Sim2Real for Environmental Neural Processes

Jonas Scholz
University of Cambridge
Cambridge, UK
js2731@cam.ac.uk

Tom R. Andersson
British Antarctic Survey
Cambridge, UK
tomand@bas.ac.uk

Anna Vaughan
University of Cambridge
Cambridge, UK
av555@cam.ac.uk

James Requeima
Vector Institute,
Toronto, ON, Canada
james.requeima@vectorinstitute.ai

Richard E. Turner
University of Cambridge
& Microsoft Research
ret26@cam.ac.uk

Abstract

Machine learning (ML)-based weather models have recently undergone rapid improvements. These models are typically trained on gridded reanalysis data from numerical data assimilation systems. However, reanalysis data comes with limitations, such as assumptions about physical laws and low spatiotemporal resolution. The gap between reanalysis and reality has sparked growing interest in training ML models directly on observations such as weather stations. Modelling scattered and sparse environmental observations requires scalable and flexible ML architectures, one of which is the convolutional conditional neural process (ConvCNP). ConvCNPs can learn to condition on both gridded and off-the-grid context data to make uncertainty-aware predictions at target locations. However, the sparsity of real observations presents a challenge for data-hungry deep learning models like the ConvCNP. One potential solution is 'Sim2Real': pre-training on reanalysis and fine-tuning on observational data. We analyse Sim2Real with a ConvCNP trained to interpolate surface air temperature over Germany, using varying numbers of weather stations for fine-tuning. On held-out weather stations, Sim2Real training substantially outperforms the same model architecture trained only with reanalysis data or only with station data, showing that reanalysis data can serve as a stepping stone for learning from real observations. Sim2Real could thus enable more accurate models for weather prediction and climate monitoring.

1 Introduction

Every day, millions of observations of the Earth system are collected by environmental sensors, such as in-situ weather stations, satellites, aircraft, oceanographic buoys, and meteorological balloons [1]. However, the spatial distance between neighbouring observations can be very large for certain variables, particularly in remote regions like Antarctica or the Himalayas. This presents a challenge for flexible deep learning models that require an abundance of data to learn realistic physical behaviour. As a result, deep learning systems developed for environmental prediction tasks often use gridded *reanalysis* datasets for training, rather than observations. Reanalysis data is produced by assimilating observations into a dynamical model to predict geophysical quantities (such as temperature) on a regular spatiotemporal grid [1]. Data-driven models trained with reanalysis data have recently made striking progress in weather forecasting [2–6].

Tackling Climate Change with Machine Learning: workshop at NeurIPS 2023.

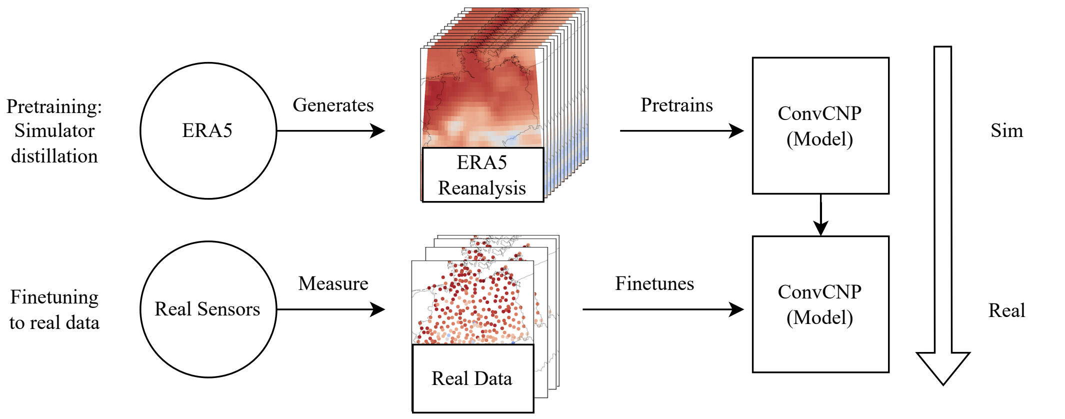


Figure 1: In the Sim2Real process, a ConvCNP is pre-trained on abundant simulator data from the ERA5 reanalysis dataset, and then fine-tuned on limited real weather station data.

A limitation of training with reanalysis target data is the mismatch between the dynamical model simulator and reality, owing to several factors, including:

- error between physical laws in the simulator and true physics,
- spatiotemporal coarseness (typically $\sim 25\,\mathrm{km}$ spatially and $\sim 1\,\mathrm{hr}$ temporally),
- not capturing real-world observation aleatoric uncertainty (e.g. sensor noise).

We call the combination of the above mismatches the 'Sim2Real gap'. Despite rapid advances in reanalysis-trained large machine learning (ML) weather forecasting models, such as GraphCast [2] and Pangu-Weather [6], the Sim2Real gap will degrade the utility of these models to some degree. A recent approach to overcoming the Sim2Real gap in weather forecasting is MetNet-3 [7], which forecasts both simulator and observation data in a multi-task setting. An important open question is if accurate ML weather models can be trained purely on environmental observations, perhaps bypassing the data assimilation step of numerical simulators and overcoming the Sim2Real gap.

This paper investigates Sim2Real transfer for Convolutional Conditional Neural Processes (ConvCNPs) [8]. ConvCNPs are flexible deep learning models that produce probabilistic predictions over $\mathbf{y}_T$ at target locations X_T conditioned on context observations $(X_C, \mathbf{y}_C)$ by computing a marginal Gaussian mean and variance at each target location. The negative log-likelihood (NLL) loss can then be calculated in closed form and directly minimised via backpropagation (Appendix C). Convolutional neural process variants have shown promising results in weather modelling tasks such as downscaling [9] and sensor placement [10]. The Sim2Real gap motivates training a ConvCNP to interpolate real station data, which could provide more realistic estimates of observation informativeness and suggest better sensor placements than a reanalysis-only ConvCNP. In this work, to quantify the benefits of Sim2Real, a ConvCNP is first pre-trained on vast amounts of reanalysis data, and then a smaller but higher-quality real weather station dataset is used to fine-tune and evaluate the model (Fig. 1).

2 Spatial Temperature Interpolation

In this experiment, a ConvCNP is trained to spatially interpolate 2-metre air temperature context observations at a given time instant $(X_C, \mathbf{y}_C)$ to predict target values $\mathbf{y}_T$ at the same time instant at arbitrary target locations X_T. ERA5 reanalysis [1] is used as the simulator data and weather stations from the German weather service (DWD) [11] are used as the real data. To aid predictions, the ConvCNP receives a second context set with high-resolution elevation data from the NASA Shuttle Radar Topography Mission [12], as well as the time of day, day of year, and normalised latitude/longitude. These auxiliary features enable the model to learn spatiotemporal non-stationarities in the data (Appendix C).

During pre-training, tasks are generated by randomly sampling ERA5 grid cells for the context and target. During fine-tuning, off-the-grid DWD weather station observations are randomly split into context and target sets. 53 DWD stations are held-out during training/validation and are used to

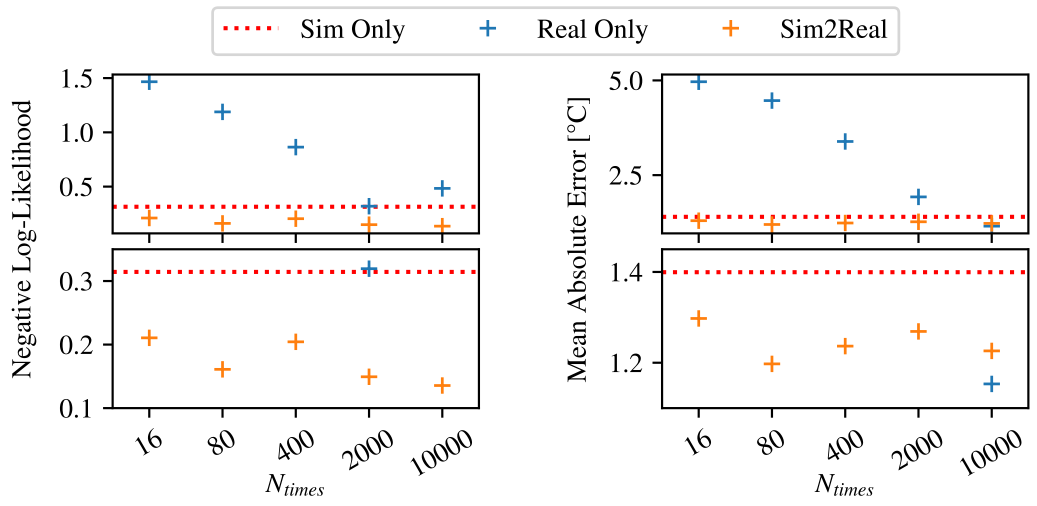


Figure 2: Sim2Real outperforms Real Only and Sim Only baselines significantly when $N_{stations} =$ 500. The bottom row is a zoomed-in version of the top row.

evaluate the generalisation abilities of the trained models. The held-out stations are taken from the VALUE protocol, which carefully selects stations that cover a wide range of geographic environments [13] (Appendix D). Different data-availability regimes are analysed, both spatially in terms of the number of stations available for training/validation $N_{stations} \in \{20, 100, 500\}$ and temporally in terms of the number of time slices used for training $N_{times} \in \{16, 80, 400, 2000, 10000\}$. Two baselines are used: 'Sim Only' (training the ConvCNP only to interpolate ERA5 and applying it directly to DWD data) and 'Real Only' (training the ConvCNP only to interpolate the DWD data). Different Sim2Real adaptation methods were investigated, mostly focusing on global fine-tuning of all parameters, and FiLM adaptation [14] (Appendix A).

3 Results

Global fine-tuning substantially outperforms FiLM adaptation in the ERA5 $\rightarrow$ DWD Sim2Real experiment. To adapt to higher-frequency correlations, we hypothesise the adaptation strategy must update the ConvCNP's convolutional filters (the smallest DWD station separation is $\sim 5\times$ smaller than the ERA5 grid spacing). FiLM adaptation does not update the CNN weights, so it struggles to adapt to shorter length scale features in the real data. This hypothesis is validated in a separate Sim2Real toy experiment using 1D Gaussian processes (Appendix B, Fig. B1).

Using global fine-tuning, Sim2Real produces substantially better NLLs (left) and mean absolute errors (MAEs) (right) than the Sim Only and Real Only baseline when there is a sufficiently large number and density of training stations ($N_{stations} = 500$). Sim2Real outperforms the Sim Only baseline even with a very small temporal window of station data ($N_{times} = 16$). However, the benefit of Sim2Real decreases as real data becomes more temporally abundant; the Real Only baseline's performance is comparable to that of Sim2Real at $N_{times} = 10000$ (Fig. 2). Furthermore, in the sparser station settings ($N_{stations} \in \{20, 100\}$), the fine-tuning stage of Sim2Real does not outperform the Sim Only pre-trained baseline as clearly (Appendix F).

The results in Figure 2 are noisy because only a single model was trained in each condition, and due to mismatches between testing and validation stations leading to poor early-stopping (Appendix D). Future work should average the results over more training runs with different random initialisations.

Sim2Real Learns High-Frequency Features When the ConvCNP is trained on ERA5, the shortest learnable correlation lengthscale is the grid spacing ℓ^{sim}_{min}. Fine-tuning the model on real data shortens ℓ_{min} to (roughly) the shortest inter-station separation, ℓ^{real}_{min}, which is a factor of 5 smaller than ℓ^{sim}_{min} for the densest DWD station setting of $N_{stations} = 500$. This leads to higher-frequency features in the model's predictions, modelling shorter-lengthscale weather phenomena (Fig. 3 a-d). The Sim Only

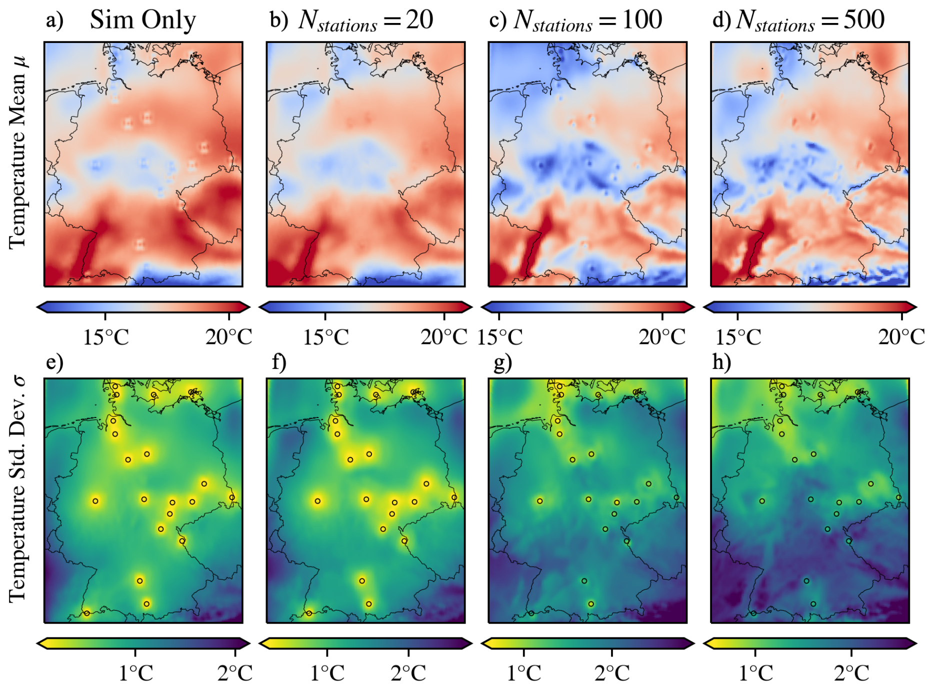


Figure 3: Increasing the density of training stations for fine-tuning increases the resolution of the ConvCNP's predictions. *a–d*, ConvCNP mean, μ. *e–h*, ConvCNP standard deviation, σ. Context weather stations are shown as black circles (not shown in *a–d* to reveal the spatial artefacts in μ).

ConvCNP produces overconfident marginal predictive uncertainties σ around context observations (Fig. 3 e). As $N_{stations}$ is increased, the model learns more appropriate conditioning behaviour for real observational data: in regions with multiple consistent measurements, uncertainty is reduced in a wide region (e.g. Northern Germany in Fig. 3 h), if nearby observations disagree, the Sim2Real ConvCNP inflates its uncertainty (e.g. central Germany). Furthermore, the average predictive uncertainties are greater. Even though Sim2Real yields these non-trivial predictive improvements, it is unable to remove the unphysical visual artefacts around context observations in Fig. 3 a-d, which we hypothesise are caused by the lack of training signal below ℓ_{min} (Appendix E).

4 Discussion

This paper explores Sim2Real with a ConvCNP in a $2\,\mathrm{m}$ temperature interpolation task over Germany, transferring from ERA5 reanalysis to real weather station observations. Our preliminary experiments paint a picture where Sim2Real is highly effective in a 'medium-data' regime. Too little real data and fine-tuning the pre-trained model has minimal effect (with the spatial data abundance being more important than the temporal abundance). Plenty of real data and the pre-training phase loses its value; starting from random initialisation is just as effective. Where Sim2Real is effective, it could bridge gaps in the finite observational data, improving downstream model usage such as active learning for sensor placement [10]. In future work, transfer learning from data-rich areas like Germany to data-sparse areas like the Himalayas or Antarctica could further alleviate data gaps and address socioeconomic disparities.

Our work identifies a limitation of ConvCNPs when training with spatially sparse data: the model can only make robust predictions in the range of length scales featured within the data. In particular, prediction artefacts appear on length scales shorter than the shortest context-target separation during training. This is likely the case for other ML models based on CNNs. Future work should explore architectural or training approaches to alleviate this.

We expect that Sim2Real can contribute to the rapidly developing future of data-driven weather and climate modelling.

Code availability

Code is made available at `https://github.com/jonas-scholz123/sim2real-downscaling`. This paper builds heavily on the `DeepSensor` [15] and `neuralprocesses` [16] packages.

Acknowledgments and Disclosure of Funding

We thank the two anonymous reviewers from the CCAI NeurIPS 2023 workshop for their helpful feedback. R.E.T. is supported by Google, Amazon, ARM, Improbable and EPSRC grant EP/T005386/1. T.R.A is supported by Wave 1 of The UKRI Strategic Priorities Fund under the EPSRC Grant EP/W006022/1, particularly the AI for Science theme within that grant and The Alan Turing Institute. A.V. acknowledges the UKRI Centre for Doctoral Training in the Application of Artificial Intelligence to the study of Environmental Risks (AI4ER), led by the University of Cambridge and British Antarctic Survey, and studentship funding from Google DeepMind.

References

[1] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

[2] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv e-prints*, pages arXiv–2212, 2022.

[3] Tung Nguyen, Johannes Brandstetter, Ashish Kapoor, Jayesh K Gupta, and Aditya Grover. Climax: A foundation model for weather and climate. *arXiv e-prints*, pages arXiv–2301, 2023.

[4] Kang Chen, Tao Han, Junchao Gong, Lei Bai, Fenghua Ling, Jing-Jia Luo, Xi Chen, Leiming Ma, Tianning Zhang, Rui Su, et al. Fengwu: Pushing the skillful global medium-range weather forecast beyond 10 days lead. *arXiv e-prints*, pages arXiv–2304, 2023.

[5] Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

[6] Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, pages 1–6, 2023.

[7] Marcin Andrychowicz, Lasse Espeholt, Di Li, Samier Merchant, Alex Merose, Fred Zyda, Shreya Agrawal, and Nal Kalchbrenner. Deep learning for day forecasts from sparse observations. *arXiv e-prints*, pages arXiv–2306, 2023.

[8] Jonathan Gordon, Wessel P Bruinsma, Andrew YK Foong, James Requeima, Yann Dubois, and Richard E Turner. Convolutional conditional neural processes. In *International Conference on Learning Representations*, 2019.

[9] Anna Vaughan, Will Tebbutt, J Scott Hosking, and Richard E Turner. Convolutional conditional neural processes for local climate downscaling. *Geoscientific Model Development*, 15(1): 251–268, 2022.

[10] Tom R Andersson, Wessel P Bruinsma, Stratis Markou, Daniel C Jones, J Scott Hosking, James Requeima, Alejandro Coca-Castro, Anna Vaughan, Anna-Louise Ellis, Matthew Lazzara, et al. Active learning with convolutional gaussian neural processes for environmental sensor placement. *Environmental Data Science*, 2023.

[11] Wetterdienst. Aktuelle stündliche stationsmessungen der lufttemperatur und luftfeuchte für deutschland., 2023.

[12] Tom G Farr, Paul A Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, et al. The shuttle radar topography mission. *Reviews of geophysics*, 45(2), 2007.

[13] Douglas Maraun, Martin Widmann, José M Gutiérrez, Sven Kotlarski, Richard E Chandler, Elke Hertig, Joanna Wibig, Radan Huth, and Renate AI Wilcke. Value: A framework to validate downscaling approaches for climate change studies. *Earth's Future*, 3(1):1–14, 2015.

[14] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[15] Tom Robin Andersson. DeepSensor: A Python package for modelling environmental data with convolutional neural processes, July 2023. URL `https://github.com/tom-andersson/deepsensor`.

[16] Wessel Bruinsma. Neuralprocesses. A framework for composing Neural Processes in Python., July 2023. URL `https://github.com/wesselb/neuralprocesses`.

[17] Jayesh K Gupta and Johannes Brandstetter. Towards multi-spatiotemporal-scale generalized pde modeling. *arXiv preprint arXiv:2209.15616*, 2022.

[18] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[19] 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*, pages 234–241. Springer, 2015.

[20] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Appendix

A Finetuning Approaches

Global Finetuning In *global* finetuning, all parameters of the pretrained model are trained on the finetuning dataset. Tuning all parameters gives the greatest degree of flexibility, but the high capacity also makes the model prone to overfitting, particularly if the finetuning dataset is small.

FiLM Adapters To reduce the model's susceptibility to overfit to small finetuning datasets, some parameters are commonly frozen and the remaining parameters are trained on the finetuning data. FiLM adaptation [14] is one data-efficient approach that has shown strong results within computer vision [14, 17].

FiLM adapters are affine transformations applied to individual feature maps $\mathbf{h}_i^{(l)}$ throughout the model:

$$\tilde{\mathbf{h}}_i^{(l)} = \gamma_i^{(l)} \times \mathbf{h}_i^{(l)} + \beta_i^{(l)}, \tag{A.1}$$

where $\gamma_i^{(l)}, \beta_i^{(l)}$ are the learnable FiLM parameters scaling and shifting the ith feature map of the lth layer, $i \in 1 \dots N_{feat}^{(l)}$. Because a FiLM layer contains only $2N_{feat}$ trainable parameters, it is relatively robust to overfitting. We apply FiLM adapters after every convolutional layer as shown in Fig. C1 (orange). During pre-training, we fix $\beta_i^{(l)} = 0, \gamma_i^{(l)} = 1$. During finetuning, we then freeze all other model parameters and train only the FiLM parameters.

B Synthetic Experiment: 1D Gaussian Progress Regression

In this synthetic experiment, we perform "Sim2Real" by generating both the "simulated" and the "real" data, both drawn from a noisy 1D Gaussian Process (GP) [18]. The GP uses a squared-exponential kernel with lengthscales ℓ^{sim}, ℓ^{real} and noise $\sigma_0^{sim}, \sigma_0^{real}$ for the "sim" and "real" data respectively. This experiment served as a stepping stone for rapid iteration and as a preliminary evaluation of adaptation methods, which partially translate to the main temperature downscaling experiments.

Baselines As baselines, we consider the ConvCNP trained in the "infinite" data regime, where we keep training it until convergence, and the 0-shot "Sim Only" baseline, where we apply the simulator-pretrained ConvCNP directly to the "real" data.

Shrinking Lengthscales In the first experiment, we keep noise fixed at $\sigma_0^{real} = \sigma_0^{sim} = 0.05$ and consider the transfer from $\ell^{sim} = 0.25$ to *shorter* lengthscale GPs, with ℓ^{real} of either 0.2, 0.1 or 0.05. This is analogous to the target domain of weather modelling, where some real measurement stations are closely separated ($\ell^{real} \sim 4\,\mathrm{km}$) and can therefore capture shorter lengthscale weather phenomena than the more coarsely gridded ERA5 simulator data with $\ell^{sim} \sim 20\,\mathrm{km}$ grid spacing.[1]

As shown in Fig. B1, both FiLM and global fine-tuning require only a very small number of tasks for effective adaptation to shorter lengthscales, when compared to the (poor) 0-shot baseline. In the more extreme $\ell = 0.25 \to 0.05$ transfer, global fine-tuning significantly outperforms FiLM. We hypothesize that the convolutional filters learned on the pre-training task extract features that are tuned to the particular $\ell^{sim} = 0.25$ – for less extreme changes in ℓ, FiLM adaptation is able to scale features to achieve similar performance to global fine-tuning, for much smaller lengthscales, the features extracted by the convolutional filters become less useful and fine-tuning the filters themselves becomes important.

Overall, this effect is smaller for smaller values of N_{tasks}^{real}. In these sparse-data regimes, the much lower capacity of FiLM adaptation makes the model less likely to overfit the "real" data. For $\ell^{real} \in (0.1, 0.2)$, and $N_{tasks}^{real} = 16$, this leads to FiLM slightly outperforming global finetuning.

[1]The analogy is not perfect, as long-lengthscale weather phenomena are still present in both real and simulator data, which is not the case for these GPs, but a part of the problem is captured nonetheless.

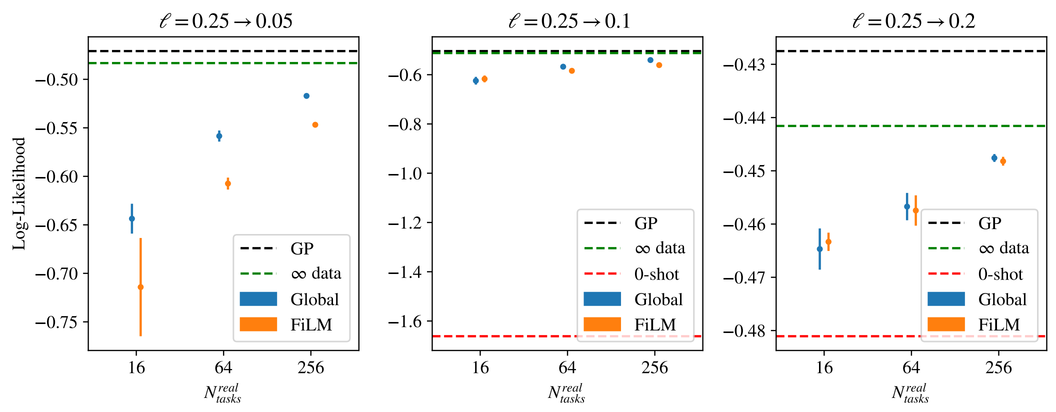


Figure B1: Test-set log-likelihoods achieved via fine-tuning on limited numbers of "real" tasks (x-axis). Global fine-tuning outperforms FiLM in the shrinking lengthscale experiments, particularly if the difference in lengthscales is large. FiLM performs slightly worse the more fine-tuning data are available. Error bars represent 95% confidence intervals and are computed by starting from the same pre-trained model and using different fine-tuning datasets. The 0-shot baseline in the left-most plot is ≈ -4.1 and is hidden to not distort the y-scale.

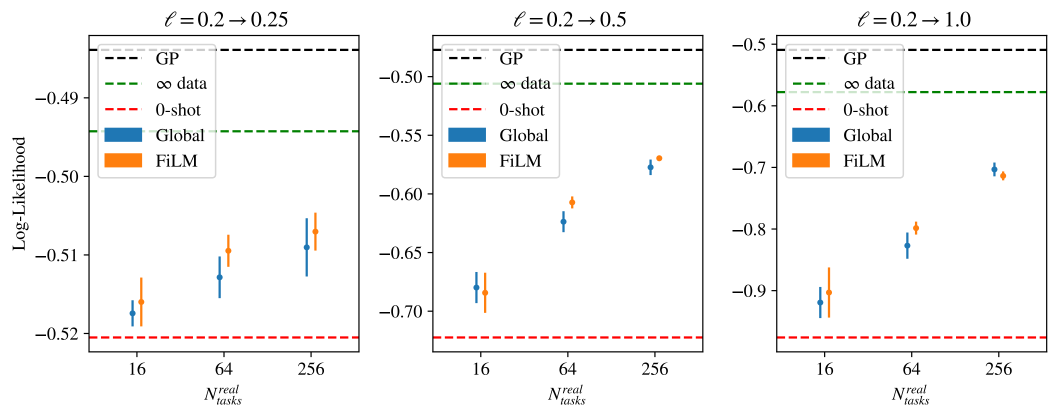


Figure B2: For growing lengthscales, FiLM outperforms global fine-tuning. Especially for sparse data settings and small Sim2Real gaps.

Growing Lengthscales If our hypothesis that FiLM layers cannot rectify low-resolution convolutional filters holds, we should see an improved FiLM performance in the inverse problem of growing lengthscales. This setting is less analogous to the real weather modelling experiment but is useful to include for generality and a more domain-agnostic approach to fine-tuning.

We therefore now consider $\ell^{sim} = 0.2$ and $\ell^{real} \in [0.25, 0.5, 1.0]$. As hypothesised, FiLM performs slightly better in this setting (Fig. B2), particularly in the sparse-data regime.

Noise Change Finally, we keep the lengthscales fixed at $\ell^{sim} = \ell^{real} = 0.25$, and instead change the level of *noise* from $\sigma_0^{sim} = 0.05$ both up to $\sigma_0^{real} \in [0.1, 0.2]$ and down to $\sigma_0^{real} \in [0.0125, 0.025]$.

This is analogous to our weather-based Sim2Real experiments, where we would expect our simulated data, ERA5 [1], to be associated with lower noise[2] than real measurements because

[2]By noise we do not mean (negligible) inaccuracies in the temperature measurement, but instead the aleatoric uncertainty due local weather phenomena (e.g. a cloud flying overhead at the time of measurement or a cold breeze passing by) that might lead to temperature changes on the order of seconds and metres, which are infeasible to model.

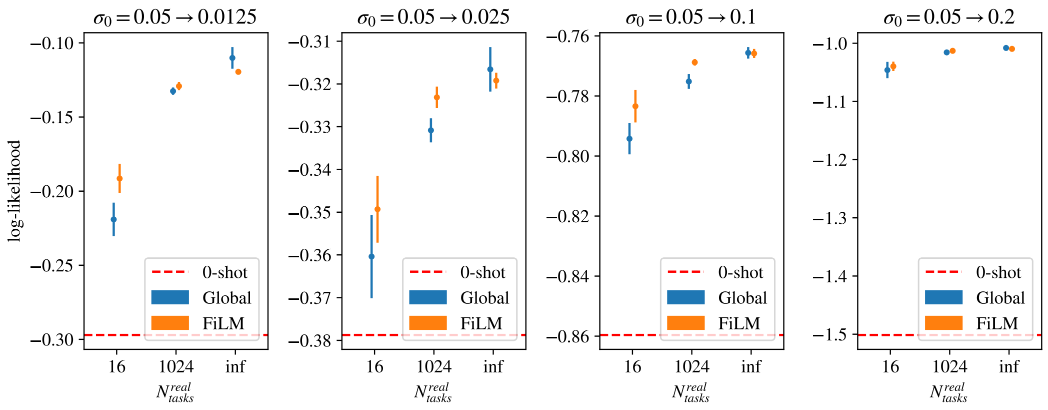


Figure B3: When adapting to different noise levels, FiLM adapters beat global fine-tuning decisively. Even in large data regimes, FiLM is only very slightly weaker.

- The simulation runs on a discrete spatiotemporal grid and therefore cannot model weather phenomena beyond its resolution.
- The data assimilation process can ingest multiple data points of observational data within one grid cell, smoothing out local measurement noise.

For generality, we both simulate an increase and a reduction in noise.

In Fig. B3, we show that in this regime FiLM outperforms global fine-tuning across different values of σ_0^{real}, even in the larger $N_{tasks}^{real} = 1024$ experiments. In the limit $N_{tasks}^{real} \rightarrow \infty$, global fine-tuning still outperforms (as it should), but not by a large margin.

These results align with our previous findings, that FiLM is a more sample-efficient fine-tuning method *unless* the frozen convolutional filters extract features of an insufficient resolution.

C Model Architecture and Experimental Details

Convolutional Conditional Neural Processes ConvCNPs [8] are spatiotemporal models with parameters θ that define the conditional distribution over target variables $\mathbf{y}_T$ at given target locations X_T as a Gaussian distribution

$$q_\theta(\mathbf{y}_T|X_T, C) = \mathcal{N}(\mathbf{y}_T; \boldsymbol{\mu}_\theta(C), \Sigma_\theta(C)). \tag{C.1}$$

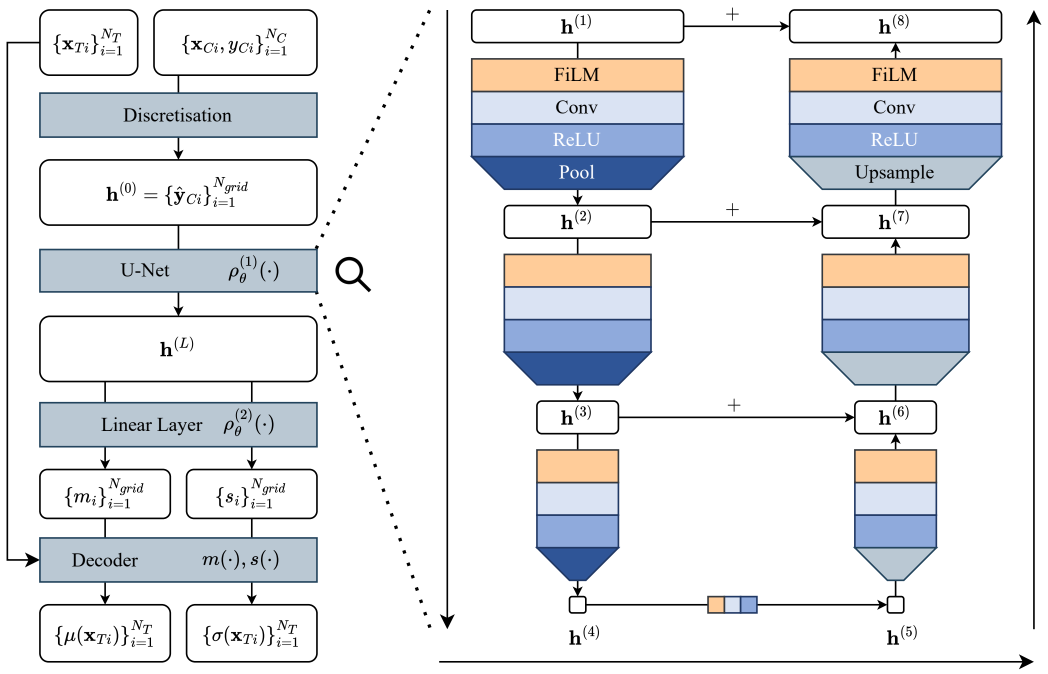
They model $\boldsymbol{\mu}$ and Σ by encoding context data $C = \{\mathbf{x}_{Ci}, y_{Ci}\}_{i=1}^{N_C}$, onto an internal gridded representation, processing that encoding into a (gridded) predictive mean and standard deviation using a Convolutional Neural Network (CNN) [20] ρ_θ, and finally decoding the gridded mean and standard deviation at any on or off-the-grid target locations. It is this CNN ρ_θ that we fine-tune in our experiments. The particular architecture we use is a U-Net [19], shown in Fig. C1.

ConvCNPs are trained by adjusting the parameters θ to minimise the Negative Log-Likelihood (NLL) of the predictive distribution over the training set, via backpropagation, to minimise the KL-Divergence between the approximate posterior predictive q_θ and the *true* posterior predictive.

Normalisation We normalise input data so that each of the two spatial coordinates is normalised to the range [0, 1]. Additionally, we normalise temperatures by subtracting their sample mean and scaling by their sample standard deviation. We save normalisation parameters during pre-training and use them during fine-tuning for consistency. This is performed using the *deepsensor* package [15].

Model Hyperparameters We use the model architecture shown in Fig. C1 with 6 layers (down and up) in the U-Net, of 96 channels each. We choose a resolution of 200 Points Per Unit[3] (PPU)

[3]Note that this means 200 $\times$ 200 grid-points per 1 $\times$ 1 unit square



$\{\mathbf{x}_{Ti}\}_{i=1}^{N_T}$
$\{\mathbf{x}_{Ci}, y_{Ci}\}_{i=1}^{N_C}$
Discretisation
$\mathbf{h}^{(0)} = \{\hat{\mathbf{y}}_{Ci}\}_{i=1}^{N_{grid}}$
U-Net $\rho_\theta^{(1)}(\cdot)$
$\mathbf{h}^{(L)}$
Linear Layer $\rho_\theta^{(2)}(\cdot)$
$\{m_i\}_{i=1}^{N_{grid}}$
$\{s_i\}_{i=1}^{N_{grid}}$
Decoder $m(\cdot), s(\cdot)$
$\{\mu(\mathbf{x}_{Ti})\}_{i=1}^{N_T}$
$\{\sigma(\mathbf{x}_{Ti})\}_{i=1}^{N_T}$
$\mathbf{h}^{(1)}$
+
$\mathbf{h}^{(8)}$
FiLM
Conv
ReLU
Pool
FiLM
Conv
ReLU
Upsample
$\mathbf{h}^{(2)}$
+
$\mathbf{h}^{(7)}$
$\mathbf{h}^{(3)}$
+
$\mathbf{h}^{(6)}$
$\mathbf{h}^{(4)}$
$\mathbf{h}^{(5)}$

Figure C1: The ConvCNP architecture, using a U-Net [19] (and linear layer) as ρ_θ. Coloured boxes with sharp corners are the processing steps and rounded white boxes are the different states along the processing pipeline.

for the internal gridded representation and a corresponding encoder and decoder lengthscale $\ell_E = \ell_D = 1/200$, which allows us to comfortably resolve the smallest station separation ($\sim 4\,\text{km}$) in normalised space. In total, our model has 3.8 million parameters, out of which 3284 (0.08%) are FiLM parameters.

Optimisation We use the Adam optimiser [21], with a learning rate of 1×10^{-4} during pre-training and 3×10^{-5} during fine-tuning, both of which were found via grid search. We use a batch size of 16 throughout. Because each point in time yields a very large combination of context and target-set combinations (which are related but distinct) an "epoch" in the traditional sense is far too large to be useful. We instead define an epoch as 200 batches (i.e. $200 \times 16 = 3200$ tasks) during pre-training. We anneal the learning rate by a factor of 3 if the validation loss stalls for more than 8 epochs, which helps for convergence at the end of training. We stop after 20 epochs without improvement. During fine-tuning, we define epochs to be smaller, each consisting of 25 batches, to monitor validation losses more frequently. This definition of epoch also allows for consistency across different N_{times} and $N_{stations}$. During fine-tuning, we stop after 30 such "epochs" without improvement.

D Data Splitting and Sampling

Spatiotemporal modelling makes splitting data significantly more complex than it is in most traditional ML domains. Generating training, validation and testing sets is not as simple as splitting all available data randomly. The most important problem is that the model can overfit both spatially and temporally – both of which can leak into the test/validation sets unnoticed unless care is taken.

The data-splitting process is further complicated by the fact that we're exploring different data splits to investigate different data-availability regimes.

Test Data We split the available real data along two dimensions: time and stations. For consistent testing, we set aside the stations from the VALUE experimental protocol [13]. It provides a standardised set of experiments to evaluate downscaling methods and has a specific selection of stations in Europe, of which we select the 53 German stations. These stations cover a wide range of geographic

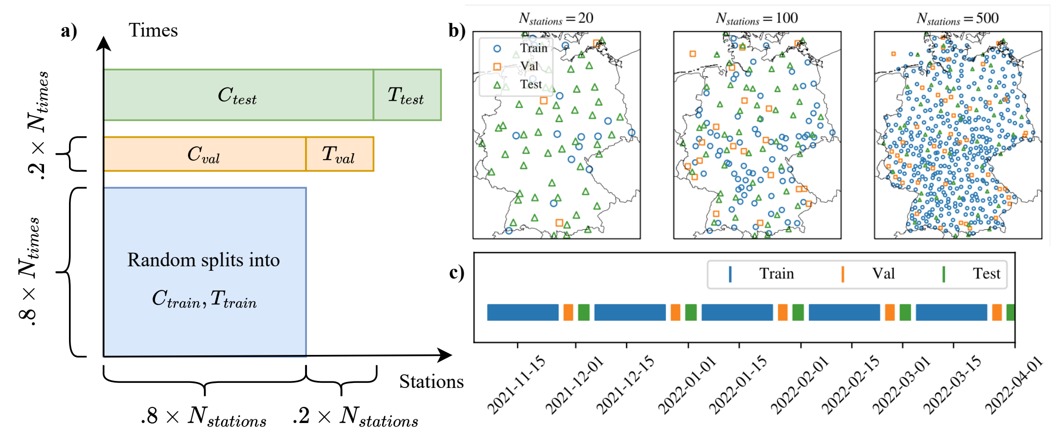


Figure D1: a) The N_{times} times and $N_{stations}$ stations each get split into 80% training and 20% validation data. For validation, we use training stations as context. For final testing, we use all available stations (train and val) to achieve the best results. The set of *test* times and stations is set aside at the beginning and is used for all evaluations. b) Train/Val/Test stations for different $N_{stations}$. c) Train/Val/Test dates are sampled throughout the available period with 2 days discarded between to avoid leakage.

features and are commonly used in downscaling experiments [9]. Because some stations available in VALUE are covered by different copyright permissions than the DWD stations, they are not available to us. In those (9 out of 53 stations) cases, we instead choose the geographically closest DWD station. The furthest discrepancy from the VALUE stations is $\sim 18\,\mathrm{km}$, with most distances below $10\,\mathrm{km}$.

D.1 Training and Validation Stations

The Train/Val stations are selected in a random order that we then keep fixed so that the stations in the $N_{stations} = 20$ experiments are a subset of the stations used for $N_{stations} > 20$ experiments, which ensures that no information is lost as we increase $N_{stations}$ (Fig. D1 b)). We also investigated choosing stations that are as far apart from each other as possible but found that this significantly hurts what the model can learn, as shorter lengthscale signals are only featured for large $N_{stations}$ in this scenario.

D.2 Training and Validation Times

To avoid distribution shift between the Train/Val/Test tasks due to macroscale changes (e.g. climate change, el Niño/la Niña events etc.), we sample times throughout the available range, cycling between 19 days of training data, 2 days of validation data, 2.5 days of testing data, each separated by 2 days that we discard to avoid partial leakage due to correlated tasks, (Fig. D1 c). In this approach, we broadly follow [7], but we increase the number of discarded days from 1 to 2 to further reduce leakage. When we restrict ourselves (artificially) to a limited number of times, we select a random subset of times from the Train/Val pool that we keep fixed across experiments.

D.3 Generating Tasks

Once we have selected $N_{stations}, N_{times}$, we try to imitate what we would do for best model performance on downstream applications, with the limited numbers of stations and times available. In such a scenario, we want to maximise the model performance using all $N_{stations}$ available stations.

Given our restricted $N_{stations}$ and N_{times}, we follow this procedure (visualised in Fig. D1 a):

1. Split $N_{stations}$ into 80% training and 20% validation stations.
2. Split N_{times} into 80% training and 20% validation times.
3. For training, generate random subsets of context and target sets C_{train}, T_{train} only from the set of training stations and times. A single task is drawn as follows:

(a) Select a random point in time t from the training times *without replacement* so that each t will be encountered equally often during training.

(b) Draw a fraction $r \sim U(0, 1)$.

(c) Denoting the number of observations at time t as $N_{stations}(t)$, we select a random subset of size $r^2 \times N_{stations}(t)$ as the *context set* C. The squaring of r makes sparser tasks more probable, which we find accelerated training for large values of $N_{stations}$.

(d) Use the remaining stations as the target set T.

(e) Note that this means a very large number of distinct (but related) tasks can be drawn from a single time t, as any combination of context stations is a “distinct" task.

4. For validation, use *all* available training stations as context C_{val} and all available validation stations as target T_{val} on unseen times from the validation times.
5. For testing, use *all* available stations, training *and* validation stations as context C_{test}, and the test stations (at test times) as targets T_{test}. This corresponds to the real-world scenario of using all available stations (train and val) for application. However, this *does* mean the model is tested in a regime that it has not encountered during training (a greater number of context stations).

E Short-Range Artefacts

A limitation associated with training a ConvCNP using a gridded simulator is the fact that the ConvCNP never receives a training signal shorter than the grid spacing of the simulator, ℓ_{min}. Attempting to predict at higher resolutions than ℓ_{min} can lead to clear visual artefacts (Fig. E1, red smears around context observations). The model is unable to interpolate the context observations smoothly.

These artefacts occur because the model's loss is never punished for predicting them: if the closest separation between two observations is given by ℓ_{min}, the model's predictions on lengthscales $\ell < \ell_{min}$ have no effect on the loss. The model has not encountered signals of $\ell < \ell_{min}$ in the data and is not endowed with any prior knowledge of temperatures on short scales, it is also unable to extrapolate to shorter lengthscales from the available $\ell > \ell_{min}$ data.

These artefacts are most egregious for Sim Only trained models (where ℓ_{min} is comparatively large), but still they remain visible after Sim2Real (Fig. 3), where ℓ_{min} is effectively given by the shortest inter-station separation.

We believe that artefacts are particularly visible because of the U-Net model we use (Fig. C1). The residual connections mean that context observations can pass through the model unmitigated. The single linear layer separating the U-Net from decoding is insufficient for “smoothing out" the unmitigated artefacts. Expanding the linear layer into a multi-layer perceptron might help mitigate these artefacts, which we believe is worth exploring in future work.

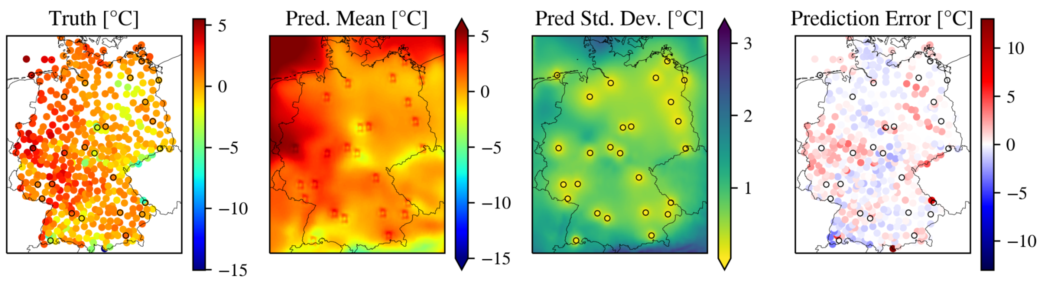


Figure E1: A ConvCNP which has been pre-trained on ERA5 simulator data and not yet fine-tuned, predicting temperature using real context measurements. Immediately around context observations, the model predicts temperatures that clearly do not align with surrounding predictions.

F Full Germany Temperature Station Interpolation Sim2Real Experiment Results

In Fig. F1, we compare the performance of our Sim2Real transferred model to that trained solely (from random initialisation) on available real data. Clearly, the initialisation at simulator-pretrained parameters is very helpful for training the model in all but the largest real data regime ($N_{times} = 10000$), regardless of $N_{stations}$, showing the utility of Sim2Real in low-data regimes.

Fig. F2 shows how the Sim2Real models compare to the Sim Only baseline. We see qualitatively different behaviours in different data availability regimes:

- In the sparse-station setup $N_{stations} = 20$, the model is unable to improve through Sim2Real, no matter the quantity of tasks (N_{times}).
- In the dense-station setup $N_{stations} = 500$, the model does improve significantly, even given very small amounts of real data (e.g. $N_{times} = 16$).
- In the middling station density $N_{stations} = 100$, the model does improve slightly given enough tasks $N_{tasks} \gtrsim 400$.

These results show that Sim2Real is not always useful, especially when the real data covers a much smaller density of context and target points than the simulator data. However, given even a modestly sized real dataset, Sim2Real can yield significant model improvements.

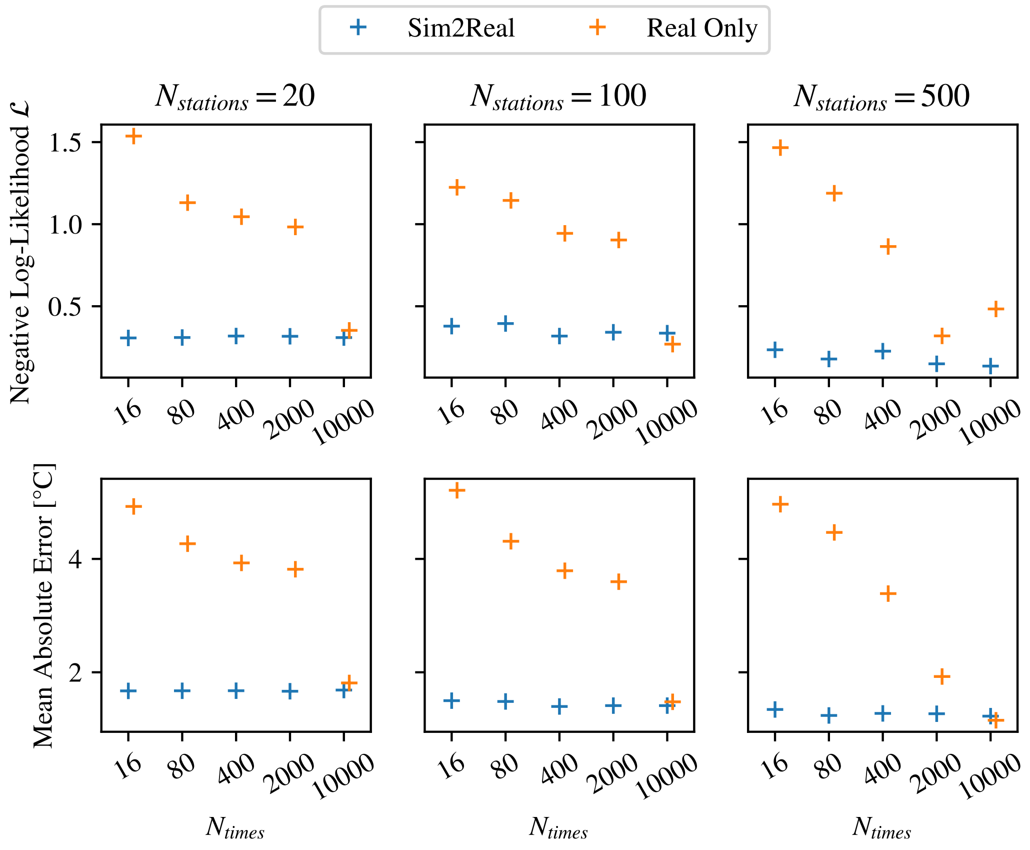


Figure F1: pre-training a model on simulator data significantly aids performance compared to starting from random initialisation. Only with large amounts of real data do the performances become comparable. Note: Fine-tuned model performances (blue) do change with additional training data, but given the scale of the y-axis, this is better shown in Fig. F2.

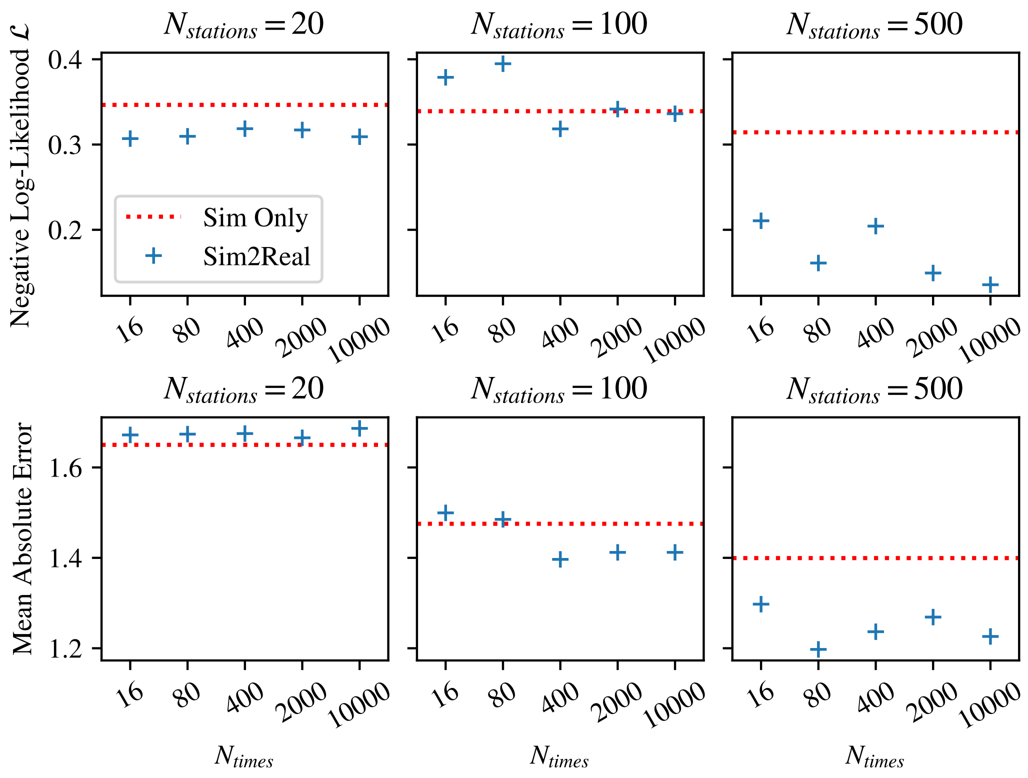


Figure F2: Fine-tuning is only useful when the fine-tuning data is sufficiently different from the simulator. In the $N_{stations} = 20$ regime, the model is unable to improve beyond the simulator, even after many training tasks. In higher data-availability regimes, fine-tuning leads to clear improvements.