Remember to Correct the Bias When Using Deep Learning for Regression!

Igel, Christian; Oehmcke, Stefan

Published in:
KI - Kunstliche Intelligenz

DOI:
10.1007/s13218-023-00801-0

Publication date:
2023

Document version
Publisher’s PDF, also known as Version of record

Document license:
CC BY

Citation for published version (APA):
Remember to Correct the Bias When Using Deep Learning for Regression!

Christian Igel¹,² · Stefan Oehmcke²

Received: 12 July 2022 / Accepted: 16 November 2022 / Published online: 18 April 2023
© The Author(s) 2023

Abstract
When training deep learning models for least-squares regression, we cannot expect that the training error residuals of the final model, selected after a fixed training time or based on performance on a hold-out data set, sum to zero. This can introduce a systematic error that accumulates if we are interested in the total aggregated performance over many data points (e.g., the sum of the residuals on previously unseen data). We suggest adjusting the bias of the machine learning model after training as a default post-processing step, which efficiently solves the problem. The severeness of the error accumulation and the effectiveness of the bias correction are demonstrated in exemplary experiments.

Keywords Deep learning · Regression · Bias correction

1 Problem Statement
We consider regression tasks in which we are not only interested in minimizing point-wise losses, but where we must avoid systematic errors that accumulate when the regression model is evaluated on many data points, for example, when we want the sum of the residuals on a test set to be of small magnitude. Our study is motivated by applications in large-scale ecosystem monitoring such as segmenting trees in satellite imagery for assessing the tree canopy cover of a country [3, 15] and learning systems trained on small patches of 3D point clouds to predict the biomass (and thus stored carbon) of large forests [11, 17]. However, there are many other application areas in which accumulated predictions matter, such as estimating the overall performance of a portfolio based on estimates of the performance of the individual assets, or predicting overall demand based on forecasts for individual consumers.

Against this background, we consider models $f: \mathcal{X} \to \mathbb{R}^d$ of the form

$$f_\theta(x) = a^T h_w(x) + b$$

with parameters $\theta = (w, a, b)$ and $x \in \mathcal{X}$. Here $\mathcal{X}$ is some arbitrary input space and w.l.o.g. we assume $d = 1$. The function $h_w: \mathcal{X} \to \mathbb{R}^F$ is parameterized by $w$ and maps the input to an $F$-dimensional real-valued feature representation, $a \in \mathbb{R}^F$, and $b$ is a scalar. If $\mathcal{X}$ is a Euclidean space and $h$ the identity, this reduces to standard linear regression. However, we are interested in the case where $h_w$ is more complex. In particular,

- $f_\theta$ can be a deep neural network, where $a$ and $b$ are the parameters of the final output layer and $h_w$ represents all other layers (e.g., a convolutional or point cloud architecture);
- $h: \mathcal{X} \to \mathbb{R}$ can be any regression model (e.g., a random forest or deep neural network) and $f_\theta$ denotes $h_w$ with an additional wrapper, where $a = 1$ and initially $b = 0$.

In the following, we call $b$ the distinct bias parameter of our model (although $w$ may comprise many parameters typically referred to as bias parameters if $h_w$ is a neural network). Given

1 For reviews of regression methods including deep neural networks for biomass prediction we refer to Morais et al. [14] and Zhang et al. [28], where also square-root of the MSE (Eq. 1) as well as the mean error (Eq. 6) are considered as evaluation criteria.
some training data \( D = \{(x_1, y_1), \ldots, (x_N, y_N)\} \) drawn from a distribution \( p_{\text{data}} \) over \( \mathbb{X} \times \mathbb{Y} \), we assume that the model parameters \( \theta \) are determined by minimizing the mean-squared error (MSE)

\[
\text{MSE}_D(f_\theta) = \frac{1}{|D|} \sum_{(x,y) \in D} (y - f_\theta(x))^2.
\] (2)

potentially combined with some form of regularization. Typically, the goal is to achieve a low expected error

\[
\text{MSE}(f_\theta) = \mathbb{E}_{(x,y) \sim p_{\text{data}}}[ (y - f_\theta(x))^2 ] = \mathbb{E}[\text{MSE}_{D_{\text{test}}}(f_\theta)],
\]

where the second expectation is over all test data sets drawn i.i.d. based on \( p_{\text{data}} \). However, here we are mainly concerned with applications where the (expected) absolute total error defined as the absolute value of the sum of residuals

\[
\Delta_{D_{\text{test}}}(f_\theta) = \sum_{(x,y) \in D_{\text{test}}} |(y - f_\theta(x))|,
\] (3)

is of high importance. That is, we are interested in the total aggregated performance over many data points. A related error measure is the relative total error given by

\[
\delta_{D_{\text{test}}}(f_\theta) = \frac{\Delta_{D_{\text{test}}}(f_\theta)}{\sum_{(x,y) \in D_{\text{test}}} y},
\] (4)

which is similar to the relative systematic error or relative systematic deviation

\[
\text{SD}_{D_{\text{test}}}^{\text{relative}}(f_\theta) = \frac{100}{|D_{\text{test}}|} \sum_{(x,y) \in D_{\text{test}}} \frac{y - f_\theta(x)}{y}
\] (5)

(in \%, e.g., [6, 11]) and the mean error

\[
\text{ME}_{D_{\text{test}}}(f_\theta) = \frac{\Delta_{D_{\text{test}}}(f_\theta)}{|D_{\text{test}}|},
\] (6)

which is the absolute value of the systematic deviation [6].

The measures defined by Eqs. 3–6 are used to quantify the prediction bias of the model, that is, how well the sum of predictions \( \sum_{(x,y) \in D_{\text{test}}} f_\theta(x) \) approximates the sum of true values \( \sum_{(x,y) \in D_{\text{test}}} y \) for a test set \( D_{\text{test}} \).

Typically it is assumed that minimizing the point-wise losses is all that matters. We consider the specific setting where we want to minimize the point-wise losses and at the same time the aggregated loss. For \( |D_{\text{test}}| \rightarrow \infty \) a constant model predicting

\[
\hat{y} = \mathbb{E}_{(x,y) \sim p_{\text{data}}}[y]
\] (7)

would minimize \( \Delta_{D_{\text{test}}}(f_\theta)/|D_{\text{test}}| \). So why do we not minimize the mean error on the training data directly? First, because we also want to have high accuracy for individual predictions (e.g., a small MSE). And second, in practice the averages on the training set \( \frac{1}{|D|} \sum_{(x,y) \in D} y \) and on a test set

\[
\frac{1}{|D_{\text{test}}|} \sum_{(x,y) \in D_{\text{test}}} y
\]

can be very different from each other and from \( \hat{y} \). For example, consider a machine learning model for predicting aboveground biomass of forests, which takes remote sensing data from small forest patches as input and aggregates biomass predictions at patch level across the forest area [17]. The training data for such a model may contain a balanced mixture of broadleaf and conifer trees. However, it may be applied to monoculture plantations. Thus the test set violates the i.i.d. assumption (this can be viewed as a special form of covariate shift or sample selection bias). Therefore, optimization of individual predictions (e.g., minimizing Eq. 2) is important and just learning \( \hat{y} \) is not sufficient.

At a first glance, it seems that low \( \mathbb{E}[\text{MSE}_{D_{\text{test}}}(f_\theta)] \) guarantees low \( \mathbb{E}[\Delta_{D_{\text{test}}}(f_\theta)] \), where the expectations are again with respect to data sets drawn i.i.d. based on \( p_{\text{data}} \). Obviously, \( \text{MSE}_D(f_\theta) = 0 \) implies \( \Delta_{D_{\text{test}}}(f_\theta) = 0 \) for any data set \( D \). More generally, optimal parameters \( \theta^* \) minimizing \( \text{MSE}_D(f_\theta) \) result in \( \Delta_{D_{\text{test}}}(f_{\theta^*}) = 0 \). Actually, \( \Delta_{D_{\text{test}}}(f_{\theta^*}) = 0 \) is necessary and sufficient for the error residuals to sum to zero and thus \( \Delta_{D_{\text{test}}}(f_{\theta^*}) = 0 \). This well known fact can be directly seen from Eq. 9 below. However, if the partial derivative of the error with respect to \( b \) is not zero, a low \( \text{MSE}_{D_{\text{test}}}(f_\theta) \) may not imply a low \( \Delta_{D_{\text{test}}}(f_\theta) \). In fact, if we are ultimately interested in the total aggregated performance over many data points, a wrongly adjusted parameter \( b \) may lead to significant systematic errors. Assume that \( f^* \) is the Bayes optimal model for a given task and that \( f_\theta \) is the model where the optimal bias parameter \( b^* \) is replaced by \( b^* - \delta_b \). Then for a test set \( D_{\text{test}} \) of cardinality \( N_{\text{test}} \) we have

\[
\sum_{(x,y) \in D_{\text{test}}} (y - f_{\theta_b}(x)) = N_{\text{test}} \cdot \delta_b + \sum_{(x,y) \in D_{\text{test}}} (y - f^*(x)).
\] (8)

That is, the errors \( \delta_b \) accumulate and thus even a very small \( \delta_b \) can have a drastic effect on aggregated quantities. While one typically hopes that errors partly cancel out when applying a model to a lot of data points, the aggregated error due to a badly chosen bias parameter increases. This can be a severe problem when using deep learning for regression, because in the canonical training process of a neural network for regression minimizing the (regularized) MSE the partial derivative of the error w.r.t. the parameter \( b \) of the final model cannot be expected to be zero:

- Large deep learning systems are typically not trained until the partial derivatives of the error w.r.t the model parameters are close to zero, because this is not necessary to achieve the desired performance in terms of MSE and/or training would take too long.
- The final weight configuration is often picked based on the performance on a validation data set (e.g., [21]), not depending on how close the parameters are to a local
optimum as measured, for example, by the maximum norm of the gradient.

- Mini-batch learning introduces a random effect in the parameter updates, and therefore in the bias parameter value in the final network.

Thus, despite low MSE, the performance of a (deep) learning system in terms of the total error as defined in Eq. 3 can get arbitrarily bad. For example, in the tree canopy estimation task described above, you may get a decently accurate biomass estimate for individual trees, but the prediction over a large area (i.e., the quantity you are actually interested in) could be very wrong.

Therefore, we propose to adjust the bias parameter after training a machine learning model for least-squares regression as a default post-processing step. This post-processing can be regarded as playing a similar role as model calibration in classification (e.g., [9]). In the next section, we show how to simply compute this correction that exactly removes the prediction bias on the training data (or a subset thereof) and discuss the consequences. Section 3 presents experiments demonstrating the problem and the effectiveness of the proposed solution.

2 Solution: Adjusting the Bias

If the sum of residuals on the training data set \( \mathcal{D} \) does not vanish, \( \Delta_{\mathcal{D}}(f_b) > 0 \), we can also not expect that the residuals will cancel each other on some test set \( \mathcal{D}_{\text{test}} \), showing a systematic error leading to a large \( \Delta_{\mathcal{D}_{\text{test}}}(f_b) \). Thus, we suggest to apply the minimal change to the model that leads to \( \Delta_{\mathcal{D}}(f_b) = 0 \), namely minimizing the MSE on \( \mathcal{D} = \{(x_1, y_1), \ldots, (x_N, y_N)\} \) w.r.t. \( b \) while fixing all other model parameters \( w \) and \( a \). For the resulting bias parameter \( b^* \) the first derivative w.r.t. \( b \) vanishes

\[
\frac{\partial \text{MSE}_D(f_b)}{\partial b} \bigg|_{b=b^*} = 0
\]

implying \( \Delta_{\mathcal{D}}(f_{(w,a,b^*)}) = 0 \). Thus, for fixed \( w \) and \( a \) we can simply solve for the new bias parameter:

\[
b^* = \frac{\sum_{i=1}^{N} (y_i - a^T h_w(x_i))}{N}
\]

\[
= \frac{\sum_{i=1}^{N} y_i - \sum_{i=1}^{N} a^T h_w(x_i)}{N}
\]

\[
= \frac{\sum_{i=1}^{N} y_i - \sum_{i=1}^{N} f_b(x_i)}{N} \quad + b^* .
\]

In practice, we can either replace \( b \) in our trained model by \( b^* \) or add \( \delta_b \) to all model predictions. The costs of computing \( b^* \) and \( \delta_b \) are the same as computing the error on the data set used for adjusting the bias.

The trivial consequences of this adjustment are that the MSE on the training data set is reduced and the residuals on the training set cancel each other. But what happens on unseen data? Intuitively, under the assumption that adjusting the single bias parameter does not lead to overfitting, the model with \( \Delta_{\mathcal{D}^\text{test}}(f_{(w,a,b^*)}) = 0 \) can be expected to have a lower \( \Delta_{\mathcal{D}^\text{test}}(f_{(w,a,b^*)}) \) on a test set \( \mathcal{D}_{\text{test}} \) than a model with \( \Delta_{\mathcal{D}^\text{test}}(f_b) > 0 \). The effect on the MSE \( \text{MSE}_{\mathcal{D}_{\text{test}}} \) is expected to be small. Adjusting the single scalar parameter \( b \) based on a lot of data is very unlikely to lead to overfitting. On the contrary, in practice we are typically observing a reduced MSE on external test data after adjusting the bias. However, this effect is typically minor. The weights of the neural network and in particular the bias parameter in the final linear layer are learned sufficiently well so that the MSE is not significantly degraded because the single bias parameter is not adjusted optimally—and that is why one typically does not worry about it although the effect on the absolute total error may be drastic.

2.1 Which Data Should Be Used to Adjust the Bias?

While one could use an additional hold-out set for the final optimization of \( b \), this is not necessary. Data already used in the model design process can be used, because assuming a sufficient amount of data, selecting a single parameter is unlikely to lead to overfitting. If there is a validation set (e.g., for early-stopping), then these data could be used. If data augmentation is used, augmented data sets could be considered. We recommend to simply use all data available for model building (e.g., the union of the training and validation sets). This minimizes the prediction bias of the model in the same way as standard linear regression. Using a large amount of data for the (typically very small) adjustment of a single model parameter that has no non-linear influence on the model predictions is extremely unlikely to lead to overfitting (as empirically shown in the experiments below), and the more data that are used to compute the correction the more accurate one can expect it to be.

2.2 How to Deal with Regularization?

So far, we have just considered empirical risk minimization. However, the bias parameter can adjusted regardless of how the model was obtained. This includes the use of early-stopping [21] or regularized risk minimization with an objective of the form \( \frac{1}{N} \sum_{i=1}^{N} (y_i - f_b(x_i))^2 + \Omega(\theta) \). Here, \( \Omega \) denotes some regularization depending on the parameters. This includes weight-decay, however, typically this type of
regularization would not consider the bias parameter $b$ of a regression model anyway (e.g., [2, p. 342]).

### 2.3 Why Not Adjust More Parameters?

The proposed post-processing serves a very well defined purpose. If the error residuals do not sum to zero on the training data set, the residuals on test data can also not be expected to do so, which leads to a systematic prediction error. The proposed adjustment of $b$ is the minimal change to the model that solves this problem. We assume that the model before the post-processing shows good generalization performance in terms of MSE, so we want to change it as little as possible. As argued above and shown in the experiments, just adjusting $b$, which has no non-linear effect on the predictions, based on sufficient data is unlikely to lead to overfitting. On the contrary, in practice an improvement of the generalization performance (e.g., in terms of MSE) is often observed (see also the experiments below).

Of course, there are scenarios where adjusting more parameters can be helpful. For example, it is straightforward to also adjust the factor $a$ in the wrapper such that the partial derivative of the MSE with respect to $a$ vanishes. This has the effect that afterwards the residuals and training inputs are uncorrelated. However, minimizing the unregularized empirical risk w.r.t. many parameters (in particular if we have non-linear effects) bears the risk of overfitting.

### 2.4 Related Work

Bias correction and calibration of regression models per se is not new. In the context of quantile regression models and regression models that predict probability distributions over the target values given an input, post-hoc calibration is used to match the observed quantiles and probability distributions to the respective quantiles and probability distributions of the model predictions (see [23], for an overview).

The post-processing step we propose can be related to updating regression models using additional labelled data (e.g., in a transfer learning setting). For example, biomass predictions are calibrated using additional reference observations (e.g., see [7, equation 13], or [16]). However, in this setting, the data used for building the model and for correcting it are typically different, with the data used for calibration being more accurate. In contrast, we suggest to use the same data for correction as used for training the model (see below). In the appendix, we describe an algorithm proposed by Rodgers et al. [22] for calibrating a regression model using additional data adapted for our setting.

Bias correction is commonly applied to power regression models, for example, allometric equations in ecology, if the model parameters are determined by MSE optimization after applying logarithmic transformations to the inputs and targets (e.g., see [27]). Because an estimator that is unbiased in the transformed space causes a bias in the original space, correction factors have been introduced that are applied after fitting the model parameters [1, 8, 27]. This type of correction becomes necessary because the optimization objective itself leads to an undesired bias. Our proposed correction become necessary because the objective function is not optimized to a point where the partial derivative with respect to a particular parameter vanishes.

### 3 Examples

In this section, we present experiments that illustrate the problem of a large total error despite a low MSE and show that adjusting the bias as proposed above is a viable solution. We start with a simple regression task based on a UCI benchmark data set [5], which is easy to reproduce. Then we move closer to real-world applications and consider convolutional neural networks for ecosystem monitoring.

#### 3.1 Gas Turbine Emission Prediction

First, we look at an artificial example based on real-world data from the UCI benchmark repository [5], which is easy to reproduce. We consider the Gas Turbine CO and NOx Emission Data Set [12], where each data point corresponds to CO and NOx (NO and NO$_2$) emissions and 9 aggregated sensor measurements from a gas turbine summarized over 1 h. The typical task is to predict the hourly emissions given the sensor measurements. Here we consider the fictitious task of predicting the total amount of CO emissions for a set of measurements.

**Experimental setup** There are 36,733 data points in total. We assumed that we know the emissions for $N_{\text{train}} = 21,733$ randomly selected data points, which we used to build our models.

We trained a neural network with two hidden layers with sigmoid activation functions having 16 and 8 neurons, respectively, feeding into a linear output layer. There were shortcut connections from the inputs to the output layer. We randomly split the training data into 16,733 examples for gradient computation and 5000 examples for validation. The network was trained for 1000 epochs using Adam [13] with a learning rate of $1 \times 10^{-2}$ and mini-batches of size 64. The network with the lowest error on the validation data was selected. For adjusting the bias parameter, we computed $\delta_b$ using Eq. 10 and all $N_{\text{train}}$ data points available for model development. As a baseline, we fitted a linear regression model using all $N_{\text{train}}$ data points.

We used Scikit-learn [19] and PyTorch [18] in our experiments. The input data were converted to 32-bit floating point precision. We repeated the experiments 10 times with 10
random data splits, network initializations, and mini-batch shufflings.

Results The results are shown in Table 1 and Fig. 1. The neural networks without bias correction achieved a higher \( R^2 \) (coefficient of determination) than the linear regression on the training and test data, see Table 1. On the test data, the \( R^2 \) averaged over the ten trials increased from 0.56 to 0.78 when using the neural network. However, the \( \Delta_D \) and \( \Delta_{D_{\text{test}}} \) were much lower for linear regression. This shows that a better MSE does not directly translate to a better total error (sum of residuals).

Correcting the bias of the neural network did not change the networks’ \( R^2 \), however, the total errors went down to the same level as for linear regression and even below. Thus, correcting the bias gave the best of both worlds, a low MSE for individual data points and a low accumulated error.

Figure 1 demonstrates how the total error developed as a function of test data set size. As predicted, with a badly adjusted bias parameter the total error increased with the number of test data points, while for the linear models and the neural network with adjusted bias this negative effect was less pronounced. The linear models performed worse than the neural networks with adjusted bias parameters, which can be explained by the worse accuracy of the individual predictions.

### 3.2 Forest Coverage

Deep learning holds great promise for large-scale ecosystem monitoring [20, 26], for example for estimating tree canopy cover and forest biomass from remote sensing data [3, 15, 17]. Here we consider a simplified task where the goal is to predict the amount of pixels in an image that belong to forests given a satellite image. We generated the input data from Sentinel 2 measurements (RGB values) and the accumulated pixels from a landcover map² as targets, see Fig. 2 for examples. Both, input and target are at the same 10 m spatial resolution, collected/estimated in 2017, and cover the country of Denmark. Each sample is a 100 × 100 large image with no overlap between images.

#### Experimental setup

From the 127,643 data points in total, 70% (89,350) were used for training, 10% (12,764) for testing, and 20% (25,529) for validation. The model architecture consisted of three fully connected layers with 1024, 512, and 256 neurons, respectively, followed by a linear output layer.

Table 1 Results for the total CO emissions prediction tasks for the different models, where “linear” refers to linear regression, “not corrected” to a neural network without bias correction, and “corrected” to the same neural network with corrected bias parameter.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>( R^2_D )</th>
<th>( R^2_{D_{\text{test}}} )</th>
<th>( \Delta_D )</th>
<th>( \Delta_{D_{\text{test}}} )</th>
<th>( \delta_{D_{\text{test}}} )</th>
<th>ME_{D_{\text{test}}}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear</td>
<td>0.56 ± 0.0</td>
<td>0.57 ± 0.0</td>
<td>0 ± 0</td>
<td>173 ± 14</td>
<td>0.49 ± 0.0</td>
<td>0.02 ± 0.0</td>
</tr>
<tr>
<td>Not corrected</td>
<td>0.78 ± 0.0</td>
<td>0.72 ± 0.0</td>
<td>1018 ± 70</td>
<td>785 ± 53</td>
<td>2.21 ± 0.15</td>
<td>0.05 ± 0.0</td>
</tr>
<tr>
<td>Corrected</td>
<td>0.78 ± 0.0</td>
<td>0.72 ± 0.0</td>
<td>0 ± 0</td>
<td>122 ± 6</td>
<td>0.34 ± 0.02</td>
<td>0.01 ± 0.0</td>
</tr>
</tbody>
</table>

The results are based on 10 trials. The mean and standard error (SE) are given; values are rounded to two decimals; \( R^2, \Delta, \delta, \) and ME denote the coefficient of determination, the absolute total error, the relative error, and the mean error; \( \delta \) is given in percent; \( D \) and \( D_{\text{test}} \) refer to data available for model development and testing, respectively.

---

² https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2/Land-cover_maps_of_Europe_from_the_Cloud.
for validation and 20% (25,529) for testing. For each of the 10 trials a different random split of the data was considered.

We employed the EfficientNet-B0 [24], a deep convolutional network that uses mobile inverted bottleneck MBConv [25] and squeeze-and-excitation [10] blocks. It was trained for 300 epochs with Adam and a batch size of 256. For 100 epochs the learning rate was set to $3 \times 10^{-5}$ and thereafter reduced to $1 \times 10^{-5}$. The validation set was used to select the best model w.r.t. $R^2$. When correcting the bias, the training and validation set were combined. We considered the constant model predicting the mean of the training targets as a baseline.

Results The results are summarized in Fig. 3 and Table 2. The bias correction did not yield a better $R^2$ result, with 0.992 on the training set and 0.977 on the test set. However, the best model w.r.t. $R^2$ for 300 epochs the learning rate was set to $3 \times 10^{-5} \pm 0.059$ and $2.5 \pm 0.1\%$. The number $K$ of neighbors is a hyperparameter. The term $\sum_{i=1}^{K} \omega_{x,k} \cdot h(x_{i,k})$ is the weighted $K$-nearest neighbor prediction for $x$ using $D'$. The idea is to correct the prediction $h(x)$ of the model by the (weighted) mean error of $h$ when applied to the $K$ nearest neighbors of $x$ in $D'$. Let $(x_{i,k}, y_{i,k})$ denote the $i$-th nearest neighbor of $x$ in a data set $D'$. The distance is measured using a metric $d$, where ties can be broken at random or deterministically. Following Bruneau and McElroy [4] and Rodgers et al. [22], we consider the Mahalanobis distance $d(x, z) = \sqrt{(x - z)'^T C^{-1} (x - z)}$ for real vectors $x$ and $z$, where $C$ is the empirical covariance matrix of the features based on the sample in $D'$. The output $h(x)$ is then corrected to $f(x)$ using

$$
\sum_{i=1}^{K} \omega_{x,k} \cdot h(x_{i,k}) - f(x) = h(x) + \frac{\sum_{i=1}^{K} \omega_{x,k} \cdot h(x_{i,k}) - y_{i,k}}{\sum_{i=1}^{K} \omega_{x,k}}.
$$

Here $\omega_{x,k}$ is a weight depending on $d(x, x_{i,k})$. Adjusting the bias such that the residuals sum to zero should be the default post-processing step when doing least-squares regression using deep learning. It comes at the cost of at most a single forward propagation of the training and/or validation data set, but removes a systematic error that accumulates if individual predictions are summed.

### 4 Conclusions

The bias correction can be related to other post-processing methods. There is a line of research that studies how the output of a model $h$ can be adjusted using an additional labelled data set $D'$ not used for building the model, and the approach by Rodgers et al. [22] resembles the recommended bias correction.

Table 2 Results of forest coverage prediction, $R^2$ and $\Delta$, $\delta$, and ME denote the coefficient of determination, absolute total error, relative error, and mean error; $D$ and $D_{test}$ are all data available for model development and testing, respectively.

<table>
<thead>
<tr>
<th>MODEL</th>
<th>$R^2_D$</th>
<th>$R^2_{D_{test}}$</th>
<th>$\Delta_D$</th>
<th>$\Delta_{D_{test}}$</th>
<th>$\delta_{D_{test}}$</th>
<th>ME$<em>{D</em>{test}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.000 ± 0.000</td>
<td>$-3 \times 10^{-5}$ ± 0.0</td>
<td>6 ± 2</td>
<td>169,666 ± 48,944</td>
<td>0.955 ± 0.272</td>
<td>665 ± 192</td>
</tr>
<tr>
<td>Not corrected</td>
<td>0.992 ± 0.027</td>
<td>0.977 ± 0.0</td>
<td>389,747 ± 77,987</td>
<td>152,666 ± 22,164</td>
<td>0.864 ± 0.124</td>
<td>598 ± 87</td>
</tr>
<tr>
<td>Corrected</td>
<td>0.992 ± 0.027</td>
<td>0.977 ± 0.0</td>
<td>3 ± 1</td>
<td>59,819 ± 10,501</td>
<td>0.338 ± 0.059</td>
<td>234 ± 41</td>
</tr>
</tbody>
</table>

The relative total error $\delta$ is given in percent. Average and standard error (SE) of these metrics are given over 10 trials for the different models, where “mean” refers to predicting the constant mean of the training set, “not corrected” to EfficientNet-B0 without bias correction, and “corrected” to the same neural network with corrected bias parameter. Values are rounded to three decimals.

---

**Fig. 2** Exemplary inputs and targets (y) for the forest coverage dataset. a Shows a scene with 75.7%, b with 2.5%, and c with 0.1% forest

---

**Fig. 3** The results are summarized in Table 2. The bias correction did not yield a better $R^2$ result, with 0.992 on the training set and 0.977 on the test set. However, the best model w.r.t. $R^2$ for 300 epochs the learning rate was set to $3 \times 10^{-5} \pm 0.059$ and $2.5 \pm 0.1\%$. The number $K$ of neighbors is a hyperparameter. The term $\sum_{i=1}^{K} \omega_{x,k} \cdot h(x_{i,k})$ is the weighted $K$-nearest neighbor prediction for $x$ using $D'$.
For constant choice of the weighting function for computing the\( K \)th nearest neighbor, obviously \( \delta_{D_{\text{test}}} = 0.38\% \pm 0.02\% \), and the Mahalanobis distance or standard Euclidean distance between raw pixels are appropriate to measure image similarity. In contrast, on the artificial problem in Sect. 3.1 with 9 inputs each representing a meaningful feature, nearest neighbor regression can be expected to work.

We applied the local bias correction to the problem in Sect. 3.1 with \( K = 3 \) as suggested by Rodgers et al. [22]. This resulted in \( R^2_{\text{test}} = 0.73 \pm 0.0, \quad \Delta D_{\text{test}} = 51 \pm 5.0, \quad \delta_{D_{\text{test}}} = 0.38\% \pm 0.02\% \), and ME\( D_{\text{test}} = 0.02 \pm 0.0 \). In this toy example, the nearest-neighbor auxiliary model performs very well. Still, while the bias correction reduced the systematic errors compared to the uncorrected neural network, it performed worse than the proposed rigorous post-processing (see Table 1).

Acknowledgements The authors acknowledge support by the Villum Foundation through the project Deep Learning and Remote Sensing for Unlocking Global Ecosystem Resource Dynamics (DeReEco, project number 34306) and the Pioneer Centre for AI, DNRF grant number P1.

Funding Open Access funding enabled and organized by Projekt DEAL.

Data availability The Gas Turbine CO and NOx Emission Data Set used in Sect. 3.1 is freely available from the UCI Machine Learning Repository. The satellite data used in Sect. 3.2 are based on the Copernicus Sentinel-2 mission and are freely available from https://www.esa.int. The selected images are available from the authors on reasonable request.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References
