity of this process with respect to k and $\vec{v}$, we calculate the outer streamlines s_o. For this, we perturb each heat pump's location y_0 by ± 10 cells orthogonal to the global flow direction, i.e., ∇p, and compute the corresponding streamlines. By incorporating $\vec{s} = (s, s_o)$ as two inputs to Step 3, it obtains a spatial, localized representation of flow paths. In future work, we aim to explore probabilistic perturbations to compute the mean and standard deviation of streamlines, which, while being computationally more expensive, could be efficiently parallelized on GPUs.

5 Results

We evaluate LGCNN's performance on the baseline dataset *3dp* for isolated Steps 1 and 3, and for the full pipeline (end-to-end), demonstrating the potential and scalability of our approach. In Section 5.2, we motivate model design choices through variational experiments. Additional metrics and visualizations can be found in Appendix A.4. Section 5.3 explains adaptations, performance, and scalability on the more realistic dataset.

5.1 Performance on the Synthetic Dataset

Isolated Steps 1 and 3 Performance metrics in Table 1 *Ours* are for Step 3 (trained and tested on simulated groundtruth $\vec{v}_{\text{sim}}$), and for Step 1 highlighted in gray. The combination of quantitative (MAE of (22.3, 32.7) m/y and 0.04 °C, and only 0.07% unconnected cells) and qualitative measures (visual assessment of Figure 3) demonstrate a strong, physically reasonable performance, especially compared to a generic UNet trained on the same number of datapoints (Figure 2).

Full Pipeline Testing the full pipeline, i.e., Steps 2 and 3 applied to $\vec{v}_{\text{pred}}$ predicted by Step 1, results in a higher test MAE of 0.09 °C compared to an isolated Step 3. This was expected as the isolated Step 3 uses groundtruth, simulated velocity fields $\vec{v}_{\text{sim}}$ as inputs. Visual assessment of representative predictions in Figure 2 reveals physically plausible heat plumes in terms of shape, extent, and heat magnitude, also measurable by a comparably low MoC of 0.09 %. Deviations in streamlines arise from smaller errors in the velocity predictions $\vec{v}_{\text{pred}}$ from Step 1, highlighting the input sensitivity of the physical problem, especially near bifurcations, where small perturbations in $\vec{v}$ can lead to an alternative path, e.g., around a clay lens in k. Compared to training end-to-end on *101dp* (see Table 1 *Standard*), our model achieves a similar MoC and at least half the accuracy while training on only 1

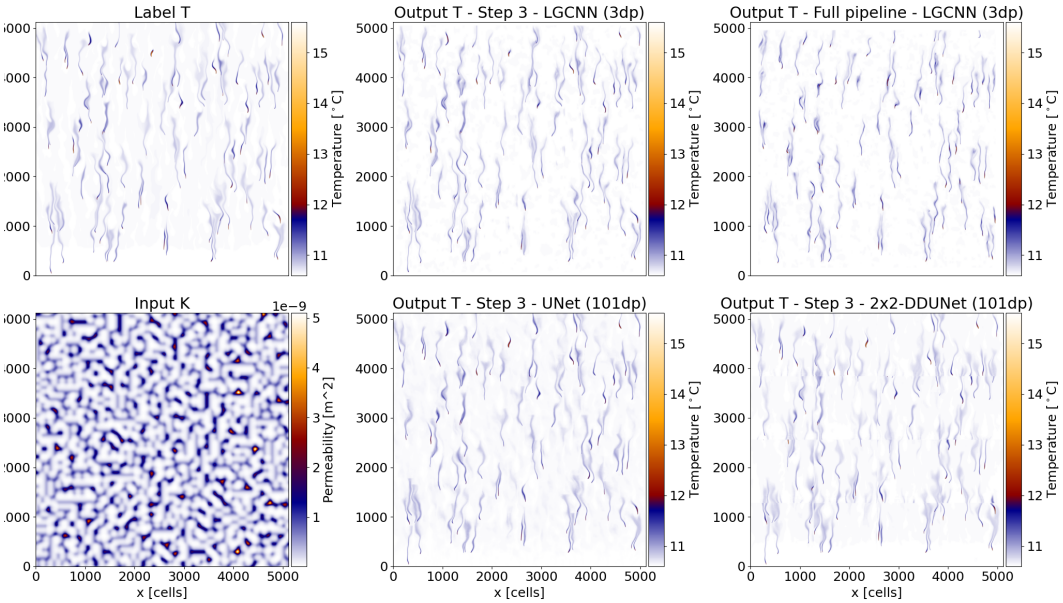

Figure 4: Predictions of T (baseline (*3dp*): scaling). 1st column: Label T and input k. 2nd-3rd column: LGCNN - Step 3 and pipeline; and Step 3 based on data-rich training (UNet, DDUNet).

versus 73 training datapoints, which strongly reduces computation time (see Table 12 for details) and data requirements.

Scaling It is expected that our model can scale to larger domains, as we train on small patches of the spatial domain and achieve high test accuracies on the full domain during validation and testing; see Section 5.2 *Partitioning Training Data* for more details. To further demonstrate the scaling, we test our model on a domain of 4× the size of the training domain with the same number of heat pumps, cf. Figure 4. LGCNN obtains a low MAE of 0.02 °C for Step 3, and 0.04 °C for the full pipeline and a very low MoC of 0.06%, see Table 1. A visual assessment of Figure 4 yields similar qualitative, physically reasonable behavior as the previous results in Figure 2, demonstrating the scalability of LGCNN.

5.2 Experiments on the Synthetic Dataset

Training in Sequence When training Step 3, we consider two alternatives for the velocity input: simulated velocities $\vec{v}_{\text{sim}}$ and velocities predicted by Step 1 $\vec{v}_{\text{pred}}$. Training the model sequentially using $\vec{v}_{\text{pred}}$ results in substantially higher MoC values compared to training with $\vec{v}_{\text{sim}}$, as shown in Table 1 (*Experiments* vs. *Ours*). As illustrated in Figure 2, using $\vec{v}_{\text{pred}}$ during training leads to physically implausible temperature fields characterized by noisy artifacts and fragmented heat plumes. We attribute this behavior to local misalignments between the predicted streamline inputs $\vec{s}$ and the corresponding temperature label T, which strongly complicates localized learning. In contrast, training with $\vec{v}_{\text{sim}}$ produces streamline fields that are physically consistent with the temperature distribution which provides informative gradients and enables stable and effective training.

Enabling Zero-Padding Using no padding in all convolutional layers produces smaller but cleaner output fields, as shown in Figure 3. This leads to a slight improvement in standard ML metrics and a substantial improvement in the MoC score (Table 1, *Ours* vs. *Experiments*), indicating fewer disconnected heat plumes.

Partitioning Training Data Hyperparameter search shows the best performance for partitioning the domain during training into 20 736 overlapping patches of 256×256 cells for Step 1 and 82 944 patches for Step 3. Although this increases per-epoch compute time compared to full-domain training, it significantly reduces the number of epochs needed to reach comparable test performance (cf. Table 1 *Experiments* vs. *Ours, Step 1*), resulting in an overall training time reduction of 67–90% (cf. Table 11) while maintaining comparable accuracy across all metrics.

Replacing Isolated Steps with (2×2-DD)UNet$_{\textit{101dp}}$ We replace each step with an optimized UNet or DDUNet trained on *101dp* (cf. Appendix A.3.3). Step 2 fails for both models. Results for Steps 1 and 3 are summarized in Table 1 *Experiments*. In Step 1, both models struggle to predict v_y due to outliers in training data. In Step 3, performance is comparable to *3dp* in standard metrics but MoC is substantially higher, and scaling results in Figure 4 show little benefit from additional training data.

Modifying Streamline Inputs of Step 3 We evaluate the influence of Step 3's inputs, focusing on the effect of modifying or omitting individual components of the streamlines $\vec{s} = (s, s_o)$. When both the central streamlines s and offset streamlines s_o are excluded, predicted plumes fail to follow flow paths beyond the CNN's receptive field. Omitting only s_o results in overly narrow plumes, whereas excluding only s produces overly smeared temperature fields. Including both but removing the fading over time along the streamlines leads to diffuse background temperatures and overly precise and too long plumes. Overall, the Huber validation loss increases by 32-132%, depending on the experiment. For extended and visual results, we refer to Appendix A.4.

5.3 Domain Transfer to Real Permeability Fields

We evaluate our method on real permeability fields k (Section 2), for which only four datapoints are available. Results for Step 3 and the full pipeline on the validation and scaling datapoints are shown in Table 1 *Domain transfer*; additional metrics are provided in Appendix A.4.

Methodological Adaptations Slight adjustments to chunk size, overlap, data split, and architecture yield better performance compared to those used on the baseline dataset, see Appendix A.3.3 for details. Visual inspection of input k in Figure 5 reveals fewer but larger-scale features. Therefore, training can benefit from a larger spatial context, which requires more training data. We increase the number of training samples by using three out of the four available datapoints for training and one for validation. For the streamlines computation, we switch to an explicit 4th-order Runge–Kutta scheme, which proved stable and sufficient for training and inference due to the lower frequencies in k.

Performance and Scaling Tests In Step 3, we reach an even lower maximum absolute error L_∞ of 0.82 °C compared to 2.90 °C of our initial model on the baseline dataset (Section 5). Other metrics confirm this except for SSIM and MoC; cf. Table 1 *Domain transfer*. In the scaling test, most errors behave as before, as expected. For the full pipeline, errors are slightly higher on the validation and scaling data compared to an isolated Step 3, consistent with our observations on the baseline dataset. Physical consistency, which is our primary objective, remains strong: Shape, magnitude, and connectivity of the predicted heat plumes are well preserved, even in the scaling test (Figure 5). Local deviations are primarily due to differences between $\vec{v}_{\text{sim}}$ and $\vec{v}_{\text{pred}}$, as evident from the discrepancy between Step 3 and full pipeline outputs.

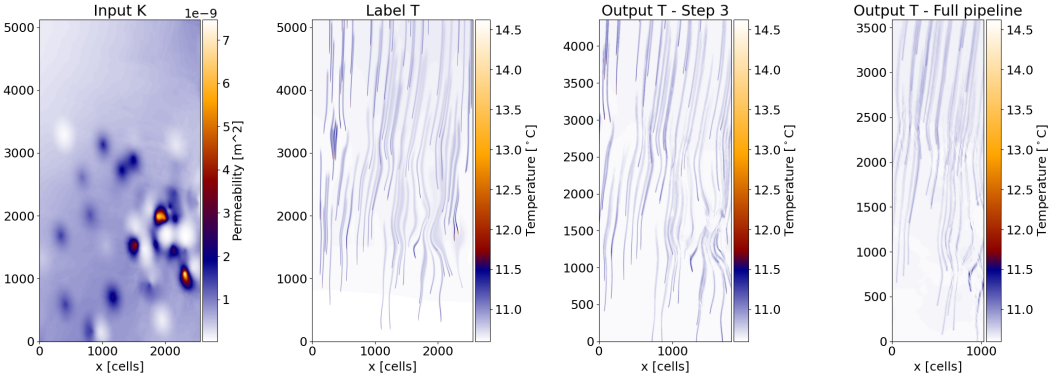

Figure 5: Predictions of T (realistic (*4dp*): scaling). Input k, label T, LGCNN: step 3, full pipeline.

Based on their poor generalization in data-scarce regimes observed on the baseline dataset, we omit generic ML architectures from the experiments on realistic k fields.

6 Conclusions and Future Work

We introduce the **Local-Global Convolutional Neural Network (LGCNN)** to address complex, data-scarce learning problems governed by both local and non-local physical interactions. Trained on only a few simulations, LGCNN accurately predicts subsurface temperature fields on much larger two-dimensional heterogeneous domains, while providing an inference-time speed-up of roughly 2,000× compared to classical high-fidelity simulations on the training data domain. By combining translation-invariant CNNs for local processes with a lightweight numerical surrogate for global transport, our method achieves data-efficient, scalable, and physically consistent predictions, outperforming conventional ML models such as UNet variants that struggle to capture long-range dependencies in data-scarce regimes. Although demonstrated for groundwater heat transport, the concept of local-global decomposition can be generalized to a broad class of advection-diffusion systems. Future work will extend the approach to three-dimensional and transient scenarios, investigate sensitivity to input uncertainty, and further accelerate streamline computation.

ACKNOWLEDGMENTS

We thank the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) for supporting this work by funding - EXC2075 – 390740016 under Germany's Excellence Strategy. We acknowledge the support by the Stuttgart Center for Simulation Science (SimTech). We also acknowledge the use of computational resources of the DelftBlue supercomputer, provided by Delft High Performance Computing Centre (Delft High Performance Computing Centre , DHPC).

Additionally, we would like to thank Fabian Böttcher and Kai Zosseder from the Technical University of Munich for their application-related input.

REFERENCES

solve_ivp — SciPy v1.14.1 Manual. https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html. accessed on Aug 30, 2024.

Chapter 2 - streamline numerical simulation theory and method. In Yao Jun and Wu Minglu (eds.), *Streamline Numerical Well Test Interpretation*, pp. 17–42. Gulf Professional Publishing, Boston, 2011. ISBN 978-0-12-386027-9. doi: https://doi.org/10.1016/B978-0-12-386027-9.00002-2. URL https://www.sciencedirect.com/science/article/pii/B9780123860279000022.

Geologica bavarica band 122: Die hydraulischen grundwasserverhältnisse des quartären und des oberflächennahen tertiären grundwasserleiters im großraum münchen. https://www.bestellen.bayern.de/shoplink/91122.htm, 2022.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

Mary P Anderson. Introducing groundwater physics. *Physics today*, 60(5):42–47, 2007.

Sheena Angra and Sachin Ahuja. Machine learning and its applications: A review. In *2017 international conference on big data analytics and computational intelligence (ICBDAC)*, pp. 57–60. IEEE, 2017.

Bayerisches Landesamt für Umwelt. Geologische Daten: Geologisches Fachinformationssystem (GEPO). https://www.lfu.bayern.de/geologie/gepo/index.htm, 2015. Accessed: 2025-05-14.

Daan Bertels and Patrick Willems. Physics-informed machine learning method for modelling transport of a conservative pollutant in surface water systems. *Journal of Hydrology*, 619:129354, 2023.

Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and George Em Karniadakis. Physics-informed neural networks (PINNs) for fluid mechanics: A review, May 2021. URL http://arxiv.org/abs/2105.09506. arXiv:2105.09506 [physics].

Byoung-Ju Choi, Hong Sung Jin, and Bataa Lkhagvasuren. Applications of the fourier neural operator in a regional ocean modeling and prediction. *Frontiers in Marine Science*, 11:1383997, 2024.

Salvatore Cuomo, Vincenzo Schiano Di Cola, Fabio Giampaolo, Gianluigi Rozza, Maziar Raissi, and Francesco Piccialli. Scientific machine learning through physics–informed neural networks: Where we are and what's next. *Journal of Scientific Computing*, 92(3):88, 2022.

Kyle Davis, Raphael Leiteritz, Dirk Pflüger, and Miriam Schulte. Deep learning based surrogate modeling for thermal plume prediction of groundwater heat pumps, 2023.

Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

Jacques W Delleur. Elementary groundwater flow and transport processes. In *The handbook of groundwater engineering*, pp. 91–120. CRC press, 2016.

Felix Dietrich and Wil Schilders. Scientific machine learning. *Mathematische Semesterberichte*, 72 (2):89–115, 2025.

M. W. M. G. Dissanayake and N. Phan-Thien. Neural-network-based approximations for solving partial differential equations. *Communications in Numerical Methods in Engineering*, 10(3):195–201, 1994. ISSN 1099-0887. doi: 10.1002/cnm.1640100303. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/cnm.1640100303. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/cnm.1640100303.

M Giselle Fernández-Godino, Wai Tong Chung, Akshay A Gowardhan, Matthias Ihme, Qingkai Kong, Donald D Lucas, and Stephen C Myers. A staged deep learning approach to spatial refinement in 3d temporal atmospheric transport. *arXiv preprint arXiv:2412.10945*, 2024.

Viktor Grimm, Alexander Heinlein, and Axel Klawonn. Learning the solution operator of two-dimensional incompressible Navier-Stokes equations using physics-aware convolutional neural networks. *Journal of Computational Physics*, pp. 114027, April 2025. ISSN 0021-9991. doi: 10.1016/j.jcp.2025.114027. URL https://www.sciencedirect.com/science/article/pii/S0021999125003109.

Manuel Hirche. Application and extension of a super resolution physics-informed convolutional neural network to groundwater modelling. Bachelorarbeit: Universität Stuttgart, Institut für Parallele und Verteilte Systeme, Simulation großer Systeme, März 2023. URL http://www2.informatik.uni-stuttgart.de/cgi-bin/NCSTRL/NCSTRL_view.pl?id=BCLR-2023-23&engl=0.

Md Amirul Islam, Sen Jia, and Neil D. B. Bruce. How much position information do convolutional neural networks encode? In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJeB36NKvB.

R. Jhaveri, A. Revathi, K. Ramana, R. Raut, and Rajesh Kumar Dhanaraj. A review on machine learning strategies for real-world engineering applications. *Mobile Information Systems*, 2022. doi: 10.1155/2022/1833507.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in neural information processing systems*, 34:26548–26560, 2021.

I.E. Lagaris, A. Likas, and D.I. Fotiadis. Artificial neural networks for solving ordinary and partial differential equations. *IEEE Transactions on Neural Networks*, 9(5):987–1000, September 1998. ISSN 1941-0093. doi: 10.1109/72.712178. Conference Name: IEEE Transactions on Neural Networks.

Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

Peter C Lichtner, Glenn E Hammond, Chuan Lu, Satish Karra, Gautam Bisht, Benjamin Andre, Richard Mills, and Jitendra Kumar. Pflotran user manual: A massively parallel reactive flow and transport model for describing surface and subsurface processes. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States); Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States); Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States); OFM Research, Redmond, WA (United States), 2015a. Accessed: 2023-09-19.

Peter C Lichtner, Glenn E Hammond, Chuan Lu, Satish Karra, Gautam Bisht, Benjamin Andre, Richard Mills, and Jitendra Kumar. Pflotran - theory guide - thermal-hydrologic mode - governing equations. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States); Sandia National Lab.(SNL-NM), Albuquerque, NM (United States); Lawrence Berkeley National Lab.(LBNL), Berkeley, CA (United States); Oak Ridge National Lab.(ORNL), Oak Ridge, TN (United States); OFM Research, Redmond, WA (United States), 2015b. Accessed: 2025-03-12.

Miguel Liu-Schiaffini, Julius Berner, Boris Bonev, Thorsten Kurth, Kamyar Azizzadenesheli, and Anima Anandkumar. Neural operators with localized integral and differential kernels. *arXiv preprint arXiv:2402.16845*, 2024.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Julia Pelzer and Lukas Piller. Optimizing groundwater heat pump placement: Extending heat plumes. In *Young Mathematicians in Model Order Reduction (YMMOR) Conference*, Stuttgart, Germany, 2024. Talk at YMMOR.

Julia Pelzer and Miriam Schulte. Efficient two-stage modeling of heat plume interactions of geothermal heat pumps in shallow aquifers using convolutional neural networks. *Geoenergy Science and Engineering*, 237:212788, 2024.

Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985.

M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

Chengping Rao, Hao Sun, and Yang Liu. Physics-informed deep learning for incompressible laminar flows. *Theoretical and Applied Mechanics Letters*, 10(3):207–212, 2020.

Bastian E. Rapp. *Péclet Number*, chapter 9.9.7. Elsevier, 2017. URL https://www.sciencedirect.com/topics/chemical-engineering/peclet-number.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015a. URL http://arxiv.org/abs/1505.04597.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pp. 234–241. Springer, 2015b.

Iqbal H Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN computer science*, 2(3):160, 2021.

Stefania Scheurer. A deep learning approach for large–scale groundwater heat pump temperature prediction, 2021.

Neha Sharma, Reecha Sharma, and Neeru Jindal. Machine learning and deep learning applications-a vision. *Global Transitions Proceedings*, 2(1):24–28, 2021.

Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

Makoto Takamoto, Timothy Praditia, Raphael Leiteritz, Daniel MacKinlay, Francesco Alesiani, Dirk Pflüger, and Mathias Niepert. Pdebench: An extensive benchmark for scientific machine learning. *Advances in Neural Information Processing Systems*, 35:1596–1611, 2022.

Bilal Thonnam Thodi, Sai Venkata Ramana Ambadipudi, and Saif Eddin Jabari. Fourier neural operator for learning solutions to macroscopic traffic flow models: Application to the forward and inverse problems. *Transportation research part C: emerging technologies*, 160:104500, 2024.

Nils Thuerey, Konstantin Weißenow, Lukas Prantl, and Xiangyu Hu. Deep learning methods for reynolds-averaged navier-stokes simulations of airfoil flows. *AIAA Journal*, 58(1):25–36, 2020.

UKB System Technology. Kabellichtlot typ 120 - ltc elektrische leitfähigkeit bis 100m. https://geotechnik-shop.de/Kabellichtlot-Typ-120-LTC-elektrische-Leitfaehigkeit-bis-100m, 2024. accessed on Aug 28, 2024.

Umweltministerium Baden-Württemberg. Arbeitshilfe zum leitfaden zur nutzung von erdwärme mit grundwasserwärmepumpen für ein- und zweifamilienhäuser oder anlagen mit energieentzug bis zirka 45.000 kwh pro jahr, October 2009. URL https://um.baden-wuerttemberg.de/fileadmin/redaktion/m-um/intern/Dateien/Dokumente/5_Energie/Erneuerbare_Energien/Geothermie/4_Arbeitshilfe_zum_Leitfaden.pdf. Accessed: 2023-03-13.

Corné Verburg, Alexander Heinlein, and Eric C. Cyr. Ddu-net: A domain decomposition-based cnn for high-resolution image segmentation on multiple gpus. *IEEE Access*, 13:66967–66983, 2025. doi: 10.1109/ACCESS.2025.3561033.

Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004. doi: 10.1109/TIP.2003.819861.

Gege Wen, Zongyi Li, Kamyar Azizzadenesheli, Anima Anandkumar, and Sally M Benson. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022.

Reza Yousefzadeh, Alireza Kazemi, Rami Al-Hmouz, and Iyad Al-Moosawi. Determination of optimal oil well placement using deep learning under geological uncertainty. *Geoenergy Science and Engineering*, 246:213621, 2025.

Ruiyang Zhang, Yang Liu, and Hao Sun. Physics-guided convolutional neural network (phycnn) for data-driven seismic response modeling. *Engineering Structures*, 215:110704, 2020.

Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation, 2018. URL https://arxiv.org/abs/1807.10165.

Kai Zosseder, Fabian Böttcher, Kyle Davis, Christine Haas, Smajil Halilovic, Thomas Hamacher, Helmut Heller, Leonhard Odersky, Viktoria Pauw, Thilo Schramm, and Miriam Schulte. Schlussbericht zum verbundprojekt geo-kw. Technical report, Bundesministerium für Wirtschaft und Klimaschutz, 2022.

A Appendix

A.1 Reproducibility

Our code to train and evaluate models, the trained and evaluated models themselves, and the datasets designed for training and testing are publicly available. Our raw data and trained models are uploaded to DaRUS `https://darus.uni-stuttgart.de/dataverse/ipvs_lgcnn/` or more specifically:

Raw datasets:

- Raw dataset of synthetic permeability with 3+1 datapoints (*3dp* + 1 for scaling tests): `https://doi.org/10.18419/darus-4156`
- Raw dataset of synthetic permeability with 101 datapoints (*101dp*): `https://doi.org/10.18419/darus-5064`
- Raw dataset of real permeability with 4+1 datapoints (*4dp* + 1 for scaling tests): `https://doi.org/10.18419/darus-5065`

Trained models (including hyperparameter settings):

- Models trained on synthetic permeability fields, *3dp*: LGCNN, vanilla UNet$_{3dp}$, PINN, dilated UNet: `https://doi.org/10.18419/darus-5080`
- Vanilla approaches trained on synthetic permeability fields, *101dp*: UNet$_{101dp}$, DDUNet, UNet++, LGCNN experiment: replace isolated steps 1 and 3 with DDUNet, UNet$_{101dp}$: `https://doi.org/10.18419/darus-5081`
- LGCNN on real permeability fields, *4dp*: `https://doi.org/10.18419/darus-5082`

Our code for raw data generation (`https://github.com/JuliaPelzer/Dataset-generation-with-Pflotran`) and model training (including dataset preparation and evaluation routines) (for LGCNN and data preparation: `https://github.com/JuliaPelzer/Heat-Plume-Prediction/tree/AllIn1/LGCNN/release25`, for DDUNet: `https://github.com/corne00/DDUNetForHeatplumePrediction`) is on GitHub.

Please download the respective raw data, then follow the instructions in the readme files to prepare the data and to train or infer a model.

A.2 Simulation Setup and Transport Regime

In Section 2, we describe how the two datasets are generated using the subsurface simulation software Pflotran (Lichtner et al., 2015a), which solves the coupled mass and energy conservation equations. Here, we provide additional technical details and modeling assumptions relevant for reproducibility. Furthermore, we show that for the selected parameters, the heat transport in our system is dominated by advection by a theoretical analysis.

Hydro-geological, Operational and Simulation Parameters Our data was generated on 12.8km$\times$12.8km$\times$5m with a cell size of 5m$\times$5m$\times$5m for the baseline simulations. To test scalability, we also simulate a domain that is twice as large in both x- and y-dimension for the baseline dataset, but only twice as large in y-dimension for the more realistic dataset due to dimension restrictions of the available input data.

The *baseline dataset* uses a constant hydraulic pressure gradient ∇p of 0.003 (Pelzer & Schulte, 2024; geo, 2022); permeability field k is generated using Perlin noise (Perlin, 1985) within (1.02E-11, 5.10E-9) m^2, and 100 randomly placed heat pumps, which are all operating with a 5 °C injection temperature difference compared to the surrounding groundwater and an injection rate of 0.00024 m^3/s. All values are based on realistic parameter ranges in the region of Munich(geo, 2022; Zosseder et al., 2022).

For the *realistic dataset*, instead of random permeability fields k, we use permeability field data that is derived from borehole measurements in the Munich region(Bayerisches Landesamt für Umwelt, 2015).

Furthermore, based on subsurface measurements, we set $\nabla p = 0.0025$ for these simulations. All other simulation parameters remain identical to the baseline. Other subsurface and operational parameters are taken directly from Pelzer & Schulte (2024), which also includes additional information about solver setup and boundaries. For mathematical details of the governing equations, we refer the reader to Lichtner et al. (2015b); Anderson (2007); Delleur (2016).

Notation Simplification For simplifications, we refer to the (hydraulic) pressure field p throughout the paper. In our paper, this field is the initial pressure field defined through the measured hydraulic head and is not the true pressure field at quasi-steady state with spatial details resulting from permeability k variations in the domain and mass injections at the heat pump positions. The true pressure field is only available after simulation (which we are avoiding for our approach) and strongly interacts with the velocity field.

Péclet Number To quantify whether heat transport in our system is dominated by advection or diffusion, we compute the dimensionless Péclet number (Rapp, 2017), which is defined as

$$\text{Pe} = \frac{L \cdot v}{\alpha},$$

with the characteristic length L, the local velocity v, and the thermal diffusivity α, defined as

$$\alpha = \frac{\kappa}{\rho c_p},$$

with κ the thermal conductivity, ρ the density, and c_p the specific heat capacity. We take the parameters of the solid phase of our simulation inputs of

- Thermal conductivity: $\kappa = 0.65\,\mathrm{W/(m \cdot K)}$
- Density: $\rho = 2800\,\mathrm{kg/m^3}$
- Specific heat capacity: $c_p = 2000\,\mathrm{J/(kg \cdot K)}$

This yields a thermal diffusivity of:

$$\alpha = \frac{0.65}{2800 \cdot 2000} \approx 1.16 \times 10^{-7}\,\mathrm{m^2/s}$$

The velocity values are derived by simulations, taken in the direction of flow (longitudinal) from the realistic k-dataset:

- Maximum: $1200\,\mathrm{m/year} \approx 3.8 \times 10^{-5}\,\mathrm{m/s}$
- Minimum: $44\,\mathrm{m/year} \approx 1.4 \times 10^{-6}\,\mathrm{m/s}$
- Mean: $330\,\mathrm{m/year} \approx 1.04 \times 10^{-5}\,\mathrm{m/s}$

With a characteristic length determined by the heat plume length of 2000–6000 m and mean x-velocity, we get:

$$Pe = \frac{L \cdot v}{\alpha} \approx L \cdot 10^2 \;\approx 10^5$$

The interpretation of the Peclet number is given by the following:

- $Pe \ll 1$: Diffusion dominates
- $Pe \approx 1$: Diffusion and advection similarly dominate
- $Pe \gg 1$: Advection dominates

Therefore, Pe $\approx 10^5$ indicates that in the simulations, the heat transport is advection-dominated at larger scales.

Reasoning for Decoupling Assumptions In LGCNN, the goal is to split properties into local and global. To build separate datasets for the isolated steps, we require reasonable inputs and labels. Hence, the from data or simulation generated physical properties are analyzed as potential interim inputs / labels. From simulation, we get $\vec{v}$ (mostly local) and T (heat is transported in an advective and diffusive manner $\rightarrow$ local and global). To split the temperature field into local and global, we handcraft a new property streamlines $\vec{s}$ that describe the central evolution of a heat plume driven only by advection, starting at each heat pump location. An important assumption underlying our approach is that the coupling between velocity and temperature can predominantly be treated as one-way in our setting. The velocity field strongly influences the temperature distribution but the temperature also influences the velocity. Since this effect mostly occurs in 3D domains with very high temperature gradients and we model 2D and very small temperature gradients, our assumption holds.

The general background velocity field influences the streamlines directly but the streamlines or rather the injection points and their mass influx also disturbs the local velocity field which in turn can influence other streamlines passing by from further upstream. By including the start locations of the streamlines, i.e., the locations of the heat pumps, as inputs to $\vec{v}$ and taking the steady-state flow field with running heat pumps from the simulation as a label, the influence of $\vec{s}$ is reduced and again decoupled as an additional input i with the influence only local and already included in the labels. Hence, a one-way coupling from positions of heat pumps $i \rightarrow \vec{v} \rightarrow \vec{s}$ is possible.

A.3 NNs, Hyperparameters and Training Details

This section provides implementation and training details for all NN models used in this work. We begin with a description of the baseline UNet architecture, which also forms the foundation of both LGCNN and DDU-Net. Then, we outline the hyperparameter optimization process and present the final configurations per model and dataset. All models are trained using PyTorch (Paszke et al., 2017).

A.3.1 UNet Architecture

The UNet architecture used in this work is based on the original design presented by Ronneberger et al. (2015b), with several modifications to tailor it to the specific needs of our task of predicting temperature fields. In this section, we introduce the key concepts that define the architecture and explain how they contribute to the model's design. These concepts will be employed in the hyperparameter search to identify the most suitable configuration, taking into account memory and data limitations.

The UNet architecture essentially consists of a series of UNet blocks. Each block consists of the layers of "Convolution - Activation - Convolution - Norm - Activation - Convolution - Activation" with convolutional layers, a batch normalization layer, and activation functions. After each block, either an upsampling or a downsampling operation is applied via "Max Pooling 2D" or "Transposed Convolution 2D" with stride 2. The *depth* of the UNet refers to the number of UNet blocks in both the encoder and decoder. The *number of initial features* refers to the number of feature maps generated by the first downsampling block. Each downsampling block produces twice as many output feature maps as input feature maps, while each upsampling block reduces the number of feature maps by half. The *number of convolutions per block* denotes how many convolutional layers are applied within each block, while *kernel size* specifies the size of the kernels used in the convolutional operations. Additionally, we explore different activation functions (ReLU, tanh, sigmoid, and LeakyReLU) and various normalization strategies (batch normalization, group normalization, and no normalization). The UNet block exists in 2 variants, depending on the hyperparameter *repeat inner*: If *repeat inner* = False, the block looks like this "Convolution - Norm - Activation", if it is True, the block looks as described above.

The training process also involves several hyperparameters. The learning rate controls how quickly the model adjusts its parameters during training. The weight decay parameter helps prevent overfitting by penalizing large weights in the model. Furthermore, the Adam optimizer (Kingma, 2014) is employed as *optimizer*. For the realistic permeability field, we additionally introduce the optimizable hyperparameter *optimizer switch*, which, when enabled, switches the optimizer to limited-memory BFGS (LBFGS) after 90 epochs.

During inference, each datapoint is processed as a whole, but during training, they are loaded in smaller patches cut out from the datapoint. Optimized hyperparameters include their overlap, i.e.,

inverse of *skip per direction*, and the patch size, called *box length*. The data split is untouched by this, i.e., separate simulation runs for train / val / test. This is important to not overlap test patches with training ones.

A.3.2 ENERGY-BASED LOSS FOR PINNS

For our PINN, we overfitted a UNet with an energy loss $\mathcal{L}_{\text{energy}}$ to a single datapoint from *3dp*.

The physics-informed loss for predicting the temperature field T considers energy conservation, including advective and conductive terms, and heat pump contributions. Let $\vec{v} = (v_x, v_y)$ be the velocity field and p the pressure. Denote the predicted temperature as $\hat{T}$. The energy loss is defined as

$$\mathcal{L}_{\text{energy}} = \text{MSE}\Big(E - E^{\text{HP}}_{\text{conv}},\, 0\Big), \tag{5}$$

where E is the local energy residual and E^{HP} the energy inflow from heat pumps.

Energy residual. The energy residual accounts for advection and thermal diffusion:

$$\rho_m = \text{molar density}(\hat{T}, p), \tag{6}$$

$$h = \text{enthalpy}(\hat{T}, p), \tag{7}$$

$$E_u = \frac{\partial}{\partial x}\Big(\rho_m\, h\, v_x - \frac{\kappa}{\Delta x}\frac{\partial \hat{T}}{\partial x}\Big), \tag{8}$$

$$E_v = \frac{\partial}{\partial y}\Big(\rho_m\, h\, v_y - \frac{\kappa}{\Delta x}\frac{\partial \hat{T}}{\partial y}\Big), \tag{9}$$

$$E = E_u + E_v, \tag{10}$$

where κ is the thermal conductivity and Δx is the uniform grid spacing.

Heat pump contribution. Heat pumps add localized energy sources based on a one-hot-encoding of the positions of heat pumps (ID) to get a spatially distributed contribution:

$$\rho = \text{density}(\hat{T}, p), \tag{11}$$

$$E^{\text{HP}} = c_p\, \rho\, \Delta T\, Q\, \frac{\text{ID}}{\Delta x^3}, \tag{12}$$

$$E^{\text{HP}}_{\text{conv}} = K * E^{\text{HP}}, \tag{13}$$

where c_p is the specific heat capacity of water, ΔT is the temperature difference at the injection well, Q is the volumetric flow rate, ID is the material indicator (1 if heat pump present, 0 otherwise), and K is a fixed, handcrafted 3×3-convolutional kernel $K = \begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$ with zero-padding used to smooth the inflow contribution. It distributes the inflow over neighboring cells while preserving directional structure and its shape matches our observations.

A.3.3 HYPERPARAMETERS AND HYPERPARAMETER OPTIMIZATION

We optimize the hyperparameters of our architecture, training process and data loading in several rounds with an automated tree-based search using Optuna (Akiba et al., 2019) and additional manual tweaking. Optuna performs optimization using a tree-structured Parzen estimator algorithm. An overlook of all considered hyperparameters, their ranges and our final choice, as well as the hyperparameters fixed during optimization is provided per used model architecture and dataset, e.g., purely data-driven approaches, LGCNN on individual steps or the full pipeline, on datasets of random k versus realistic k.

Vanilla UNet on 3dp The vanilla UNet_{3dp} was trained using the following hyperparameters: a batch size of 20, kernel size of 5, and a network depth of 3. The number of initial features was set to 32, with a stride and dilation of 1. We used the ReLU activation function in combination with batch normalization. The inner block was not repeated (`repeat_inner = False`). No cutouts are applied during training (`bool_cutouts = False`), i.e., the model is trained on the whole datapoint at once.

The inputs to the model are pki (pressure field, permeability field, and location of heat pumps). For training, the MAE loss function was used, and optimized with the Adam optimizer. The learning rate is set to 1×10^{-5}. The model is trained for 10,000 epochs.

UNet Variants on 101dp The values considered during hyperparameter optimization for the UNet and DDUNet, trained on 73 datapoints and performed using Optuna (Akiba et al., 2019), along with the best settings found, are listed in Table 2. Certain hyperparameters were fixed: we set the number of epochs to 750, with an early stopping criterion based on validation loss and a patience of 80 epochs. Additionally, we note that some hyperparameter combinations (e.g., 32 initial features, depth 6, and 3 convolutions per layer with a kernel size of 7) caused memory issues, leading to their exclusion from the hyperparameter search.

Table 2: Overview of used hyperparameters for the UNet_{101dp} and 2×2 DDUNet_{101dp} their search ranges (if applicable), and best values across training stages. Note that the number of communicated feature maps in the vanilla UNet is simply an extra convolution layer in the coarsest part of the UNet (without communication).

Hyperparameter	**Range**	$pki \rightarrow v_x v_y$ (Step 1)		$iv_x v_y ss_o k \rightarrow T$ (Step 3)		$pki \rightarrow T$ (Full)	
		UNet_{101dp}	$2 \times 2\text{DDUNet}_{101dp}$	UNet_{101dp}	$2 \times 2\text{DDUNet}_{101dp}$	UNet_{101dp}	$2 \times 2\text{DDUNet}_{101dp}$
Dataset							
Batch size (train)	4, 6, 8	4	6	6	4	6	6
Include pressure field	True, False	False	False	-	-	False	False
Encoder-decoder properties							
Depth	4, 5, 6	6	5	6	6	5	5
No. initial features	8, 16, 32	8	16	8	8	8	16
No. convs. per block	1, 2, 3	1	1	3	3	3	3
Kernel size	3, 5, 7	7	7	3	5	7	7
Communication Network							
No. comm. feature maps	64, 128, 256	64	128	256	64	256	256
Training							
Learning rate	[1e-5, 1e-3]	0.00024	0.00100	0.00017	0.00030	0.00024	0.00024
Weight decay	0.0, 0.001	0.0	0.0	0.0	0.0	0.0	0.0
Train loss	MSE, L1	MSE	MSE	L1	MSE	MSE	MSE

After the hyperparameter search, the values corresponding to the best-performing model (based on Huber loss for the validation dataset) were selected. With these values fixed, five models were trained using different randomly sampled initializations to evaluate sensitivity to random initialization, for these values see Tables 7 and 8.

UNet+++ and dilated UNet Hyperparameters of our UNet++ and dilated UNet trainings are listed in Table 3. As far as possible, we stuck to the previously excessively optimized, well-working architecture: For the dilated CNN, we based them on the vanilla UNet *3dp* and only varied the dilation in [2, 4, 8, 10, 16, 32, 100] with 10 producing the lowest validation error and hence listed here. UNet++ is based on the set of hyperparameters from the UNet++-paper (Zhou et al., 2018) with parameters that were adapted to fit our scenario listed below, based on the hyperparameter search of $\text{UNet}_1 01dp$. For example, the weight decay of the Adam optimizer is set to 0.0 and the training loss to MSE. The number of epochs for UNet++ is so small due to limited training time on the server.

PINN and FNO Hyperparameters of our PI-CNN and FNO trainings are listed in Table 4. For PI-CNN, we used the optimized, well-working architecture from before; for FNO a transferred set of hyperparameters with an additional hyperparameter search on layers, modes, inputs and optimizer.

Hyperparameters for LGCNN on random permeability The values considered during hyperparameter optimization with Optuna (Akiba et al., 2019) and the best settings found for both steps of LGCNN are listed in Table 5. Although the optimization was originally run for 100 epochs, the optimum was consistently found within the first 25 epochs. Therefore, to reduce computation cost,

Table 3: Baseline hyperparameters for end-to-end temperature prediction.

Parameter	**Dilated CNN (3dp)**	**UNet++ (101dp)**
Parameters of the Dataset		
dataset size	3	101
inputs	pik	ik
cutouts / patches	full domain	full domain
Parameters of Training		
batch size	20	2
training loss	MAE	MSE
initial learning rate	10^{-4} (manual decay)	2×10^{-4}
optimizer	Adam	Adam
epochs	1000	190
Parameters of the Network		
architecture	Dilated CNN (zero-padded)	UNet++
kernel size	5	*default*
depth	3	*default*
initial features	32	*default*
stride	1	*default*
dilation	10	*default*
activation function	ReLU	*default*
normalization	BatchNorm	*default*
repeat inner blocks	False	-

Table 4: Baseline hyperparameters for overfitting experiments on a single datapoint (dp id 0).

Parameter	**FNO (overfit test)**	**PINN (overfit test)**
Parameters of the Dataset		
dataset size	1	1
inputs	pik	$pikv_xv_y$
box length	480	256
Parameters of Training		
batch size	8	20
training loss	MSE	MSE (physics-only)
initial learning rate	10^{-3} (LBFGS)	10^{-4} (manual decay)
optimizer	LBFGS	Adam
epochs	1000	500
Parameters of the Network		
architecture	FNO	UNet (zero-padding)
modes / kernel size	64 modes	4×4
depth	6	4
initial features	16	32
stride	1	1
dilation	1	1
activation function	ReLU	ReLU
normalization	BatchNorm	BatchNorm
repeat inner blocks	False	False

we therefore conservatively lowered the maximum number of epochs to 50. This adjustment does not affect any of the reported metrics in the paper.

Table 5: LGCNN-Random k: Hyperparameter optimization: Parameter ranges and best configurations.

Parameter	**Range**	**Step 1** (v)	**Step 3** (T)
	Parameters of the Dataset		
inputs	$v : (p, i, k)$ $T : (i, v_x, v_y, s, s_o, k)$	pik	$iv_xv_yss_ok$
skip per direction	v : 4, 8, 16, 32, 64 T : 8, 16, 32, 64	16	8
box length	64, 128, 256, 512	256	256
	Parameters of Training		
loss function (training)	MAE, MSE	MSE	MAE
optimizer	Adam, SGD	Adam	Adam
	Parameters of the Network		
No. initial features	8, 16, 32, 64, 128	32	32
kernel size	3, 4, 5	5	4
depth	1, 2, 3, 4	4	4

Fixed parameters for this hyperparameter search are the learning rate (fixed at 10^{-4}), ReLU as activation function, the batch size of 20, and the use of a batch normalization layer within the inner blocks of the UNet architecture. The validation loss used for selecting the optimal model is the MAE.

Hyperparameters for LGCNN on real permeability fields The values considered during hyperparameter optimization on the dataset with a more realistic permeability field were selected using Optuna (Akiba et al., 2019), and are summarized in Table 6, along with the best configurations found for both steps of the LGCNN. The optimization was run for up to 100 epochs. For more background on the network architecture and the various hyperparameters, cf. A.3.1. Several hyperparameters were fixed during this process. These include a constant learning rate schedule, an Adam optimizer with a weight decay of 10^{-4}, and, when enabled, a switch to LBFGS after 90 epochs. Fixed architectural parameters include a convolutional stride and dilation of 1. During training, the inputs were cut out from the full datapoints. For model comparison, the validation loss was consistently computed using the Huber loss.

A.4 Additional Experimental Results

This section provides additional experimental results. While the main results section focused only on the test and scaling datasets, we also include here the metric values on training and validation datasets. For easier comparison, the test and scaling metrics are re-listed as well.

Vanilla UNet and DDUNet on *101dp*: Metrics of training and ablation study We present metrics for predicting the temperature field directly from the inputs pki using a data-driven approach, evaluated on the training, validation, and test datasets. The results are provided for two models: (1) UNet trained on 73 datapoints (73-18-10 train-validation-test split), and (2) DDUNet trained on the same 73 datapoints dataset, operating on 2×2 subdomains. To assess the model's sensitivity to random initialization, the training of the same architecture was repeated five times. Based on these repetitions, the mean and standard deviation of the performance metrics were computed using the following equations:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

Table 6: LGCNN-Real k: Hyperparameter optimization: Parameter ranges and best configurations.

Parameter	Range	model $\vec{v}$	model T
	Parameters of the Dataset		
inputs	$v : \nabla py, p, k, i$ $T : iv_x v_y ss_o k$	pik	$iv_x v_y ss_o k$
batch size	2, 4, 8, 16	8	8
skip per direction	256, 128, 64, 32, 16, 8	8	8
box length	1280, 640	1280	1280
	Parameters of Training		
loss function (training)	MSE, MAE	MSE	MSE
optimizer switch	True, False	False	False
learning rate	1e-3, 5e-4, 1e-4, 5e-5	1e-4	1e-4
	Parameters of the Network		
No. initial features	8, 16, 32	16	16
kernel size	3, 5	5	5
depth	4, 5, 6	6	6
repeat inner	True, False	False	False
activation function	relu, tanh, sigmoid, leakyrelu	relu	relu
layer norm	batch-, group-, None	batch-	batch-

where x_i denotes the metric value from the i-th training run, and $n = 5$ is the number of runs. These results are summarized in Table 7. The choice of $n = 5$ was made empirically to balance computational effort and statistical reliability. The standard deviations in Table 7 were used as validation: they are neither excessively large (indicating instability) nor unrealistically small (indicating insufficient sampling).

In addition to testing the UNet and DDUNet trained on 73 datapoints on the *101dp* test dataset, we also evaluate these models on the same datapoint used to test the UNet_{3dp}.

Table 7: Statistical error metrics for end-to-end prediction of T with *101dp* UNet and DDUNet. Errors in [°C], MSE in [°C^2], and SSIM unitless. The LGCNN-test dataset corresponds to the 1 datapoint used for testing the LGCNN approach. Mean $\pm$ standard deviation reported.

Model	Data	Case	Huber	L_∞	MAE	MSE
UNet	*101dp*	train	0.0011 ± 0.0002	4.37 ± 0.20	0.0182 ± 0.0020	0.0023 ± 0.0004
		val	0.0055 ± 0.0003	4.36 ± 0.16	0.0454 ± 0.0019	0.0114 ± 0.0006
		test	0.0052 ± 0.0003	4.35 ± 0.23	0.0441 ± 0.0019	0.0110 ± 0.0006
	*	LGCNN-test	0.0049 ± 0.0002	4.30 ± 0.16	0.0470 ± 0.0010	0.0102 ± 0.0004
	*	scaling	0.0017 ± 0.0001	4.48 ± 0.15	0.0208 ± 0.0014	0.0035 ± 0.0002
DDUNet	*101dp*	train	0.0014 ± 0.0003	4.11 ± 0.25	0.0203 ± 0.0026	0.00300 ± 0.0007
		val	0.0079 ± 0.0002	4.20 ± 0.25	0.0564 ± 0.0008	0.01648 ± 0.0005
		test	0.0075 ± 0.0001	4.05 ± 0.22	0.0552 ± 0.0008	0.01580 ± 0.0002
	*	LGCNN-test	0.0057 ± 0.0003	4.00 ± 0.20	0.0526 ± 0.0015	0.01171 ± 0.0006
	*	scaling	0.0025 ± 0.0001	4.04 ± 0.20	0.0251 ± 0.0007	0.00514 ± 0.0002

Vanilla UNet on *3dp*: Metrics of training and ablation study We present metrics for predicting T directly from the inputs pki with UNet_{3dp} in Table 8, evaluated on training, validation, test and scaling datapoints. To assess the model's sensitivity to random initialization, the training of the same architecture was repeated five times. Other models were not trained again to reduce computational effort and because the results of all vanilla approaches indicate that the performance of UNet_{101dp}, UNet_{3dp}, and DDUNet_{101dp} can be viewed as a representative of the general performance of purely data-driven methods.

LGCNN: Metrics of training and ablation study The results of all steps/e2e, evaluated on both synthetic and realistic permeability fields, for the training, validation, testing, and scaling datasets are shown in Table 9.

LGCNN+random k: Performance of Step 1 The model generally obtains good results in Figure 3, even for cells that are far away from injection points.

Table 8: Statistical error metrics for predicting with different models and datasets. Errors in [°C] or [m/y], MoC in [%], and SSIM unitless. Mean ± standard deviation reported. *e2e* stands for end-to-end application.

Architecture	Data	Split	Scaling	Step	MAE	RMSE	Huber	SSIM	MoC
UNet[a]	synth.: 3dp	train	No	e2e	0.0155 ± 0.0064	0.0732 ± 0.0310	0.0025 ± 0.0021	0.9645 ± 0.0157	1.6770 ± 1.9093
UNet[a]	synth.: 3dp	val	No	e2e	0.1114 ± 0.0021	0.2177 ± 0.0039	0.0219 ± 0.0007	0.6251 ± 0.0067	6.9180 ± 0.7236
UNet[a]	synth.: 3dp	test	No	e2e	0.1043 ± 0.0016	0.1917 ± 0.0035	0.0176 ± 0.0006	0.6239 ± 0.0048	7.7775 ± 1.5033
LGCNN[b]	synth.: 3dp	train	No	e2e	0.0720 ± 0.0012	0.1736 ± 0.0019	0.0137 ± 0.0003	0.7993 ± 0.0044	0.0933 ± 0.0430
LGCNN[b]	synth.: 3dp	val	No	e2e	0.1026 ± 0.0014	0.2095 ± 0.0032	0.0204 ± 0.0006	0.6812 ± 0.0061	1.6610 ± 0.0797
LGCNN[b]	synth.: 3dp	test	No	e2e	0.0965 ± 0.0013	0.1866 ± 0.0030	0.0167 ± 0.0005	0.6724 ± 0.0059	0.1443 ± 0.0388
LGCNN[b]	synth.: 3dp	test	Yes	e2e	0.0430 ± 0.0007	0.1249 ± 0.0016	0.0072 ± 0.0002	0.7908 ± 0.0151	0.0951 ± 0.0317
UNet	synth.: 3dp	train	No	3	0.0354 ± 0.0007	0.0770 ± 0.0024	0.0029 ± 0.0002	0.9084 ± 0.0032	0.1516 ± 0.0203
UNet	synth.: 3dp	val	No	3	0.0491 ± 0.0005	0.0916 ± 0.0008	0.0042 ± 0.0001	0.8251 ± 0.0033	0.1970 ± 0.0381
UNet	synth.: 3dp	test	No	3	0.0452 ± 0.0005	0.0829 ± 0.0011	0.0034 ± 0.0001	0.8381 ± 0.0040	0.1938 ± 0.0907
UNet	synth.: 3dp	test	Yes	3	0.0179 ± 0.0005	0.0399 ± 0.0007	0.0008 ± 0.0000	0.8809 ± 0.0138	0.1148 ± 0.0497
UNet	synth.: 3dp	train	No	1 vx	14.6004 ± 10.3108	16.7442 ± 9.9912	14.1096 ± 10.3058	0.9884 ± 0.0020	-
UNet	synth.: 3dp	train	No	1 vy	14.7965 ± 6.3568	17.9164 ± 6.3728	14.3037 ± 6.3526	0.9868 ± 0.0051	-
UNet	synth.: 3dp	val	No	1 vx	28.7237 ± 6.9461	39.5616 ± 5.9981	28.2283 ± 6.9444	0.9734 ± 0.0025	-
UNet	synth.: 3dp	val	No	1 vy	27.8241 ± 2.0130	39.7497 ± 2.7446	27.3289 ± 2.0125	0.9743 ± 0.0020	-
UNet	synth.: 3dp	test	No	1 vx	27.4355 ± 5.3603	36.6786 ± 4.8168	26.9402 ± 5.3588	0.9727 ± 0.0025	-
UNet	synth.: 3dp	test	No	1 vy	30.8755 ± 5.5264	42.5114 ± 6.4231	30.3799 ± 5.5256	0.9734 ± 0.0042	-
UNet	synth.: 3dp	test	Yes	1 vx	28.3820 ± 5.8382	38.2430 ± 4.8683	27.8864 ± 5.8367	0.9770 ± 0.0017	-
UNet	synth.: 3dp	test	Yes	1 vy	28.0927 ± 2.1725	39.6320 ± 2.8592	27.5972 ± 2.1719	0.9717 ± 0.0025	-
LGCNN[b]	real: 4dp	train	No	e2e	0.0550 ± 0.0025	0.0960 ± 0.0014	0.0046 ± 0.0001	0.6744 ± 0.0249	1.8951 ± 0.6214
LGCNN[b]	real: 4dp	val	No	e2e	0.0827 ± 0.0029	0.1553 ± 0.0031	0.0120 ± 0.0005	0.6337 ± 0.0195	2.0679 ± 0.4008
LGCNN[b]	real: 4dp	test	Yes	e2e	0.0486 ± 0.0075	0.0714 ± 0.0056	0.0026 ± 0.0004	0.6257 ± 0.0396	2.7049 ± 0.6239
UNet	real: 4dp	train	No	3	0.0213 ± 0.0051	0.0295 ± 0.0061	0.0005 ± 0.0002	0.8895 ± 0.0208	1.9966 ± 0.2618
UNet	real: 4dp	val	No	3	0.0239 ± 0.0023	0.0395 ± 0.0034	0.0008 ± 0.0001	0.8982 ± 0.0169	2.9922 ± 0.3038
UNet	real: 4dp	test	Yes	3	0.0218 ± 0.0042	0.0326 ± 0.0044	0.0005 ± 0.0001	0.8734 ± 0.0190	1.6682 ± 0.2054
UNet	real: 4dp	train	No	1 vx	22.0019 ± 5.7393	27.7180 ± 6.8232	21.5061 ± 5.7380	0.9731 ± 0.0280	-
UNet	real: 4dp	train	No	1 vy	13.4570 ± 9.2203	16.8320 ± 10.8123	12.9699 ± 9.2128	0.9607 ± 0.0629	-
UNet	real: 4dp	val	No	1 vx	24.8187 ± 4.0315	29.6583 ± 4.7384	24.3217 ± 4.0310	0.9641 ± 0.0220	-
UNet	real: 4dp	val	No	1 vy	12.9956 ± 9.9800	15.4509 ± 10.3446	12.5082 ± 9.9727	0.9485 ± 0.0748	-
UNet	real: 4dp	test	Yes	1 vx	122.781 ± 32.645	135.767 ± 36.810	122.282 ± 32.644	0.9128 ± 0.0511	-
UNet	real: 4dp	test	Yes	1 vy	25.5754 ± 6.2873	34.6947 ± 8.2358	25.0801 ± 6.2858	0.9445 ± 0.0834	-

[a] newly trained with other validation loss for comparability in this table
[b] with fixed first step predictions

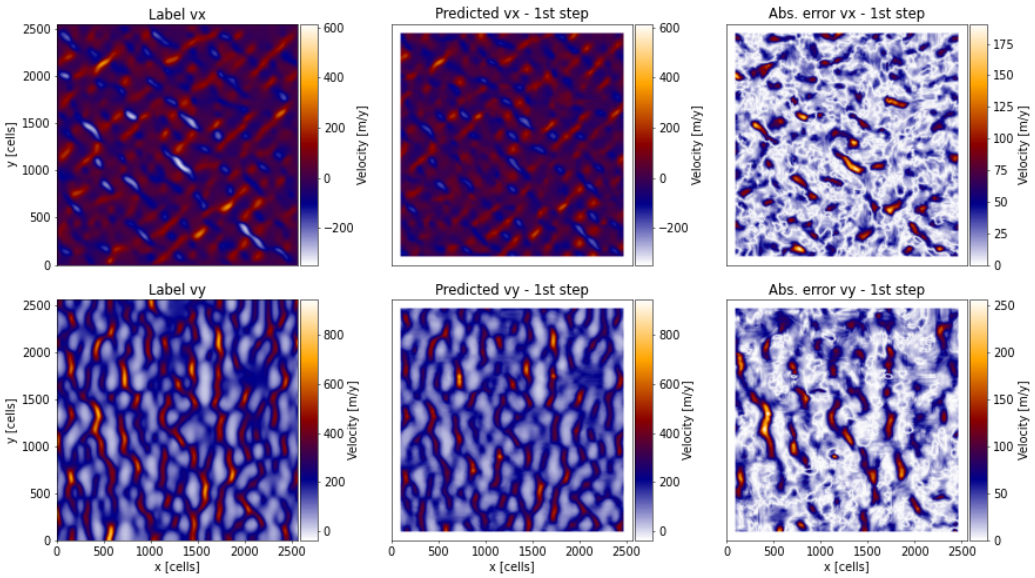

Figure 6: Prediction of $\vec{v}$ with LGCNN. Left: Test datapoint with labels v_x and v_y. Middle: Step 1 prediction for v_x, v_y. Right: Error of the v_x, v_y prediction.

Table 9: Performance metrics for predicting with different models and datasets. Errors in [°C] or [m/y], MoC in [%], and SSIM unitless.

Architecture	Data	Split	Scaling	Step	MAE	RMSE	Huber	SSIM	MoC
UNet	synthetic: 3dp	train	No	end-to-end	0.1181	0.5537	0.0558	0.8454	2.29
UNet	synthetic: 3dp	val	No	end-to-end	0.1930	0.5795	0.0742	0.6515	11.32
UNet	synthetic: 3dp	test	No	end-to-end	0.1908	0.6190	0.0736	0.6671	10.85
DilatedCNN	synthetic: 3dp	train	No	end-to-end	0.1180	0.2494	0.0270	0.5483	11.28
DilatedCNN	synthetic: 3dp	val	No	end-to-end	0.1273	0.2513	0.0287	0.5183	10.67
DilatedCNN	synthetic: 3dp	test	No	end-to-end	0.1200	0.2236	0.0237	0.5004	10.95
PINN (overfit)	synthetic: 1dp	train	No	3	0.7532	0.8593	0.3454	0.1793	0
LGCNN	synthetic: 3dp	train	No	end-to-end	0.0657	0.1640	0.0121	0.8209	0.08
LGCNN	synthetic: 3dp	val	No	end-to-end	0.0965	0.1996	0.0184	0.6933	1.54
LGCNN	synthetic: 3dp	test	No	end-to-end	0.0916	0.1738	0.0146	0.6841	0.09
LGCNN	synthetic: 3dp	test	Yes	end-to-end	0.0413	0.1189	0.0065	0.7911	0.06
UNet	synthetic: 3dp	train	No	1 (vx)	9.9620	13.0768	9.4732	0.9868	-
UNet	synthetic: 3dp	val	No	1 (vx)	22.7179	33.2020	22.2241	0.9748	-
UNet	synthetic: 3dp	test	No	1 (vx)	22.3178	31.1860	21.8237	0.9739	-
UNet	synthetic: 3dp	test	Yes	1 (vx)	24.9261	34.7004	24.4314	0.9784	-
UNet	synthetic: 3dp	train	No	1 (vy)	11.8902	16.6018	11.4005	0.9890	-
UNet	synthetic: 3dp	val	No	1 (vy)	27.1026	39.0450	26.6078	0.9770	-
UNet	synthetic: 3dp	test	No	1 (vy)	32.7444	45.0703	32.2488	0.9733	-
UNet	synthetic: 3dp	test	Yes	1 (vy)	26.2795	38.2599	25.7847	0.9761	-
UNet	synthetic: 3dp	train	No	3	0.0282	0.0632	0.0020	0.9376	0.11
UNet	synthetic: 3dp	val	No	3	0.0452	0.0869	0.0038	0.8454	0.11
UNet	synthetic: 3dp	test	No	3	0.0417	0.0762	0.0029	0.8540	0.07
UNet	synthetic: 3dp	test	Yes	3	0.0168	0.0373	0.0007	0.8895	0.08
UNet, in sequence	synthetic: 3dp	train	No	3	0.0402	0.1039	0.0052	0.9019	0.08
UNet, in sequence	synthetic: 3dp	val	No	3	0.1013	0.2025	0.0191	0.6682	1.15
UNet, in sequence	synthetic: 3dp	test	No	3	0.0919	0.1695	0.0139	0.6714	1.30
UNet, with zero-padding	synthetic: 3dp	train	No	3	0.0440	0.0788	0.0031	0.8104	0.44
UNet, with zero-padding	synthetic: 3dp	val	No	3	0.0548	0.0908	0.0041	0.7546	0.70
UNet, with zero-padding	synthetic: 3dp	test	No	3	0.0487	0.0795	0.0031	0.7784	0.57
UNet, no partitioning	synthetic: 3dp	train	No	1 (vx)	3.6714	4.6374	3.1983	0.9959	-
UNet, no partitioning	synthetic: 3dp	val	No	1 (vx)	25.7367	37.7048	25.2418	0.9764	-
UNet, no partitioning	synthetic: 3dp	test	No	1 (vx)	23.8584	33.8297	23.3646	0.9765	-
UNet, no partitioning	synthetic: 3dp	train	No	1 (vy)	3.5177	4.6358	3.0496	0.9971	-
UNet, no partitioning	synthetic: 3dp	val	No	1 (vy)	30.1247	42.7471	29.6294	0.9762	-
UNet, no partitioning	synthetic: 3dp	test	No	1 (vy)	35.9249	49.9040	35.4289	0.9721	-
UNet, no partitioning	synthetic: 3dp	train	No	3	0.0323	0.0736	0.0027	0.9245	0.14
UNet, no partitioning	synthetic: 3dp	val	No	3	0.0449	0.0861	0.0037	0.8333	0.12
UNet, no partitioning	synthetic: 3dp	test	No	3	0.0429	0.0793	0.0031	0.8349	0.29
UNet	real: 4dp	train	No	1 (vx)	14.1819	18.4573	13.6890	0.9915	-
UNet	real: 4dp	val	No	1 (vx)	15.4095	19.5032	14.9187	0.9869	-
UNet	real: 4dp	test	Yes	1 (vx)	110.0078	116.4903	109.5079	0.9436	-
UNet	real: 4dp	train	No	1 (vy)	9.5619	11.2571	9.0680	0.9945	-
UNet	real: 4dp	val	No	1 (vy)	10.6605	12.1757	10.1675	0.9932	-
UNet	real: 4dp	test	Yes	1 (vy)	17.6051	24.8298	17.1118	0.9905	-
UNet	real: 4dp	train	No	3	0.0139	0.0212	0.0002	0.9361	1.88
UNet	real: 4dp	val	No	3	0.0175	0.0319	0.0005	0.9405	3.30
UNet	real: 4dp	test	Yes	3	0.0189	0.0287	0.0004	0.8942	1.77
LGCNN	real: 4dp	train	No	end-to-end	0.0534	0.0997	0.0049	0.6900	2.76
LGCNN	real: 4dp	val	No	end-to-end	0.0841	0.1659	0.0137	0.6511	1.65
LGCNN	real: 4dp	test	Yes	end-to-end	0.0394	0.0665	0.0022	0.6755	3.50

LGCNN+random k: Experiment on Inputs to Step 3 Additional tests show that excluding other inputs, either i alone or both i and k, raises prediction error by 58–121%. We also evaluated alternative time-stepping schemes for solving the IVP. Replacing the 5th-order implicit Runge–Kutta method with explicit 2nd- or 4th-order schemes accelerates computation, but increases prediction error by 16–23%—a moderate degradation compared to the complete removal of streamline inputs. Nonetheless, we retain the implicit scheme for its superior accuracy and stability. Quantitative and qualitative results for the predictions are shown in Figure 7 and Table 10.

Table 10: Experiment on 3rd step: Test metrics for predicting T with different input combinations. Errors in [°C], MSE in [°C^2], and SSIM unitless.

Inputs	Huber	L_∞	MAE	MSE	SSIM
ikv_xv_y	0.0070	2.2990	0.0712	0.0139	0.7662
ikv_xv_ys	0.0041	1.8674	0.0545	0.0083	0.8368
$ikv_xv_ys_o$	0.0057	2.6623	0.0598	0.0114	0.8423
$ikv_xv_yss_o$ (not faded)	0.0072	2.3039	0.0744	0.0144	0.7837
$ikv_xv_yss_o$[a]	0.0031	1.8364	0.0442	0.0062	0.8828
$v_xv_yss_o$	0.0066	2.1301	0.0647	0.0132	0.8681
$kv_xv_yss_o$	0.0049	2.0925	0.0587	0.0097	0.8732
explicit RK, order 4	0.0038	3.2636	0.0486	0.0076	0.8871
explicit RK, order 2	0.0036	2.2034	0.0463	0.0072	0.8830

[a] new run to be comparable to the others in this experiment: trained with Huber validation loss, hence the results differ slightly wrt. to Table 1.

LGCNN+realistic k: Performance of 3rd step and full pipeline The qualitative performance is observable in Figure 8, where we see coherent streamlines and plume structures for both the isolated 3rd step and the full pipeline.

Training and inference times Table 11 summarizes the training and inference times, number of epochs, and dataset splits (train:val:test) for each of the three steps in our pipeline, both for the LGCNN trained on partitioned and full datasets. Although both approaches exhibit similar inference times, they show significant differences in training time, with the partitioned approach yielding better results.

Table 11: Training measurements on *3dp* dataset.

	Data Points (train:val:test)	Epochs[a]	Training Time[a]	Inference Time
1st Step (partitioned)	20,736:1:1	19	9.5 min	0.02 s
1st Step (full)	1:1:1	9,688	92.6 min	0.02 s
2nd Step	1:1:1	-	-	9.82 s
3rd Step (partitioned)	82,944:1:1	14	31.5 min	0.03 s
3rd Step (full)	1:1:1	9,671	92.1 min	0.02 s

[a]Early stopping: measurements until best validation loss.

In Table 12, the number of epochs and the total training time for the data-driven approaches are shown. For Step 1, both the UNet and DDUNet need many epochs and comparable training time to converge; however, for the third step and the full pipeline, the DDUNet significantly reduces both the number of epochs and the total training time required to reach convergence.

A.5 Hardware Specifications

The vanilla models that were trained on the large data-driven dataset of 101 samples were trained and evaluated on a server using NVIDIA V100 GPUs with 32 GB memory. All training was conducted using PyTorch 2.1.0 with CUDA 11.6 acceleration.

Training and evaluation of all other models (dilated UNet, LGCNN single steps, full pipeline and ablation studies, FNOs, PINNs) were performed on a single NVIDIA A100-SXM4 GPU. Data generation was carried out on a dual-socket system equipped with AMD EPYC 9274F CPUs.

A.6 Glossary

A list of the most relevant physical properties used in our paper is provided in Table 13.

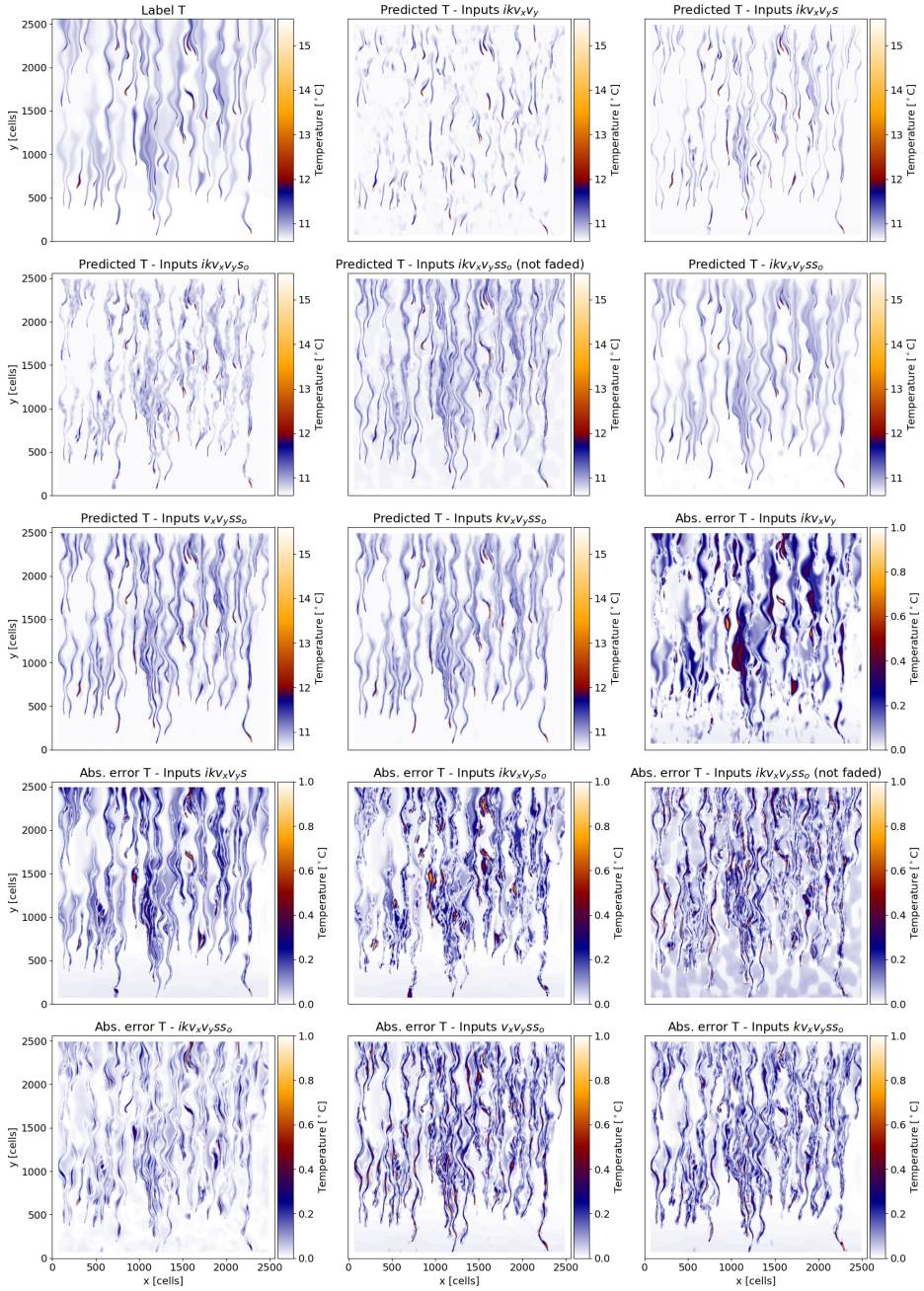

Figure 7: 1st Column: Label, input k, 2nd-5th: Without (s, s_o), (s_o), (s), not-faded streamlines (s, s_o), 6th: include all inputs, 7th-8th: Without (i, k), (i). Absolute errors capped at 1°C for better visualizations. Maximum errors are listed in Table 10.

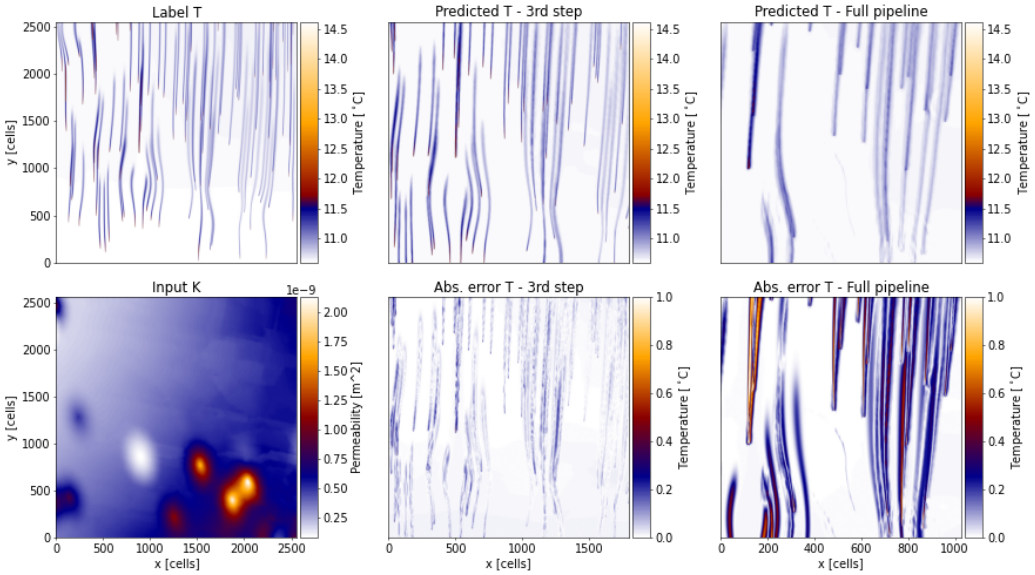

Figure 8: 1st Column: Label, input k. 2nd Column: 3rd step prediction of T and error with respect to the label. 3rd Column: Predicted T and error of the full pipeline.

Table 12: Training measurements for the data-driven approaches trained on the *101dp* dataset: UNet$_{101dp}$ and 2×2 DDUNet$_{101dp}$.

	Epochs[a]	**Training Time**[a]
1st Step		
UNet$_{101dp}$	738	5.497 hours
2×2 DDUNet$_{101dp}$	735	4.787 hours
3rd Step		
UNet$_{101dp}$	726	8.343 hours
2×2 DDUNet$_{101dp}$	303	3.508 hours
Full Pipeline		
UNet$_{101dp}$	267	3.680 hours
2×2 DDUNet$_{101dp}$	97	1.222 hours

[a]Early stopping: measurements until best validation loss.

Table 13: Glossary of Abbreviations.

Abbr.	**Parameter**
t	time
$X(t_0)$	property X at initial time
$X(t_{\text{end}})$	property X at quasi steady-state
X_{pred}	predicted property X
X_{sim}	simulated property X
i	positions of heat pumps
Q_{inj}	injected mass rate
ΔT_{inj}	injected temperature difference
k	permeability
p	hydraulic pressure
$g = \nabla p$	hydraulic pressure gradient
$\vec{v} = (v_x, v_y)$	flow velocity
$\vec{s}$	both streamline fields
s	central streamlines, starting at all i
s_o	streamlines starting with $\pm$ 10 cells transversal offset compared to s
T	temperature