INFERENCE FROM REAL-WORLD SPARSE MEASUREMENTS

SUPPLEMENTARY MATERIAL

Anonymous authors
Paper under double-blind review

1 DETAILED RESULTS

Here, we present a breakdown of the detailed results for the various models, presented in terms of the specific metrics reported in their original study. This is in contrast to the result table found in the main paper, where all models are reported in terms of RMSE. In table 2 of the main paper, we provide the results for GEN and CNP based on their original implementations, which do not involve sharing the position encoding. However, for TFS and MSA, we include the results obtained with shared position encoding. To ensure comprehensiveness and to differentiate the enhanced performance attributed to the latent representation from that resulting from the improved data encoding approach, we also apply the shared encoding technique to GEN and CNP.

In most cases, we observe that employing a shared encoding for position tends to improve the performance of all models, except for CNP where it seems to decrease the overall performance in some cases. Additionally, it seems that MSA surpasses its competitors in the majority of instances, consistently exhibiting superior performance across various data sizes. Moreover, when the shared encoding for positions fails to achieve the optimal performance, the model's performance is generally comparable to that without shared encoding, often falling within a standard deviation range.

2 DATASET DESCRIPTIONS

2.1 WIND NOWCASTING

The wind nowcasting experiment uses the same dataset as Pannatier et al. (2021). It is available at: `https://zenodo.org/record/5074237`. It is an important task for ATC as it involves the crucial prediction of high-altitude winds using real-time data transmitted by airplanes. This prediction is essential for ensuring efficient airspace management the airspace. It is important to note that in the vertical direction (z), the majority of wind measurements are typically taken at elevated altitudes, specifically between 4,000 and 12,000 meters. Regarding the horizontal dimensions, the airspace extends over 600 km from north to south and approximately 500 km from east to west. As ground-based measurements are not accessible at these heights, our dependence lies on the measurements gathered directly from airplanes. As planes do not record the wind in the z direction, the input space corresponds to wind speed in the x, y plane $\mathbb{I} \subseteq \mathbb{R}^2$, and the output space is the wind speed measured later, $\mathbb{O} \subseteq \mathbb{R}^2$. Here the underlying space is European airspace represented as $\mathbb{X} \subseteq \mathbb{R}^3$ In this particular setup, the models need to extrapolate in time, based on a set of the last measurements.

Airplanes measure wind speed with a sampling frequency of four seconds. We split the dataset into time slices, where each slice contains one minute of data, such that each time slice contains between 50 and 1500 data points. We tried different time intervals but noticed that having a longer time slice did not improve the quality of the forecasts. The objective for the different models is to output a prediction of the wind at different query points 30 minutes later. We evaluate performance using the Root Mean Square Error (RMSE) metric, as in previous work.

2.2 POISSON EQUATION

This experiment uses the same dataset as Alet et al. (2019). It is available at `https://github.com/FerranAlet/graph_element_networks/tree/master/data`. The Poisson equation models the heat diffusions over the unit square with sources represented by $\phi(x, y) \in \mathbb{R}$ and fixed boundary conditions $\omega \in \mathbb{R}$. The equation is given by:

$$\begin{cases}\Delta\phi(x,y)=\psi(x,y) & \text{if } (x,y)\in(0,1)^2\\ \phi(x,y)=\omega & \text{if } (x,y)\in\partial[0,1]^2\end{cases} \tag{1}$$

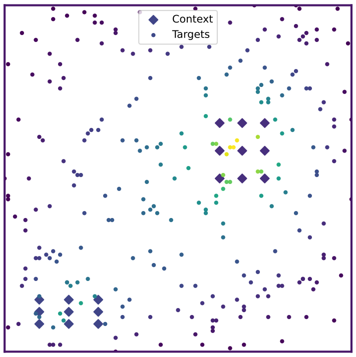


Figure 1: A sample of the Poisson equation dataset Alet et al. (2019). ● represents the targets, The context values are comprised of points on the boundaries and points on the sources and sink are represented by ◆ which corresponds to their thermal coefficient.

Table 1: Results of the High-Altitude Wind Nowcasting, Poisson, Navier Stokes and Darcy Flow equation and the weather forecasting task. Each model ran for respectively 10, 2000, 1000, 100, and 100 epochs on an NVIDIA GeForce GTX 1080 Ti. The low number of epochs for wind nowcasting is due to the amount of data which is considerably larger than in the other experiments. The standard deviation is computed over 3 runs. We choose the configuration of the models so that every model has a comparable number of parameters. We underlying the best models for each size and we put in bold the best model overall. In this table, we kept the results in the same metric as originally reported. The first half of the table corresponds to the case without the novel encoding scheme, as opposed to the second one. In table 2 of the main paper, we recomputed all metrics in terms of Root Mean Square Error (RMSE).

		Wind Nowcasting		**Poisson Equation**		**Navier Stokes Equation**		**Darcy Flow Equation**		**ERA 5**	
Model	**Size**	**Train RMSE (↓)**	**Val RMSE (↓)**	**Train MSE (↓)**	**Val MSE (↓)**	**Train MSE (↓)**	**Val MSE (↓)**	**Train RMSE (↓) `E-4`**	**Val RMSE (↓)**	**Train MSE (↓)**	**Val MSE (↓)**
Default encoding for positions											
CNP	5k	11.94± 0.78	10.19± 0.21	.127± .0179	.130± .0134	.485± .0033	.492± .0033	9.02± 0.54	9.65± 0.47	4.33± 0.01	4.48± 0.01
	20k	10.19± 1.83	10.11± 0.20	.084± .0057	.105± .0012	.438± .0015	.452± .0015	7.88± 0.16	8.71± 0.11	4.25± 0.01	4.44± 0.00
	100k	10.17± 1.24	10.20± 0.26	.021± .0040	.105± .0024	.398± .0010	.430± .0010	6.93± 0.08	8.17± 0.06	4.17± 0.01	4.43± 0.01
GEN	5k	11.02± 3.19	9.84± 2.92	.011± .0006	.017± .0011	.362± .0012	.365± .0012	8.49± 0.17	9.25± 0.18	4.32± 0.02	4.47± 0.02
	20k	9.98± 0.76	9.24± 0.35	.006± .0006	.020± .0054	.354± .0007	.359± .0007	7.76± 0.11	8.74± 0.09	4.24± 0.01	4.44± 0.00
	100k	9.56± 0.21	9.23± 0.44	.011± .0122	.048± .0389	.339± .0006	.355± .0006	7.01± 0.26	8.66± 0.09	4.10± 0.01	4.46± 0.00
TFS (Ours)	5k	9.86± 0.21	8.75± 0.14	.022± .0021	.055± .0248	.365± .0033	.367± .0033	6.50± 0.60	7.46± 0.55	4.29± 0.02	4.46± 0.00
	20k	9.69± 0.38	8.70± 0.06	.005± .0008	.017± .0014	.353± .0019	.357± .0019	5.48± 0.10	6.69± 0.14	4.08± 0.03	**4.38± 0.01**
	100k	9.55± 0.19	8.67± 0.07	.001± .0001	.016± .0045	.314± .0011	.350± .0011	4.45± 0.17	7.47± 0.24	3.78± 0.01	4.42± 0.01
MSA (Ours)	5k	8.86± 0.01	8.40± 0.10	.022± .0062	.048± .0186	.359± .0057	.361± .0054	6.91± 0.50	7.74± 0.48	4.28± 0.02	4.44± 0.02
	20k	7.94± 0.03	8.47± 0.12	.005± .0012	.047± .0107	.341± .0042	**.346± .0042**	5.72± 0.11	6.76± 0.10	4.13± 0.01	4.39± 0.01
	100k	6.67± 0.02	8.98± 0.22	.000± .0001	.030± .0081	.310± .0015	.350± .0015	4.91± 0.17	7.33± 0.22	4.19± 0.01	4.41± 0.01
Sharing encoding for positions											
CNP	5k	10.41± 0.03	10.33± 0.19	.154± .0010	.165± .0023	.703± .0001	.706± 0.0003	16.94± 0.06	17.26± 0.04	4.38± 0.00	4.53± 0.01
	20k	9.48± 0.02	10.12± 0.21	.133± .0004	.160± .0011	.700± .0001	.706± 0.0001	16.68± 0.10	17.06± 0.06	4.30± 0.01	4.47± 0.00
	100k	8.60± 0.05	10.26± 0.24	.107± .0008	.158± .0006	.696± .0001	.705± 0.0001	16.36± 0.02	16.87± 0.02	4.23± 0.01	4.43± 0.01
GEN	5k	10.04± 0.13	8.79± 0.23	.005± .0004	.017± .0023	.369± .0007	.372± 0.0008	5.93± 0.15	6.83± 0.17	4.42± 0.01	4.54± 0.01
	20k	9.75± 0.09	9.08± 0.47	.003± .0007	.018± .0036	.362± .0002	.366± 0.0000	5.65± 0.06	6.75± 0.05	4.36± 0.01	4.50± 0.01
	100k	9.84± 0.02	9.07± 0.00	.001± .0000	.019± .0008	.345± .0005	.361± 0.0000	4.97± 0.22	7.19± 0.12	4.34± 0.02	4.50± 0.01
TFS (Ours)	5k	8.78± 0.02	8.30± 0.03	.009± .0015	.025± .0121	.364± .0024	.365± 0.0026	6.74± 0.93	7.58± 0.84	4.41± 0.01	4.53± 0.01
	20k	7.94± 0.03	8.20± 0.04	.002± .0002	.010± .0013	.352± .0012	.355± 0.0009	5.56± 0.10	**6.64± 0.14**	4.26± 0.01	4.45± 0.01
	100k	6.57± 0.02	8.38± 0.13	.000± .0000	.035± .0054	.312± .0006	.349± 0.0015	4.51± 0.24	7.25± 0.19	4.09± 0.01	4.41± 0.00
MSA (Ours)	5k	8.54± 0.03	8.07± 0.11	.008± .0024	.014± .0014	.355± .0013	.357± 0.0013	6.74± 0.61	7.51± 0.61	4.38± 0.03	4.52± 0.03
	20k	7.73± 0.02	**7.98± 0.03**	.002± .0003	**.007± .0004**	.342± .0014	.347± 0.0016	5.79± 0.34	6.74± 0.38	4.25± 0.01	4.44± 0.01
	100k	6.58± 0.05	8.18± 0.14	.000± .0000	.010± .0017	.310± .0008	.347± 0.0008	4.76± 0.12	6.97± 0.24	4.14± 0.02	4.40± 0.01

It should be noted that the boundary constant and sources function $\phi(x, y)$ conditions
can change for each sample.

The dataset used in Alet et al. (2019) uses three dimensions for the context values $\mathbf{c}_y \in \mathbb{I} \subseteq \mathbb{R}^3$: either sources values $\phi(x, y) = \mu$ inside the domain encoded as $\mathbf{c}_x, \mathbf{c}_y = (x, y), (\mu, 0, 0)$ or boundary conditions $\phi(x, y) = \omega$ on the boundaries, encoded as $\mathbf{c}_x, \mathbf{c}_y = (x, y), (0, \omega, 1)$. The target space is one-dimensional $\mathbf{t}_y \in \mathbb{O} \subseteq \mathbb{R}$ and corresponds to the solution of the Poisson equation at that point. A sample is depicted in fig. 1.

We ran all models for three different sizes on this particular setup and found that MSA with the novel encoding scheme outperform other models by a significant margin as can be seen in table 1. We initialized GEN with a 7 × 7 regularly initialized grid on the $[0, 1]^2$ as in the original work Alet et al. (2019).

2.3 NAVIER-STOKES EQUATION

Similarly to the darcy flow task, we adapted the Navier-Stokes equation as described in Li et al. (2021) to an irregular setup.

The Navier-Stokes equation describes a real fluid and is described with the following PDE :

$$\begin{cases} \partial_t w(x,t) + u(x,t) \cdot \nabla w(x,t) = \nu \Delta w(x,t) + f(x), x \in (0,1)^2 & \text{if } t \in (0,T] \\ \nabla \cdot u(x,t) = 0, x \in (0,1)^2 & \text{if } t \in [0,T] \\ w(x,0) = w_0(x) & \text{if } x \in (0,1)^2 \end{cases} \tag{2}$$

For more detailed information regarding the notation, please refer to the original work by Li et al. Li et al. (2021), specifically section 5.3.

In this study, our objective is to forecast the future state of the vorticity quantity, specifically 50 time steps ahead, based on measurements of vorticity at different spatial locations. This approach differs from the original work, where the model was provided with ten initial vorticity grids and required to predict the complete system evolution.

To create our dataset, we subsampled the complete dataset, which consisted of the evolution of the Navier-Stokes equation solved by a numerical solver. The full dataset had dimensions of 1000 × 1024 × 1024.

For our purposes, we selected pairs of slices that were separated by 50 timesteps and performed spatial subsampling. We took 64 context measurements and 256 targets as our subsampled data points. This experiment adapts the dataset available in Li et al. (2021). It is available at `https://github.com/neural-operator/fourier_neural_operator` and can be processed with the code given to rearrange it in the irregular setup.

2.4 DARCY FLOW

We evaluated the performance of various models in solving the Darcy Flow equation on the unit grid with null boundary conditions, as described in Li et al. (2021). Specifically, we aimed to predict the value of the function u, given the diffusion function a, with the two related implicitly through the PDE given by:

$$\begin{cases} -\nabla \cdot (a(x) \nabla u(x)) = 1 & \text{if } x \in (0,1)^2 \\ u(x) = 0 & \text{if } x \in \partial [0,1]^2. \end{cases} \tag{3}$$

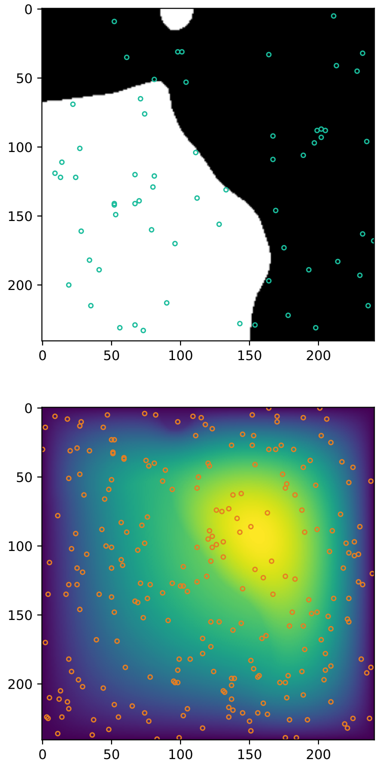
The dataset used in our study is adapted from that used in Li et al. (2021), and originally generated by a traditional high-resolution PDE solver. The dataset consists of a 1024 × 1024 grid, which was subsampled uniformly at random and arranged into context-target pairs. The context is comprised of the evaluations of the diffusion coefficient: $(\mathbf{c}_x, \mathbf{c}_y) = ((x, y), a(x, y))$, and the target is the solution of the Darcy Flow equation at a position, $(\mathbf{t}_x, \mathbf{t}_y) = ((x, y), u(x, y))$.

The results are presented in table 1, they are coherent with the rest of the experiment and show that the MSA and TFS models are able to outperform all competing models. We used the same initialization for GEN as for the Poisson Equation.

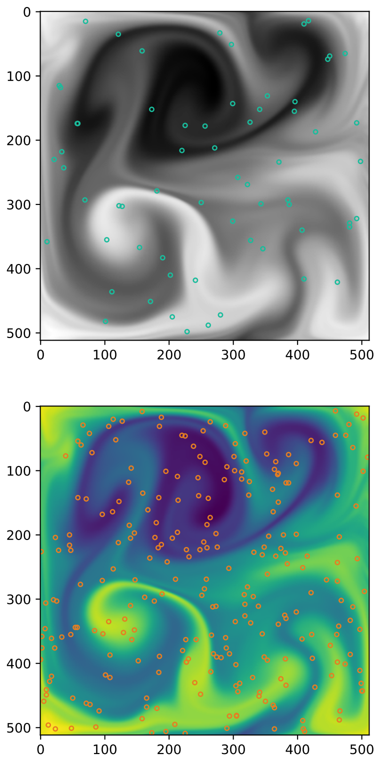
This experiment adapts the dataset available in Li et al. (2021). It is available at `https://github.com/neural-operator/fourier_neural_operator` and can be processed with the code provided to rearrange it in the irregular setup.

2.5 TWO-DAYS WEATHER FORECASTING

In this task, we want to evaluate our models on the task of two-days weather forecasting based on irregularly sampled data in space. The requested data collection description focuses on climate reanalysis from the ERA5 dataset, which is available through the Copernicus Climate Data Store (CDS). The dataset contains various parameters related to wind, surface pressure, temperature, and cloud cover.



(a) A sample of the Darcy Flow dataset Li et al. (2021). Blue dots correspond to the context set and orange dots to the targets.



(b) A sample of the Navier-Stokes dataset Li et al. (2021). Blue dots correspond to the context set and orange dots to the targets.

To access the data, please visit the following URL: `https://cds.climate.copernicus.eu/cdsapp#!/dataset/reanalysis-era5-single-levels?tab=form`.

We selected the following variables:

- 10m u-component of wind: This refers to the eastward wind component at a height of 10 meters above the surface.
- 10m v-component of wind: This represents the northward wind component at a height of 10 meters above the surface.
- 100m u-component of wind: This denotes the eastward wind component at a height of 100 meters above the surface.
- 100m v-component of wind: This signifies the northward wind component at a height of 100 meters above the surface.
- Surface pressure: This represents the atmospheric pressure at the Earth's surface.
- 2m temperature: This indicates the air temperature at a height of 2 meters above the surface.
- Total cloud cover: This refers to the fraction of the sky covered by clouds. The data collection includes measurements for all times of the day and all days available in the dataset. However, for training purposes, we selected the data from the year 2000. For validation, we use the data from the year 2010, specifically the months of January and September.

Next, we proceeded to extract the corresponding grib files. To create our dataset, we paired time slices that had a time difference of two days, equivalent to 48 time steps since ERA5 data has a one-hour resolution over the whole world.

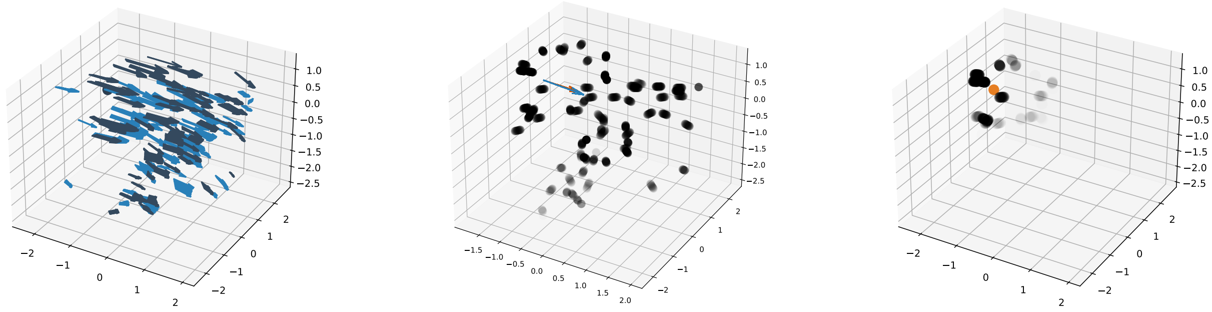
For the context slice, we performed subsampling at 1024 different positions and retained all seven variables mentioned earlier. As for the target slice, we subsampled it at 1024 different positions and kept only the two components of the wind at 100m.

3 SPECIFIC ASPECTS OF OUR APPROACH IN RELATION TO EXISTING WORKS

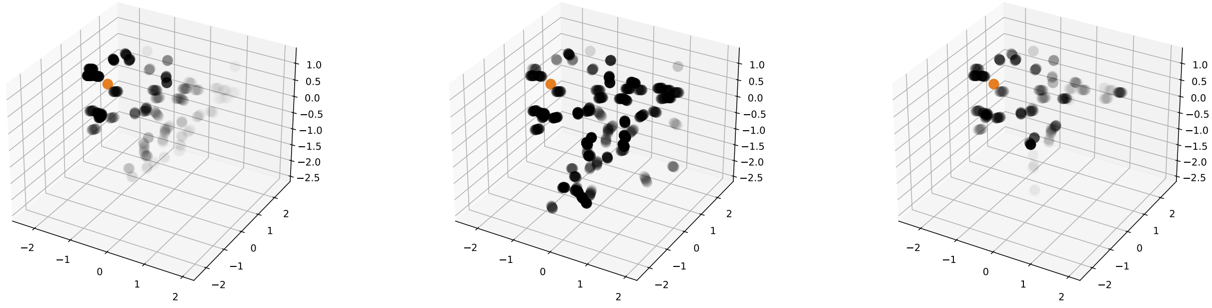
In our research, we address a distinct challenge that sets our work apart from conventional mesh-based simulators Pfaff et al. (2020); Janny et al. (2023); Li et al. (2022). Our focus is on extrapolating sets of sparse and variably-positioned measurements, a context markedly different from the consistent mesh structures employed in traditional approaches. Unlike mesh-based models where values are accessible at fixed points on an irregular mesh, our method tackles scenarios where the quantity and locations of measurements can change dynamically from one data set to another. This unique aspect of our work necessitates a departure from the conventional mesh framework. Mesh-based simulators, while effective in their domains, are not equipped to handle the sporadic and non-uniform nature of our data. For instance, in applications such as airspace monitoring, data points represent measurements taken only where planes are present, leaving vast areas without data. Our methodology, therefore, diverges fundamentally from

Dataset	dim($\mathbb{X}$)	dim($\mathbb{I}$)	dim($\mathbb{O}$)	# Train. points	# Val. points	URL
Wind Nowcasting	3	2	2	26.956.857	927.906	Pannatier et al. (2021) Dataset page
Poisson Equation	2	3	1	409.600	102.400	Alet et al. (2019) Github Repository
Navier Stokes Equation	2	1	1	48.883.20	11.673.600	Li et al. (2021) Github Repository
Darcy Flow Equation	2	1	1	409.600	102.400	Li et al. (2021) Github Repository
ERA5	2	7	2	8.648.704	2.107.392	Hersbach et al. (2023) Copernicus Climate Dataset Store
Random	2	1	1	640.000	64.000	Randomly generated
Sine	2	1	1	640.000	64.000	Randomly generated

Table 2: Description of the different datasets used in this study.



(a) Context, Targets (b) MSA (Ours) (c) GKA Pannatier et al. (2021)



(d) GEN Alet et al. (2019) (e) CNP Garnelo et al. (2018) (f) TFS (Ours)

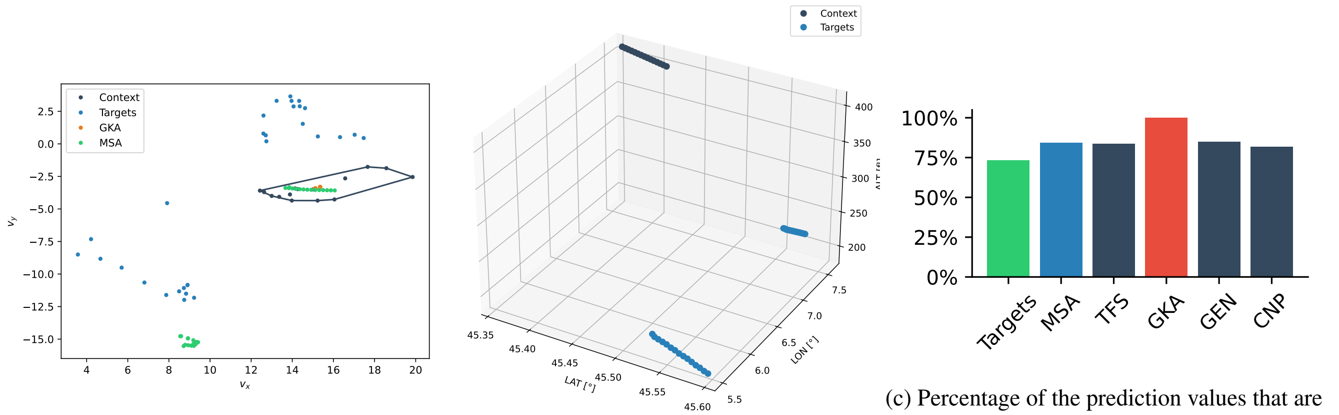
Figure 3: Displaying the importance given to points in the context to do the prediction for the different models for a given query point (orange). We used the norm of the input gradients for that purpose and highlight the context values that have the largest gradient with respect to the output. The opacity of the dots corresponds to the relative magnitude of the input gradients compared to other points in the context.

mesh-based techniques, as it requires innovative handling of sparse and irregular data inputs, rather than relying on a fixed, uniform mesh structure.

Furthermore, our approach markedly differs from trajectory prediction models Yuan et al. (2021); Nayakanti et al. (2023); Girgis et al. (2022). While these models excel in forecasting future points along established trajectories, our work diverges in both intent and application. Our primary concern is not the sequential prediction of trajectories but rather the interpretation and forecasting of phenomena in a spatially and temporally sparse environment. In the context of wind nowcasting, for example, our model excels at processing limited and scattered data points from multiple trajectories to generate a comprehensive forecast. This capability to assimilate sparse measurements from varied locations and synthesize them into a coherent prediction sets our work apart from traditional trajectory-focused models. These models, such as Wayformer Nayakanti et al. (2023) and Agentformer Yuan et al. (2021), are primarily designed for time series forecasting and trajectory completion.

4 ATTENTION MATRIX FOR WIND NOWCASTING

For all models, we can plot the norm of the input gradients to see how changing a given value would impact the prediction [fig. 3]. We see that Transformers seem to find a trade-off between taking into account the neighboring nodes and the global context.



(a) Context, targets, and forecast in wind speed space.

(b) Context and targets in the 3D space.

(c) Percentage of the prediction values that are contained in the convex hull of the measurement in the context in percent.

Figure 4: Context and targets represented in wind speed and 3D space. Both the context and targets correspond to 1-minute slices of data. The targets capture wind measurements recorded 30 minutes following the context data. Additionally, the forecasts generated by the MSA and GKA models are depicted. Notably, the GKA model is confined to predicting within the convex hull of the context values, which is visually highlighted in grey. Conversely, the MSA model is unrestricted by this limitation and can make predictions beyond the convex hull of the context values.

5 COMPARISON WITH GRAPH KERNEL AVERAGING

Smart averaging strategies such as GKA are limited by design to the range of the values in their context. However, these baselines are useful as they are not prone to overfitting and in the case of time series forecasting (if the underlying variable has some persistence), the last values would in general be the best predictor of the next value. We hypothesize that over sufficiently short horizons, these methods would achieve performance similar to that from more complicated models, but when the prediction horizon rises, this design limitation would become more pronounced.

Since, for extrinsic forecasting, target points need not lie in the convex hull of context points, the greater flexibility of MSA and TFS can make correct forecasts that are impossible with GKA. fig. 4a demonstrates this fact, with a plot of the target variable, the context points and the context convex hull in wind speed space.

But greater capacity means also more failure modes. As we saw in the metrics table of table 3 of the main paper, MSA and TFS are the only models to outperform GKA whereas other models, even if they are more flexible, fail to beat this baseline. On average 27% of the predictions were outside the convex hull of the context. In fig. 4c we analyze the percentage of measurements outside the convex hull produced by the different models. We can see that GKA is limited as 100% of the measurements lies within the convex hull whereas the other models can compensate and predict values outside it.

5.1 DECODER AND CONDITIONING FUNCTION

In the main paper, we presented a comparison of the key distinctions between our attention-based architectures. However, it could be that the performance differences observed could be attributed to minor variations in the architecture design. For the sake of completeness, in this section, we provide experimental results to address this aspect by comparing the remaining differences between GEN and TFS. Specifically, we focus on analyzing the impact of the number of decoder layers and the conditioning function. It is important to note that this analysis does not apply to MSA, as it does not possess an encoder-decoder structure.

GEN and TFS differ not only in their latent representation but also on two other points: (1) the way that they access the encoded information in the decoder and (2) how much computing power is used in the decoder stack.

In all three models, we tried using either a distance-based function or cross-attention to combine the target position with the encoded latents. CNP concatenate the same context vector to all target queries, which we model as equivalent to averaging the latents based on the distance in the case that there is only one latent.

Specifically, each latent is associated with a position $\mathbf{l}_x$ in the underlying space with corresponding value $\mathbf{l}_y$. Conditioning is made by averaging the latent features based on their distance with the query $\mathbf{t}_x$. Its formula is given by:

$$z = \sum_{\mathbf{l} \in \text{latents}} r(\text{dist}(\mathbf{l}_x, \mathbf{t}_x))\mathbf{l}_y \tag{4}$$

Where $r : \mathbb{R} \to [0, 1]$ is a function that maps distances to weights, and $\mathbf{l}_x$ represents the position associated with the latent nodes with value $\mathbf{l}_y$. This vector z is then concatenated to the target position $\mathbf{t}_x$ and fed to the decoder which is a MLP in this case.

Table 3: Results of the ablation of the conditioning function used to combine the latents with the target position. We adapted the architecture so that they used both a distance-based conditioning function, which combines the query position with a weighted average of the nearest latents and standard cross-attention. We show in italics hybrid architecture where we had to switch the default conditioning method. For GEN and CNP replacing the conditioning method does not impact the performance, but for TFS switching to a distance-based conditioning function impacts the performance drastically.

Model	**Distance-based**	**Cross-attention**
CNP	.115 ± .015	*.119 ± .014*
GEN	.024 ± .004	*.022 ± .006*
TFS	*.042 ± .023*	**.020 ± .011**

Table 4: Results of the decoder-layer ablation. One difference between TFS and GEN is that by default TFS use multiple layers in its decoder whereas GEN uses one, and delegates all processing to the encoder. We see that having multiple decoder layers helps the transformer whereas it impacts the performance of the GEN.

	# Decoder layers		
Model	**1**	**2**	**4**
GEN	.021 ± .002	.040 ± .012	.106 ± .004
TFS	.020 ± .011	.019 ± .005	**.013 ± .001**

GEN use this distance-based aggregation function by default whereas TFS uses cross-attention. We tried using cross-attention for CNP and GEN and using the same distance-based aggregating function for TFS using the distance between context position and target position as a reference for this scheme. For GEN and CNP using cross-attention give approximately the same results, but using distance-based conditioning for TFS hurts the performance significantly.

We also ran ablations that created hybrid cases for both transformers and GEN and concluded that adding decoding layers helped TFS to reach better performance. Adding layers to the GEN decoder drastically reduces model performance, which seems to be coherent with the fact that GEN were shown to be more prone to overfitting in the results section.

Finally, we want to highlight that MSA outperforms all the models presented in this section by a large margin table 1.

6 WIND NOWCASTING METRICS

This section defines the metrics used for wind nowcasting.

Root Mean Square Error (RMSE) It takes the square root of the square distance between the prediction and the output. It has the advantage of having the same units as the forecasted values.

$$\text{RMSE}(\hat{t}, t) = \sqrt{\frac{\sum_k^N (\hat{t}_k - t_k)^2}{N}} \tag{5}$$

Mean Absolute Error for angle (angle MAE) It is interesting and sometimes more insightful to decompose the error made by the models in their angular and norm components. This is the role of this metric and the following.

$$\text{angle MAE}(\hat{t}, t) = ||(\alpha(\hat{t}) - \alpha(t))||_{L_1} \tag{6}$$

where $\alpha \left(\vec{x} = \begin{pmatrix} x \\ y \end{pmatrix} \right) = \arctan(y, x) * \frac{360}{2\pi}$

Mean Absolute Error for norm (norm MAE) Following the explanation of the last paragraph:

$$\text{norm MAE}(\hat{t}, t) = \sum_k \mid || \hat{t}_k ||_2 - || t_k ||_2 \mid \tag{7}$$

Relative Bias (rel BIAS) x,y Additionally, we used weather metrics as in Ghiggi et al. (2019). The relative bias measure if the considered model under or overestimate one quantity. It is defined as:

$$\text{rel BIAS}(\hat{t}, t) = \frac{\text{mean}(\hat{t}_k - t_k)}{\text{mean}(\hat{t}_k)} \tag{8}$$

Ratio of standard deviation (rSTD) The ratios of standard deviation indicate whether the dispersion of the output of the model matches the target distribution. It has an optimal value of 1.

$$\text{rSTD}(\hat{t}, t) = \frac{\text{std}(\hat{t})}{\text{std}(t)} \tag{9}$$

NSE The last domain metric used is the Nash-Sutcliffe efficiency (NSE), which compares the error of the model with the average of the target data. A score of 1 is ideal and a negative score indicates that the model was worse than the average prediction on average.

$$\text{NSE}(\hat{t}, t) = 1 - \frac{\text{MSE}(\hat{t}, t)}{\text{MSE}(t, \text{mean(t)})} \tag{10}$$

7 INFLUENCE OF THE TIGHTNESS OF THE MEASUREMENTS ON THE PERFORMANCES

To evaluate the influence of the tightness of the measurements on the results we designed the following experiment: We start with a pair of context (c_x, c_y) and target points (t_x, t_y). We pick one point from the context and restrict the context to a small number of measurements close to this point. Then, we calculate the error made by the model when it is given only this restricted context and rank them by their distance to the context's center. As the dimensions (longitude, latitude, altitude) differ, we first normalized the data along each axis. We repeat this process and average the results.

Table 5: Influence of the tightness of the measurements on the performance. We restrict the context to measurements that are close to the query point and then measure the error as a function of the distance from this query point averaged over the whole dataset. We estimate the standard deviation using three random seeds.

Normalized Dist	**MSA**	**TFS**
0.5–1.0	5.56 ± 0.49	7.73 ± 0.37
1.0–1.5	6.62 ± 0.35	7.35 ± 0.53
1.5–2.0	7.57 ± 0.43	7.87 ± 0.36
2.0–2.5	9.52 ± 0.47	9.48 ± 0.66
2.5–end	10.35 ± 0.49	10.53 ± 0.61

We see that the error increases with the distance to the context. Now in practice, the context is not restricted and contains points all over the space (so the mean minimum normalized distance to the target points is usually often below 1.0)

We assessed general uncertainty by training the model using various random seeds, resulting in an ensemble of models from which we can determine the standard deviation and added it to the results table of the previous experiment.

8 INFLUENCE OF THE TARGET POSITIONS

In this section, we assess the impact of the target position on the training of the MSA model.

The context data for the Poisson equation is heavily prescribed (where context points belonged to the source or sink or boundary conditions), while for the Darcy Flow equation, we constructed a similarly-shaped data set by subsampling the data set from Alet et al. (2019) to create the context and targets. Thus, the Darcy flow test problem gives an ideal testbed for examining the influence of target positions on the training. To examine the impact of the context position, we designed an additional experiment where we sampled the context from the bottom left quadrant of the unit cube and the targets from the upper right quadrants of the unit cube. We conduct our analysis for small (5k parameters) and large (100k parameters) models. We report the test loss as a function of the percentage of the context from the upper-right quadrants (where all targets are located, the remaining from the bottom-left quadrant).

In the first column (0% of the data from the first quadrant), we see that the MSA model struggled to learn when the context and targets are disjoint. The error improves with greater overlaps between context and target, with p = 100% corresponding to the case in the paper (albeit on one-quarter of the data). Similar dynamics hold for both model sizes. We believe that this is due to the Darcy Flow simulation being highly location-dependent, as can be seen in some examples from the dataset fig. 2a.

Table 6: Influence of the target position on the training. We report the test error for the MSA model at two different sizes. We devised an experiment in which we extracted the context from the bottom left quadrant of the unit cube, while the targets were sampled from the upper right quadrants of the same unit cube.

	0%	**2%**	**25%**	**50%**	**100%**
MSA 5k	0.14	0.15	0.14	0.09	0.03
MSA 100k	0.14	0.14	0.13	0.09	0.03

Regarding the training performance, we noticed no substantial difference in the total time to a solution, they all took approximately 1000 epochs in all cases. Larger models were always able to overfit the data.

Table 7: Influence of the target position on the training. We report the train error for the MSA model at two different sizes. Both models were able to overfit the data.

	0%	**2%**	**25%**	**50%**	**100%**
MSA 5k	0.07	0.07	0.07	0.05	0.02
MSA 100k	0.00	0.00	0.00	0.00	0.00

9 DIFFERENT MESSAGE PASSING SCHEME

We try here different message-passing schemes even one using attention, to help us demonstrate the fact that the whole family of models that encodes the space as a single graph suffers from the same bottlenecking effect. Here are the results of the wind nowcasting task:

Table 8: Performance of GEN with different message passing schemes for the wind nowcasting task. Various message-passing schemes, including those with attention, were explored using PyTorch Geometric Fey and Lenssen (2019). The best-performing GEN model with the default message-passing scheme is indicated with a reference line. While certain schemes showed improvements over GEN's performance, the overall performance remained relatively consistent and was still outperformed by MSA. For additional information on the different message-passing schemes and related references, please refer to the PyTorch Geometric documentation.

Model	**Score**
TransformerConv	9.2830
GATv2Conv	9.3353
GeneralConv	9.3699
GATConv	9.7744
SAGEConv	9.7837
ARMAConv	9.8391
TAGConv	9.8653
SGConv	9.9465
SuperGATConv	10.0961
GCNConv	10.1344
LEConv	10.6062
GENConv	23.9847

Although certain message-passing schemes enhance the performance of the GEN model, it still falls short of the performance achieved by MSA. We attribute this difference to the bottlenecking effect caused by the latent graph. For additional information on the various message-passing schemes and relevant references, please consult the PyTorch Geometric documentation available at `https://pytorch-geometric.readthedocs.io/en/latest/modules/nn.html#convolutional-layers`.

10 BROADER IMPACTS

We do not anticipate any significant adverse effects on society due to this work. Generally, having an additional method for weather nowcasting may lead to an increase in computational resources required, but this is a common consequence of many deep learning systems.

We believe that enhancing the reliability of weather forecasts and dynamical systems has a more positive impact on society. By improving the accuracy and precision of these predictions, we can provide valuable information for various sectors, such as agriculture, transportation, disaster management, and overall planning and decision-making processes.

11 LIMITATIONS

Our method, like other attention-based models, suffers from quadratic scaling. In the case of MSA, it is slightly more computationally intensive compared to TFS due to the combination of target and context in the same sequence. This results in an attention mechanism that scales with $\mathcal{O}((N_c + N_t)^2)$, which is asymptotically equivalent to the scaling of transformers, which is $\mathcal{O}(N_c^2 + N_t^2 + N_c N_t)$, but in practice introduces an additional $N_c N_t$ term. However, in all our experiments, this computational overhead did not pose a problem. We acknowledge the issue in the scaling of the model and refer to possible solutions outlined in the related work to address this challenge.

Another drawback of this study is the absence of a comparison between the models and other established models used for modeling dynamical systems on a grid, as discussed in the work by Li et al. (2021). We anticipate that our approach may not perform as effectively as competing models on grids due to two reasons. Firstly, the previously mentioned scaling issue becomes significant when dealing with larger grid sizes, such as 1024×1024 grids. Additionally, our approach lacks certain inductive biases that aid in grid-based modeling. Nonetheless, we believe that attention-based models still hold promise for grid-based systems, as demonstrated by the success of Vision Transformers (ViT) in other domainsDosovitskiy et al. (2020). However, tokenization may be necessary for their effective implementation.

12 EXPERIMENT AND TRAINING DETAILS

In this supplementary material, we provide the code required to process the dataset and reproduce the experiments. We utilized Hydra as a configuration manager Yadan (2019) and include the precise configuration for each case. The experiments were executed multiple times on a GPU cluster, but each experiment can be conducted on a single GPU in a relatively short timeframe, ranging from a few hours to a maximum of a few days.

13 ERROR BARS

We ran multiple runs of our experiments and report the standard deviation in all concerned tables.

14 AMOUNT OF COMPUTE NEEDED TO REPLICATE THE EXPERIMENTS

Training smaller models (5k, 20k) usually takes a few hours. Training larger models (except for CNP which is considerably faster) runs in at most two GPU days. We estimate the number of GPU days to replicate all experiments for all models to be on the order of 100 GPU days.

15 REPRODUCTIBILITY

We provide the link to the dataset and the whole code base for processing it and running the experiments.

16 LICENSES

All concerned databases are openly accessible on the web and have permissive licenses, we give a link to each dataset in table 2

REFERENCES

Ferran Alet, Adarsh Keshav Jeewajee, Maria Bauza Villalonga, Alberto Rodriguez, Tomas Lozano-Perez, and Leslie Kaelbling. Graph element networks: adaptive, structured computation and memory. In *International Conference on Machine Learning*, pages 212–222. PMLR, 2019.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/garnelo18a.html`.

G. Ghiggi, V. Humphrey, S. I. Seneviratne, and L. Gudmundsson. Grun: an observation-based global gridded runoff dataset from 1902 to 2014. *Earth System Science Data*, 11(4):1655–1674, 2019. doi: 10.5194/essd-11-1655-2019. URL `https://essd.copernicus.org/articles/11/1655/2019/`.

Roger Girgis, Florian Golemo, Felipe Codevilla, Martin Weiss, Jim Aldon D'Souza, Samira Ebrahimi Kahou, Felix Heide, and Christopher Pal. Latent variable sequential set transformers for joint multi-agent motion prediction. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=Dup_dDqkZC5`.

Hans Hersbach, Barbara Bell, Paul Berrisford, Giovanni Biavati, András Horányi, Joaquín Muñoz Sabater, Julien Nicolas, Carole Peubey, Razvan Radu, Ivan Rozum, Dries Schepers, Adrian Simmons, Cornel Soci, Dick Dee, and Jean-Noël Thépaut. ERA5 hourly data on single levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS), 2023. Accessed on 17-05-2023.

Steeven Janny, Aurélien Bénéteau, Madiha Nadri, Julie Digne, Nicolas THOME, and Christian Wolf. EAGLE: Large-scale learning of turbulent fluid dynamics with mesh transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL `https://openreview.net/forum?id=mfIX4QpsARJ`.

Zongyi Li, Nikola Borislavov Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=c8P9NQVtmnO`.

Zongyi Li, Daniel Zhengyu Huang, Burigede Liu, and Anima Anandkumar. Fourier neural operator with learned deformations for pdes on general geometries, 2022.

Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S. Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2980–2987, 2023. doi: 10.1109/ICRA48891.2023.10160609.

Arnaud Pannatier, Ricardo Picatoste, and François Fleuret. Efficient wind speed nowcasting with gpu-accelerated nearest neighbors algorithm, 2021. URL `https://arxiv.org/abs/2112.10408`.

Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter Battaglia. Learning mesh-based simulation with graph networks. In *International Conference on Learning Representations*, 2020.

Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL `https://github.com/facebookresearch/hydra`.

Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M. Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9813–9823, October 2021.