Estimating Bias on Model Outputs Using Convolutional Neural Networks (CNNs)

Georgios Britzolakis

A scholarly paper in partial fulfillment of the requirements for the degree of

Master of Science

May 2023

Department of Atmospheric and Oceanic Science, University of Maryland
College Park, Maryland

Advisor: Dr. Kayo Ide

Table of Contents

Abstract

Bias and uncertainty are key contributors of prediction errors in numerical weather and climate prediction. Biases can arise from various sources, including noise in the data, approximations in the observation operators used to simulate the data, or limitations of the assimilating algorithm. Although ensemble prediction systems can be computationally expensive as they require running multiple simulations in parallel, statistical post-processing techniques are often used to estimate bias and often used to inexpensively improve the prediction quality of the ensemble system. Many methodologies has been proposed in the literature to estimate bias and most of them are based on a mathematical formula for that quantification either processing the analysis increments, using the background residuals or by injecting the bias term in the cost function (Bias-Aware) techniques. On the other hand, the evolution of Machine Learning and more specifically Convolutional Neural Networks (CNNs) provide an intelligent way of detecting artifacts on images, classifying images, recognize patterns, perform texture segmentation etc. Certain work has been proposed using CNNs to detect bias/uncertainty but there is more that need to be done in the CNN architecture implementation to improve bias estimation in post processing. The approach of this work is focused on reviewing the literature on current methods for estimating bias in on Data Assimilation systems using conventional methods and Convolutional Neural Networks.

Acknowledgements

List of Tables

List of Figures

List of Symbols

x: Model state

y: Observation

B: Background Covariance Matrix

R: Observation Covariance Matrix

$\boldsymbol{h}(\cdot)$: Is the function that relates the state with the observations

***J*(x)**: The cost function

$\mathbf{x}^{\alpha}$: Analysis (solution that minimizes the cost function)

K: Kalman Gain

$<\cdot>$: Linear Average

$\tilde{\mathbf{w}}_k^f$: Unbiased forecast at time t_k (i.e $\mathbf{w}_k^f - \hat{\mathbf{b}}_k^-$ model forecast minus bias)

$\hat{\mathbf{b}}_k$: Updated estimate of forecast bias

$\mathbf{L}_k$: Bias-gain matrix

Superscript $^|$: Set of selected unbiased observations

$\mathbf{w}_k^{o|}$: Vector of **unbiased** observations at time t_k

$\mathbf{H}_k^|$: Linearized observation operator for the selected **unbiased** observations

$\mathbf{P}_k^f$: Forecast error covariance matrix (i.e $< (\tilde{\mathbf{w}}_k^f - \mathbf{w}_k^t)(\tilde{\mathbf{w}}_k^f - \mathbf{w}_k^t)^T >$)

$\mathbf{P}_k^{b-}$: Covariance matrix of forecast bias (i.e $< (\hat{\mathbf{b}}_k^- - \mathbf{b}_k^f)(\hat{\mathbf{b}}_k^- - \mathbf{b}_k^f)^T >$)

$\mathbf{R}_k^|$: Observational covariance matrix for the selected unbiased observations at time t_k

$\mathbf{K}_k$: Kalman Gain matrix at time t_k

$\mathbf{A}_k$: Known matrix that relates the previous analysis with the current forecast

$\hat{\mathbf{b}}_k^-$: Sequential forecast bias

$\hat{\mathbf{b}}_{k-1}$: Previous Forecast Bias Estimate

$\mathbf{g}$: Some non-linear operator

$\mathbf{w}_k^f$: Previous Model Analysis

$\mathbf{w}_k^f$: Model Forecast valid for time t_k

Chapter 1 Introduction

Ensemble prediction systems are a common method for quantifying uncertainty in weather forecasts, especially for predicting extreme weather events. These systems consist of running multiple simulations, or trajectories, of a numerical weather model with perturbed initial conditions and/or model parameters. The resulting ensemble of simulations provides a range of possible outcomes, allowing forecasters to estimate the likelihood and range of possible weather conditions.

However, ensemble prediction systems can be computationally expensive, as they require running multiple simulations in parallel. To improve the efficiency of these systems, statistical post-processing techniques are often used to inexpensively improve the raw prediction quality of the ensemble. These techniques can include methods such as bias correction, calibration, and downscaling.

Bias correction involves adjusting the raw ensemble predictions to correct for any systematic biases in the model. This can be done by comparing the ensemble predictions to observations and making appropriate adjustments to the ensemble members.

Calibration involves transforming the raw ensemble predictions to improve their statistical properties, such as their mean and variance. This can be done using various statistical methods, such as linear regression or Bayesian methods.

Downscaling involves using statistical techniques to extract more detailed information from the raw ensemble predictions, such as local weather conditions or extreme events. This can be done using methods such as spatial interpolation or statistical modeling of the relationships between different variables.

By using ensemble prediction systems and statistical post-processing techniques, forecasters can better quantify the uncertainty associated with weather forecasts and provide more accurate and reliable predictions, especially for extreme weather events.

Many bias estimation methods have been proposed in the literature and most of them were thoroughly summarized in (Dee, 2005). All of these proposed methods use a mathematical formula in quantifying Bias. On the other hand, recent advances in Artificial Intelligence (AI) and Machine Learning (ML) and more specifically the use of Convolutional Neural Networks (CNNs) have been used to a variety of tasks including scene classification, object detection, segmentation and last but not least Image Processing. In more detail a U-Net Convolutional Neural Network (CNN) Architecture was proposed in (Ronneberger, Fischer and Brox, 2015) which was effectively used to perform image segmentation on Biomedical images. This work was then evolved in post-processing ensemble weather forecasts for temperature at 850hPa (T850) and geopotential at 500hPa (Z500) using ECMWF data (Grönquist *et al.*, 2021).

Chapter 2. Conventional Bias Estimation Techniques

Traditional or conventional bias estimation techniques typically use a mathematical formula to estimate bias in the data. These techniques can be broadly categorized into two types: bias-blind and bias-aware methods.

Bias-blind methods do not explicitly account for bias in the data and assume that the data are unbiased. These methods typically involve simple statistical techniques, such as mean or median value estimation, or linear regression analysis.

In contrast, bias-aware methods explicitly account for bias in the data and attempt to correct for it. These methods can involve more complex statistical techniques, such as data assimilation or Bayesian inference, and may require a priori knowledge of the sources of bias in the data.

The choice of bias estimation technique depends on the application and the available data. In some cases, bias-blind methods may be sufficient if the data are known to be relatively unbiased. However, in many cases, bias-aware methods may be necessary to accurately estimate bias and improve the quality of the data as described in (Dee, 2005).

2.1 Bias – Blind Data Assimilation

In data assimilation we are mostlyc interested in minimizing the following cost function J(**x**):

$$J(\mathbf{x}) = (\mathbf{x}^b - \mathbf{x})^T \mathbf{B}^{-1} (\mathbf{x}^b - \mathbf{x}) + [\mathbf{y} - \mathbf{h}(\mathbf{x})]^T \mathbf{R}^{-1} [\mathbf{y} - \mathbf{h}(\mathbf{x})] \qquad (1)$$

Where:

x: model state

y: observations

$\boldsymbol{h}(\cdot)$: Is the function that relates the state with the observations

B: Is the Background error Matrix

R: Is the observation error Matrix

As we can see from the above there is not account for bias and so we can describe this data assimilation as Bias-Blind data assimilation

In order to minimize the cost function above, we can take the gradient of J(x) and set it to zero. The solution $\mathbf{x}^{\alpha}$ also called the analysis and based on that analysis we define the analysis increment as $\mathbf{dx} = \mathbf{x}^{\alpha} - \mathbf{x}^{\mathbf{b}}$ and also $\mathbf{dy} = \mathbf{y}^{\alpha} - \mathbf{h}(\mathbf{x}^{\mathbf{b}})$ as the vector of the observed-minus the background residuals. Also by minimizing the cost function we can obtain the Kalman Gain and the relate **dx** and **dy** with the Kalman Gain with the following equations:

$$\boldsymbol{K} = \boldsymbol{B}\boldsymbol{H}^{\boldsymbol{T}}(\boldsymbol{H}\boldsymbol{B}\boldsymbol{H}^{\boldsymbol{T}} + \boldsymbol{R})^{-\mathbf{1}} \qquad (2)$$

$$\boldsymbol{dx} = \boldsymbol{K}\boldsymbol{dy} \qquad (3)$$

$$\boldsymbol{H} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}}\Big|_{x=x^{b}} \qquad (4)$$

The errors from the above equations can be defined as:

$$\boldsymbol{e}^{a} = \boldsymbol{x}^{a} - \boldsymbol{x} \qquad (5)$$

$$\boldsymbol{e}^{b} = \boldsymbol{x}^{b} - \boldsymbol{x} \qquad (6)$$

$$\boldsymbol{e}^{o} = \boldsymbol{y} - h(\boldsymbol{x}) \qquad (7)$$

By combining equation (3) above with equations (5) – (6) we have:

$$\boldsymbol{e}^{a} \approx \boldsymbol{K}\boldsymbol{e}^{o} + [\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}]\boldsymbol{e}^{b} \qquad (8)$$

And if we apply linear average (i.e $<\cdot>$) on (8) we have:

$$< \boldsymbol{e}^{a} > \approx < \boldsymbol{K}\boldsymbol{e}^{o} > + < [\boldsymbol{I} - \boldsymbol{K}\boldsymbol{H}]\boldsymbol{e}^{b} > \qquad (9)$$

2.1.1 Analysis increments Bias Estimation

Bias detection using analysis increments is a common technique used in data assimilation to diagnose the presence of biases in the input data. The technique involves comparing the difference between the observed data and the assimilated model output, known as the analysis increment, with the expected uncertainty in the observations.

In a perfect assimilation system, the analysis increment should be consistent with the expected observation uncertainty, and any deviations from this consistency can be attributed to biases in the data. For example, if the analysis increment consistently underestimates the expected observation uncertainty, this may indicate a bias in the input data that is not being fully accounted for in the assimilation process.

The analysis increment can be calculated using a variety of statistical techniques, such as the Kalman filter or variational assimilation, and can be applied to a wide range of input data, including atmospheric and oceanic observations, as well as other types of geophysical data.

Bias detection using analysis increments can be a powerful tool for improving the accuracy of data assimilation systems, as it can help to identify and correct for biases in the input data. However, the technique can also be sensitive to errors in the model

and the assimilation methodology, and must be used with caution to ensure that any detected biases are not simply artifacts of the assimilation process itself.

In quantitative terms the bias from the analysis increments is defined as:

$$< \boldsymbol{dx} > \approx < \boldsymbol{Ke}^{o} > - < \boldsymbol{KHe}^{b} > \quad (10)$$

In a bias-free situation equation (10) is close to zero.

2.1.2 Background Residuals Bias Estimation

Another common technique used in data assimilation to detect biases is bias detection using background residuals. This technique involves comparing the difference between the model background prediction and the observed data with the expected uncertainty in the observations.

The model background prediction is the model output generated from the previous time step or forecast, and is used as the initial condition for the assimilation of new observations. The difference between the model background prediction and the observed data is known as the background residual, and any deviation from the expected uncertainty in the observations can be attributed to biases in the input data.

Similar to bias detection using analysis increments, the bias detection using background residuals technique requires knowledge of the expected observation uncertainty, which can be estimated from the observation error statistics.

The bias detection using background residuals technique can be applied to a wide range of input data, including atmospheric and oceanic observations, and can be used in conjunction with a variety of assimilation methodologies, such as the Kalman filter or variational assimilation.

While bias detection using background residuals can be a powerful tool for improving the accuracy of data assimilation systems, it can also be sensitive to errors in the model and assimilation methodology, and must be used with caution to ensure that any detected biases are not simply artifacts of the assimilation process itself.

In quantitative terms the bias using background residuals is defined as follow:

$$< \boldsymbol{dy} > \approx < \boldsymbol{e}^{o} > - < \boldsymbol{He}^{b} > \qquad (11)$$

The term above in a Bias-Free situation is supposed to be zero.

2.2 Bias – Aware Data Assimilation

Bias aware data assimilation is an approach that explicitly accounts for biases in the input data during the assimilation process. This approach recognizes that biases in the input data can significantly impact the accuracy of the assimilated data and can lead to incorrect model predictions.

There are several methods for implementing bias aware data assimilation. One common approach is to use bias correction methods to adjust the input data prior to

assimilation. Bias correction involves applying a correction factor to the input data to account for known biases, and can be done either before or after assimilation.

Another approach is to include bias estimation and correction as part of the assimilation process itself. This involves estimating the bias in the input data during the assimilation process and adjusting the assimilated data accordingly. This approach can be computationally expensive but has the advantage of being able to account for time-varying biases.

In addition, there are methods for estimating and correcting biases in the model itself. These methods involve comparing the model predictions with independent observations to identify and correct biases in the model output.

Bias aware data assimilation has been shown to significantly improve the accuracy of model predictions, particularly for extreme events and in regions where biases are known to be large. However, implementing bias aware data assimilation requires accurate knowledge of the biases in the input data, which can be challenging to obtain, and may also require significant computational resources.

In order to convert the cost function above to a bias aware version, it is possible to represent the state vector with the following format:

$$\boldsymbol{z}^T = [\boldsymbol{x}^T \beta^T] \quad (12)$$

$$J(\boldsymbol{z}) = (\boldsymbol{z}^b - \boldsymbol{z})^T \boldsymbol{Z}^{-1} (\boldsymbol{z}^b - \boldsymbol{z}) + [\boldsymbol{y} - \widetilde{\boldsymbol{h}}(\boldsymbol{z})]^T \boldsymbol{R}^{-1} [\boldsymbol{y} - \widetilde{\boldsymbol{h}}(\boldsymbol{z})] \qquad (13)$$

In a such way the cost function becomes a bias aware function because of equation 12.

2.3 Bias Correction Methods with Data Assimilation

One of the direct ways of properly accounting for bias in a statistical analysis is by estimating the forecast bias and then correcting the forecast prior to analysis. Several algorithms have been proposed in the literature for estimating forecast bias by means of data assimilation based on an unbiased subset of the observing system (Dee and Da Silva, 1998). These algorithms can be categorized as off-line when the forecast biases are based on previously produced assimilated datasets or on-line which are based on approximating information about forecast and observation error covariances. These 3 algorithms proposed by (Dee and Da Silva, 1998) follow:

2.3.1 Off–line forecast-bias Estimation

In Off-line forecast bias estimation we have estimation of forecast bias on existing assimilated data sets. The bias estimation is performed in two stages on the first state we have the bias prediction while on the second stage we have the bias update which is used on the next iteration for the bias prediction:

$$\hat{\mathbf{b}}_k^- = \hat{\mathbf{b}}_{k-1} + \mathbf{g}(\hat{\mathbf{b}}_{k-1}, \mathbf{w}_{k-1}^a) \tag{14}$$

$$\tilde{\mathbf{w}}_k^f = \mathbf{w}_k^f - \hat{\mathbf{b}}_k^- \tag{15}$$

$$\hat{\mathbf{b}}_k = \hat{\mathbf{b}}_k^- - \mathbf{L}_k \left[\mathbf{w}_k^{o|} - \mathbf{H}_k^{|} \tilde{\mathbf{w}}_k^f \right] \tag{16}$$

$$\mathbf{L}_k = \mathbf{S}_k^{b^-} \mathbf{H}_k^{|^T} \left[\mathbf{H}_k^{|} \mathbf{S}_k^{b^-} \mathbf{H}_k^{|^T} + \mathbf{H}_k^{|} \mathbf{S}_k^f \mathbf{H}_k^{|^T} + \mathbf{R}_k^{|} \right]^{-1} \tag{17}$$

A concise description of the notation above is in paragraph 2.3.4

2.3.2 On–line forecast-bias Estimation and correction

In the case of On-line forecast bias estimation and correction we basically have the implementation of the Off-line forecast bias estimation in on-line format which is basically in parallel with the bias-blind state estimator which is used as input in this algorithm. The description of the algorithm is ass follow:

$$\mathbf{w}_k^f = \mathbf{A}_k \mathbf{w}_{k-1}^a \tag{18}$$

$$\hat{\mathbf{b}}_k^- = \hat{\mathbf{b}}_{k-1} + \mathbf{g}(\hat{\mathbf{b}}_{k-1}, \tilde{\mathbf{w}}_{k-1}^a) \tag{19}$$

$$\tilde{\mathbf{w}}_k^f = \mathbf{w}_k^f - \hat{\mathbf{b}}_k^- \tag{20}$$

$$\hat{\mathbf{b}}_k = \hat{\mathbf{b}}_k^- - \mathbf{L}_k \left[\mathbf{w}_k^{o|} - \mathbf{H}_k^| \tilde{\mathbf{w}}_k^f \right] \tag{21}$$

$$\mathbf{L}_k = \mathbf{S}_k^b \ {\mathbf{H}_k^|}^T \left[\mathbf{H}_k^| {\mathbf{S}_k^b}^- {\mathbf{H}_k^|}^T + \mathbf{H}_k^| \mathbf{S}_k^f {\mathbf{H}_k^|}^T + \mathbf{R}_k^| \right]^{-1} \tag{22}$$

$$\mathbf{w}_k^a = \mathbf{w}_k^f + \mathbf{K}_k \left[\mathbf{w}_k^o - \mathbf{H}_k \mathbf{w}_k^f \right] \tag{23}$$

$$\mathbf{K}_k = \mathbf{S}_k^f \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{S}_k^f {\mathbf{H}_k}^T + \mathbf{R}_k^| \right]^{-1} \tag{24}$$

$$\tilde{\mathbf{w}}_k^a = \mathbf{w}_k^a - \left[\mathbf{I} - \mathbf{K}_k \mathbf{H}_k \right] \hat{\mathbf{b}}_k \tag{25}$$

The same as in paragraph 2.3.1 the description of the notation is in paragraph 2.3.4.

2.3.3 On-line Forecast-bias estimation and correction with Feedback

In the case of On-line Forecast-bias estimation and correction with feedback we have a more efficient on-line algorithm which is utilizing the most recent forecast-bias estimates as soon as they become available to produce bias-corrected forecasts and analyses. A step by step

$$\mathbf{w}_k^f = \mathbf{A}_k \tilde{\mathbf{w}}_{k-1}^a \tag{26}$$

$$\hat{\mathbf{b}}_k^- = \hat{\mathbf{b}}_{k-1} + \mathbf{g}(\hat{\mathbf{b}}_{k-1}, \tilde{\mathbf{w}}_{k-1}^a) \tag{27}$$

$$\tilde{\mathbf{w}}_k^f = \mathbf{w}_k^f - \hat{\mathbf{b}}_k^- \tag{28}$$

$$\hat{\mathbf{b}}_k = \hat{\mathbf{b}}_k^- - \mathbf{L}_k \left[\mathbf{w}_k^{o|} - \mathbf{H}_k^| \tilde{\mathbf{w}}_k^f \right] \tag{29}$$

$$\mathbf{L}_k = \mathbf{S}_k^{b^-} \mathbf{H}_k^{|^T} \left[\mathbf{H}_k^| \mathbf{S}_k^{b^-} \mathbf{H}_k^{|^T} + \mathbf{H}_k^| \mathbf{S}_k^f \mathbf{H}_k^{|^T} + \mathbf{R}_k^| \right]^{-1} \tag{30}$$

$$\tilde{\tilde{\mathbf{w}}}_k^f = \mathbf{w}_k^f - \hat{\mathbf{b}}_k \tag{31}$$

$$\tilde{\mathbf{w}}_k^a = \tilde{\tilde{\mathbf{w}}}_k^f + \mathbf{K}_k \left[\mathbf{w}_k^o - \mathbf{H}_k \tilde{\tilde{\mathbf{w}}}_k^f \right] \tag{32}$$

$$\mathbf{K}_k = \mathbf{S}_k^f \mathbf{H}_k^T \left[\mathbf{H}_k \mathbf{S}_k^f \mathbf{H}_k^T + \mathbf{R}_k \right]^{-1} \tag{33}$$

2.3.4 Notation explanation

$\hat{\mathbf{b}}_k^-$: Sequential forecast bias

$\hat{\mathbf{b}}_{k-1}$: Previous Forecast Bias Estimate

$\mathbf{g}$: Some non-linear operator

$\mathbf{w}_k^f$: Previous Model Analysis

$\mathbf{w}_k^f$: Model Forecast valid for time t_k

$\tilde{\mathbf{w}}_k^f$: Unbiased forecast at time t_k (i.e $\mathbf{w}_k^f - \hat{\mathbf{b}}_k^-$ model forecast minus bias)

$\hat{\mathbf{b}}_k$: Updated estimate of forecast bias

$\mathbf{L}_k$: Bias-gain matrix

Superscript $^|$: Set of selected unbiased observations

$\mathbf{w}_k^{o|}$: Vector of **unbiased** observations at time t_k

$\mathbf{H}_k^|$: Linearized observation operator for the selected **unbiased** observations

$\mathbf{P}_k^f$: Forecast error covariance matrix (i.e $< (\tilde{\mathbf{w}}_k^f - \mathbf{w}_k^t)(\tilde{\mathbf{w}}_k^f - \mathbf{w}_k^t)^T >$)

$\mathbf{P}_k^{b-}$: Covariance matrix of forecast bias (i.e $< (\hat{\mathbf{b}}_k^- - \mathbf{b}_k^f)(\hat{\mathbf{b}}_k^- - \mathbf{b}_k^f)^T >$)

$\mathbf{R}_k^|$: Observational covariance matrix for the selected unbiased observations at time t_k

$\mathbf{K}_k$: Kalman Gain matrix at time t_k

$\mathbf{A}_k$: Known matrix that relates the previous analysis with the current forecast

2.4 Bias Correction in model Prediction

Bias correction methods aim to remove or reduce the biases in the observations or model predictions, allowing for more accurate and reliable estimates of the true state of the system. These methods can be applied to different types of data, such as satellite measurements, weather station observations, or oceanographic measurements.

There are several approaches to bias correction in the data assimilation community. Some commonly used methods include:

1. Additive Bias Correction: This method involves estimating and subtracting a constant bias term from the observations or model predictions. The bias term is typically determined using historical data or independent calibration data. This method has been adopted/adapted in a variety of studies (Vila *et al.*, 2009), (Kornelsen and Coulibaly, 2015), (Balmaseda *et al.*, 2007), (Su *et al.*, 2014)
2. Multiplicative Bias Correction: In this approach, a multiplicative bias factor is applied to the observations or model predictions to account for the systematic errors. The bias factor is usually derived from historical or calibration data. A detailed discussion of both Additive and Multiplicative Bias Correction techniques was done in (Su *et al.*, 2014)
3. Empirical Bias Correction: This method involves developing empirical relationships between observations and model predictions based on historical data. These relationships are then used to correct the biases in the current observations or model predictions. Several studies have been proposed in the literature empirical bias correction techniques. More specifically (Guldberg *et al.*, 2005) initially

proposed a two step empirical method using as a first step a nudged simulation followed by a climatological seasonal cycle of the correction term calculation. This method was later applied (Kharin and Scinocca, 2012) on GCM experiments.

4. Model Output Statistics (MOS): MOS is a statistical technique that combines observational data and model output to improve the accuracy of model predictions. It involves developing statistical relationships between the model output and observations, including bias correction terms. Post processing statistics in general have shown great potential in removing defects in Numerical Weather Prediction (NWP) by statistically processing their output (Wilks, Daniel S, 2011). A combination of Perfect Prog and MOS ((Klein, Lewis and Enger, 1959; Glahn and Lowry, 1972) was proposed by (Marzban, Sandgathe and Kalnay, 2006)
5. Ensemble-based Bias Correction: Ensemble methods, such as ensemble Kalman filtering, can be used to estimate and correct biases in the observations or model predictions. The ensemble members are perturbed to represent the bias, and the filtering process helps estimate and correct these biases.

The choice of bias correction method depends on various factors, including the nature of the biases, the availability of historical data, and the specific requirements of the application. It is important to carefully assess the impact of bias correction on the assimilation results and to validate the effectiveness of the chosen method through rigorous evaluation and verification techniques.

Chapter 3. Convolutional Neural Networks in Bias Estimation

Convolutional Neural Networks (CNNs) have recently been used for bias estimation in data assimilation systems. CNNs are a type of deep neural network that are particularly well-suited for image processing tasks, but have also been shown to be effective in other domains.

In the context of bias estimation, CNNs can be trained on input data that includes both the original data and known biases, and then used to estimate the biases in new data. The input data can be in the form of model output or observational data, and the CNN can learn to identify patterns that are associated with biases in the data.

One advantage of using CNNs for bias estimation is that they can capture complex, nonlinear relationships between the input data and biases. This is particularly important in cases where biases are not well-understood or have a complicated relationship with the input data.

In addition, CNNs can be trained on large datasets, which can improve the accuracy of bias estimates. This is important for data assimilation systems, which often have to process large volumes of data.

Overall, the use of CNNs for bias estimation shows promise for improving the accuracy of data assimilation systems, particularly in cases where biases are difficult to identify using conventional methods. However, further research is needed to fully understand the

strengths and limitations of this approach, and to develop methods for integrating CNN-based bias estimation into existing data assimilation frameworks.

3.1 Features of Convolutional Neural Networks

3.1.1 Local Receptive Fields

One of the main differences between Normal Neural Networks and Convolutional Neural Networks is that every region in the input is connected with the every region in the hidden layer (Figure 1). On the other hand on a Convolutional Neural Network only a small region in the input layer is connects to the neurons in the hidden layer (Figure 2).

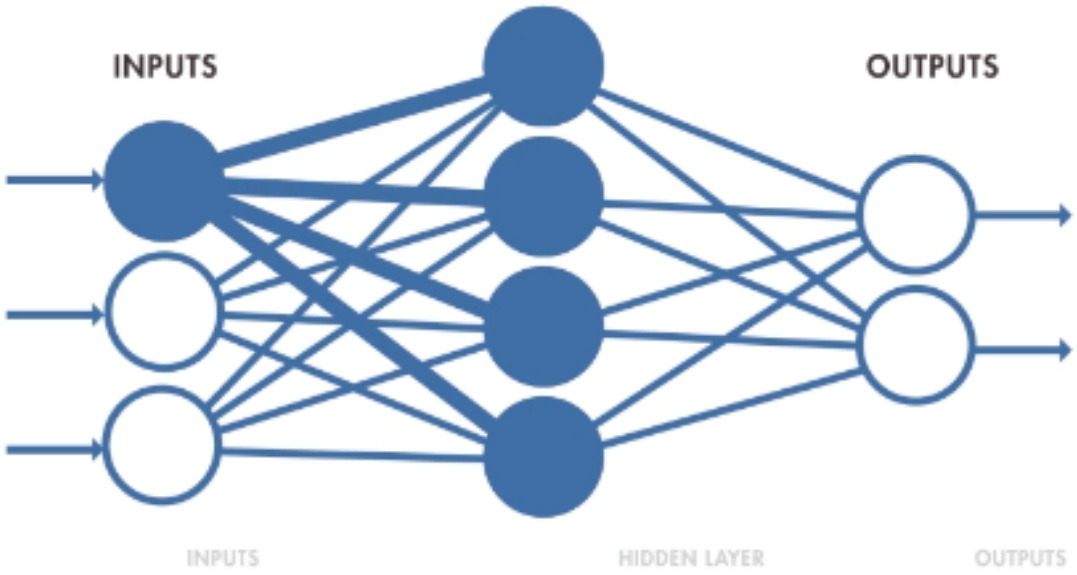

Figure 1 – Typical Neural Network Architecture – Every region in the input is connected to every region in the hidden layer (‘Matlab’, 2023).

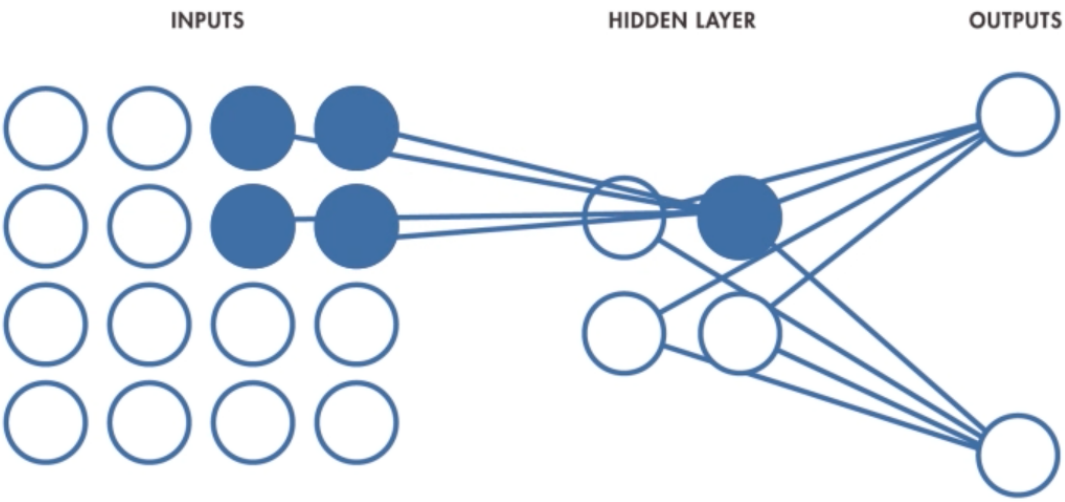

Figure 2 – Convolutional Neural Network – Only a small region in the input is connected to a small region in the hidden layer ('Matlab', 2023).

3.1.2 Shared Weights and Biases

In convolutional neural networks (CNNs), the same set of weights and biases are shared across all units of a given layer. This means that each unit in the layer applies the same set of weights to its input, and adds the same bias term before applying a non-linear activation function.

Sharing weights and biases in this way allows the CNN to learn spatially invariant features in the input data. For example, if a CNN is trained to recognize faces, the same set of weights and biases can be applied to different parts of an image to identify facial features regardless of their location.

In addition to spatial invariance, weight sharing also reduces the number of parameters in the network, making it more computationally efficient and easier to

train. This is particularly important in CNNs, which can have many layers and require large amounts of training data.

Overall, weight sharing is a key feature of CNNs that allows them to effectively process high-dimensional inputs such as images and video, and has been a major factor in the success of deep learning for a wide range of applications.

A comparison between conventional Neural Network and Convolutional Neural network in regard to the shared weights and biases feature can be seen in Figures 3, 4.

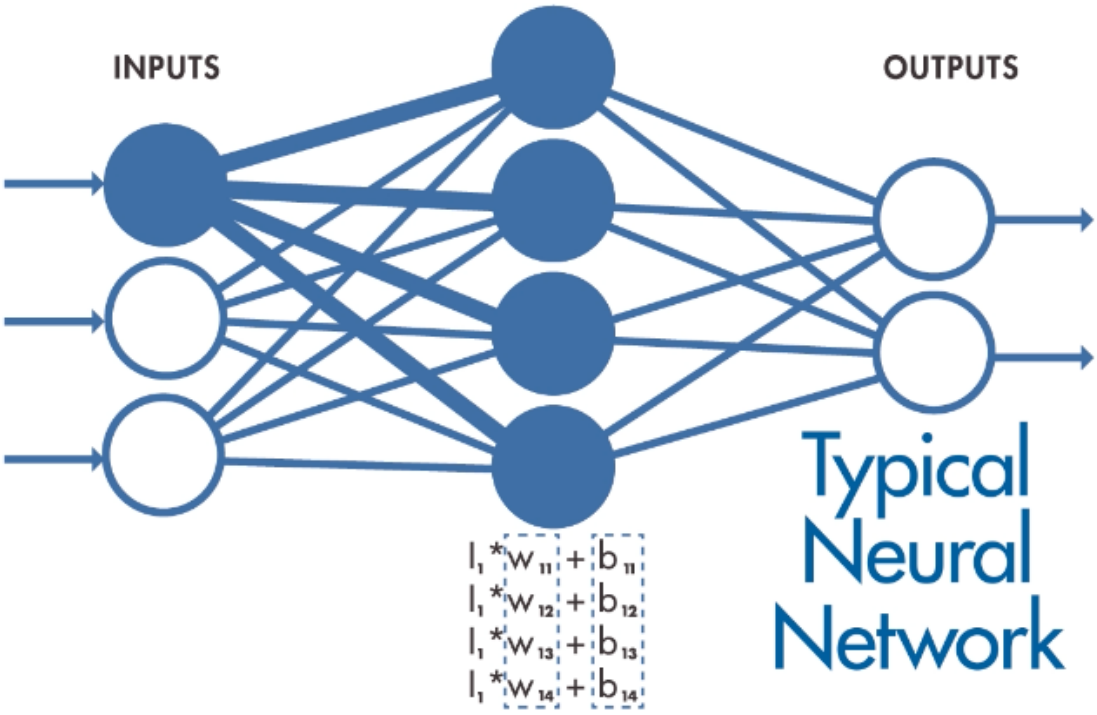

Figure 3 – Weights and Biases on a Typical Neural Network. Every connection of the input on a node in the hidden layer has different weights and biases ('Matlab', 2023).

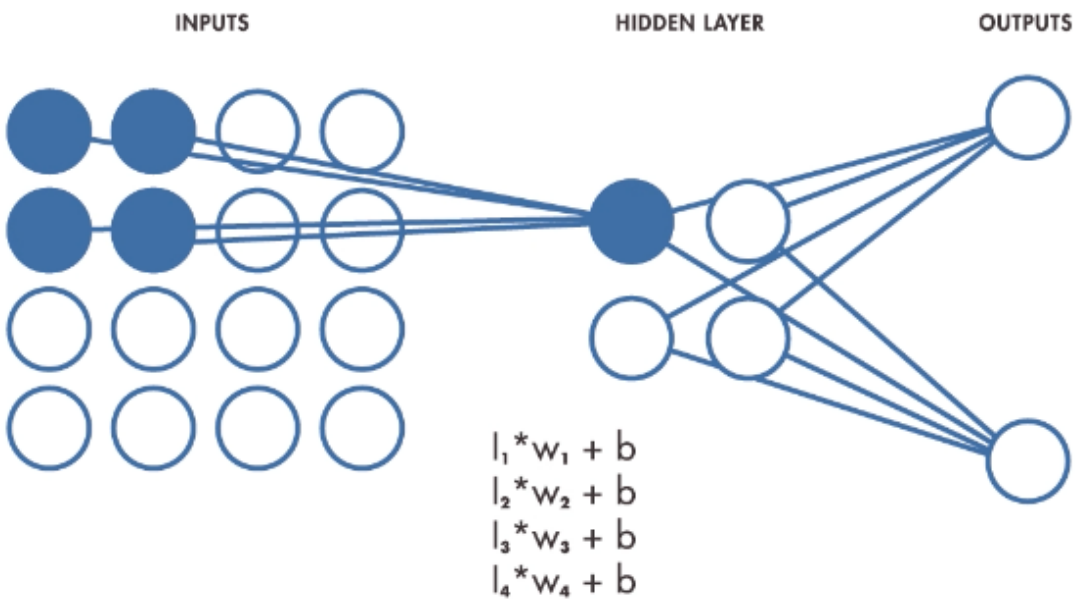

Figure 4 – Shared Weights and Biases on a Convolutional Neural Network. Weights and Biases values are the same for all hidden neurons in a given layer (‘Matlab’, 2023).

This feature is exclusively CNNs are very promising in estimating bias in data assimilation.

3.2 Operations in Convolutional Neural Networks

3.2.1 Convolution

One of the major task that CNNs perform is convolution. Convolution is a mathematical operation that involves taking the dot product of a small "filter" or "kernel" matrix with a larger input matrix or image, and sliding the filter over the input matrix in a specified pattern.

In a CNN, convolution is typically used to extract spatially-local features from an input image or other high-dimensional data. The filters applied in convolutional

layers of a CNN are learned during training, and can be thought of as feature detectors that identify patterns in the input data that are relevant to the task at hand.

The outputs of convolutional layers are typically passed through non-linear activation functions, and may be pooled or downsampled to reduce their spatial dimensionality and extract more abstract features. This process of convolution, activation, and pooling is repeated in multiple layers of the CNN, with the output of each layer serving as input to the next.

Overall, convolution is a fundamental operation in CNNs that allows them to effectively extract features from complex input data such as images and video, and has been a key factor in the success of deep learning for a wide range of applications. This operation is demonstrated in Figure 5.

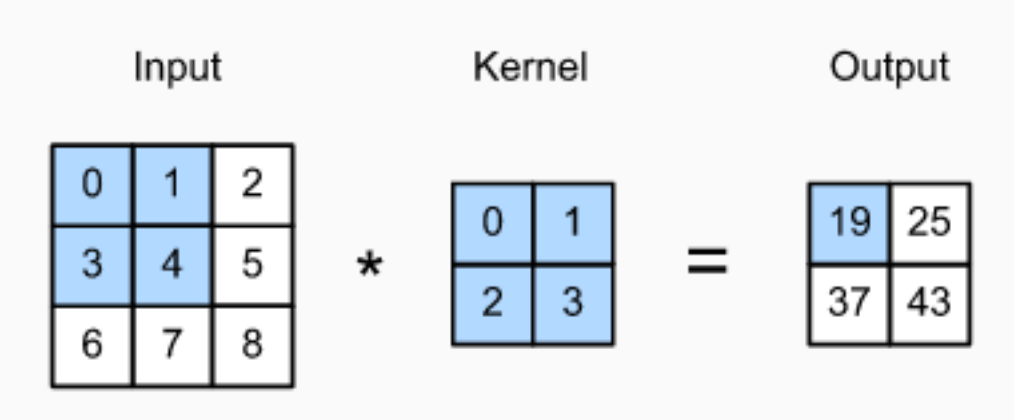

Figure 5 – Basic Demonstration of Convolution operation in Convolutional Neural Networks (‘Matlab’, 2023).

3.2.2 Activation and Pulling

Activation and pooling are two important operations that are commonly performed after convolution in convolutional neural networks (CNNs).

Activation functions are used to introduce non-linearity into the output of a convolutional layer. Without an activation function, the output of a convolutional layer would be a linear combination of the input pixels or features, which may not be sufficient for capturing complex patterns in the data. Common activation functions used in CNNs include ReLU (rectified linear unit), sigmoid, and hyperbolic tangent.

Pooling is a down-sampling operation that reduces the spatial dimensionality of the feature maps output by the convolutional layers. This is typically done by dividing the feature maps into non-overlapping regions, and computing a summary statistic such as the maximum or average value within each region. The result is a smaller set of feature maps that summarize the most important information in the original feature maps, while reducing the computational requirements for subsequent layers.

Pooling can also introduce a degree of translational invariance into the output of the CNN, which means that small shifts in the input image or data will not result in large changes to the output. This is achieved by summarizing the output of nearby pixels or features into a single value, which helps to reduce the sensitivity of the CNN to small variations in the input.

Overall, activation and pooling are important operations in CNNs that help to introduce non-linearity and reduce the spatial dimensionality of the feature maps, respectively. They play a key role in allowing CNNs to effectively extract features from complex input data and achieve state-of-the-art performance on a wide range of tasks.

A schematic demonstration of an activation and pulling operation is shown in Figure 6. Initially an activation function is applied to screen out the negative values and then pooling is applied to pull max values from predefined sections of the image.

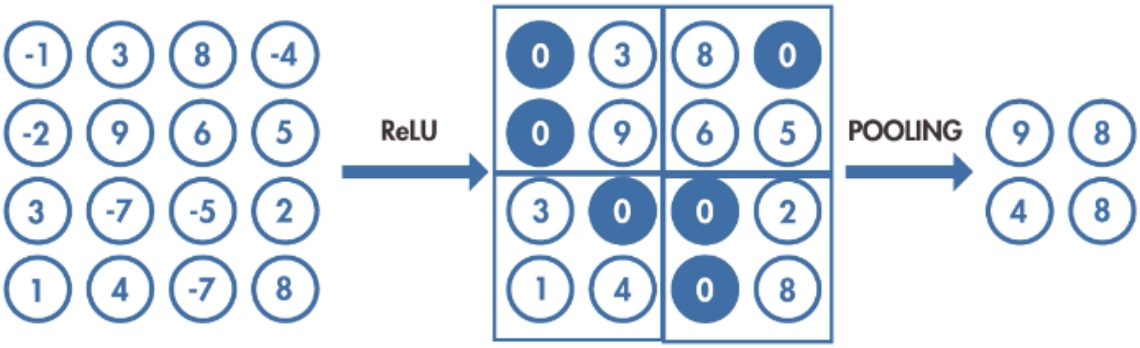

Figure 6 – Activation and Pulling operation in CNNs – First an activation function is applied to screen out negative values and then a pooling operation is applied to pull max values (‘Matlab’, 2023)

3.3 The U-Net architecture for Biomedical Imaging

The U-Net architecture is a popular CNN architecture for biomedical imaging applications, particularly for image segmentation tasks (Ronneberger, Fischer and Brox, 2015).

The U-Net architecture consists of an encoder and a decoder network, which are connected through skip connections. The encoder network consists of a series of

convolutional and pooling layers that progressively reduce the spatial resolution of the input image while increasing the number of feature maps. The decoder network is a mirror image of the encoder network, consisting of a series of upsampling and convolutional layers that gradually increase the spatial resolution of the feature maps while decreasing the number of feature maps.

The skip connections between the encoder and decoder networks are used to preserve the high-resolution features from the encoder network, which are important for accurate segmentation. Specifically, the feature maps from the encoder network are concatenated with the corresponding feature maps in the decoder network at the same spatial resolution. This allows the decoder network to access both the high-resolution features from the encoder network and the low-resolution features that it has generated itself.

The U-Net architecture has been shown to achieve state-of-the-art performance on a wide range of biomedical image segmentation tasks, including cell segmentation, nuclei segmentation, and brain tumor segmentation. Its success is attributed to the combination of the encoder and decoder networks with skip connections, which allows for the efficient extraction of both high- and low-level features from the input image.

The original U-Net implementation (Ronneberger, Fischer and Brox, 2015), shown in Figure 7, consists of 5 levels. On first level, 64 variations of a 3x3 filter is applied

producing 64 image channels (blue arrow) and a step function operation is applied. The original image from 572x572 pixels is reduced to 570x570 pixels. Then another 3x3 kernel is applied on the output producing another 64 channels of 568x568 pixel image. After that a 2x2 max pool operation is applied reducing the 568x568 to a half image 284x284 image and then the same convolution scheme was applied until the image is reduced to a 28x28 image before starting the upscaling process.

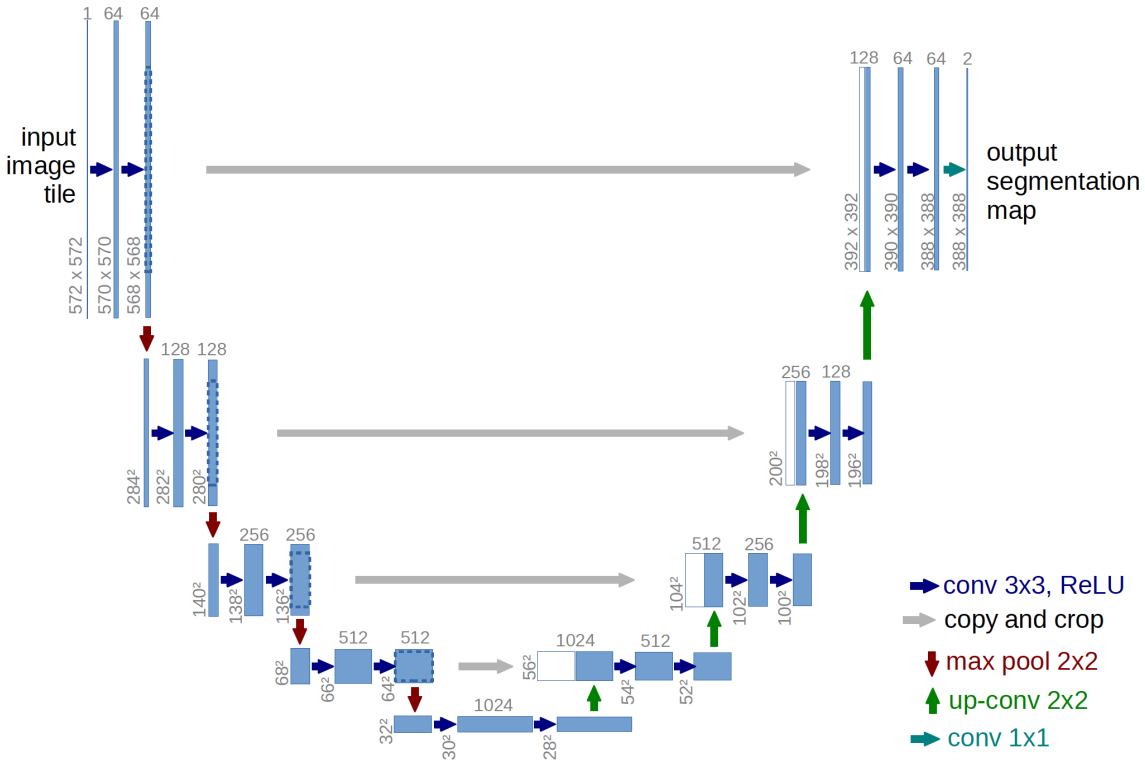

Figure 7 – U-Net Original Implementation (Ronneberger, Fischer and Brox, 2015)

On Level 0 after the last convolution (64 image 568x568pixel channels) the copy and crop operation is applied which is basically crops the images from 568x568pixels to 392x392pixels which is the upscale resolution.

The results of applying a U-Net on segmenting a Biomedical Image are shown in Figure 8.

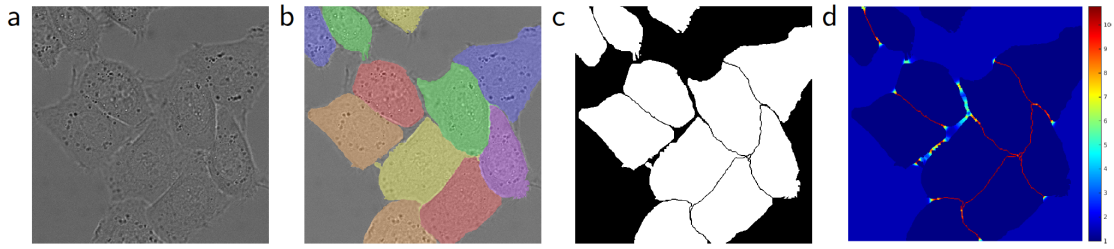

Figure 8 – U-Net Original Implementation Results (a) the raw Biomedical Image is Shown, (b) overlay with ground truth (i.e truth result), (c) the generated segmented image, (d) the differences between the truth and segmented images (Ronneberger, Fischer and Brox, 2015)

In our case we are not interested in the segmented image but rather than extracting the bias parameter b as shown in Figure 4.

3.4 Bias and Uncertainty Estimation in Post Processing Ensemble Weather Forecasts

Based on the previous U-Net implementation, a recent work (Ronneberger, Fischer and Brox, 2015) has been proposed in the literature estimating Bias in Ensemble Weather Forecasts (Grönquist *et al.*, 2021). In this approach the U-Net was restructured and reduced to a 3-level U-Net, because the 5-level original structure was causing overfitting. The fully redesigned U-Net architecture is shown in Figure 9.

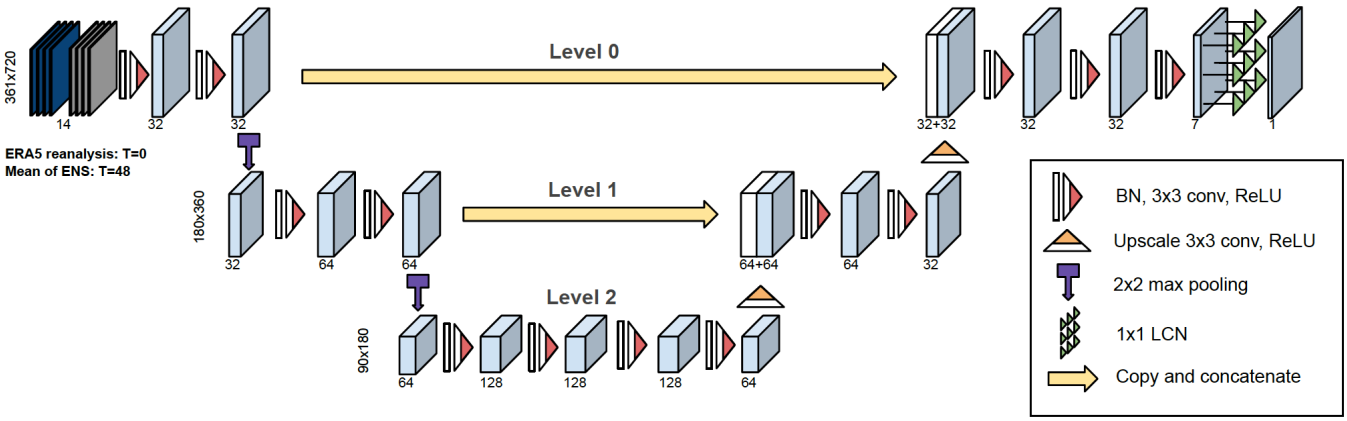

Figure 9 – U-Net for Post-Processing Weather Forecasts – The redesigned U-Net architecture used for Estimating Bias in Post Processing Ensemble Weather Forecasts (Grönquist *et al.*, 2021)

For inputs ECMWF Reanalysis ERA5 was used as ground truth at T=0 and mean of 10 Ensemble Forecasts at T=48. All 7 Fields for Temperature (T), Geopotential (Z), U Component of wind (U), V component of wind (V), Divergence (D), Vertical Velocity (W), Specific Humidity (Q) were used in the training process for the appropriate pressure level we are aiming to predict (Temperature, Geopotential). Temperature at 850mb (T850), and geopotential at 500mb (Z850) were used as the prediction parameters. On a separate experiment only predicted fields were used for training but when all fields were used there were better results. Certain pre-processing, such as standardization of the input to zero mean and unit variance, was implemented to prepare the data for processing. This U-Net implementation have shown some very promising results for the prediction of T850 to the order of 7.59% improvement when all fields were used and 6.90% improvement only when T850 was used. For the geopotential at 500mb (Z500), there was a 2.37% improvement

when all fields were used and 2.06% when only the geopotential at 500mb was used. In the same paper, uncertainty quantification was evaluated using a different Deep Learning architecture. There was a 16.4% and 12.3% improvement for T850 prediction when all Fields and Predicted fields were used respectively for the Uncertainty quantification architecture scheme. Also, there was a 12.9% and 12.8% improvement for Z500 prediction when all Fields and Predicted fields were used respectively for the same Uncertainty Quantification scheme.

3.5 Why U–Net?

U–Net architecture offers several advantages over other CNN architectures when it comes to image segmentation in cases with extreme weather phenomena (i.e ability to detect and segment features in that phenomenon). A summary of the key advantages of a U-Net vs other CNNs is the following:

1. U-Net Architecture: The U-Net architecture has a unique and intuitive structure that enables effective information flow. It consists of a contracting path, which captures context and reduces spatial dimensions, and an expansive path, which recovers spatial resolution for precise segmentation. This U-shaped architecture allows for the integration of high-resolution features from the contracting path with localized information in the expansive path, aiding in accurate segmentation.
2. Skip Connections: U-Net incorporates skip connections that directly connect corresponding layers in the contracting and expansive paths. These skip connections help to preserve fine-grained details and contextual information, facilitating precise localization and segmentation. They also address the problem

of information loss during downsampling and upsampling, promoting better feature propagation.

3. Dense Feature Extraction: U-Net is effective at capturing and representing intricate features in images due to its deep architecture and successive convolutional layers. This dense feature extraction capability enables the network to learn complex patterns and capture both local and global context, leading to improved segmentation accuracy.
4. Data Efficiency: U-Net can achieve good segmentation performance with relatively small training datasets. The combination of skip connections and dense feature extraction allows the network to make effective use of available training samples and generalize well to unseen data. This data efficiency is particularly beneficial when working with limited annotated data.
5. Versatility: While U-Net is widely recognized for image segmentation tasks, its architecture can be adapted for other tasks such as image-to-image translation, image denoising, or image super-resolution. The inherent flexibility of U-Net makes it a versatile choice for various computer vision tasks, allowing for easy modifications and extensions.
6. Real-Time Inference: U-Net's architecture, skip connections, and dense feature extraction enable efficient and parallelizable computations, making it feasible to achieve real-time inference on modern hardware. This advantage is particularly valuable in applications where low-latency segmentation results are required, such as real-time medical image analysis or autonomous driving systems.

3.6 Cross Validation of U-Net vs Conventional methods

Initially cross validation against the Lorenz 63 model should be implemented using a 3-DAR, 4-DVAR and other data assimilation algorithms. From equations (10), (11) we can obtain the bias. If bias from (10), (11) is very close to 0 we can inject bias by adding noise with certain mean and standard deviation. The bias estimate (a scalar) from this conventional approach should be the mean of the injected noise is added or defined by the difference in equations (10), (11).

On the Convolutional Neural Network approach using U-Net we could feed the background and the observation vectors in the architecture, a separate U-Net architecture can be used for each of those. If bias is zero as in the conventional approach some noise could be added. After the model is trained we can extract the bias vectors and produce the zonal averages as it was done in equations (10), (11). Certain comparison of the estimates should be conducted in both the conventional and U-Net implementations should be conducted.

After experimenting with the Lorenz 63 and the results are promising the study should be extended using real data 2D data.

3.7 Bias Correction using Deep Learning

Bias correction using deep learning involves leveraging deep neural networks to estimate and correct systematic biases in data or model predictions. Here are some approaches commonly used in the deep learning community for bias correction:

1. Generative Adversarial Networks (GANs): GANs consist of a generator and a discriminator network. The generator network learns to generate synthetic data samples, while the discriminator network learns to distinguish between real and synthetic samples. By training GANs on biased data or model outputs, the generator can learn to generate unbiased samples, effectively correcting the bias in the generated data.
2. Conditional Variational Autoencoders (CVAEs): CVAEs are deep generative models that learn the underlying distribution of the data and can generate new samples based on conditional information. By training CVAEs on biased data or model predictions, the model can learn to generate unbiased samples conditioned on the provided information, thus correcting the bias.
3. Domain Adaptation: Deep learning models can be trained to adapt from a biased source domain to an unbiased target domain. By using techniques such as adversarial learning or domain adaptation algorithms, the model can learn to minimize the discrepancy between the source and target domains, effectively reducing the bias in the predictions.
4. Residual Learning: Residual learning involves training deep neural networks to learn residual mappings, focusing on the difference between predicted outputs and ground truth. By training models to directly predict the bias or the bias-corrected output, deep learning models can effectively estimate and correct biases in the predictions.
5. Meta-Learning: Meta-learning techniques aim to learn the ability to adapt quickly to new tasks or domains. By training deep learning models on diverse datasets with

known biases, the model can learn to generalize and correct biases in new datasets or domains with similar biases.

6. Adversarial Training: Adversarial training involves training deep learning models in the presence of adversarial examples or biases. By incorporating an adversarial component into the learning process, the model can learn to be robust to biases and make more accurate predictions in the presence of systematic errors.

3.8 Future work and Research

Deep learning techniques have shown great potential in estimating bias in post-processing Ensemble Weather Forecasts. There is further experimentation starting from a Toy model needs to be done to restructure/redesign the U-Net Convolutional Neural Network (CNN) and be able to improve results. As a comparison the conventional methods for bias estimation for the implemented toy model need to be implemented. In addition, since ARMA models used in predicting the analysis increment very well a better ARMA model need to be identified for predicting the analysis increments. There are certain statistical tests such as the Akaike's Information Criterion (AIC) (Akaike, 1974) and Bayesian Information Criterion (BIC) that can be used to identify the appropriate ARMA model. Comparison between the conventional methods and the Deep Learning Methods need to be implemented in the Toy Model implementation. Extend the study in other variables other than T850 and Z500 and have as a goal to apply this on Ice Products for the Great Lakes. Apply shallow neural networks and compare results.

Chapter 4. Summary and conclusions

In this scholarly paper we investigated conventional Bias Estimation Techniques on postprocessing model outputs. As an alternative Convolutional Neural Networks were presented that basically can be used to estimate bias. The initially proposed work by (Ronneberger, Fischer and Brox, 2015) implementing a U-Net CNN architecture for biomedical image segmentation. In continuation of this work a modified U-Net for Bias estimation was presented (Grönquist *et al.*, 2021).

A general review on how Data Assimilation has used bias correction conventional approaches were discussed. At the same time how machine learning techniques were used in bias estimation were discussed. A methodology for comparing bias estimation using conventional methods vs U-Net was also presented. Further enhancements were proposed starting from experimenting with toy models and restructuring the U–Net architecture to improve results to extend the study with real data hopefully on detecting bias on ice products for the Great lakes.

References

Akaike, H. (1974) ‘A new look at the statistical model identification’, *IEEE Transactions on Automatic Control*, 19(6), pp. 716–723. Available at: https://doi.org/10.1109/TAC.1974.1100705.

Balmaseda, M.A. *et al.* (2007) ‘A multivariate treatment of bias for sequential data assimilation: Application to the tropical oceans’, *Quarterly Journal of the Royal Meteorological Society*, 133(622), pp. 167–179. Available at: https://doi.org/10.1002/qj.12.

Dee, D.P. (2005) ‘Bias and data assimilation’, *Quarterly Journal of the Royal Meteorological Society*, 131(613), pp. 3323–3343. Available at: https://doi.org/10.1256/qj.05.137.

Dee, D.P. and Da Silva, A.M. (1998) ‘Data assimilation in the presence of forecast bias’, *Quarterly Journal of the Royal Meteorological Society*, 124(545), pp. 269–295. Available at: https://doi.org/10.1002/qj.49712454512.

Glahn, H.R. and Lowry, D.A. (1972) ‘The Use of Model Output Statistics (MOS) in Objective Weather Forecasting’, *Journal of Applied Meteorology*, 11(8), pp. 1203–1211. Available at: https://doi.org/10.1175/1520-0450(1972)011<1203:TUOMOS>2.0.CO;2.

Grönquist, P. *et al.* (2021) ‘Deep Learning for Post-Processing Ensemble Weather Forecasts’, *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 379(2194), p. 20200092. Available at: https://doi.org/10.1098/rsta.2020.0092.

Guldberg, A. *et al.* (2005) ‘Reduction of systematic errors by empirical model correction: impact on seasonal prediction skill’, *Tellus A: Dynamic Meteorology and Oceanography*, 57(4), p. 575. Available at: https://doi.org/10.3402/tellusa.v57i4.14707.

Kharin, V.V. and Scinocca, J.F. (2012) ‘The impact of model fidelity on seasonal predictive skill: IMPACT OF FIDELITY ON SKILL’, *Geophysical Research Letters*, 39(18). Available at: https://doi.org/10.1029/2012GL052815.

Klein, W.H., Lewis, B.M. and Enger, I. (1959) ‘OBJECTIVE PREDICTION OF FIVE-DAY MEAN TEMPERATURES DURING WINTER’, *Journal of Meteorology*, 16(6), pp. 672–682. Available at: https://doi.org/10.1175/1520-0469(1959)016<0672:OPOFDM>2.0.CO;2.

Kornelsen, K.C. and Coulibaly, P. (2015) ‘Reducing multiplicative bias of satellite soil moisture retrievals’, *Remote Sensing of Environment*, 165, pp. 109–122. Available at: https://doi.org/10.1016/j.rse.2015.04.031.

Marzban, C., Sandgathe, S. and Kalnay, E. (2006) 'MOS, Perfect Prog, and Reanalysis', *Monthly Weather Review*, 134(2), pp. 657–663. Available at: https://doi.org/10.1175/MWR3088.1.

'Matlab' (2023). Natick, Massachusetts, United States: The MathWorks Inc. (Deep Learning Toolbox). Available at: https://www.mathworks.com.

Ronneberger, O., Fischer, P. and Brox, T. (2015) 'U-Net: Convolutional Networks for Biomedical Image Segmentation', in N. Navab et al. (eds) *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. Cham: Springer International Publishing (Lecture Notes in Computer Science), pp. 234–241. Available at: https://doi.org/10.1007/978-3-319-24574-4_28.

Su, C.-H. *et al.* (2014) 'Beyond triple collocation: Applications to soil moisture monitoring: Beyond triple collocation', *Journal of Geophysical Research: Atmospheres*, 119(11), pp. 6419–6439. Available at: https://doi.org/10.1002/2013JD021043.

Vila, D.A. *et al.* (2009) 'Statistical Evaluation of Combined Daily Gauge Observations and Rainfall Satellite Estimates over Continental South America', *Journal of Hydrometeorology*, 10(2), pp. 533–543. Available at: https://doi.org/10.1175/2008JHM1048.1.

Wilks, Daniel S (2011) *Statistical methods in the atmospheric sciences*. Academic press.

MATHWORKS online webinars. What are the Convolutional Neural Networks? Introduction to Deep Learning: