Meta-Learning Deep Kernels for Turbulence Forecast Verification

Grzegorz Zakrzewski[1,2] and Jacek Mańdziuk[1,3]
[1]*Faculty of Mathematics and Information Science, Warsaw University of Technology*, Warsaw, Poland
[2]*Institute of Meteorology and Water Management – National Research Institute*, Warsaw, Poland
[3]*Faculty of Computer Science, AGH University of Krakow*, Krakow, Poland

***Abstract*—Accurate turbulence forecasts help prevent aviation accidents, but improving forecasts requires reliable ground truth for evaluation. Constructing the best available estimate of the true atmospheric turbulence state – called reanalysis – is not a trivial task, as turbulence encounters are sparse, confined to flight paths, and measured in heterogeneous units. We tackle this problem using Gaussian Processes, which learn spatial and temporal covariances directly from the data and provide uncertainty estimates. To capture complex patterns, we combine standard RBF and linear kernels with deep kernels, where a neural network extracts features before covariance computation. We show that meta-learning of kernel hyperparameters and network weights across many reanalysis construction tasks outperforms per-task optimization. We also demonstrate that our approach is competitive with standard reanalysis construction methods.**

***Index Terms*—Gaussian Process, deep kernel learning, deep kernel transfer, turbulence forecast verification**

I. Introduction

Atmospheric turbulence causes aviation accidents, damages aircraft, and injures passengers and crew [1]. Accurate turbulence forecasts enable planning of safer routes, reducing both human and economic costs. As new forecasting methods emerge, we need reliable ways to evaluate and compare them.

Meteorologists verify forecasts in two ways: by comparing them against direct observations or against an analysis [2]. Observations are point measurements from instruments at specific locations and times. An analysis estimates the atmospheric state across the entire spatial domain on a regular grid. To construct an analysis, numerical weather predictions (NWP) are combined with observations through data assimilation [3] – a process that corrects the forecast using measurements. When assimilation is retrospective and incorporates observations collected after the nominal time of the analysis, the result is called reanalysis (see Figure 1). Since turbulence observations are noisy and sparse, reanalysis is better suited for turbulence verification, yet, to our knowledge, no high-resolution turbulence reanalysis exists that would enable reliable verification of regional turbulence forecasts.

Measuring turbulence poses unique challenges. The standard metric, Eddy Dissipation Rate (EDR), quantifies turbulence intensity [4], but its direct applicability is limited as only some aircraft carry instruments to measure it [1]. Some planes instead report Derived Equivalent Vertical Gust (DEVG), which requires conversion to EDR [5]. Ground-based radars offer another data source, as they track the plane's motion, and one can derive EDR from their measurements [6]. However, radar data often contain errors and require careful quality control. Pilots also submit manual reports (AIREPs) with subjective turbulence assessments like "moderate" or "severe" [1]. All aircraft-based observations share a fundamental limitation: they are restricted to specific flight paths, resulting in data-rich corridors and data-poor regions outside of them. Flight traffic also varies throughout the day, causing temporal gaps. These characteristics make turbulence verification exceptionally difficult.

We tackle this challenge by developing a method to construct turbulence reanalysis from sparse, noisy, and heterogeneous observations. Our approach fuses four data sources: in situ DEVG measurements, Mode-S radar data, manual AIREP reports, and semi-automated special air reports (ARS). Each source brings some kind of uncertainty or noise – Mode-S data suffer from measurement noise, AIREPs and ARS provide only categorical values, and all sources require conversion to EDR, which introduces additional inaccuracies. A forecast verification process must explicitly handle the above uncertainty.

Gaussian Processes (GPs) [7] provide a natural framework for tackling this issue. Within a GP, we can use a numerical weather forecast as a prior mean, and then update this prior with observations. The GP model learns spatial covariances and the observation noise directly from data, and produces both predictions and uncertainty estimates at any location.

Standard GP kernels, however, may lack the flexibility to capture complex patterns in turbulence data. Deep kernel learning [8], [9] addresses this limitation by passing inputs through a neural network before applying a base kernel, expanding the kernel's expressive power. Patacchiola et al. [10] combined deep kernels with meta-learning in a few-shot setting to solve regression and classification benchmarks. Instead of fitting kernel parameters separately for each task, their framework learns shared parameters across many tasks. This approach naturally fits our setting. Each hourly reanalysis constitutes one task, and the meta-learner selects hyperparameters that transfer across hours (see Figure 3). Zanetta et al. [11] recently applied a similar framework to a different task than ours, namely to estimate surface winds at sub-kilometer resolution.

In summary, the main contribution of this work is five-fold.

(i) We are the first to incorporate deep kernels for turbulence reanalysis construction, demonstrating their benefits in

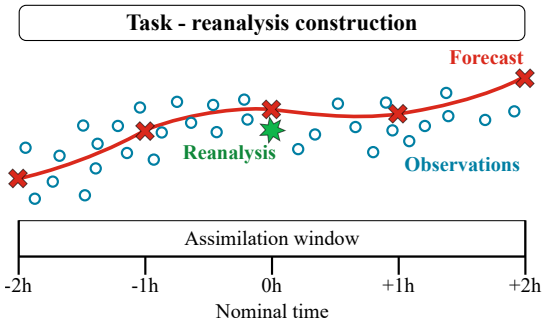

Fig. 1. Visualization of a reanalysis construction task, with a 4-hour-long data assimilation window, and a weather forecast with hourly resolution.

this setting.

(ii) We extend the framework of Zanetta et al. [11] to handle temporally irregular observations by introducing kernels that operate on the time dimension.

(iii) We demonstrate that meta-learning yields more accurate and stable results than optimizing each hourly task independently. This confirms the ability of meta-learning to capture the underlying structure of turbulence across tasks, whereas per-task optimization fails to deliver consistent performance.

(iv) We show that our approach outperforms common baselines for reanalysis construction (*climatology*, *background forecast*) and is competitive with SOTA methods: *Optimal Interpolation* (OI) and *Ensemble Kalman Filter* (EnKF).

(v) We highlight the practical value of utilizing GP uncertainty estimates in forecast verification.

The source code is available in Zenodo repository [12].

II. Related Work

Calandra et al. [8] and Wilson et al. [9] showed that deep kernel GPs outperform GPs with standard kernels such as RBF or spectral mixture. They also demonstrated that deep kernels are superior to a two-stage approach where a GP operates on features from an independently trained neural network. Deep kernels can adapt to sharp changes in covariance structure – a property we expect to be important for turbulence, where spatial correlations vary with atmospheric conditions. Modern libraries like GPyTorch [13] and PyTorch [14] make deep kernel GPs straightforward to implement; we use these packages in our experiments.

Ober et al. [15] raised concerns about deep kernel learning: optimizing the marginal likelihood in overparameterized models can lead to pathological overfitting. They attribute the practical success of deep kernels largely to regularization from stochastic minibatching. Wilson et al. [16] recently challenged this critique, identifying an oversight in the mathematical analysis that invalidates the main argument. Still, they emphasize that careful training is essential.

Patacchiola et al. [10] proposed Deep Kernel Transfer (DKT) for few-shot learning, where deep kernel parameters are shared across many small, related tasks. Although DKT optimizes the marginal likelihood, it samples a single task per training iteration. We hypothesize that this task-level stochasticity introduces gradient noise analogous to data mini-batching, acting as a regularizer that mitigates overfitting. This view aligns with Wilson et al.'s emphasis on careful training.

Zanetta et al. [11] applied DKT to estimate surface winds at sub-kilometer resolution. They modified DKT by sampling a mini-batch of tasks rather than a single task in the model training loop. Their framework involves spatial RBF, deep, linear kernels, and their products. However, in their study, observations arrive at fixed times, and they do not model temporal covariance.

Modern weather forecasting increasingly relies on ensemble methods, where multiple (different) forecasts are made simultaneously to account for uncertainty in initial conditions and model physics [2]. These ensembles produce forecast distributions rather than single deterministic predictions. Our GP-based reanalysis also produces probabilistic estimates, raising the question of how to compare two distributions – the forecast and the analysis. Ferro [17] argues against direct distribution-to-distribution comparison: it can favor forecasts that match the observation uncertainty rather than the true state of the atmosphere. A perfect forecast of the truth might score worse than one that simply reproduces observation noise. Ben Bouallegue et al. [18] show how to handle observation uncertainty in practice. They propose perturbing each ensemble member f_i with an independent sample ϵ_i from the observation error distribution: $f_i' = f_i + \epsilon_i$. Their method requires fitting a parametric model to characterize observation errors. In contrast, in our approach such distribution is provided directly by the GP model, eliminating the need for additional fitting.

III. Datasets

We fuse four data sources to construct turbulence reanalysis: Mode-S data, AMDAR reports, manual (human) AIREP reports, and semi-automated ARS reports.

Mode-S radar data constitutes the largest source, providing several dozen thousand records per day. Many Mode-S records have clearly unrealistic values, so we apply strict quality control measures and discard about one-third of the observations. We derive EDR from Mode-S records using methods I and II (Eqs. 3 and 5) from Kopeć et al. [6]. Both methods require several consecutive clean records within a short time window – we set a threshold of a minimum of four messages in a 30-second window. Since the two methods detect different turbulence encounters, we use them simultaneously. After filtering, we obtain approximately 10,000 turbulence measurements per day, most of which indicate very weak turbulence.

AMDAR reports provide in situ DEVG measurements, which we convert to EDR using method II (Eq. 8) from Kim et al. [5]. Only a fraction of AMDAR reports carry DEVG, yielding about 20 turbulence observations per day.

Human AIREPs and semi-automated ARS reports – which partially overlap – contain categorical turbulence assessments such as "moderate" or "severe". We assign these categories numerical EDR values of 0.24 and 0.54, respectively, following thresholds from the NOAA Aviation Weather Center [19].

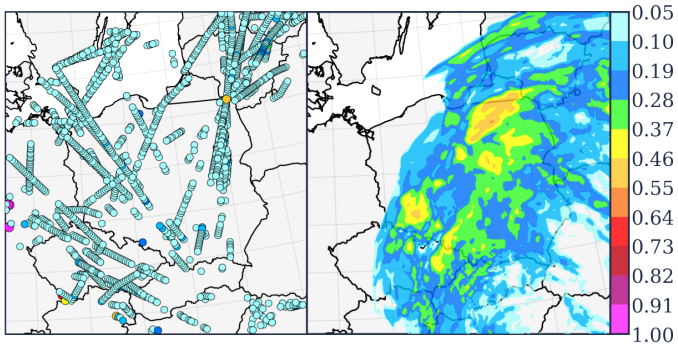

Fig. 2. Turbulence observations from all data sources between 2025-11-25 22:00 UTC and 2025-11-26 02:00 UTC (left) and EDR ensemble forecast mean for 2025-11-26 00:00 UTC (right), with color scale representing turbulence intensity.

Together, these sources contribute about 10 observations per day.

Each of the four data sources spans a three-month period from 25 September 2025 to 21 December 2025. We filter the observations vertically between 8,700 and 12,500 meters to focus on cruise-phase turbulence encounters. Horizontally, observations cover Poland and its surroundings, spanning 0.65°–10.125° longitude and −2.4°–7.7° latitude in a rotated pole projection with the pole located at 170°W, 40°N. This matches the forecast domain of the COSMO Time-Lagged Ensemble (TLE) model [20]. The ensemble comprises 20 members and runs four times daily at 00, 06, 12, and 18 UTC. Each run produces hourly forecasts; we use the first six hours of each run for reanalysis construction. The model operates on a regular grid of 2.8 km resolution. We interpolate the EDR forecast, taken as the maximum over the 8,700–12,500 m altitude range (as turbulence forecasts are often used to identify the most severe scenarios) at observation locations. A sample of observations and a forecast are visualized in Figure 2.

IV. Methods

A. Turbulence reanalysis construction task

We define a *task* $\mathcal{T}_t$ as a turbulence reanalysis construction problem for a nominal time t, always a full hour (e.g., 26 November 2025, 00:00 UTC). Each task combines an NWP forecast valid at time t with observations from a data assimilation window $[t-w, t+w]$. We set $w = 2$ hours; for example, a task at 00:00 UTC assimilates observations from 22:00 UTC the previous day to 02:00 UTC. The GP target variable is the difference between observed EDR and the NWP ensemble mean at the observation location. If the forecast is unbiased, this target variable distribution is symmetric around zero, and the GP prediction acts as a correction to the NWP forecast.

B. Covariance modeling

We consider four kernel types. The *spatial* kernel is a Radial Basis Function (RBF) with Automatic Relevance Determination (ARD) operating on observation coordinates. The *temporal* kernel is an RBF acting on the time difference between observation and nominal reanalysis time. The *linear* kernel is a scaling component based on the NWP ensemble mean. The *deep* kernel passes inputs through a neural network before applying an RBF with ARD; the network receives 24 features: two coordinates, time difference, ensemble mean, and all 20 ensemble forecasts from the COSMO TLE model.

We evaluate several kernel products (Table I). Starting from spatial kernels – which capture the most fundamental dependencies – we progressively add temporal, linear, and deep components. This design lets us compare standard kernels against deep kernels and assess whether combining them improves performance.

TABLE I
KERNEL PRODUCTS AND THEIR ABBREVIATIONS.

Kernel	Abbreviation
spatial	*S*
spatial × temporal	*ST*
spatial × temporal × linear	*STL*
deep	*D*
deep × spatial	*DS*
deep × spatial × temporal	*DST*
deep × spatial × temporal × linear	*DSTL*

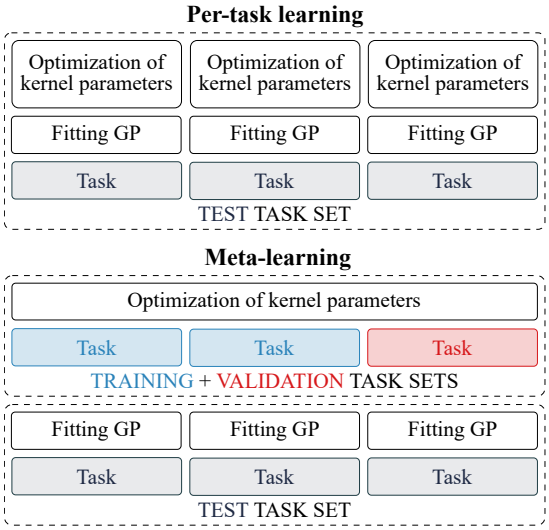

Fig. 3. Comparison of per-task learning, where kernel parameters are optimized separately for each task, with meta-learning, where parameters are optimized and shared across many tasks.

C. Hyperparameter optimization approaches

Each kernel introduces specific hyperparameters: *length-scale* for RBF kernels, *variance* for linear kernels, and neural network weights for deep kernels. Following [8], [9], we optimize all parameters jointly by minimizing the negative marginal log-likelihood.

We compare two optimization strategies (Figure 3). In the *per-task* approach, we optimize hyperparameters separately for each task. In the *meta-learning* approach, we optimize

hyperparameters once for all training tasks. At each iteration, we sample a mini-batch of 16 tasks, compute the marginal log-likelihood for each of them, and backpropagate based on their average.

We split our dataset chronologically: training spans 25 September to 16 November 2025 (1068 tasks), validation covers 17–24 November (168 tasks), and test runs from 25 November to 21 December (672 tasks). The training and validation task sets serve only the meta-learning approach. Both optimization strategies share the same test set.

In either case, we subsequently fit a separate GP per task.

D. Implementation details

To mitigate sensitivity to initialization, we repeat each hyperparameter optimization process with 20 Optuna [21] trials (with a sampler using Tree-structured Parzen Estimator algorithm [22], [23]), which propose learning rates and initial kernel parameters; PyTorch initializes the neural network weights (Kaiming uniform initialization). We select the best hyperparameter configuration based on validation performance.

We evaluate kernels using 5-fold cross-validation within each test task: we fit the GP on four folds of turbulence observations and measure performance on the held-out fold, rotating through all five splits. For the per-task approach, hyperparameter optimization takes place on training folds (with 15% held out for validation) before fitting – that yields 100 optimization runs per test task (20 trials $\times$ 5 folds).

We train for up to 5000 epochs, halving the learning rate after 10 epochs without improvement and stopping early after 25 epochs of stagnation. We keep these patience parameters small for computational reasons, as the per-task approach requires many optimization runs. In all experiments, the neural network architecture is a Multilayer Perceptron with $24 \times 64 \times 32 \times 4$ (input, two hidden layers, output) neurons.

V. Results

A. Domain-specific benchmark methods

We compare GP-constructed reanalyses against two baselines (*climatology* and *background forecast*) and two established data assimilation methods (OI and EnKF) [2], [3]. All these methods use the same evaluation protocol as GP-based approaches.

1) Climatology: the average of COSMO TLE EDR forecasts over the training data (from 25 September to 16 November 2025); serves as a natural baseline.

2) Background forecast: the raw COSMO TLE EDR forecast without assimilation of turbulence observations; any method performing worse indicates incorrect data assimilation.

3) Optimal Interpolation: minimizes the following cost function

$$\mathcal{J}(x) = ||x - x_b||^2_{B^{-1}} + ||y - H(x)||^2_{R^{-1}} \tag{1}$$

where x is the model state, x_b - the background forecast, B - the model error covariance matrix, y - observations, H - the observation operator, R - the covariance matrix of observation error, and $||z||^2_A = z^T A z$. With linear H (i.e., a matrix), the eq. (1) is minimized by

$$x_a = x_b + BH^T \left(HBH^T + R\right)^{-1} (y - Hx_b) \tag{2}$$

For turbulence reanalysis, H interpolates from the model grid at observation locations. Matrix B is calculated once and does not vary over time. We estimate B from the training data using the NMC method [24], based on differences between forecasts valid at the same time, but initialized at different times (e.g., forecast starting from time t valid at t+10h vs. the one starting from time t+6h valid at t+4h).

4) Ensemble Kalman Filter: is formulated analogously to OI, but estimates B based on all $L = 20$ member forecasts of COSMO TLE:

$$B = \frac{1}{L-1} \sum_{l=1}^{L} \left(x_b^{(l)} - \bar{x}_b\right) \left(x_b^{(l)} - \bar{x}_b\right)^T \tag{3}$$

where $\bar{x}_b$ is the ensemble mean.

Both OI and EnKF require the covariance matrix of observation error - R. Lacking prior knowledge of instrument uncertainty, independent errors with uniform variance are assumed. We tuned R on the validation set, testing several diagonal values; the best performance was achieved by setting the observation error standard deviation to 10% of the measured turbulence (but not less than 0.05) for OI, and 5% (but not less than 0.01) for EnKF.

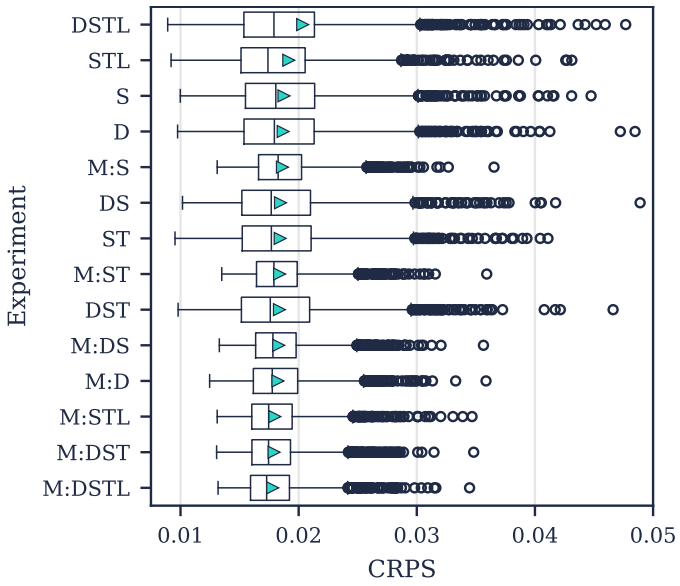

Fig. 4. Boxplot of CRPS values from 5-fold cross-validation on tasks from the test set, sorted by mean (teal triangles). For readability, the x-axis is limited to 0.05 CRPS; therefore, not all outliers are displayed. The "M:" prefix denotes meta-learning; no prefix denotes per-task optimization.

B. Overall results

Table II presents the results averaged over 3360 held-out folds (672 test tasks $\times$ 5 cross-validation splits). For our proposed method, we report two probabilistic metrics:

TABLE II

MEAN AND STANDARD DEVIATION OVER 3360 FOLDS (5 FOLDS × 672 TASKS). BOLDED VALUES INDICATE THE BEST RESULTS. CRPS AND JOINT NLPD METRICS CANNOT BE CALCULATED FOR TWO DOMAIN-SPECIFIC METHODS AS THEY DO NOT PROVIDE UNCERTAINTY ESTIMATES.

Experiment	CRPS	Joint NLPD	Joint NLPD fails	RMSE	MAE
climatology	n/a	n/a	n/a	0.0555 ± 0.0135	0.0406 ± 0.0031
background forecast	n/a	n/a	n/a	0.0473 ± 0.0198	0.0234 ± 0.0087
OI	0.0281 ± 0.0022	-1.2695 ± 0.0873	0	0.0423 ± 0.0177	0.0204 ± 0.0037
EnKF	0.0186 ± 0.0073	1.8481 ± 6.6779	0	0.0463 ± 0.0194	0.0225 ± 0.0075
spatial	0.0188 ± 0.0045	-1.6393 ± 0.5211	28	0.0436 ± 0.0175	0.0212 ± 0.0037
spatial × temporal	0.0184 ± 0.0044	-1.6545 ± 0.5437	3	0.0431 ± 0.0173	0.0208 ± 0.0035
spatial × temporal × linear	0.0192 ± 0.0313	-1.6362 ± 0.5754	49	0.0446 ± 0.0368	0.0217 ± 0.0322
deep	0.0187 ± 0.0046	-1.5839 ± 0.7148	42	0.0472 ± 0.0194	0.0216 ± 0.0039
deep × spatial	0.0185 ± 0.0045	-1.6103 ± 0.6607	9	0.0465 ± 0.0190	0.0213 ± 0.0038
deep × spatial × temporal	0.0184 ± 0.0044	-1.6133 ± 0.6897	2	0.0463 ± 0.0191	0.0212 ± 0.0037
deep × spatial × temporal × linear	0.0203 ± 0.0354	-1.6123 ± 0.6452	46	0.0470 ± 0.0421	0.0226 ± 0.0371
meta: spatial	0.0186 ± 0.0028	-1.6944 ± 0.3612	0	0.0429 ± 0.0167	0.0210 ± 0.0035
meta: spatial × temporal	0.0184 ± 0.0027	-1.7030 ± 0.3392	0	0.0420 ± 0.0165	0.0207 ± 0.0035
meta: spatial × temporal × linear	0.0180 ± 0.0027	-1.6803 ± 0.3869	0	0.0424 ± 0.0174	0.0204 ± 0.0032
meta: deep	0.0182 ± 0.0029	-1.6907 ± 0.3984	0	0.0438 ± 0.0173	0.0211 ± 0.0036
meta: deep × spatial	0.0183 ± 0.0027	-1.6933 ± 0.3530	0	0.0423 ± 0.0166	0.0209 ± 0.0035
meta: deep × spatial × temporal	0.0179 ± 0.0026	**-1.7068 ± 0.3604**	0	**0.0419 ± 0.0167**	0.0204 ± 0.0033
meta: deep × spatial × temporal × linear	**0.0178 ± 0.0026**	-1.6830 ± 0.3927	0	0.0421 ± 0.0173	**0.0203 ± 0.0031**

1) Continuous Ranked Probability Score (CRPS) [25]:

$$\text{CRPS}\left(\mathcal{N}_{\mu,\sigma^2}, y\right) = \sigma\left[z\left(2\Phi(z)-1\right) + 2\phi(z) - \frac{1}{\sqrt{\pi}}\right], \quad (4)$$

where $z = \frac{y-\mu}{\sigma}$ is the standardized prediction error, Φ is the cumulative distribution function and ϕ is the probability density function of the standard normal distribution, and

2) Joint Negative Log-Probability Density (NLPD) [7]:

$$\text{NLPD} = \frac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{y}-\boldsymbol{\mu}) + \frac{1}{2}\log|\boldsymbol{\Sigma}| + \frac{N}{2}\log(2\pi) \quad (5)$$

of the observation vector $\mathbf{y}$ given the predictive mean vector $\boldsymbol{\mu}$ and predictive covariance matrix $\boldsymbol{\Sigma}$, where N is the number of test points and $|\boldsymbol{\Sigma}|$ is the determinant of the covariance matrix.

Both metrics use uncertainty estimates – standard deviation and covariance matrix – provided by the GPs. We also count Joint NLPD calculation failures, which occur when the predictive covariance matrix is not positive semi-definite. Additionally, we report Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE), computed from mean predictions only.

Meta-learning combined with expressive kernel products– *DST* and *DSTL* in Table I–achieves the best results across all metrics. Meta-learning consistently outperforms per-task optimization, but the most striking difference lies in the variance of the results: standard deviations for meta-learning are 1.5–2 times smaller than for per-task optimization. Figure 4 clearly illustrates this difference. Please note that in per-task optimization, the predictive covariance matrix sometimes is not positive semi-definite, preventing Joint NLPD computation; such cases are excluded from the reported averages.

These results reveal an important insight. Per-task optimization can occasionally achieve excellent fits – the lower whiskers in Figure 4 extend below those of meta-learning. However, such cases do not dominate. More often, per-task optimization yields results that are similar to or worse than those of meta-learning. We further examine this in Section V-C. Meta-learning, by contrast, delivers consistent performance. This suggests that there is a one-fits-all hyperparameter set that captures the underlying physical structure of turbulence.

The results also demonstrate that combining kernels improves performance. Among meta-learning approaches, *ST* outperforms *S*, *STL* outperforms *ST*, and kernels incorporating the deep component – *DST* and *DSTL* – perform best.

Furthermore, all GP-based approaches outperform climatology; only the worst per-task optimized kernels match the background forecast or EnKF. In terms of RMSE and MAE, OI achieves accuracy comparable to meta-learning but requires careful estimation of B and R matrices.

C. An example of per-task optimization weak performance

One might expect per-task optimization to produce better fits, since it tailors hyperparameters to each individual task. We designed our experimental setup to support per-task optimization by running 20 Optuna trials per task to reduce sensitivity to initialization and selecting the best configuration based on validation performance.

Despite these safeguards, optimization failures occur and inflate the error distribution for per-task kernels, with the strongest effect on *STL* and *DSTL* (Figure 4). This instability affects even the simplest kernels. Figure 5 illustrates the worst case of the *S* kernel (1st cross-validation split for 2025-12-02 01:00 UTC). Per-task optimization achieves CRPS $= 0.0628$ – much worse than meta-learning's CRPS $= 0.0201$ for the same time period. The per-task reanalysis misses a high-turbulence encountered near the western boundary and simultaneously overestimates turbulence elsewhere.

VI. CONCLUSIONS

We developed a Gaussian Process framework for constructing turbulence reanalysis, enabling verification of regional tur-

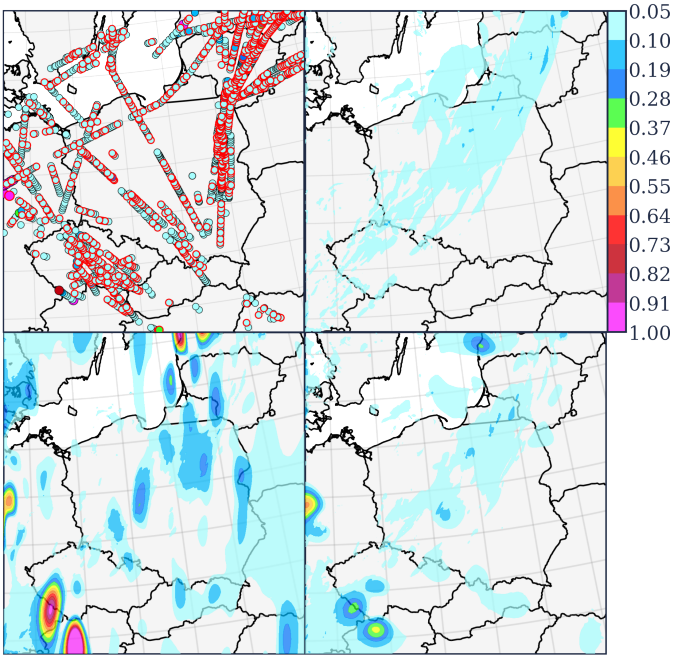

Fig. 5. Turbulence observations from all data sources between 2025-12-01 23:00 UTC and 2025-12-02 03:00 UTC, with observations belonging to the test fold in the 1st cross-validation split marked with red borders (top left); EDR ensemble forecast mean for 2025-12-02 01:00 UTC (top right); reanalysis constructed after per-task optimization with S kernel (bottom left); and reanalysis constructed after meta-learning optimization with S kernel (bottom right); color scale represents turbulence intensity.

bulence forecasts. GP-based reanalysis is competitive against established data assimilation methods while also providing uncertainty estimates alongside predictions, which enables the proper handling of observation error in forecast verification.

We compared two hyperparameter optimization strategies: per-task and meta-learning. Although per-task optimization occasionally achieves excellent fits, it just as often fails to find good hyperparameters. Meta-learning achieves consistent performance with lower variance, indicating that a single hyperparameter set captures the physical structure of turbulence across various atmospheric conditions. Meta-learning also reduces computational cost, as it requires only one optimization run rather than separately for each reanalysis.

We tested spatial, temporal, linear, and deep kernels, as well as their products. Combining kernels progressively improves performance. This suggests a direction for future work: exploring more expressive deep kernel architectures and training on larger datasets with a broader range of input features.

References

[1] R. Sharman and T. Lane, Eds., *Aviation Turbulence*. Cham: Springer International Publishing, 2016.

[2] R. Stull, "Numerical Weather Prediction," in *Practical Meteorology: An Algebra-Based Survey of Atmospheric Science*. University of British Columbia, 2015, pp. 745–792.

[3] G. Evensen, F. C. Vossepoel, and P. J. Van Leeuwen, *Data Assimilation Fundamentals: A Unified Formulation of the State and Parameter Estimation Problem*, ser. Springer Textbooks in Earth Sciences, Geography and Environment. Cham: Springer International Publishing, 2022.

[4] R. D. Sharman, L. B. Cornman, G. Meymaris, J. Pearson, and T. Farrar, "Description and derived climatologies of automated in situ eddy-dissipation-rate reports of atmospheric turbulence," *Journal of applied meteorology and climatology*, vol. 53, no. 6, pp. 1416–1432, 2014.

[5] S.-H. Kim, H.-Y. Chun, J.-H. Kim, R. D. Sharman, and M. Strahan, "Retrieval of eddy dissipation rate from derived equivalent vertical gust included in Aircraft Meteorological Data Relay (AMDAR)," *Atmospheric Measurement Techniques*, vol. 13, no. 3, pp. 1373–1385, 2020.

[6] J. M. Kopeć, K. Kwiatkowski, S. De Haan, and S. P. Malinowski, "Retrieving atmospheric turbulence information from regular commercial aircraft using Mode-S and ADS-B," *Atmospheric Measurement Techniques*, vol. 9, no. 5, pp. 2253–2265, 2016.

[7] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, 3rd ed., ser. Adaptive Computation and Machine Learning. Cambridge, Mass.: MIT Press, 2008.

[8] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, "Manifold Gaussian processes for regression," in *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2016, pp. 3338–3345.

[9] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *Artificial Intelligence and Statistics*. PMLR, 2016, pp. 370–378.

[10] M. Patacchiola, J. Turner, E. J. Crowley, M. O'Boyle, and A. J. Storkey, "Bayesian meta-learning for the few-shot setting via deep kernels," *NeurIPS*, vol. 33, pp. 16 108–16 118, 2020.

[11] F. Zanetta, D. Nerini, M. Buzzi, and H. Moss, "Efficient modeling of sub-kilometer surface wind with Gaussian processes and neural networks," *Artificial Intelligence for the Earth Systems*, 2025.

[12] 2026. [Online]. Available: https://doi.org/10.5281/zenodo.18442613

[13] J. Gardner, G. Pleiss, K. Q. Weinberger, D. Bindel, and A. G. Wilson, "Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration," *NeurIPS*, vol. 31, 2018.

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, and L. Antiga, "Pytorch: An imperative style, high-performance deep learning library," *NeurIPS*, vol. 32, 2019.

[15] S. W. Ober, C. E. Rasmussen, and M. van der Wilk, "The promises and pitfalls of deep kernel learning," in *Uncertainty in Artificial Intelligence*. PMLR, 2021, pp. 1206–1216.

[16] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Response to Promises and Pitfalls of Deep Kernel Learning," no. arXiv:2509.21228, 2025.

[17] C. A. T. Ferro, "Measuring forecast performance in the presence of observation error," *Quarterly Journal of the Royal Meteorological Society*, vol. 143, no. 708, pp. 2665–2676, 2017.

[18] Z. Ben Bouallegue, T. Haiden, N. J. Weber, T. M. Hamill, and D. S. Richardson, "Accounting for Representativeness in the Verification of Ensemble Precipitation Forecasts," *Monthly Weather Review*, vol. 148, no. 5, pp. 2049–2062, 2020.

[19] "Aviation weather center," https://aviationweather.gov/gfa/help/#products.

[20] G. Duniec, W. Interewicz, A. Mazur, and A. Wyszogrodzki, "Operational setup of the soil-perturbed, time-lagged Ensemble Prediction System at the Institute of Meteorology and Water Management–National Research Institute," *Meteorology Hydrology and Water Management. Research and Operational Applications*, vol. 5, 2017.

[21] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage AK USA: ACM, 2019, pp. 2623–2631.

[22] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11. Red Hook, NY, USA: Curran Associates Inc., 2011, p. 2546–2554.

[23] S. Sieradzki and J. Mańdziuk, "Modified Adaptive Tree-Structured Parzen Estimator for Hyperparameter Optimization," 2025. [Online]. Available: https://arxiv.org/abs/2502.00871

[24] D. F. Parrish and J. C. Derber, "The National Meteorological Center's Spectral Statistical-Interpolation Analysis System," *Monthly Weather Review*, vol. 120, no. 8, pp. 1747–1763, 1992.

[25] T. Gneiting and A. E. Raftery, "Strictly Proper Scoring Rules, Prediction, and Estimation," *Journal of the American Statistical Association*, vol. 102, no. 477, pp. 359–378, 2007.