
AI-BASED DATA ASSIMILATION: LEARNING THE FUNCTIONAL OF ANALYSIS ESTIMATION

A PREPRINT

 **Jan D. Keller***

Research and Development
Deutscher Wetterdienst
Offenbach, Germany
jan.keller@dwd.de

 **Roland Potthast**

Research and Development
Deutscher Wetterdienst
Offenbach, Germany
roland.potthast@dwd.de

ABSTRACT

The integration of observational data into numerical models, known as data assimilation, is fundamental for making Numerical Weather Prediction (NWP) possible, with breathtaking success over the past 60 years (Bauer et al. [2015]). Traditional data assimilation methods, such as variational techniques and ensemble Kalman filters, are basic pillars of current NWP by incorporating diverse observational data. However, the emergence of artificial intelligence (AI) presents new opportunities for further improvements. AI-based approaches can emulate the complex computations of traditional NWP models at a reduced computational cost, offering the potential to speed up and improve analyses and forecasts dramatically (e.g. Pathak et al. [2022], Bi et al. [2023], Lam et al. [2023], Bouallegue et al. [2023]). AI itself plays a growing role in optimization (e.g. Fan et al. [2024]), which offers new possibilities also beyond model emulation.

In this paper, we introduce a novel AI-based variational data assimilation approach (AI-Var) designed to replace classical methods of data assimilation by leveraging deep learning techniques. Unlike previous hybrid approaches, our method integrates the data assimilation process directly into a neural network, utilizing the variational data assimilation framework. This innovative AI-based system, termed AI-Var, employs a neural network trained to minimize the variational cost function, enabling it to perform data assimilation without relying on pre-existing analysis datasets.

We present a proof-of-concept implementation of this approach, demonstrating its feasibility through a series of idealized and real-world test cases. Our results indicate that the AI-Var system can efficiently assimilate observations and produce accurate initial conditions for NWP, highlighting its potential to carry out the data assimilation process in weather forecasting. This advancement paves the way for fully data-driven NWP systems, offering a significant leap forward in computational efficiency and flexibility.

Keywords AI · neural networks · data assimilation · variational methods · reanalysis · self-supervised learning · optimization

1 Introduction

In the dynamic and intricate field of Numerical Weather Prediction (NWP), the integration of observational data into numerical models stands as a cornerstone for making high forecast accuracy and reliability possible. This process, known as data assimilation, melds observations with model outputs to furnish refined analyses and initial conditions, thereby making the predictive prowess of NWP systems with high quality feasible. Traditional methods such as variational data assimilation techniques, the ensemble Kalman filter and its variants and recent developments on particle filters are the basis of current systems in this domain. There is a breathtaking improvement of forecast accuracy through algorithmical improvements and by adeptly incorporating a wide range of diverse observational data, e.g. [Houtekamer and Mitchell, 1998, 2001, 2005, Anderson, 2001, Lorenc et al., 2000, Lorenc, 2003, Kalnay, 2003, Hunt et al., 2007,

* www.dwd.de

Evensen, 2009, Anderson and Moore, 2012, van Leeuwen et al., 2015, Reich and Cotter, 2015, Nakamura and Potthast, 2015, Bishop, 2016, Vetra-Carvalho et al., 2018, van Leeuwen et al., 2019, Potthast et al., 2019].

Current developments in artificial intelligence (AI)-based approaches have the potential to considerably enhance NWP [e.g., Lam et al., 2023, Bi et al., 2023]. These sophisticated tools, leveraging deep learning techniques, are designed to replicate the complex physics-based calculations of traditional NWP models but at a significantly reduced computational cost for carrying out the forecasting step (called inference in the framework of machine learning). These NWP emulators unlock a potential not previously thought possible in terms of speeding up the forecasts generated by weather forecasting models. Weather predictions with lead times of a week or more normally taking hours to be performed can be calculated in seconds by AI emulators. This advancement not only expedites the NWP cycle, but also ensures that meteorologists can deliver timely and reliable weather predictions. On the other hand, classical data assimilation methods can thus become a bottleneck in the weather forecasting chain - taking up a much larger percentage of time needed to perform a NWP cycle. As the volume of meteorological data continues to grow, the gap in computing time needed between AI-based NWP models and the data assimilation step may even further increase.

Therefore, the advent of AI in NWP should also indicate a new era in data assimilation strategies, fostering the development of systems that intertwine well-tested traditional methodologies with the cutting-edge potential of machine learning. These innovative models aim to transcend the limitations of conventional techniques, offering a leap forward in refining initial conditions for NWP through AI-augmented assimilation processes [Gottwald and Reich, 2021, Dong et al., 2023, Arcucci et al., 2021, Bonavita et al., 2021]. One example is the proposition of merging machine learning with 4D-Var, suggesting a seamless integration of AI into the fabric of established data assimilation frameworks integrates deep learning methods into the data assimilation process. Already now, continuous learning of uncertain components of the forecasting model based on observations have been introduced into operational state-of-the-art forecasting systems such as the ICON model of DWD [Zängl, 2023], leading to much improved forecasting for surface variables. An important part of data assimilation systems is the mapping between model fields and the observations, known as *observation operators*. Using AI for improving or constructing observation operators has been a topic of research for many years, leading e.g. to an improved simulation of satellite radiances within RTTOV [Scheck, 2021, Baur et al., 2023], or for assimilating cloud images [Reinhardt et al., 2023].

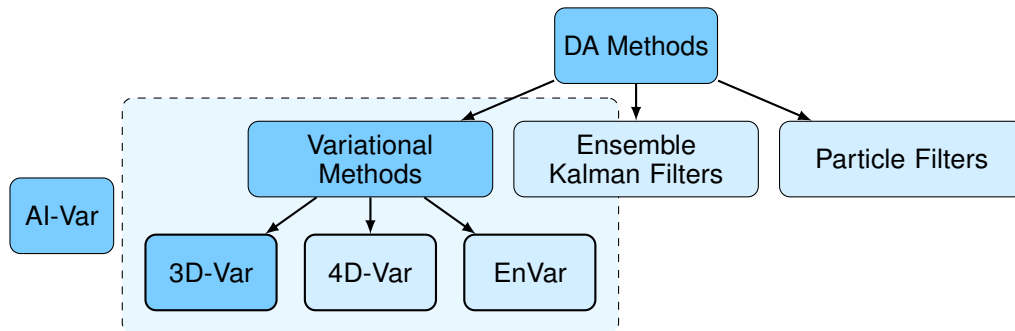


Figure 1: We will approach the task to develop an AI-based data assimilation method based on the 3D-Var functional.

An alternative to hybrid AI approaches, which are currently investigated, is to include observations directly in the AI emulators for NWP, thus, replacing the data assimilation step altogether [Geer, 2021]. However, current approaches to apply machine learning techniques in data assimilation aim to either substitute parts of the assimilation process with AI-based methods to improve performance [Farchi et al., 2021] or to extend the capabilities of current implementations to enhance the quality of the analysis [Buizza et al., 2022, Chipilski, 2023, Qu et al., 2024].

In this paper, however, we strive to replace the classical data assimilation scheme by training a neural network to perform the data assimilation task itself. Specifically, we introduce a concept of AI-based data assimilation which avoids the need of a long time-series of analyses as training set. This approach promises to completely replace a classical data assimilation systems and, thus, opening the field for fully data driven NWP systems in the future.

Integrating data assimilation into neural networks following variational approaches can build on recent advances in machine learning that have demonstrated the potential of neural networks to learn the complex functional mappings required for optimization, bypassing the computational bottlenecks of traditional methods. By leveraging these capabilities, neural networks can be trained to learn the cost function minimization process inherent in variational data assimilation, offering significant improvements in computational speed and solution accuracy.

In the broader literature, different approaches for solving this problem have emerged in recent years [Fan et al., 2024]. The relevant approach here is to train neural networks to map input parameters directly to optimal solutions [Liu et al., 2022]. Solutions are for example presented for integrating optimization for quadratic problems into neural network architectures [e.g., Effati and Ranjbar, 2011, Amos and Kolter, 2021]. The field of AI-based optimization includes the use of neural networks for guiding and speeding up the minimization such as *reinforcement learning* where the neural network learns a policy for optimization [Deng et al., 2022, Sukhija et al., 2023] and the task of *meta-training* which employs trained models to help training or minimizing other models [Wichrowska et al., 2017, Li and Malik, 2016]. One key approach for practical applications is function-based learning [e.g., Briden et al., 2023] that trains neural networks using decision-focused loss functions. Overall, integrating neural networks into variational data assimilation leverages recent advances in machine learning to offer a path toward more efficient and accurate state estimation.

Starting with the basic equations of 3D-Var, we present a proof-of-concept for our new AI-based algorithmic data assimilation approach that enables the utilization of deep learning architectures for the assimilation process, where the full AI training can be carried out based on observations and first guess fields only. Being based on the concept of variational data assimilation, we implement the algorithm directly into the training concept of an AI based assimilation system, thus, effectively embodying an *AI-Var*.

The remainder of the paper is structured as follows. Section 2 is dedicated to the description of our algorithmic approach, with Section 3 outlining our experiment framework. Our results are presented in Section 4, followed by the discussion and conclusions in Section 5.

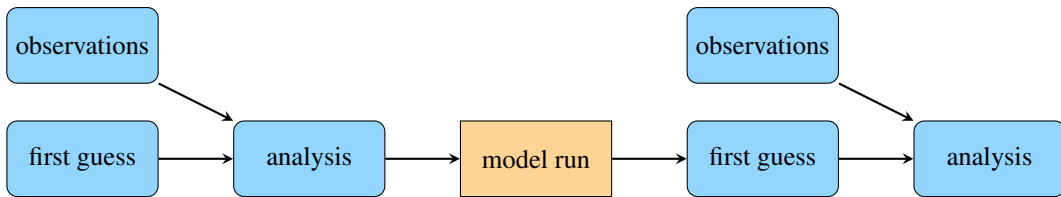


Figure 2: The classical DA cycle, where we replace the step which integrates observations and first guess into an analysis by a direct AI-based assimilation step.

2 Methodology

Current approaches to apply machine learning techniques in data assimilation aim to either substitute parts of the assimilation process with AI-based methods to improve performance and / or to extend the capabilities of current implementations to enhance the quality of the analysis. In this paper, however, we develop a concept to replace the classical data assimilation scheme by training a neural network to perform the task.

2.1 AI-based Data Assimilation Concepts

While previous approaches to utilize AI in data assimilation focused on either extending or substituting aspects of the process, we are here striving to fully integrate data assimilation into a neural network. While machine learning is in general based on the idea of learning relationships from large data sets, we follow a different approach.

A standard approach of many machine learning applications is to use the AI-based emulators to construct fast and flexible mappings, given input data and the respective desired outcome. For the case of data assimilation, this approach is shown in Figure 3 as *Approach 1*. In this first approach, a training data set of triples is needed which for each time step consists of *observations* y_ξ , *first guess* x_ξ^b , and *analysis* x_ξ^a , such that we would use the set

$$\mathcal{S}_1 := \{s_\xi = (y_\xi, x_\xi^b, x_\xi^a), \quad \xi = 1, \dots, n_t\} \quad (1)$$

with the number n_t of training samples. While this approach appears to be straightforward, it will always require a classical analysis system as basis for the training. Approach 1 is therefore characterized through a strong dependence on the conventional approaches.

Instead of learning the relationship between input data (i.e., first guess x^b , observation y) and the desired outcome, i.e. the analysis x^a , through training the network on the analysis field as a target, we set the AI model to learn the functional $f(x^b, y) \mapsto x^a$ itself. Therefore, we will make use of the internal loss function of the neural network model. While the loss function commonly only measures the difference between the model output and the provided target in the training, we replace it with a term which penalizes the departure of the outcome (i.e., the analysis) to the first guess plus a term

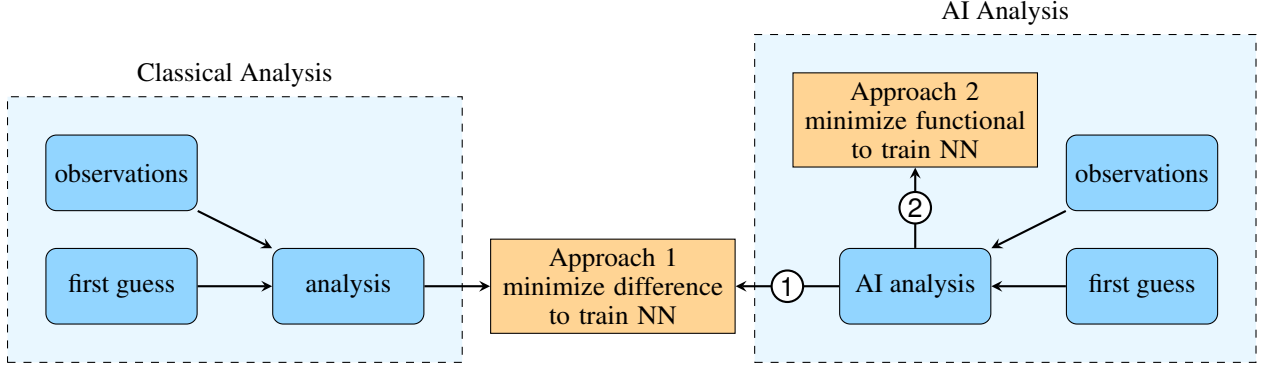


Figure 3: The conventional approach to training a neural network is to minimize the difference between a prescribed analysis and the AI analysis, here Approach 1. The alternative is to minimize a functional as in classical variational minimization, such that the classical analysis is not needed to carry out the minimization.

which measures the difference of the model output to the provided observations with weights given by the background and observation error covariance, respectively. Therefore, we end up with a loss function which is exactly the cost function that is commonly used in data assimilation with 3D-Var (compare Figure 1):

$$l = (\hat{\mathbf{x}} - \mathbf{x}^b)^T \mathbf{B}^{-1} (\hat{\mathbf{x}} - \mathbf{x}^b) + (\hat{\mathbf{y}} - \mathbf{y})^T \mathbf{R}^{-1} (\hat{\mathbf{y}} - \mathbf{y}) \quad (2)$$

where $\delta\hat{\mathbf{x}} = \hat{\mathbf{x}} - \mathbf{x}^b$ is the analysis increment as the output of the neural network with $\hat{\mathbf{x}}$ as the analysis state, \mathbf{B} the background error covariance matrix, \mathbf{y} the observations, $\hat{\mathbf{y}} = H(\hat{\mathbf{x}})$ the model equivalents of $\hat{\mathbf{x}}$, i.e., the output transformed into observations, and \mathbf{R} the observation error covariance matrix. $\delta\hat{\mathbf{x}}$ is obtained as the model output by minimizing the functional (2).

We make use of the fact that both variational data assimilation algorithms as well as the AI training algorithm minimize a functional. If we employ the same functional (2) as loss function for training our neural network, we construct the AI-based emulator for data assimilation such that it solves a particular minimization problem – the data assimilation problem of 3D-Var.

When we train a neural network with the loss function (2), we do not need to prescribe an analysis x_ξ^a for the input samples x_ξ^b and y_ξ . In this case, our training set reduces to

$$\mathcal{S}_2 := \{s_\xi = (y_\xi, x_\xi^b), \quad \xi = 1, \dots, n_t\} \quad (3)$$

with the number n_t of training samples. This means that we do not need to prescribe an analysis for training. But we need to discuss the role of the first guess fields x^b in the whole data assimilation cycle shown in Figure 2. We first note that in classical data assimilation the analysis depends on the observation and the first guess, and of course the first guess is generated from an earlier analysis – there is a cyclic dependence of the analysis and the first guess through the model propagation in the data assimilation cycle. Usually, when starting a data assimilation cycle, one would try to generate an analysis with a preliminary first guess followed by a spin-up period for the cycle.

The challenge is even stronger for the AI-based analysis algorithms, since we do not solve the minimization problem in each step, but we train an emulator to solve this minimization problem – and the training is carried out by solving many minimization problems. However, when a reasonable first guess is available, we can use it for training. When no first guess is available, we might work with $x^b = 0$ initially. When a first version of an analysis algorithm after training with $x^b = 0$, one can

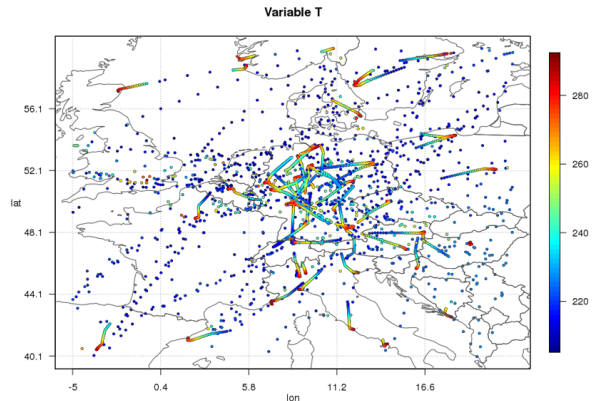


Figure 4: Examples of highly dynamical observations from airplanes over central Europe within one hour. Observations are AIREP upper air temperature in Kelvin.

construct a better analysis algorithms based on the available first guess fields calculated from available analyses in an iterative manner. Here, we will focus on the analysis step itself, showing that indeed the above approach is feasible, further refinements through iterations will be a topic of future research.

Approach 1 is using the *supervised learning* technique in the sense that the target solution x^a is prescribed and the AI-based analysis is trained to get as close to the target as possible. Unsupervised learning algorithms carry out their tasks without such prescribed targets or labels, respectively. Our Approach 2 is a self-supervised learning approach that is a sub-category of unsupervised learning algorithms. For further details on these categorizations we refer to [James et al. \[2013\]](#), [Bishop \[2006\]](#), [Goodfellow et al. \[2016\]](#), [Hastie et al. \[2001\]](#). For our conceptional tests we have chosen three basic configurations:

- a) a *one-dimensional* idealized configuration,
- b) a *two-dimensional* idealized configuration,
- c) a *real-world* test case (2m temperature analysis).

In addition, we will investigate two basic *observational setups* that are encountered in real world data assimilation problems:

- (1) the observations are available in a *fixed* configuration - static observation locations,
- (2) the places where observations are taken are *flexible* - dynamic observation locations.

Both observational setups reflect real-world problems of high relevance. For example, our real-world case when assimilating SYNOP two-meter temperature measured at standard operational weather stations falls into category (1). When we want to assimilation measurements of commercial airplanes as shown in Figure 4, which are important observations of today’s global or regional forecasting systems, the observational setup would be of category (2).

2.2 Assimilation of fixed-position observations

For testing the feasibility of the new Approach 2 we have used PyTorch, the well-known open-source machine learning library developed by Facebook’s AI Research lab. We employed PyTorch version 2.2.0 for neural network training and experimentation, see [PyTorch \[2024\]](#). Pseudo code for the core procedure is given in the Appendix A. Conceptually, we need to provide (a) *input fields*, (b) implement our choice of a neural architecture with input and output dimensions and (c) define a loss function which the minimizer of pytorch can use. Further, we need to provide a training and evaluation data set for the minimization framework to run.

In the basic configuration a) defined in Section 2.1, where observations are at fixed locations, we can calculate observation equivalents from model fields based on a fixed set of parameters given by the observation locations. The observation operators usually carry out an interpolation of model fields to observation locations, often in combination with further physical processes such as the transmission of radiation. Here, we have chosen to stay with a simple framework where the model field consists of one variable and we measure the model field in a selection of model grid locations. In this case, the observational parameters are a list of indices of the model grid points where the values are measured. The observations are modeled by a simple array of scalar values. In the case of m temperature observations at particular locations, this array will consist of m values, for which the model equivalents can be obtained taking the m indices of the model grids where the measurement took place, and evaluating the corresponding field values.

(a) For PyTorch, the input is a combined array of values (x^b, y) where x^b is the first guess field transformed into the inherent PyTorch tensor structure, and y is the array of observations. With n gridpoints and a field of one variable only, the tensor (x^b, y) has $n + m$ values.

(b) The neural architecture has been adapted to the three cases a)-c) and will be described in more detail in the sections below. We basically used dense networks for our test cases as shown in Figure 5.

(c) The cost function can be provided in PyTorch as a function based on its own data structures, the PyTorch tensors. The tensorial structure allows us define the complex loss function (2) as an implementation of the Mahalanobis distance (see [Mahalanobis \[2019\]](#)) easily even in the multi-dimensional case.

Finally, we need to define a training data set. For the test configurations a) and b) we have chosen set of prescribed *true* functions for simulation, used a random number generator to select a set of indices to fix observation locations, and calculated artificial observations based on this data set. We have split the data set into a training data set and an evaluation data set. With these ingredients the setup of the training loop based on a data loader, an optimizer, and the optimization criterion is straightforward in PyTorch.

2.3 Assimilation of flexible-position observations

For the case of flexible-position observations a simple approach is to add the location of the input fields to the input vectors given to the AI-based system. This means that we have a parameter vector p_ξ which depends on the sample index ξ which together with the first guess x_ξ^b and the observations y_ξ , $\xi = 1, \dots, N$ with N being the number of training samples is given as input to the system. The set of training samples now takes the form

$$\mathcal{S}_3 := \{s_\xi = (y_\xi, p_\xi, x_\xi^b), \quad \xi = 1, \dots, N\}, \quad (4)$$

where in the simplest possible case of linear indices the dimension of y_ξ and p_ξ is identical. Depending on the particular format of indexing chosen, of course p_ξ can take tensorial form. The PyTorch system is able to deal with any of these, as long as the structures are transformed into appropriate PyTorch tensors.

We remark that adding the locations to the system increases the computational complexity of the learning task considerably. Initially, we naively added the indices, but in these cases our training tasks failed. As described in [Liu et al., 2022], for the case where the order of the observations does not change the minimization problem, deep sets and *permutation invariant* neural architectures can be used to reduce the complexity of the training task. Here, we employed a simple approach to such methods by ordering the observations based on their location in the grid, which leads to a uniform and permutation invariant structure for all permutations of observations in the data assimilation task. With this approach we were successful to achieve meaningful results.

3 Experiment setup

In this section, we describe the data used in the three test cases for our data assimilation learning approach. We employ two idealized cases, one- and two-dimensional, respectively, as well as a real world example.

3.1 1D setup

The first case examines our data assimilation learning algorithm using data from one-dimensional functions. We use two different basic functions to generate these data.

In our setup, the x-axis values represent a linearly spaced sequence of $n_x = 50$ values in the interval $[0, 2\pi]$. With $f \sim \text{Uniform}(0.5, 1.5)$ a random forcing value and $x_t = x - t \cdot \Delta t$ the shifted x values along the x-axis for n_t number of time steps, we define *sinusoidal curves* as $z_t^s = \sin x_t$ and *parabolas* using $z_t^p = f \cdot x_t^2$. The data set is constructed such that, at each time step, the data is randomly chosen from each of the two types of curves. Examples of the resulting curves are shown in the results section in Figure 7.

To test the functionality of the data assimilation approach in the 1D idealized case, we apply the following experiment setup. First, we sample our observations from the training data set of true model states. We choose the first guess state to be zero and, then force the model to reconstruct the true fields by assimilating the observations. In this case, to give the observations an appropriate impact reflecting a large uncertainty of the background, we introduce artificial weights $w_b = 0.001$ for the background term and $w_o = 0.999$ from the observation term into the loss function.

In order to investigate the performance of our model for the data assimilation purposes, we employ two common scenarios - a static and a dynamic layout of observation locations. We then investigate the robustness of the model for the different scenarios with respect to three hyperparameters. First, we train models with different number of observations $n_o = 5, 10, 15, 20$ to represent cases from a sparsely to densely observed space. Second, we use different Gaussian kernel widths sigma ($s = 0.5, 1., 2., 4.$) for the B-matrix to account for various radii of influence for the observations on the state space. Last, we train the model with different training sample sizes of $n_t = 250, 500, 1000, 2500$. For all experiments we use 100 independent samples for evaluation. The samples are drawn without replacement using the different random forcings f .

Our architecture consist of a simple feed forward neural network with three hidden layers as depicted in Figure 5. The input of the neural network is the first guess field (size n_x) as well as the observations (size n_o) and the observation indices in the spatial dimension (also size n_o), made permutation invariant as described in the methodology section. For

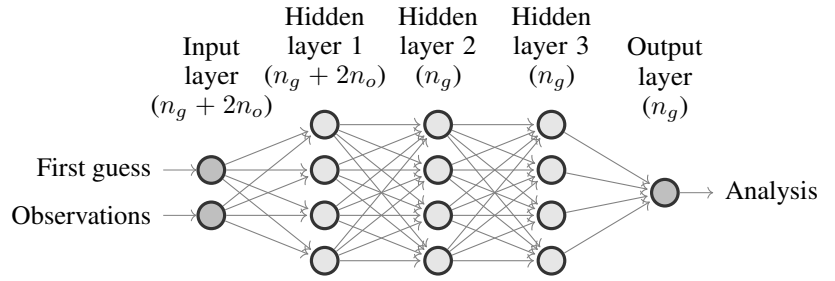


Figure 5: Network architecture for the idealized cases. Below the layer names, the dimension of each layer is provided. Here, n_g is size of the grid space (i.e., $n_g = n_x$ for the 1D case and $n_g = n_x \cdot n_y$ for the 2D case) and n_o is the number of observations.

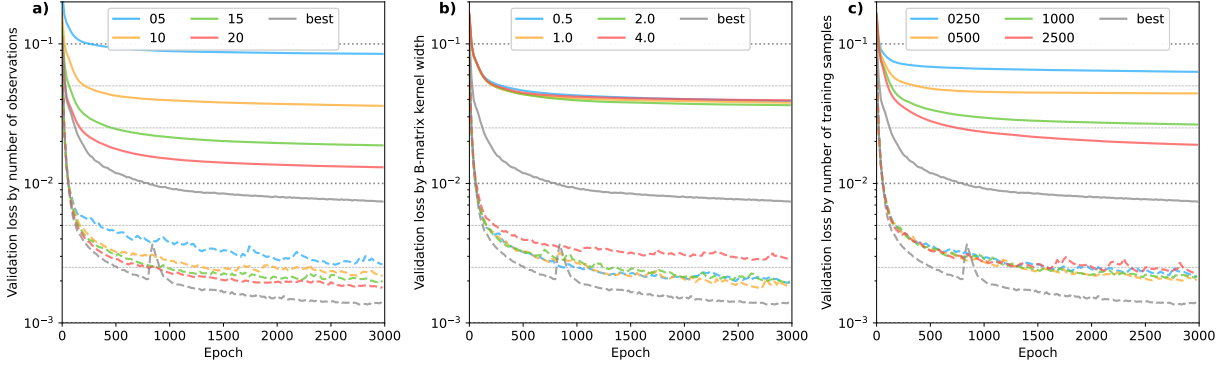


Figure 6: Validation loss curves for the 1D idealized case for the two scenarios - static (dashed lines) and dynamic (solid lines) observation locations. The curves represent the mean validation loss over the group of respective training runs with the same a) number of observations, b) B-matrix kernel width, and c) number of training samples. Therefore, each line is the average over 16 data sets, respectively.

the implementation of the algorithm, we use the python package PyTorch. The used optimizer is Adam and the learning rate is set to 0.001.

3.2 2D setup

To systematically investigate the functionality of our approach for a more realistic data assimilation problem, we employ a 2D setup using trigonometric functions which is designed similar to the 1D case. Here, we use a 2D grid described by the coordinates x and y . Then, we define the factors f_x and f_y determining the different displacement in x and y directions for the samples through $c_x = f_x \cdot \pi \cdot t/n_t$ and $c_y = f_y \cdot \pi \cdot t/n_t$, respectively. The 2D field z is then calculated by $z = \sin[(\pi/5) \cdot (x - c_x[t])] \cdot \cos[(\pi/8) \cdot (y - c_y[t])]$. Finally, the array z is permuted along the sample axis. The idealized 2D experiments follow the exact same setup as for the 1D case (cf. section 3.1) including the model architecture and parameter settings, including the *permutation invariance* described above. Examples for the resulting fields are shown in the results section in Figure 10.

3.3 Real case setup

In order to evaluate the potential of our AI-Var approach in a real world setting, we add another experiment setup where we apply our approach to assimilate 2-meter temperature (T2M) observations from synoptic observation sites into T2M fields from an NWP model. This setup corresponds to our 2D idealized static observation location experiment. As gridded data, we choose hourly T2M fields from DWD’s COSMO-REA6 regional reanalysis [Bollmeyer et al., 2015]. Instead of using the full data set, a subdomain of 20×25 grid points (at 6km horizontal resolution) over central Germany is selected to reduce the complexity. While conventional observations were assimilated in the generation process of the reanalysis, T2M measurements have only been introduced indirectly via a soil moisture reanalysis once a day. Therefore, analyses of the reanalysis T2M estimates showed considerable biases, RMSE and correlation mismatches [Keller and Wahl, 2021].

The observational data comes from DWD’s climatological archive where synoptic measurements are stored for German stations. The subdomain includes 25 stations with hourly observations. Times where data from any station are missing have been removed from the whole data set.

Our test period comprises four years (2010-2013) where we only use the summer months to concentrate on Northern hemisphere summer. The data set comprises 8658 samples which are randomly permuted along the time axis. We retain 500 data points as validation data set and use the remaining 8158 samples for training with a batch size of 4096. We employ the same setup as for the 2D case except for a few tuning choices. (1) To better account for the complexity of the underlying dynamics, we provide the model surface height (which is closely connected to T2M) as a 2D field as well as time of day as a scalar to the model. (2) We adapt the neural network architect by increasing the number of neurons in hidden layer 1 and 2 to $n_g \times 4$ and $n_g \times 2$, respectively.

(3) We still use a Gaussian B-matrix which might not be the best choice but should suffice in this proof-of-concept, but we set the B-matrix kernel width to 0.2 and the observation error to 0.1.

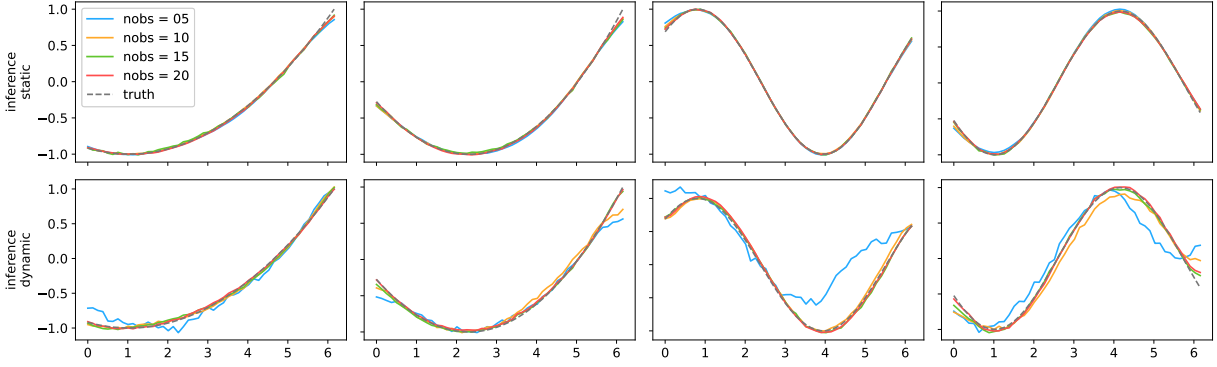


Figure 7: Examples of the 1D idealized experiments. The plots depict the 1D fields for four inference samples of the validation data set (columns) from the static (top row) and the dynamic observation locations (bottom row). The grey dashed lines denote the respective true state.

In a secondary test setup, we withhold observations to allow for an independent verification data set. In this cross-validation setup, we perform five model training runs with $n_o = 20$ by each time keeping a subset of five observations from the original data set for evaluation in such a way that we have an independent model for each observation.

4 Results

In this section, we show the results from our proof-of-concept implementation of an AI-based data assimilation algorithm. We first present the results for the idealized 1D and 2D cases and conclude this section with a simple real-world 2D example.

4.1 Idealized 1D case

We first look at the loss curves of the training of our data assimilation AI model. Specifically, we are interested in the validation loss, as the aim of the approach is to be applied to independent data. The three plots in Figure 6 show the average validation loss for all training runs with a specific parameter setting as described in Section 3.1, i.e. each colored line represents the average over 16 different models. The dashed lines are determined using a static observation setup whereas the solid lines indicate the average over models with dynamic observation positions. The grey lines indicate the best performing model for both cases.

The training loss (not shown) as well as the validation loss converges for all models. The comparison (Fig. 6) further shows that the models with static observation locations perform better than the ones with dynamic observation locations (i.e., the former are exhibiting a lower average validation loss). This was expected as the static structure allows for the neural network to better learn the characteristic of the underlying fields with respect to the provided observations.

Looking at the parameter "number of observations" (Fig. 6a), we find a strong deviation in performance between the models especially for the dynamic setup with 20 observations showing a validation loss nearly one order of magnitude lower than the models with only 5 observations. Remembering that our 1D space has 50 grid points, the case with 20 observations prescribes already nearly half of the state space, thus, significantly reducing the number of grid points to be determined compared to the 5 observation case. For the static observation locations, the validation loss varies only slightly with respect to the number of observations (between 0.002 and 0.004 at the end of the training), again, with 20 observations exhibiting the lowest validation loss.

For the various B-matrix kernel width settings (Fig. 6b), we find that the parameter does only exhibit a very small impact on the performance of the model with dynamic observation locations. For the static setup however, we find that while the lower settings have similar validation losses, a value of 4.0 seems to imply a too strong smoothing which increases the validation loss.

The number of training samples seems to be a crucial parameter for the performance of the models with dynamic observation locations. Figure 6c shows that there is strong variation among the different settings with the largest number of samples (2500) having the smallest validation loss. This is founded in the ability of the model to generalize from the changing observation network layout by more samples and, thus, a larger number of different combinations of

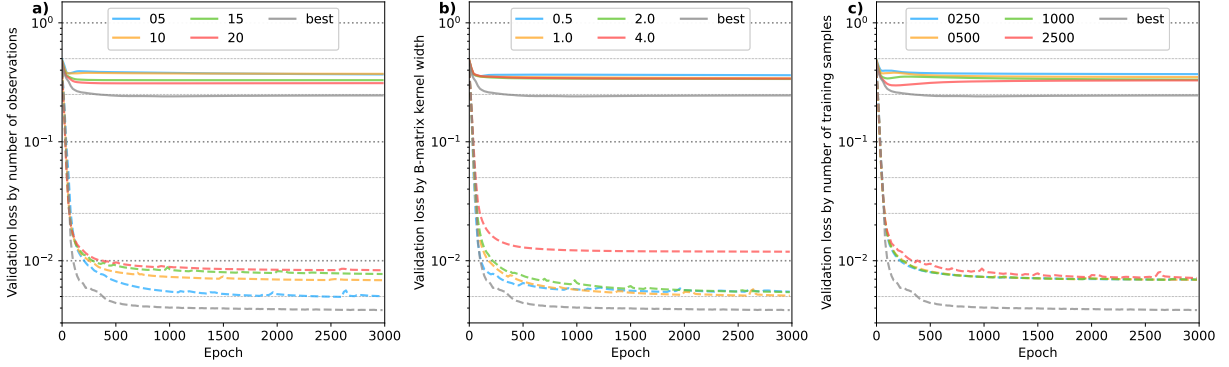


Figure 8: Validation loss curves for the 2D idealized case for the two scenarios - static (dashed lines) and dynamic (solid lines) observation locations. The curves represent the mean validation loss over the group of respective training runs with the same a) number of observations, b) B-matrix kernel width, and c) number of training samples. Therefore, each line is the average over 16 data sets, respectively.

observation locations. The impact on the static observation location setup is, however, only small, as a limited number of samples is sufficient for the model to understand the underlying structure of the fields.

To take a closer look at the generated fields, Figure 7 shows four exemplary samples from the validation data set (columns) - two parabolas (left) and two sinus curves (right) with the respective inference results for all four numbers of observations n_o with $n_s = 2500$ and $s = 2.0$. In the top row, the curves are reconstructed with the static observation locations setup whereas the bottom row depicts the results from the dynamic observation locations.

The results indicate that our AI-Var approach is able to very closely reconstruct the true state for the static case for the given parameter settings independent of the number of observations. For the dynamic case, we find that for $n_o = 15, 20$, the model is also nearly identical to the truth except for some data points close to the boundaries when there are no nearby observations. While the estimates slightly deviate from the truth for $n_o = 10$, we find that for $n_o = 5$, the inference is not very smooth anymore and shows larger discrepancies from the original state. This indicates that for a less dense observation network, a larger training data set can compensate this issue for the static case, but it seems to be much more complicated to achieve good results for the dynamic case.

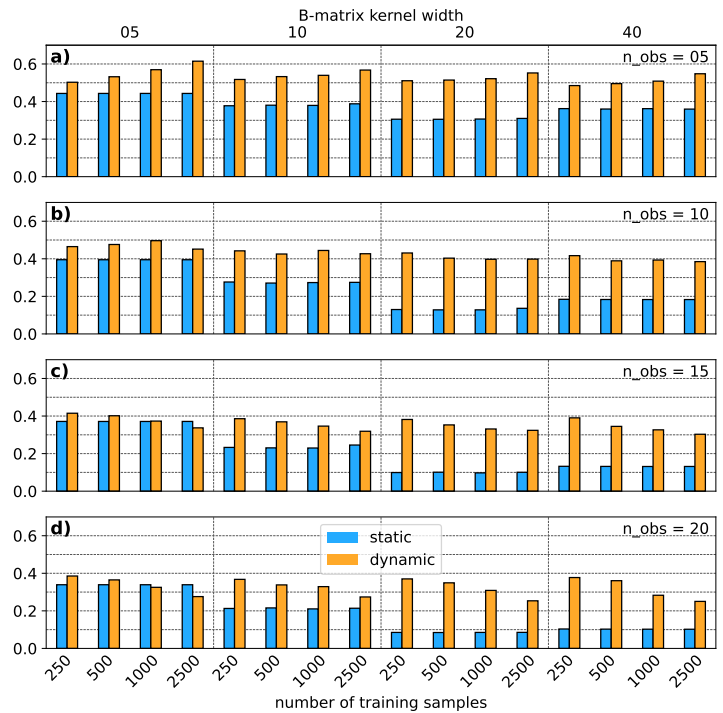


Figure 9: RMSE for inference against truth over all 100 validation samples and all grid points for the static (blue) and dynamic (yellow) observation locations. The four plots represent the results for the models based on a) 5, b) 10, c) 15 and d) 20 observations.

4.2 Idealized 2D case

We now analyze the results with respect to the idealized 2D case. Figure 8 depicts the average validation loss over the training epochs for all training runs similar to the 1D case (cf. Fig. 6). As expected, we find that independent of the hyperparameter settings, the static observation locations exhibit lower validation losses compared to the dynamic setup. This difference is considerably increased in comparison to the 1D case which can be attributed to the more complex 2D situation.

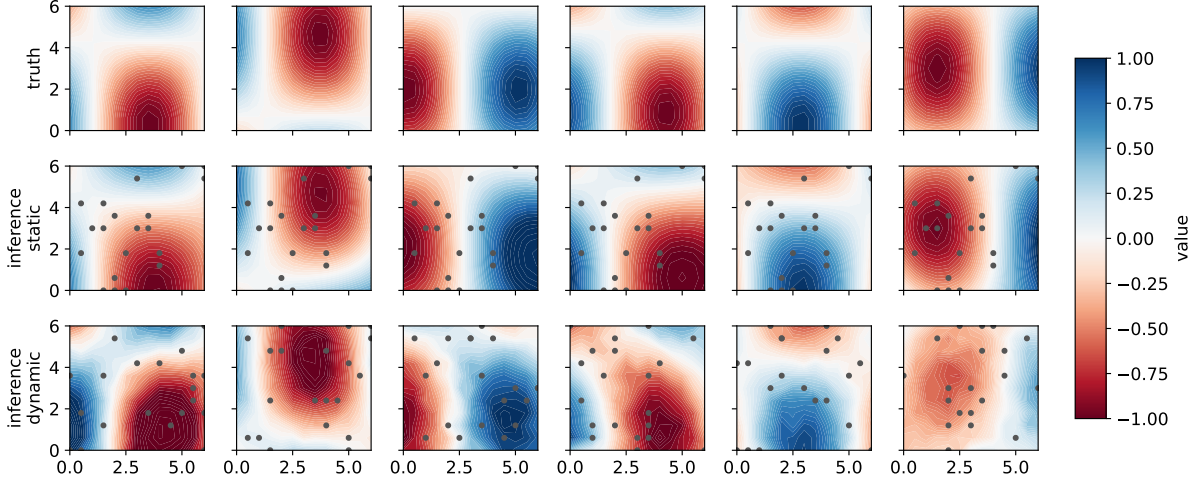


Figure 10: Examples of the 2D idealized experiments. The plots depict the 2D fields for six samples of the validation data set (columns) from the truth (top row), the inference (i.e. the AI based analysis) from the static (middle row) and the dynamic observation locations (bottom row). The grey dots denote the respective observation locations.

In Figure 8a, we find that the impact of the number of observations is far more pronounced for the static case than for the dynamic observation locations. For the latter, we see that the best performing models are those with the largest number of observations. However, for the static locations, this effect is reversed with lower number of observations showing a lower validation loss. For the B-matrix kernel width (Figure 8b), most of the respective models exhibit a similar performance except for a width of 4 for the static case where the strong smoothing appears to hinder the ability of the neural network to estimate the observations, thus leading to a considerably higher validation loss. For the training sample size (Figure 8c), we find that the results are similar for the static observation locations, indicating that such a setup is robust also against smaller sample sizes. For the dynamic locations, we find that larger sample sizes lead to smaller validation losses. However, especially for 2500 samples, we see that the average validation loss decreases until a minimum is reached (around epoch 150) and then increases until convergence is reached. As we look at the validation loss here, this indicates that while larger sample sizes can improve the model quality, there is a considerable risk of overfitting the model towards the training samples.

To further investigate the performance of the AI-based DA approach in a 2D idealized setting, we perform the inference on the validation data set with all models and compare the RMSE of the estimated fields to the truth. Figure 9 shows the average RMSE over all grid points and all samples for the static (blue) and dynamic (yellow) observation locations setup. The four plots indicate the different number of observations as these are usually given in a real world setting. In general, we find that the model for the static setup is much less insusceptible towards the B-matrix kernel width whereas the models for the dynamic locations perform much better with larger B-matrix kernel widths. For the latter, a setting of 2 seems to be the best of the choices for this case. In comparison, the number of training samples has a much larger influence on the performance of the models for static locations than for the dynamic locations. The influence on the former also depends on the number of observations. With 5 observations, smaller samples produce higher RMSEs whereas for 20 observations, the best results can be achieved with larger number of samples. This clearly indicates that the model’s tendency for overfitting is dependent on the complexity of the situation (i.e., the number of observations).

Six examples of the reconstruction from the validation data set for both the dynamic and the static case can be found in Figure 10. The inference is performed for the models with 2500 training samples, 20 observations and a B-matrix kernel width of 2.0. The top row depicts the original fields, whereas the center and bottom row show the reconstruction for the

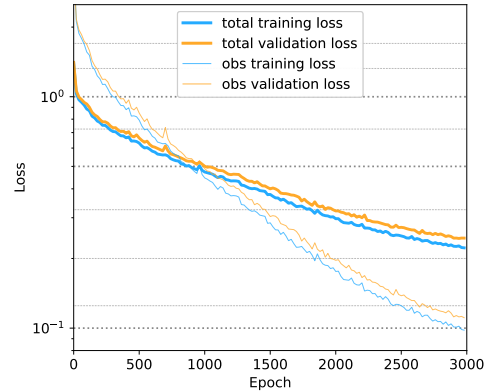


Figure 11: Loss function for the real test case. The thick lines depict the total loss whereas the thin lines are the loss coming from the observation term. Blue indicates training loss and yellow denotes validation loss.

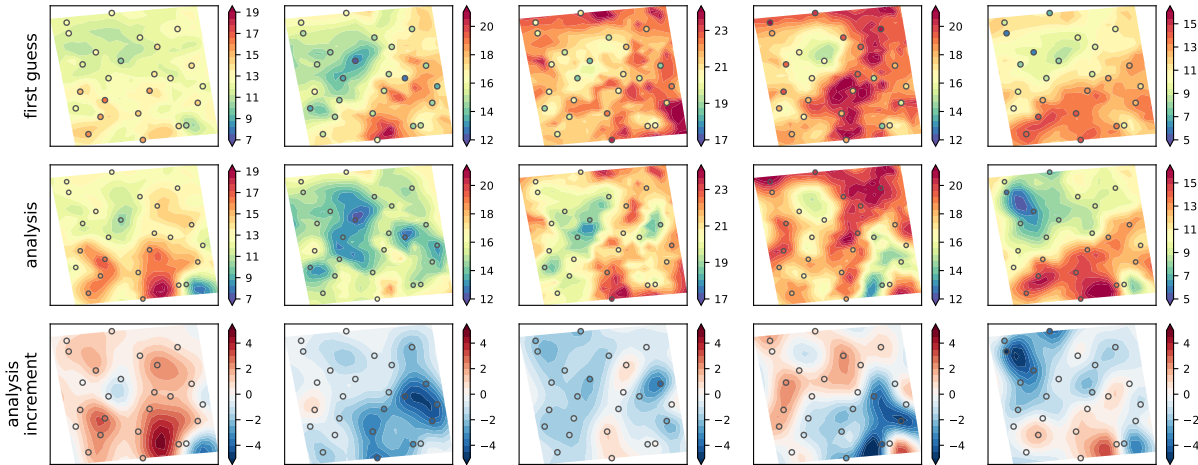


Figure 12: Five examples from the validation data of the real case experiment. The contours show the first guess field in the top row and the AI-Var analysis in the center row. The circles denote the respective T2M observations. The bottom row depicts the contours of the analysis increments and the deviation of the observations from the first guess in the circles.

static and dynamic observation locations, respectively. The dots denote the observation location for each example. We find that the models are able to quite reasonably reproduce the 2D structure of the truth. While in the static case, the differences between truth and reconstruction are only marginal, some deviations become apparent for the dynamic case.

In summary, the results for 2D idealized experiments are very promising indicating the potential for the application in a real test case.

4.3 Real test case

Using the experiment setup described in section 3.3, we apply our AI-Var to a real world example for data assimilation.

As usual, the loss is decreasing with increasing epochs as shown in Figure 11. Here, training and validation loss move in parallel, i.e., an improvement in the model as indicated by a lower training loss also leads to a decrease of the validation loss. However, with the training loss still trending downward induced by the steep reduction of the observation loss term, the model is not yet fully trained and may further improve as there is no evidence of overfitting yet, i.e., the validation loss keeps decreasing, too. Due to the complexity, a larger number of training epochs may be reasonable. However, as the following results indicate, the model reaches a sufficient level of quality for your proof-of-concept approach.

Figure 12 shows the application of the trained model to the validation data set. The contour plots depict the first guess fields from the reanalysis (top row) and the analysis from our AI-Var (center row). The circles denote the observation locations with the colors indicating the observed value in the same color range as the contour plots. The bottom row depicts the filled contours of the respective analysis increments with the circles showing the deviation of the observations from the first guess.

While some observations agree with the first guess fields, there is often a mismatch between the observed and estimated T2M in the first guess. From the center row, we find that AI-Var is able to locally correct for these deviations such that the observation match the analysis field very well. This can also be deduced from the bottom row where we find that for almost all observations, the analysis increment matches the error of the first guess to the observed values.

Another evaluation is given in the left plot in Figure 13 which depicts the diurnal cycle of the mean absolute deviations of the observations from the first guess (blue) and AI-Var analysis (yellow). The single observations are denoted as thin lines, whereas the thick lines are the average over all observations. The upper plot shows the result for the validation period of our real case experiment. We find that there is a clear diurnal cycle of first guess errors with respect to the observations with larger deviations becoming larger during the night time. This is due to a nightly warm bias in the COSMO-REA6 reanalysis which is well known. However, with our AI-Var, we are not only able to significantly reduce the error but also eliminate the diurnal cycle of the T2M errors from the first guess field (80%-90% reduction).

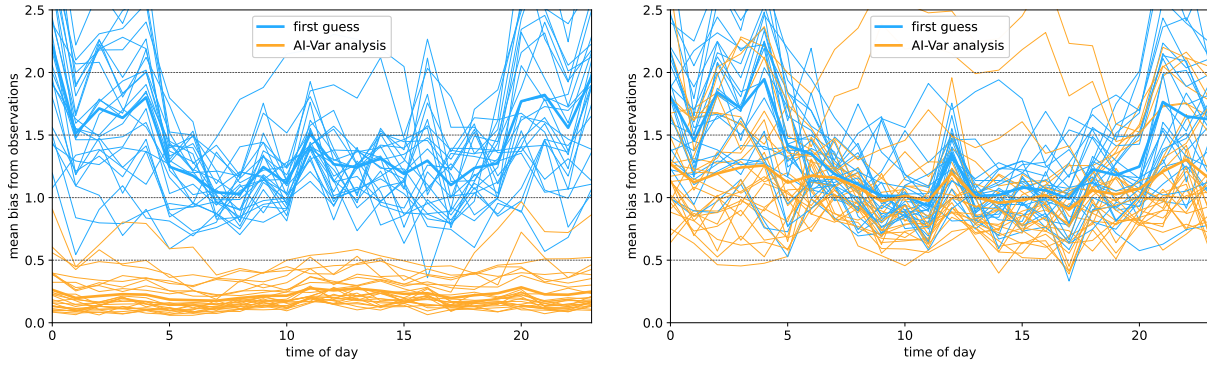


Figure 13: Diurnal cycle of first guess (blue) and AI-Var analysis (yellow) deviations from the observations in the validation data set. Thin lines indicate the individual 25 observation sites whereas the thick line denotes the mean over all stations. The left plot shows the experiment using all observations, the right plot the cross-validation experiment.

In our cross-validation experiment for the real case, we try to evaluate the merit of our AI-Var system using independent observations. The evaluation of the five training runs withholding five different observations from the full observational data set in each run, are combined to confirm the validity of our approach. We find a reduction of root mean square error from first guess to analysis in 20 of the 25 observations with an average reduction of 26% in these 20 observations and an increase of about 16% in the remaining 5 observations. It is important to note that as a proof-of-concept, we only employed a Gaussian B-matrix which might not be an optimal choice for such heterogeneous fields as T2M. Despite this strong constraint, the AI-Var is still able to enhance the T2M representation in the cross-validation experiment.

We find further evidence of this in the right hand side plot in Figure 13 which shows the results from the cross-validation experiments for the diurnal cycle of the deviations from the observations. Here, the differences are calculated from the independent observations from the five trained models, respectively. While for some observations the errors even increase in the analysis, on average we still find a reduction of the deviations through data assimilation and an attenuation of the nightly warm bias. It is important to note that as a proof-of-concept, we only employed a Gaussian B-matrix which might not be an optimal choice for such heterogeneous fields as T2M. Despite this strong constraint, the AI-Var is still able to enhance the T2M representation in the cross-validation experiment.

5 Conclusions

In this paper, we have presented a novel AI-based data assimilation approach that has the potential to replace the classical data assimilation scheme by training a neural network to perform the data assimilation task itself. This proof-of-concept study has shown promising results across both idealized and real-world scenarios.

Our investigations in one-dimensional (1D) and two-dimensional (2D) idealized cases indicate that the AI-Var approach is - even with a zero first guess - capable of effectively reconstructing the original state from observations. In particular, we have noticed that while the reconstruction can be nearly perfect for static observation setups, the performance decreases only modestly for the more challenging dynamic observation locations case. This indicates that combining a sufficient number of observations with a large training data set can reasonably compensate for the added complexity introduced by dynamic observation locations.

Our real-world test case, which involved the assimilation of 2-meter temperature (T2M) observations from synoptic observation sites, has demonstrated that our AI-Var system can indeed improve estimation accuracy. Specifically, it can correct for biases in the first guess fields, such as the nocturnal warm bias in the COSMO-REA6 reanalysis, and reduce errors significantly. These improvements were to a lesser extent also evident in the cross-validation experiment with independent observations, though the Gaussian B-matrix employed may not be the optimal choice for heterogeneous fields like T2M.

As with any novel approach, there still remains a large potential for improvement with further research and development necessary before AI-based data assimilation systems can be fully operational. Future work may focus on: First, *enhanced neural architectures* are needed to handle the increased complexity of high-dimensional data assimilation problems. Second, *advanced training strategies* including iterative training approaches and meta-learning will be needed to further reduce computational costs and improve the quality of solutions. Third, *realistic 3D applications* will be an important next step extending the approach to three-dimensional as well as multi-variate data sets to better emulate real-world meteorological scenarios and increase the realism of the results. Fourth, *optimized covariance models* have been important to improve quality in traditional algorithms. Investigating alternative B-matrix representations

beyond the Gaussian kernel such as climatological and ensemble-based approaches to better capture the variability and heterogeneity of different meteorological variables will be an important development step. We note that from a conceptional viewpoint the use of an ensemble based B-Matrix in the loss functional (2) as in the EnVar [e.g., Buehner et al., 2013] is straightforward. This needs, however, the generation of an appropriate ensemble and, thus, if one aims to have a fully data-driven analysis, further progress in AI based ensemble generation. Fifth, it is important to explore the *operational feasibility* of AI-based data assimilation, integrating this approach into existing NWP systems, possibly through hybrid data assimilation strategies that blend traditional and AI-based methods.

In conclusion, our AI-based data assimilation method shows a clear path forward for leveraging machine learning to handle traditionally complex and computationally expensive tasks. By fully integrating neural networks into the assimilation process, we open up opportunities for faster, more efficient, and eventually more accurate weather prediction systems—paving the way for fully data-driven NWP systems in the future.

Disclaimer

This Work has not yet been peer-reviewed and is provided by the contributing Author(s) as a means to ensure timely dissemination of scholarly and technical Work on a noncommercial basis. Copyright and all rights therein are maintained by the Author(s) or by other copyright owners. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each Author’s copyright. This Work may not be reposted without explicit permission of the copyright owner.

A AI-Var Pseudo Code

1. Set Input Data:

```

1 z := nt x ng array containing the grid input data on the grid
2 o := nt x no array containing the observations input data
3 i := nt x no array containing the locations of the observations
  on the grid

```

2. Define Input Fields and Vectors for Training:

```

1 sort_indices = sort i in ascending order
2 input_field  = scale and convert z to tensor
3 input_vector = stack sorted o and normalized i into tensor
4 b_cov       = calculate covariance matrix with Gaussian kernel
  width sigma
5 regularize b_cov

```

3. Initialize Training Data and DataLoader:

```

1 train_set = create training dataset with input_field and
  input_vector
2 train_loader = create DataLoader for training with batch_size and
  shuffling based on train_set

```

4. Define Data Assimilation Neural Network and Loss Function:

```

1 class DataAssimilationNN:
2     initialize with dimensions ng, n_obs, obs_error
3     define three fully connected layers
4     forward pass:
5         concatenate input first guess, observations and their
           locations
6         apply relu activation after each fully connected layer
7         reshape and return the output
8
9 class DataAssimilationLoss:
10    initialize with ng, b_matrix, obs_err
11    forward pass:
12        calculate background mismatch using Mahalanobis distance

```

```

13     sample observations from model output
14     calculate observation mismatch
15     combine losses and return

```

5. Prepare Training:

```

1 model      = initialize DataAssimilationNN with ng, n_obs
2 criterion  = initialize DataAssimilationLoss with
              regularized_b_cov, ng, obs_err
3 optimizer  = initialize Adam optimizer with learning rate 0.001

```

6. Training Loop:

```

1 for epoch in range(num_epochs):
2     for batch in train_loader:
3         clear optimizer gradients
4         output = model(batch)
5         loss   = criterion(output, batch)
6         backpropagate loss
7         update model parameters with optimizer
8         store training loss

```

References

- B. Amos and J. Z. Kolter. Optnet: Differentiable optimization as a layer in neural networks, 2021. URL <https://arxiv.org/abs/1703.00443>.
- B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Dover Books on Electrical Engineering Series. Dover Publications, Incorporated, 2012. ISBN 9780486136899. URL <http://books.google.de/books?id=iYMqLQp49UMC>.
- J. L. Anderson. An ensemble adjustment Kalman filter for data assimilation. *Monthly Weather Review*, 129(12): 2884–2903, 2001. doi:[10.1175/1520-0493\(2001\)129<2884:AEAKFF>2.0.CO;2](https://doi.org/10.1175/1520-0493(2001)129<2884:AEAKFF>2.0.CO;2).
- R. Arcucci, J. Zhu, S. Hu, and Y.-K. Guo. Deep data assimilation: Integrating deep learning with data assimilation. *Applied Sciences*, 11(3), 2021. ISSN 2076-3417. doi:[10.3390/app11031114](https://doi.org/10.3390/app11031114). URL <https://www.mdpi.com/2076-3417/11/3/1114>.
- P. Bauer, A. Thorpe, and G. Brunet. The quiet revolution of numerical weather prediction. *Nature*, 525:47–55, 2015. doi:[10.1038/nature14956](https://doi.org/10.1038/nature14956).
- F. Baur, L. Scheck, C. Stumpf, C. Köpken-Watts, and R. Potthast. A neural-network-based method for generating synthetic 1.6 μ m near-infrared satellite images. *Atmospheric Measurement Techniques*, 16(21):5305–5326, 2023. doi:[10.5194/amt-16-5305-2023](https://doi.org/10.5194/amt-16-5305-2023). URL <https://amt.copernicus.org/articles/16/5305/2023/>.
- K. Bi, L. Xie, H. Zhang, X. Chen, X. Gu, and Q. Tian. Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 619(7970):533–538, July 2023. ISSN 1476-4687. doi:[10.1038/s41586-023-06185-3](https://doi.org/10.1038/s41586-023-06185-3). URL <http://dx.doi.org/10.1038/s41586-023-06185-3>.
- C. H. Bishop. The GIGG-EnKF: Ensemble Kalman filtering for highly skewed non-negative uncertainty distributions. *Quart. J. Roy. Meteor. Soc.*, 142:1395–1412, 2016. doi:[10.1002/qj.2742](https://doi.org/10.1002/qj.2742).
- C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 2006. URL <https://link.springer.com/book/10.1007/978-0-387-45528-0>.
- C. Bollmeyer, J. D. Keller, C. Ohlwein, S. Wahl, S. Crewell, P. Friederichs, A. Hense, J. Keune, S. Kneifel, I. Pscheidt, S. Redl, and S. Steinke. Towards a high-resolution regional reanalysis for the European CORDEX domain. *Quarterly Journal of the Royal Meteorological Society*, 141(686):1–15, 2015. doi:[10.1002/qj.2486](https://doi.org/10.1002/qj.2486).
- M. Bonavita, R. Arcucci, A. Carrassi, P. Dueben, A. J. Geer, B. L. Saux, N. Longépé, P.-P. Mathieu, and L. Raynaud. Machine learning for earth system observation and prediction. *Bulletin of the American Meteorological Society*, 102(4):E710 – E716, 2021. doi:[10.1175/BAMS-D-20-0307.1](https://doi.org/10.1175/BAMS-D-20-0307.1). URL <https://journals.ametsoc.org/view/journals/bams/102/4/BAMS-D-20-0307.1.xml>.
- Z. B. Bouallegue, M. Alexe, M. Chantry, M. Clare, J. Dramsch, S. Lang, C. Lessig, L. Magnusson, A. P. Nemesio, F. Pinault, B. Raoult, and S. Tietsche). Artificial intelligence for forecasting system (aifs), 2023. URL <https://www.ecmwf.int/en/about/media-centre/aifs-blog>. Accessed: 2024-05-20.

- J. Briden, C. Choi, K. Yun, R. Linares, and A. Cauligi. Constraint-informed learning for warm starting trajectory optimization, 2023. URL <https://api.semanticscholar.org/CorpusID:266520898>.
- M. Buehner, J. Morneau, and C. Charette. Four-dimensional ensemble-variational data assimilation for global deterministic weather prediction. *Nonlinear Processes in Geophysics*, 20(5):669–682, 2013. doi:10.5194/npg-20-669-2013. URL <https://npg.copernicus.org/articles/20/669/2013/>.
- C. Buizza, C. Quilodrán Casas, P. Nadler, J. Mack, S. Marrone, Z. Titus, C. Le Cornec, E. Heylen, T. Dur, L. Baca Ruiz, C. Heaney, J. A. Díaz Lopez, K. S. Kumar, and R. Arcucci. Data learning: Integrating data assimilation and machine learning. *Journal of Computational Science*, 58:101525, Feb. 2022. ISSN 1877-7503. doi:10.1016/j.jocs.2021.101525. URL <http://dx.doi.org/10.1016/j.jocs.2021.101525>.
- H. G. Chipilski. Exact nonlinear state estimation, 2023. URL <https://arxiv.org/abs/2310.10976>.
- M. Deng, J. Wang, C.-P. Hsieh, Y. Wang, H. Guo, T. Shu, M. Song, E. Xing, and Z. Hu. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In Y. Goldberg, Z. Kozareva, and Y. Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391, Abu Dhabi, United Arab Emirates, Dec. 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.emnlp-main.222. URL <https://aclanthology.org/2022.emnlp-main.222>.
- R. Dong, H. Leng, C. Zhao, J. Song, J. Zhao, and X. Cao. A hybrid data assimilation system based on machine learning. *Frontiers in Earth Science*, 10:Article 1012165, 2023. doi:10.3389/feart.2022.1012165. URL <https://doi.org/10.3389/feart.2022.1012165>. Published on 05 January 2023, Section Atmospheric Science.
- S. Effati and M. Ranjbar. A novel recurrent nonlinear neural network for solving quadratic programming problems. *Applied Mathematical Modelling*, 35(4):1688–1695, 2011. doi:10.1016/j.apm.2010.10.001.
- G. Evensen. *Data Assimilation: The Ensemble Kalman Filter*. Springer, Berlin, Heidelberg, 2 edition, 2009. doi:10.1007/978-3-642-03711-5.
- Z. Fan, B. Ghaddar, X. Wang, L. Xing, Y. Zhang, and Z. Zhou. Artificial intelligence for operations research: Revolutionizing the operations research process, 2024.
- A. Farchi, P. Laloyaux, M. Bonavita, and M. Bocquet. Using machine learning to correct model error in data assimilation and forecast applications. *Quarterly Journal of the Royal Meteorological Society*, 147(739):3067–3084, July 2021. ISSN 1477-870X. doi:10.1002/qj.4116. URL <http://dx.doi.org/10.1002/qj.4116>.
- A. J. Geer. Learning earth system models from observations: machine learning or data assimilation? *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), Feb. 2021. ISSN 1471-2962. doi:10.1098/rsta.2020.0089. URL <http://dx.doi.org/10.1098/rsta.2020.0089>.
- I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, Cambridge, MA, 2016. URL <http://www.deeplearningbook.org>.
- G. A. Gottwald and S. Reich. Combining machine learning and data assimilation to forecast dynamical systems from noisy partial observations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(10), Oct. 2021. ISSN 1089-7682. doi:10.1063/5.0066080. URL <http://dx.doi.org/10.1063/5.0066080>.
- T. Hastie, J. H. Friedman, and R. Tibshirani. *The Elements of Statistical Learning*. Springer, New York, 2001. doi:10.1007/978-0-387-21606-5.
- P. L. Houtekamer and H. L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 126(3):796–811, 1998. doi:10.1175/1520-0493(1998)126<0796:DAUAEK>2.0.CO;2.
- P. L. Houtekamer and H. L. Mitchell. A sequential ensemble Kalman filter for atmospheric data assimilation. *Monthly Weather Review*, 129(1):123–137, 2001. doi:10.1175/1520-0493(2001)129<0123:ASEKFF>2.0.CO;2.
- P. L. Houtekamer and H. L. Mitchell. Ensemble Kalman filtering. *Quarterly Journal of the Royal Meteorological Society*, 131(613):3269–3289, 2005. doi:10.1256/qj.05.135.
- B. R. Hunt, E. J. Kostelich, and I. Szunyogh. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform Kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2):112–126, 2007. doi:10.1016/j.physd.2006.11.008.
- G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning*. Springer, New York, 2013. doi:10.1007/978-1-4614-7138-7.
- E. Kalnay. *Atmospheric Modeling, Data Assimilation and Predictability*. Cambridge University Press, Cambridge, 2003. doi:10.1017/CBO9780511802270.
- J. D. Keller and S. Wahl. Representation of climate in reanalyses: An intercomparison for Europe and North America. *Journal of Climate*, 34(5):1667–1689, 2021. doi:10.1175/JCLI-D-20-0609.1.

- R. Lam, A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed, and P. Battaglia. Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421, 2023. doi:[10.1126/science.adi2336](https://doi.org/10.1126/science.adi2336). URL <https://www.science.org/doi/abs/10.1126/science.adi2336>.
- K. Li and J. Malik. Learning to optimize, 2016. URL <https://api.semanticscholar.org/CorpusID:13395552>.
- X. Liu, Y. Lu, A. Abbasi, M. Li, J. Mohammadi, and S. Kolouri. Teaching networks to solve optimization problems, 2022. URL <https://arxiv.org/abs/2202.04104>.
- A. C. Lorenc. The potential of the ensemble Kalman filter for NWP: A comparison with 4D-Var. *Quarterly Journal of the Royal Meteorological Society*, 129(595):3183–3203, 2003. doi:[10.1256/qj.02.132](https://doi.org/10.1256/qj.02.132). URL <https://doi.org/10.1256/qj.02.132>.
- A. C. Lorenc, S. P. Ballard, R. S. Bell, N. B. Ingleby, P. L. F. Andrews, D. M. Barker, J. R. Bray, A. M. Clayton, T. Dalby, D. Li, T. J. Payne, and F. W. Saunders. The Met. Office global three-dimensional variational data assimilation scheme. *Quarterly Journal of the Royal Meteorological Society*, 126(570):2991–3012, 2000. doi:[10.1002/qj.49712657002](https://doi.org/10.1002/qj.49712657002). URL <https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.49712657002>.
- P. C. Mahalanobis. Reprint of: On the generalised distance in statistics. *Sankhya A*, 80(Suppl 1):1–7, 2019. doi:[10.1007/s13171-019-00164-5](https://doi.org/10.1007/s13171-019-00164-5). Originally published in 1936.
- G. Nakamura and R. Potthast. *Inverse Modeling: An Introduction to the Theory and Methods of Inverse Problems and Data Assimilation*. G - Reference, Information and Interdisciplinary Subjects Series. IOP Publishing, 2015. ISBN 9780750312196. URL https://books.google.de/books?id=toR_jgEACAAJ.
- J. Pathak, S. Subramanian, P. Harrington, S. Raja, A. Chattopadhyay, M. Mardani, T. Kurth, D. Hall, Z. Li, K. Aziz-zadenesheli, P. Hassanzadeh, K. Kashinath, and A. Anandkumar. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators, 2022. URL <https://arxiv.org/abs/2202.11214>.
- R. Potthast, A. Walter, and A. Rhodin. A localized adaptive particle filter within an operational nwp framework. *Monthly Weather Review*, 147(1):345 – 362, 2019. doi:[10.1175/MWR-D-18-0028.1](https://doi.org/10.1175/MWR-D-18-0028.1).
- PyTorch. Pytorch: An imperative style, high-performance deep learning library. <https://pytorch.org/>, 2024. Accessed: 2024-05-31.
- Y. Qu, J. Nathaniel, S. Li, and P. Gentine. Deep generative data assimilation in multimodal setting, 2024. URL <https://arxiv.org/abs/2404.06665>.
- S. Reich and C. J. Cotter. *Probabilistic Forecasting and Bayesian Data Assimilation*. Cambridge University Press, Cambridge, 2015. doi:[10.1017/CBO9781107706804](https://doi.org/10.1017/CBO9781107706804).
- M. Reinhardt, S. Y. Schoger, F. Kurzrock, and R. Potthast. Convective-scale assimilation of cloud cover from photographs using a machine learning forward operator. *Artificial Intelligence for the Earth Systems*, 2(2):e220025, 2023. doi:[10.1175/AIES-D-22-0025.1](https://doi.org/10.1175/AIES-D-22-0025.1).
- L. Scheck. A neural network based forward operator for visible satellite images and its adjoint. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 274:107841, 2021. ISSN 0022-4073. doi:<https://doi.org/10.1016/j.jqsrt.2021.107841>.
- B. Sukhija, N. Köhler, M. Zamora, S. Zimmermann, S. Curi, A. Krause, and S. Coros. Gradient-based trajectory optimization with learned dynamics, 2023. URL <https://arxiv.org/abs/2204.04558>.
- P. J. van Leeuwen, Y. Cheng, and S. Reich. *Nonlinear Data Assimilation*. Springer, Berlin, Heidelberg, 2015. doi:[10.1007/978-3-319-18347-3](https://doi.org/10.1007/978-3-319-18347-3).
- P. J. van Leeuwen, H. R. Künsch, L. Nerger, R. Potthast, and S. Reich. Particle filters for high-dimensional geoscience applications: A review. *Quarterly Journal of the Royal Meteorological Society*, 145(723):2335–2365, 2019. doi:[10.1002/qj.3551](https://doi.org/10.1002/qj.3551). URL <https://doi.org/10.1002/qj.3551>.
- S. Vetra-Carvalho, P. J. van Leeuwen, L. Nerger, A. Barth, M. U. Altaf, P. Brasseur, P. Kirchgessner, and J.-M. Beckers. State-of-the-art stochastic data assimilation methods for high-dimensional non-Gaussian problems. *Tellus A: Dynamic Meteorology and Oceanography*, 70(1):1–43, 2018. doi:[10.1080/16000870.2018.1445364](https://doi.org/10.1080/16000870.2018.1445364).
- O. Wichrowska, N. Maheswaranathan, M. W. Hoffman, S. G. Colmenarejo, M. Denil, N. de Freitas, and J. Sohl-Dickstein. Learned optimizers that scale and generalize, 2017. URL <https://arxiv.org/abs/1703.04813>.
- G. Zängl. Adaptive tuning of uncertain parameters in a numerical weather prediction model based upon data assimilation. *Quarterly Journal of the Royal Meteorological Society*, 149:2861–2880, 2023. doi:[10.1002/qj.4535](https://doi.org/10.1002/qj.4535).