Twin neural network regression is a semi-supervised regression algorithm

IOP Publishing Mach. Learn.: Sci. Technol. 3 (2022) 045007 https://doi.org/10.1088/2632-2153/ac9885

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
7 July 2022

REVISED
14 September 2022

ACCEPTED FOR PUBLICATION
7 October 2022

PUBLISHED
20 October 2022

PAPER

Twin neural network regression is a semi-supervised regression algorithm

Sebastian J Wetzel[1,*], **Roger G Melko**[1,2,3] and **Isaac Tamblyn**[3,4]

[1] Perimeter Institute for Theoretical Physics, Waterloo, Ontario N2L 2Y5, Canada
[2] University of Waterloo, Waterloo, Ontario N2L 3G1, Canada
[3] Vector Institute for Artificial Intelligence, Toronto, Ontario M5G 1M1, Canada
[4] University of Ottawa, Ottawa, Ontario K1N 6N5, Canada
[*] Author to whom any correspondence should be addressed.

E-mail: swetzel@perimeterinstitute.ca

Keywords: artificial neural networks, regression, semi-supervised learning

Abstract

Twin neural network regression (TNNR) is trained to predict differences between the target values of two different data points rather than the targets themselves. By ensembling predicted differences between the targets of an unseen data point and all training data points, it is possible to obtain a very accurate prediction for the original regression problem. Since any loop of predicted differences should sum to zero, loops can be supplied to the training data, even if the data points themselves within loops are unlabelled. Semi-supervised training improves TNNR performance, which is already state of the art, significantly.

1. Introduction

Regression is the process of estimating the relationship between feature variables to outcome variables. It is one of the most central machine learning tasks in a wide range of scientific, analytic and industrial research and development disciplines. Regression analysis is considered to be a supervised machine learning task, where one is given labelled training data from which a model is trained to make predictions of the labels of unlabelled data points. In many applications, there is few labelled training data available, while unlabelled data is much easier to obtain. In addition to labelled data, unlabelled data can be leveraged to train machine learning models in the context of semi-supervised learning [1, 2]. There are two different cases of semi-supervised learning, which differ by the availability of the data for which one wants to obtain predictions during the training phase. The goal of inductive semi-supervised learning is to find the function that maps a feature variable to its outcome variable from a combined set of labelled and unlabelled data. This function can then be used to make predictions on new data points that are not available during the training phase. The goal of transductive semi-supervised learning is to infer the correct labels for the given unlabelled data, which is already present during the training phase.

Most of existing semi-supervised learning algorithms focus on classification problems, which are primarily based on continuity and cluster assumptions. Continuity assumes that neighboring data points likely share the same label, clustering of the data makes it possible to draw decision boundaries in low density regions [1, 2]. Continuity and cluster assumptions cannot be leveraged in semi-supervised regression to the same extent as in semi-supervised classification. That is why it is more difficult to develop semi-supervised regression methods. This obstacle explains the scarcity of publications on semi-supervised regression and why research in semi-supervised regression was not able to achieve the same level of success as semi-supervised classification.

Semi-supervised regression was a research area before neural networks became popular in 2012 [3]. However, only very few research articles have since included neural networks as part of their proposed semi-supervised regression pipelines, mostly combined with other machine learning algorithms [4, 5]. While neural networks are not always the optimal choice, their performance ceiling is much higher due to the

universal approximation theorem [6, 7], the incorporation of feature extraction into the learning problem, and the efficient training through gradient descent and backpropagation.

In this article we develop a semi-supervised training procedure for TNNR [8]. TNNR is based on an architecture similar to Siamese neural networks [9, 10] in the sense that it takes two inputs. TNNR is trained to predict the differences between the labels of the input pair. The solutions of the original regression problem is then obtained by creating an ensemble of all predicted differences between a new data point and all labelled training data points plus their labels. TNNR is normally trained on pairs, here we explain how TNNR can be trained on triples of data points which can be unlabelled. These triples form a loop along which predictions must sum to zero. One can implement this constraint using a suitable loss function to transform TNNR into a semi-supervised regression algorithm. The strengths of TNNR can be summarized as follows:

(a) TNNR attempts to circumvent the bias-variance tradeoff by internal ensembling which translates to a smaller expected error and experimentally confirmed performance boost [8].
(b) TNNR provides uncertainty estimates through the variance of predictions and violations of consistency conditions [8].
(c) TNNR can be trained in a semi-supervised manner on loops containing unlabelled data points by enforcing loop consistency [this work].

We demonstrate the performance of semi-supervised TNNR at six commonly used benchmark data sets for regression, for example the well-known Boston Housing (BH) data set. Furthermore, we test our algorithm on science data sets that were generated from mathematical equations including descriptions of electrical circuits.

2. Prior work

Twin neural networks are inspired by Siamese neural networks, these networks were introduced to solve an infinite class classification problem as it occurs in finger print recognition or signature verification [9, 10]. Siamese neural networks consist of two identical neural networks which project an input pair into a latent space. The similarity of two inputs is determined based on the distance in latent space. The twin neural network in TNNR also takes a pair of inputs to predict the difference between the labels [8]. These networks differ from Siamese networks in two properties: First, there is no (or only minimal) weight sharing between neurons acting on similar parts of each member of the input pair. Second, they do not project into a latent space but are fully connected. TNNR predictions can be transformed to a solution of the original regression problem via averaging over all predicted differences between an unlabelled data point and all labelled training data points, plus their labels. TNNR attempts to circumvent the bias-variance tradeoff, since the bias of neural networks is low given sufficiently many parameters and the variance can be reduced by the implicit ensemble of different predictions, this argument is supported by the bias and variance decomposition for ensembles as outlined in appendix D. TNNR also provides prediction uncertainty estimates through the variance of the predictions.

Most previous research in semi-supervised regression can be classified into three categories: Co-training, kernel and graph based regression methods [11]. Co-training denotes alternating training between two different regressors (the original idea is to use different feature sets for each regressor) where the predictions of one helps create a training set for the other, this process is repeated until convergence [12]. While initially invented for semi-supervised classification, there has been progress to adapt co-training to semi-supervised regression [13–15]. Semi-supervised support vector machines [16, 17] have been extended for semi-supervised regression [18]. In graph based semi-supervised regression methods labelled and unlabelled data are considered nodes on a graph. Similar nodes are connected with edges along which information propagates [19–21].

3. Twin neural network regression

3.1. Reformulation of the regression problem

We are given a training set of n labelled data points $X^{train} = (x_1^{train}, \ldots, x_n^{train})$ with target values $Y^{train} = (y_1^{train}, \ldots, y_m^{train})$. The goal is to find a function f such that $f(x_i) = y_i$, which generalizes to unseen data X^{test} to make an accurate prediction for the labels Y^{test}. TNNR aims to solve a reformulation of the original regression problem, see figure 1. Given a pair of data points $(x_i^{train}, x_j^{train})$ we train a neural network to find a function F to predict the difference:

$$F(x_i, x_j) = y_i - y_j. \tag{1}$$

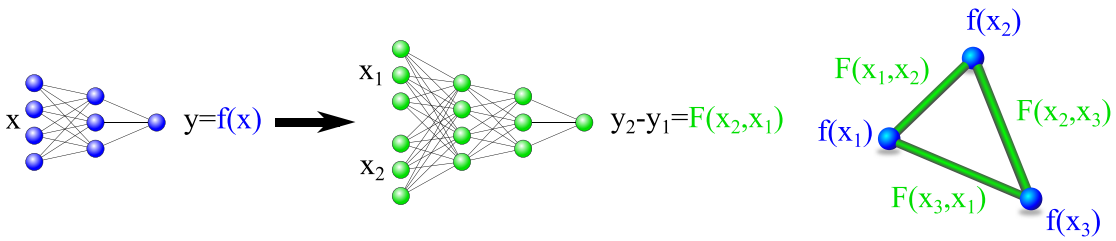

Figure 1. TNNR formulation of a regression problem: in order to solve a traditional regression problem, a neural network is trained to map a data point x to its target value $f(x) = y$. A TNN takes two inputs x_1 and x_2 and is trained to predict the difference between the target values $F(x_2, x_1) = y_2 - y_1$. This difference can be employed as an estimator for $y_2 = F(x_2, x_1) + y_1$ given a labelled anchor point (x_1, y_1). In contrast to a traditional neural network f, a TNN F must satisfy loop consistency, predictions along each loop sum to zero: $F(x_1, x_2) + F(x_2, x_3) + F(x_3, x_1) = 0$.

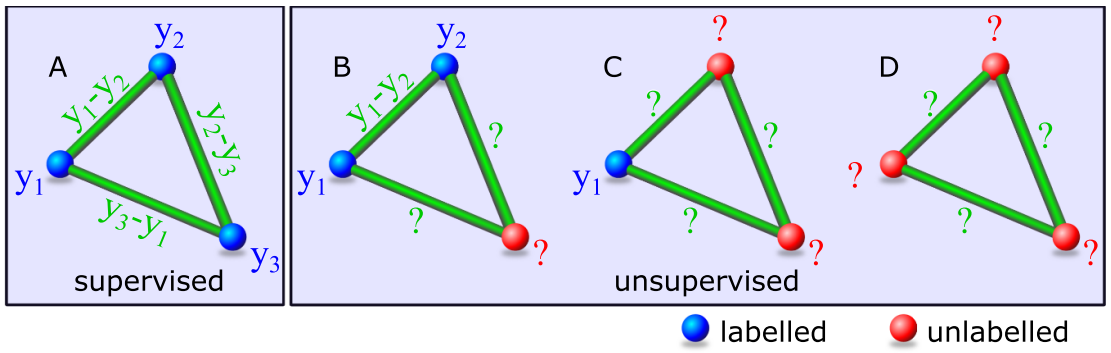

Figure 2. Training Data: A triple of labelled data points (x_1, x_2, x_3) with labels (y_1, y_2, y_3) translates to a TNNR training loop which has labels $(y_1 - y_2, y_2 - y_3, y_3 - y_1)$, depicted as loop A. Loop A is used as supervised training data for TNNR with the objective to minimize the MSE between the TNN outputs and each label of loop A. In loops B, C and D at least one data point is unlabelled. On these loops, TNNR is trained in unsupervised manner to enforce loop consistency $F(x_1, x_2) + F(x_2, x_3) + F(x_3, x_1) = 0$.

The TNN F provides a solution to the original regression problem of predicting y_i by evaluating $y_i = F(x_i, x_j) + y_j$, where (x_j, y_j) is an *anchor* whose label is known. Every training data point $x_j^{train} \in X^{train}$ can be employed as anchor. Since ensembles of predictions are more accurate than single predictions, the best estimate for the solution of the original regression problem is obtained by averaging:

$$\begin{aligned} y_i^{pred} &= \frac{1}{m}\sum_{j=1}^{m}\left(F(x_i, x_j^{train}) + y_j^{train}\right) \\ &= \frac{1}{m}\sum_{j=1}^{m}\left(\frac{1}{2}F(x_i, x_j^{train}) - \frac{1}{2}F(x_j^{train}, x_i) + y_j^{train}\right). \end{aligned} \tag{2}$$

This ensemble contains twice the size of the training set of predictions of differences $y_i - y_j$ for a single prediction of y_i.

Since training data points are not just separated by infinitesimal perturbations the created ensemble is more diverse than a pseudo ensemble generated through infinitesimal perturbations of model weights [22]. Ensembles of different TNN models, each containing an implicit ensemble of predictions themselves is even more accurate than an implicit ensemble from a single TNN [8].

3.2. Semi-supervised learning on loops

If we have in addition to a labelled training data set of size m an unlabelled data set of size n available during the training phase, TNNR can be used as a semi-supervised regression method. TNNR can be trained on unlabelled data points which are connected along loops, see figure 2. Loops consist of several data points which can be labelled or unlabelled, the corresponding loop labels express the differences between the target labels of each original data point. For simplicity, we restrict ourselves to loops of size three. Since all higher order loops can be combined from loops of size three, we do not believe they would lead to an improved performance. In our case, a loop can be expressed as triple (x_i, x_j, x_k) with data labels (y_i, y_j, y_k) and corresponding loop labels $(y_i - y_j, y_j - y_k, y_k - y_i)$, if a label of a data point is unknown during the training

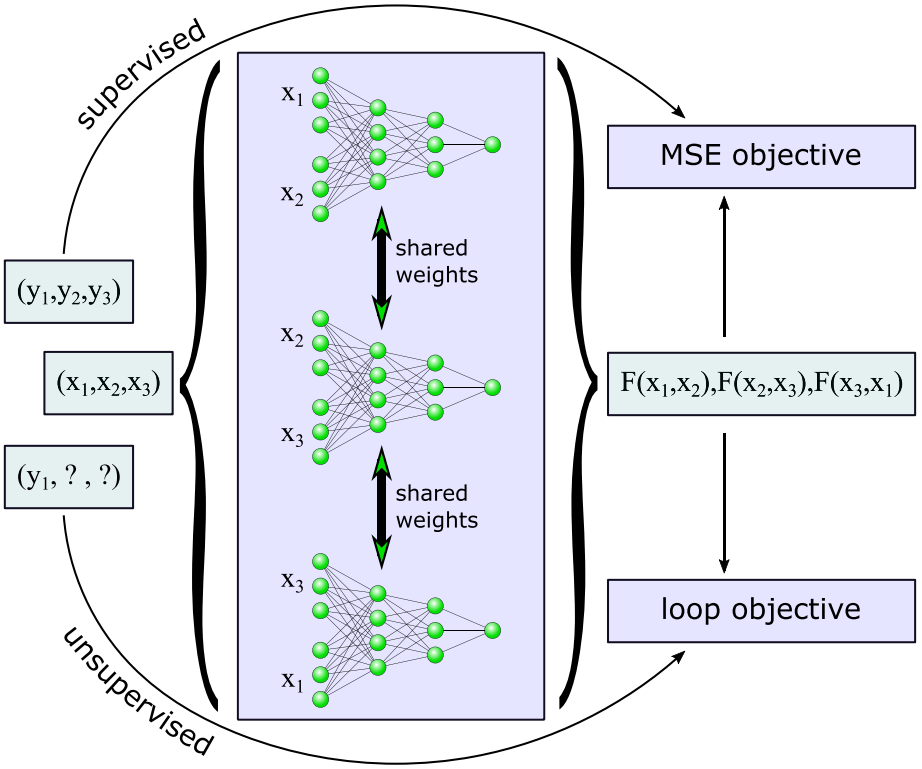

Figure 3. Training architecture: In order to train TNNR in a semi-supervised manner, three copies of the main TNN are embedded in the training architecture. The inputs are loops, meaning triples of data points (x_1, x_2, x_3) which are distributed among the three TNNs to predict differences between the target values $(y_1 - y_2, y_2 - y_3, y_3 - y_1)$. If training data is labelled, the three TNNs are optimized through the MSE objective; if the training data is unlabelled or partially labelled, the three TNNs are optimized through the loop objective. TNNs are trained simultaneously on both labelled and unlabelled/partially labelled loops contained in the same training batch.

phase the corresponding loop labels are left blank. As long as the labels are known, each pair of original data points (x_i, x_j) drawn from the triple can be used to train the TNN to predict the difference $y_i - y_k$. Furthermore, predictions on along each loop containing (x_i, x_j, x_k) points should satisfy the loop consistency condition:

$$0 = F(x_i, x_j) + F(x_j, x_k) + F(x_k, x_i). \tag{3}$$

This condition can be used in order to train the TNN in an unsupervised manner since it does not require the presence of any labels. As shown in figure 2 we consider four types of loops. We only use loops where all x_i have a label y_i as supervised training data (loop A). If at least one of the x_i is unlabelled (loops B, C, D) we choose to use the loop as unsupervised training data. TNNR cannot be used as a purely unsupervised regression algorithm, some labelled data points must be contained in the training set.

3.3. Training architecture

The architecture used to train the TNN is shown in figure 3. Three copies of the main TNN are trained simultaneously. During training these networks share their weights. The input of the full architecture are loops along triples (x_i, x_j, x_k), which can either be fully labelled or unlabelled/partially labelled. The input of the three TNNs are the three possible pairs drawn from the loop, so that the prediction of the full training model is a triple $(F(x_i, x_j), F(x_j, x_k), F(x_k, x_i))$. Depending on the loop type figure 2 of the input, labelled loops (loop A) are used in the mean squared error loss function (MSE objective) to train the TNN to predict the differences between each data label $(y_i - y_j, y_j - y_k, y_k - y_i)$. Unlabelled loops (loops B, C, D) are used to train the TNN to enforce loop consistency (3). For this purpose the weights and biases in the TNNs are updated via gradient descent by minimizing the batchwise estimate of:

$$loss_{MSE} = \frac{1}{n^2} \sum_{ij} \left(F(x_i, x_j) - (y_i - y_j) \right)^2, \tag{4}$$

if the loops are labelled and:

$$loss_{loop} = \frac{1}{(m+n)^3} \times \sum_{ijk} \big(F(x_i,x_j) + F(x_j,x_k) + F(x_k,x_i)\big)^2, \tag{5}$$

if the loops are unlabelled or partially labelled. The loop loss function can be seen as a MSE objective with noisy labels for each of the $F(x_i,x_j)$ provided by the other two predictions $y_i - y_j \approx -F(x_j,x_k) - F(x_k,x_i)$. The combined loss function is:

$$loss = loss_{MSE} + \Lambda\, loss_{loop}, \tag{6}$$

where the optimal loop weight Λ is a hyperparameter that needs to be optimized by evaluating the loss on a validation set. Each update step containing a number of gradient signals equal to the batch size b is constructed from $b/2$ labelled data pairs indexed by $\mathcal{P}$ and $b/2$ loops each with at least one unlabelled data point indexed by $\mathcal{L}$:

$$loss_{mini-batch} = \frac{2}{b}\sum_{ij\in\mathcal{P}} \big(F(x_i,x_j) - (y_i - y_j)\big)^2 + \frac{2\Lambda}{b}\sum_{ijk\in\mathcal{L}} \big(F(x_i,x_j) + F(x_j,x_k) + F(x_k,x_i)\big)^2. \tag{7}$$

Our TNNs all consist of two hidden layers, each with 128 neurons in these layers. Each hidden layer uses rectified linear units as activation functions while the output is a linear neuron. We do not employ any regularization. We note, it is very inconvenient to store all possible loop combinations as input data for the training architecture. For this reason, we employ a generator which randomly samples all possible loops containing all original data points, labelled and unlabelled. Each batch generated by the generator contains the same number of supervised loops and unsupervised loops. The training batch size is set to 16, where half of the batch is used for supervised learning and the other half for unsupervised learning. We train for 2000 epochs using the adadelta optimizer. We employ learning rate decay callbacks and early stopping callbacks stop the training if the validation loss stops improving.

For labelled and unlabelled data set sizes of m and n, respectively, the time complexity estimate for traditional neural networks scales like $\mathcal{O}(m)$ for training and $\mathcal{O}(1)$ for inference. This translates to a training time complexity for TNNs of $\mathcal{O}((m+n)^2)$, since the number of possible differences scales quadratically with the number original data points. Since the loop loss function behaves similar to the MSE loss function with noisy labels, we expect quadratic scaling in both supervised and semi-supervised training. Ensembling the predictions for the differences between all labelled training data points and a new data point during inference time scales like $\mathcal{O}(m)$. For training and inference only a subset of all possible differences can be sampled to reduce time at the cost of accuracy.

Let us discuss if it is possible to relate TNNR to the existing classes of semi-supervised regression methods which are co-training, kernel or graph based [11]. TNNR cannot be classified into either of these classes; however, some ideas might appear similar. In a vague way, it is possible to argue that TNNR is similar to co-training since in the loop loss function two predictions act as a noisy label for the third prediction. Kernel methods make use of some sort of similarity kernel function, TNNR provides a similarity measure in the label space through (1). Further, TNNR can be connected to graph based methods, since the TNNR training loops are minimal sub graphs containing three nodes and three edges.

4. Experiments

4.1. Data preparation

We examine the performance of TNNR on different regression data sets: bio conservation (BC), BH, concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), red wine quality (WN), test function (TF), red WN, Wheatstone bridge (WSB) and yacht hydrodynamics (YH). The common data sets can be found online at [23] and their structures are listed in table A1. The scientific data sets are simulations of mathematical equations and physical systems. TF is a polynomial function combined with a sine function. RCL is a simulation of the electric current in an RCL circuit and WSB a simulation of the WSB voltage.

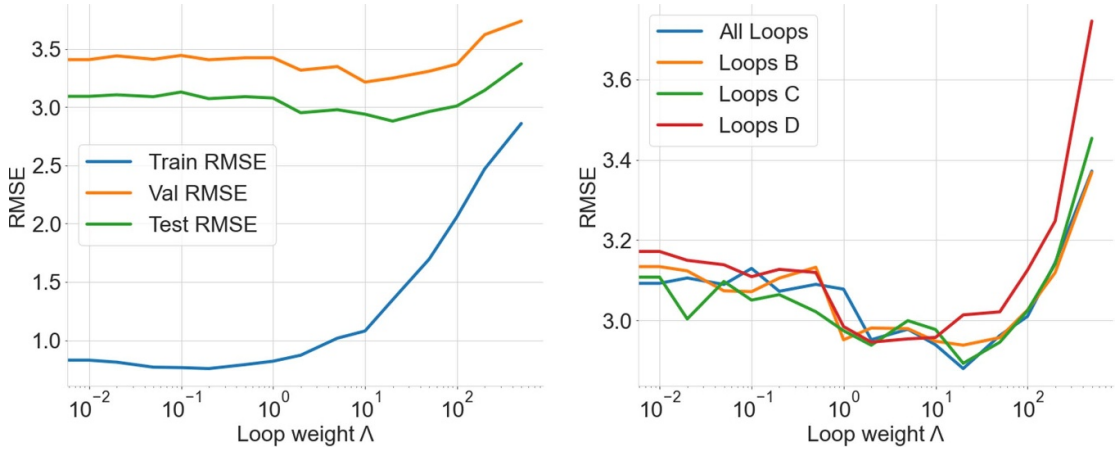

Figure 4. Effect of semi-supervised training on RMSE at the example of the Boston Housing data set. The loop weight denotes the strength of the loop constraint versus the MSE loss function, see (6). Left: increasing the loop weight Λ to a suitable value causes the validation and test RMSE to decrease while at the same time the train RMSE increases. Right: different loop types reduce the test RMSE in a similar manner. All loops together perform slightly better.

We perform two different experiments to examine TNNR as transductive and inductive semi-supervised learning algorithm. To focus solely on transductive learning, in the first experiments all data is split into 80% labelled training, 10% unlabelled validation and 10% unlabelled test data, all of which is used during the training phase. To examine inductive learning in addition to transductive learning we perform a second experiment with 30% labelled training, 50% unlabelled validation and 10% unlabelled test data, which is used during training and 10% unlabelled data on which the performance of inductive semi-supervised learning is evaluated. We normalize and centre the input features to a range of $[-1, 1]$ based on the training data. As outlined in section 3.3, labelled and unlabelled data are combined to form loops, which are supplied to the training architecture figure 3 through a generator from which all possible loops are sampled. All resulting RMSEs in the figures and tables are produced by repeating the experiment 25 times which random splits of the underlying data, the same random splits are used while varying the hyperparameter Λ.

4.2. Experiment 1: transductive semi-supervised learning

The first experiment focuses on transductive semi supervised learning, where the goal is to predict labels of unlabelled data which is present during the training phase. First, for the example of the BH data set, we include all possible loop types depicted in figure 2 into the training data and examine the effect on the RMSEs of the training, validation and test sets. In figure 4, we can see that the validation and test set RMSE can be lowered by tuning the loop weight Λ to a certain finite value. However, at that point the training RMSE increases above its base value. This indicates that including the loops containing unlabelled data points is similar to regularization and reduces overfitting to the labelled training set. Secondly, we determine the effects of including several loop types into the training phase at the example of the BH data set. In figure 4 we can see that all loops have a positive effect on the RMSE, however, some loops seem to have a slightly better effect than others. Loops B and C tend to reduce the RMSE more than Loop D. This is most likely because Loops B and C contain the final predictions of differences between elements of the labelled training set and the unlabelled test set. Based on this figure we choose to train on all possible loops in all further experiments.

The results of semi-supervised transductive learning for different data sets are presented in table 1 where they are compared with purely supervised results. Supervised learning is equivalent to setting the loop weight $\Lambda = 0$. The transductive results are produced choosing the optimal loop weight $\Lambda \in [0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100, 200, 500]$ based on the validation RMSE. The RMSE curves are visualized in figure C1. Due to noise, sometimes the optimal choice of Λ based on the validation RMSE slightly differs from the optimum for the test RMSE. In some cases (CS, WSB, YH) this prevents us from uniquely identifying an optimal finite Λ. In these cases we cannot claim an improvement of semi-supervised learning over purely supervised learning. In some data sets the RMSE shows two local minima as a function of Λ, this effect is separate from noise and can be reproduced, but we do not know of the cause of this observation.

On 6 out of 9 data sets the test RMSE can be (statistically) significantly reduced by semi-supervised transductive learning. An interesting observation is that the data sets which did not benefit from internal TNNR ensembling in the original TNNR paper [8], namely BC and red WN, are among those that benefit most from semi-supervised learning. That means TNNR ensembling combined with semi-supervised

Table 1. Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), Boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), red wine quality (WN), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 100% of the available data where 80% is labelled training data, 10% is unlabelled validation data and 10% is unlabelled test data whose labels are predicted using TNNR as a transductive semi-supervised learning method.

	80% labelled training data		
	Supervised	Transductive	Gain
BC	0.8121±0.0219	0.7221±0.0178	11.1%
BH	3.0716±0.1678	2.9205±0.1642	4.9%
CS	4.3043±0.1476	4.3043±0.1476	0.0%
EE	0.6143±0.0236	0.5647±0.0179	8.1%
RCL	0.0145±0.0002	0.0143±0.0002	1.3%
TF	0.0044±0.0002	0.0038±0.0002	13.6%
WN	0.7037±0.0100	0.6334±0.0084	10.0%
WSB	0.0214±0.0011	0.0214±0.0011	0.0%
YH	0.5597±0.0592	0.5597±0.0592	0.0%

Table 2. Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), Boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 90% of the available data where 30% is labelled training data, 10% is unlabelled validation data and 50% is unlabelled test data whose labels are predicted using TNNR as a transductive semi-supervised learning method. The labels of the 10% of the data which was not used during training are inferred using TNNR as an inductive semi-supervised learning method.

	30% labelled training data					
	Supervised	Transductive	Gain	Supervised	Inductive	Gain
BC	0.9382±0.0137	0.7960±0.0056	15.2%	0.8996±0.0280	0.7721±0.0175	14.2%
BH	4.1357±0.1229	3.8228±0.0951	7.6%	3.6830±0.2337	3.5521±0.2281	3.6%
CS	6.0777±0.0773	5.9088±0.0616	2.8%	6.0467±0.1412	6.0905±0.1260	−1.0%
EE	1.5084±0.0317	1.4194±0.0409	5.9%	1.4794±0.0416	1.3902±0.0459	6.0%
RCL	0.0200±0.0003	0.0194±0.0003	3.0%	0.0203±0.0004	0.0195±0.0004	3.9%
TF	0.0066±0.0004	0.0063±0.0004	4.5%	0.0064±0.0004	0.0059±0.0004	7.8%
WN	0.7841±0.0047	0.6511±0.0027	17.0%	0.7868±0.0087	0.6534±0.0075	17.0%
WSB	0.0341±0.0012	0.0341±0.0012	0.0%	0.0368±0.0018	0.0368±0.0018	0.0%
YH	1.2203±0.0616	1.2203±0.0616	0.0%	1.1170±0.0910	1.1170±0.0910	0.0%

training on loops improved the RMSE on all considered data sets significantly compared to previous state-of-the art regression methods.

4.3. Experiment 2: transductive and inductive semi-supervised learning

In the second experiment, the goal is to compare transductive and inductive semi-supervised learning using TNNR. The transductive RMSEs are evaluated on the unlabelled test data which was supplied to the training phase, while the inductive RMSEs are evaluated on the unlabelled test set which was not contained in the training loops. Further, the validation set was part of the training loops. In table 2 the improvements due to semi-supervised training are displayed, while the RMSE curves can be found in figures C2 and C3. From this table we conclude that TNNR can be successfully used as a transductive or inductive semi-supervised learning method.

4.4. Comparison with co-training

We compare the results of semi-supervised TNNR with co-training of traditional feed-forward neural networks. Co-training is the most similar semi-supervised learning algorithm since it augments data with proposed labels from different predictors and the underlying algorithm can be chosen to be neural networks as it is the case in TNNR. We describe the co-training procedure and its results in appendix B in tables B1 and B2. Let us first note that in order to use co-training we require the training of an ensemble of neural networks on different bootstrapped data sets. In the case of TNNR we only use a single neural network, however using an ensemble of TNNRs would of course also increase its predictive power [8]. Setting aside hyperparameter tuning, that means that during co-training we train typically around 100–1000 neural networks for each co-training run while for TNNR we only train one single neural network.

Semi-supervised learning with TNNR improves the RMSE by an average of 5.4% in transductive learning with 80% of the data labelled, 6.2% in transductive learning with 30% of the data labelled and 5.7% in inductive learning with 30% of the data labelled. Semi-supervised learning with Co-training improves the

RMSE by an average of 0.4% in transductive learning with 80% of the data labelled, 1.4% in transductive learning with 30% of the data labelled and 1.0% in inductive learning with 30% of the data labelled. In 23 of 27 experiments the final RMSE of TNNR was lower while in 4 of 27 experiments the final co-training RMSE was lower. While these results are supporting the power of semi-supervised TNNR it is important to look at where these results were achieved. Co-training tends to perform better in very low dimensional data sets, in [13] co-training achieved double digit percentage improvements only on one dimensional data sets. While we do not explore one dimensional data sets, we observe a stronger improvement through co-training in the two-dimensional TF data set. We also note that the TNNR performance boost is very pronounced in only three of our data sets by achieving a double digit performance gain.

5. Summary

We have presented a method to train TNNR in a semi-supervised manner. While TNNR is already a state of the art regression method, we observe a significant improvement by semi supervised training on loops containing differences between labelled and/or unlabelled data. On 6 out of 9 data sets the test RMSE can be significantly reduced by semi-supervised learning. Data sets which do not benefit from internal ensembling [8] benefit among the most from semi-supervised learning. Thus, TNNR ensembling combined with semi-supervised training on loops beat all considered supervised state-of-the-art methods on all considered data sets. This outperformance is persistent no matter if it is achieved via transductive or inductive semi-supervised learning. The effective cost of TNNR compared to traditional supervised neural networks is the quadratic time scaling of the former versus the linear time scaling of the latter.

It might be possible to improve the results by continuously increasing the loop weight Λ during training. It is very likely that certain specific loops improve the performance more than others, after identifying these loops the learning process could be weighted in their favour. The increased performance measured by the RMSE comes largely from data points in high density regions figure E4, an increased weighting of data points in low density regions might also increase the performance there. While in this work only the training process is assisted by loops, it might be fruitful to explore if the loops can help during the inference phase. Neural networks tend to be composed of custom building blocks using standard libraries that enable a straightforward implementation of the semi-supervised twinned training. It is possible to implement a similar semi-supervised training method for other regression, however, this would most likely require a rewriting of these algorithms beyond using standard libraries.

The code supporting this publication is available at [23, 24].

Data availability statement

The data that support the findings of this study are available upon reasonable request from the authors.

Acknowledgments

Research at Perimeter Institute is supported in part by the Government of Canada through the Department of Innovation, Science and Economic Development Canada and by the Province of Ontario through the Ministry of Economic Development, Job Creation and Trade. We thank the National Research Council of Canada for their partnership with Perimeter on the PIQuIL. R G M and I T acknowledge NSERC. R G M is supported by the Canada Research Chair Program. We also acknowledge Compute Canada for computational resources.

Appendix A. Data sets

Table A1. Data sets.

Name	Key	Size	Features	Type
Bio concentration	BC	779	14	Discrete, continuous
Boston housing	BH	506	13	Discrete, continuous
Concrete strength	CS	1030	8	Continuous
Energy efficiency	EF	768	8	Discrete, continuous
RCL circuit current	RCL	4000	6	Continuous
Test function	TF	1000	2	Continuous
Red wine quality	WN	1599	11	Discrete, continuous
Wheatstone bridge voltage	WSB	200	4	Continuous
Yacht hydrodynamics	YH	308	6	Discrete

The test function (TF) data set created from the equation:

$$F(x_1, x_2) = x_1^3 + x_1^2 - x_1 - 1 + x_1 x_2 + \sin(x_2), \tag{A1}$$

and zero noise.

The output in the RCL circuit current data set (RCL) is the current through an RCL circuit, modelled by the equation:

$$I_0 = V_0 \cos(\omega t)/\sqrt{R^2 + (\omega L - 1/(\omega C))^2}, \tag{A2}$$

with added Gaussian noise of mean 0 and standard deviation 0.1.

The output of the Wheatstone Bridge voltage (WSB) is the measured voltage given by the equation:

$$V = U(R_2/(R_1 + R_2) - R_3/(R_2 + R_3)), \tag{A3}$$

with added Gaussian noise of mean 0 and standard deviation 0.1.

Appendix B. Co-training

In order to compare the TNNR results to the most similar baseline we train traditional neural networks with co-training [12–15]. In our case the co-training procedure trains an ensemble of 10 neural networks. Each of these networks have an architecture similar to the TNNs. All consist of two hidden layers, each with 128 neurons in these layers. Each hidden layer uses rectified linear units as activation functions while the output is a linear neuron. We do not employ any regularization and train on batches of size 16. We train for 10 000 epochs using the adadelta optimizer with learning rate decay unless early stopping callbacks stop the training if the validation loss stops improving. The explicit variant of our co-training variant is described in algorithm 1. The bootstrapping procedure is implemented by randomly sampling from a data set with replacement in addition to maintaining a weighting of at least 1 for every data instance.

Algorithm 1. Co-training for regression.

Input: Ensemble members E
stepwise data size D
validation threshold $T_{val} = \infty$
Data: Labelled data set $L = (X_{train,L}, Y_{train,L})$
Unlabelled data set $U = (X_{train,U})$
Validation set $V = (X_{val}, Y_{val})$
L_B =bootstrap E different datasets from L
initialize E different machine learning models $M = (m_1, \ldots, m_E)$
while $length(U) \geqslant D$ **do**
  for i in $[1, \ldots, E]$ **do**
    train $M[i]$ on $L_B[i]$
  $P_{val} = [M[i](X_{val}), \ldots, M[i](X_{val})]$
  $P_U = [M[i](X_{train,U}), \ldots, M[i](X_{train,U})]$
  $P_{val,av}$ =mean across ensemble members (P_{val})
  $P_{U,av}$ =mean across ensemble members (P_U)
  $P_{U,var}$ =variance across ensemble members(P_U)
  Π = (D elements in U with smallest variance in $P_{U,var}$)
  $U = U - \Pi$
  for i in $[1, \ldots, E]$ **do**
    $L_B[i] = L_B[i]$+bootstrap$(\Pi, P_{U,av}[\Pi])$
  if $RMSE(P_{val,av}, Y_{val}) \geqslant T_{val}$ **then**
    break
  else
    $T_{val} = RMSE(P_{val,av}, Y_{val})$
    $M_{final} = M$
Output: Trained Models $M_{final} = (m_1, \ldots, m_E)$

Table B1. Co-Training: Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), Boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), red wine quality (WN), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 100% of the available data where 80% is labelled training data, 10% is unlabelled validation data and 10% is unlabelled test data whose labels are predicted using co-training as a transductive semi-supervised learning method.

	80% labelled training data		
	Supervised	Transductive	Gain
BC	0.7247±0.0162	0.7359±0.0175	−1.5%
BH	2.9448±0.1726	2.9572±0.1697	−0.4%
CS	4.7400±0.1234	4.6505±0.1217	1.9%
EE	0.7623±0.0273	0.7528±0.0256	1.3%
RCL	0.0146±0.0001	0.0146±0.0001	0.0%
TF	0.0036±0.0004	0.0035±0.0004	2.9%
WN	0.6417±0.0092	0.6375±0.0089	0.7%
WSB	0.0338±0.0015	0.0338±0.0015	0.0%
YH	0.5339±0.0601	0.5396±0.0622	−1.0%

Table B2. Co-training: Best estimates for test RMSEs belonging to different data sets. Our confidence on the RMSEs is determined by their standard error. Data sets: bio conservation (BC), Boston housing (BH), concrete strength (CS), energy efficiency (EE), RCL circuit (RCL), test function (TF), red wine quality (WN), Wheatstone bridge (WSB) and yacht hydrodynamics (YH). We train on 90% of the available data where 30% is labelled training data, 10% is unlabelled validation data and 50% is unlabelled test data whose labels are predicted using co-training as a transductive semi-supervised learning method. The labels of the 10% of the data which was not used during training are inferred using co-training as an inductive semi-supervised learning method.

	30% labelled training data					
	Supervised	Transductive	Gain	Supervised	Inductive	Gain
BC	0.8493±0.0104	0.8507±0.0110	−0.2%	0.8021±0.0198	0.8036±0.0200	−0.2%
BH	3.7778±0.0826	3.7567±0.0796	0.6%	3.5291±0.1997	3.4815±0.1894	1.4%
CS	6.1157±0.0537	6.0886±0.0553	0.4%	6.3099±0.1299	6.2510±0.1405	0.8%
EE	1.6834±0.0424	1.6632±0.0407	1.2%	1.6438±0.0496	1.6277±0.0477	1.0%
RCL	0.0253±0.0003	0.0252±0.0003	0.4%	0.0256±0.0004	0.0256±0.0004	0.0%
TF	0.0113±0.0006	0.0110±0.0010	2.7%	0.0102±0.0006	0.0100±0.0010	2.0%
WN	0.7021±0.0047	0.6976±0.0047	0.6%	0.6945±0.0092	0.6921±0.0089	0.3%
WSB	0.0584±0.0015	0.0569±0.0013	2.6%	0.0601±0.0029	0.0591±0.0024	1.7%
YH	1.8360±0.0970	1.7624±0.0925	4.2%	1.6621±0.1140	1.6310±0.1179	1.9%

Appendix C. Analysis of semi-supervised TNNR

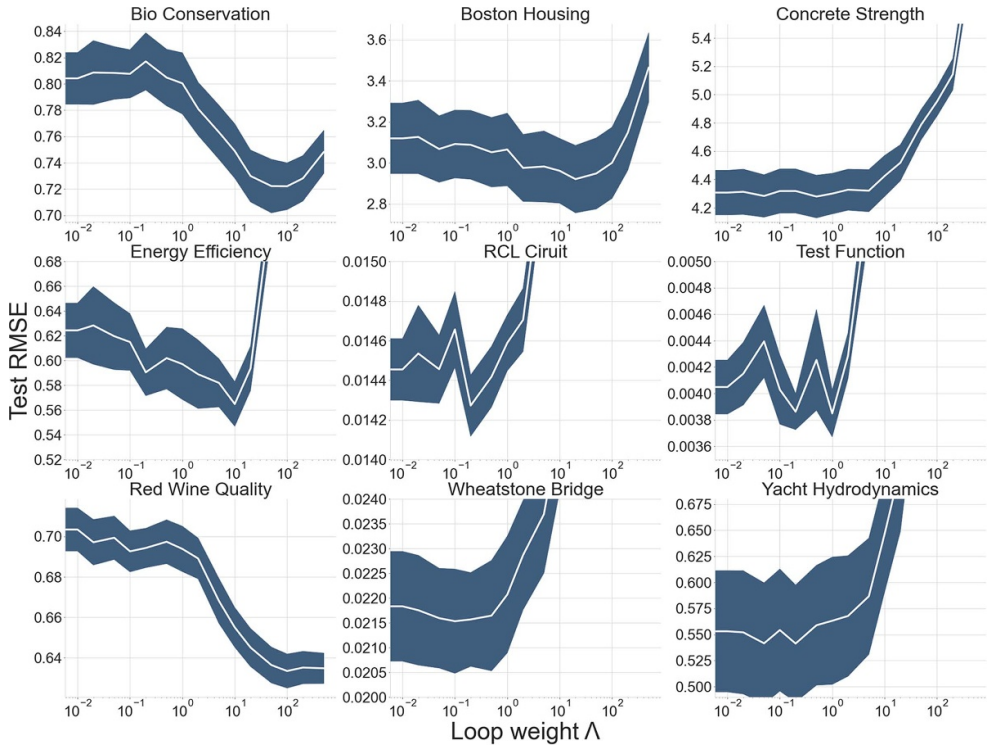

Figure C1. Effect of transductive semi-supervised training on RMSEs of different data sets. We train on 80% labelled training data, 10% unlabelled validation data and 10% is unlabelled test data. The loop weight denotes the strength of the loop constraint versus the MSE loss function, see (6). The blue area is indicative for the standard error.

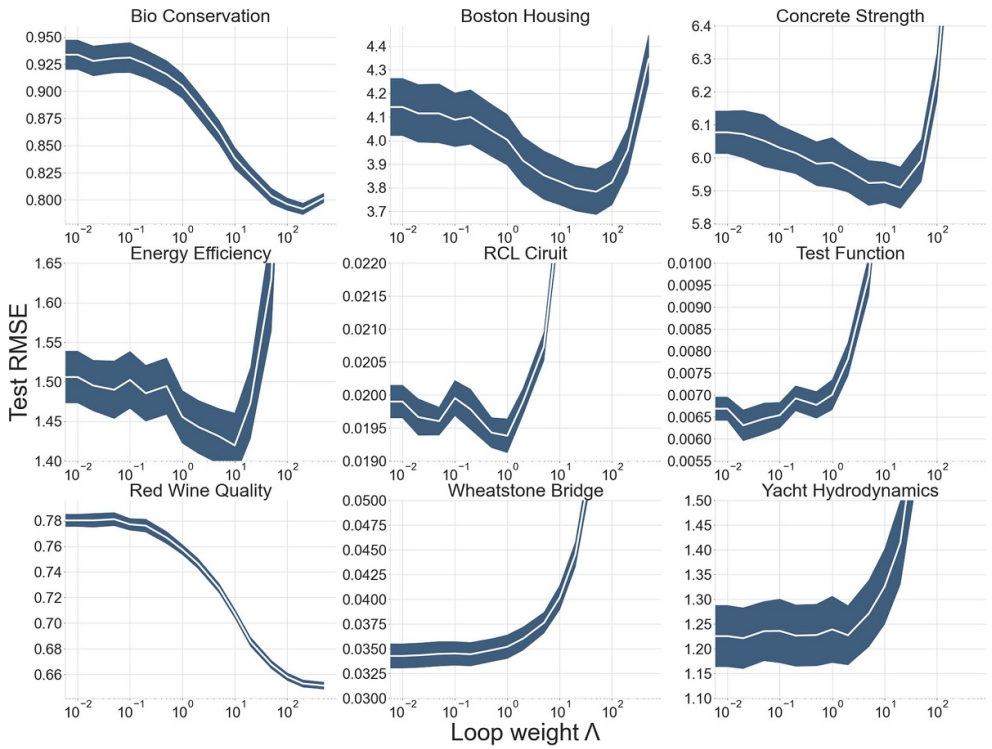

Figure C2. Effect of transductive semi-supervised training on RMSEs of different data sets. We train on 30% labelled training data, 10% unlabelled validation data and 50% is unlabelled test data, whose targets are predicted using TNNR as a transductive semi-supervised learning method. The remaining 10% unlabelled test data is not used for training, its labels are predicted using TNNR as an inductive semi-supervised learning method. The loop weight denotes the strength of the loop constraint versus the MSE loss function, see (6). The blue area is indicative for the standard error.

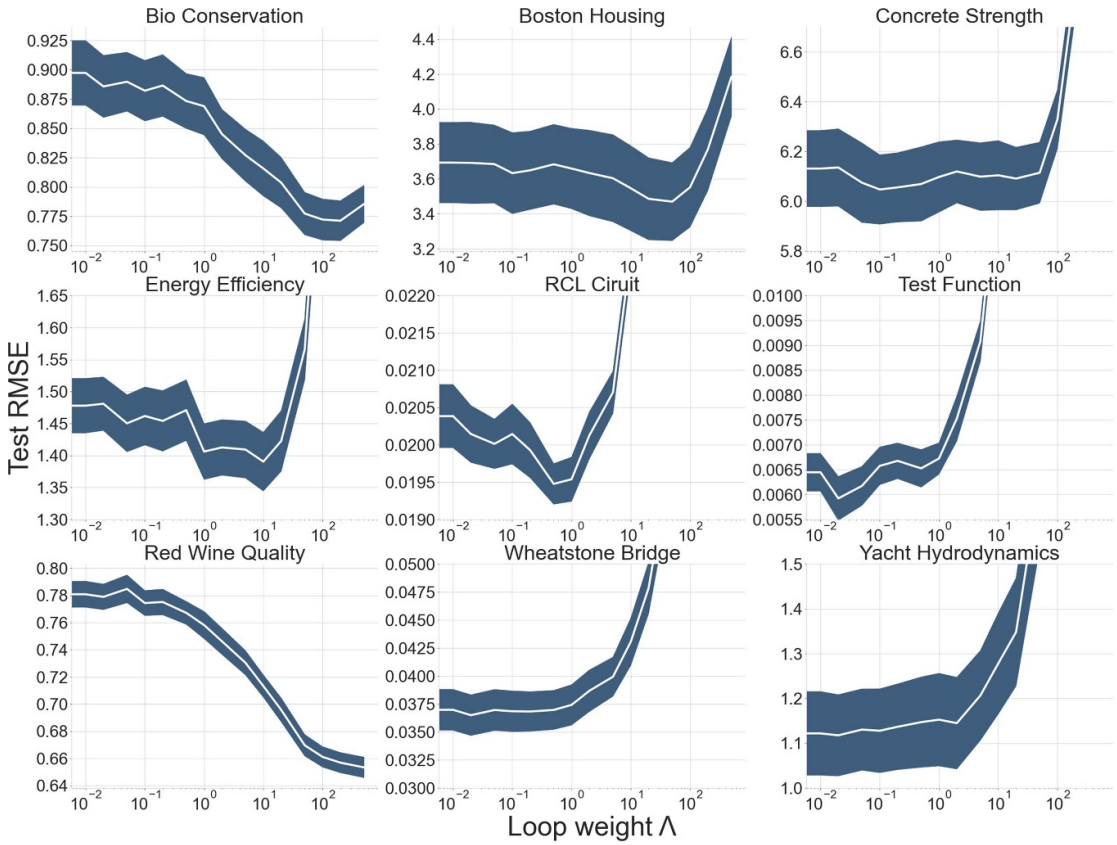

Figure C3. Effect of inductive semi-supervised training on RMSEs of different data sets. We train on 30% labelled training data, 10% unlabelled validation data and 50% is unlabelled test data, whose targets are predicted using TNNR as a transductive semi-supervised learning method. The remaining 10% unlabelled test data is not used for training, its labels are predicted using TNNR as an inductive semi-supervised learning method. The loop weight denotes the strength of the loop constraint versus the MSE loss function, see (6). The blue area is indicative for the standard error.

Appendix D. Bias-variance tradeoff and ensembles

In a regression problem, one is tasked with finding the true labels on yet unlabelled data points through the estimation of a function $f(x) = y$. Given a finite training data set D we denote this approximation $\hat{f}(x;D)$. The expected mean squared error can be decomposed by three sources of error, bias error $\mathrm{Bias}_D[\hat{f}(x;D)]$, variance error $\mathrm{Var}_D\left[\hat{f}(x;D)\right]$ and intrinsic error of the data set σ:

$$\mathrm{MSE} = \mathrm{E}_x\left\{\mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[\hat{f}(x;D)\right]\right\} + \sigma^2. \tag{D1}$$

If we replace the estimator by an ensemble of two functions $\hat{f}(x;D) = 1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)$, each exhibiting the same bias and variance as the original estimator, then we can decompose the MSE:

$$\mathrm{MSE} = \mathrm{E}_x\left\{\mathrm{Bias}_D[1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)]^2 + \mathrm{Var}_D\left[1/2\hat{f}_A(x;D) + 1/2\hat{f}_B(x;D)\right]\right\} + \sigma^2 \tag{D2}$$

$$= \mathrm{E}_x\left\{\mathrm{Bias}_D[\hat{f}(x;D)]^2 + \mathrm{Var}_D\left[1/2\hat{f}_A(x;D)\right] + \mathrm{Var}_D\left[1/2\hat{f}_B(x;D)\right]\right. \tag{D3}$$

$$\left. + 2\,\mathrm{Cov}_D\left[1/2\hat{f}_A(x;D), 1/2\hat{f}_B(x;D)\right]\right\} + \sigma^2 \tag{D4}$$

$$= \mathrm{E}_x\left\{\mathrm{Bias}_D[\hat{f}(x;D)]^2 + 1/2\,\mathrm{Var}_D\left[\hat{f}_A(x;D)\right] + 1/2\,\mathrm{Cov}_D\left[\hat{f}_A(x;D), \hat{f}_B(x;D)\right]\right\} + \sigma^2. \tag{D5}$$

The more uncorrelated $\hat{f}_A(x;D)$ and $\hat{f}_B(x;D)$ are, the smaller is the ratio between variance and covariance. Thus an ensemble consisting of weakly correlated ensemble members reduce the MSE by circumventing the bias-variance tradeoff. By induction this argument extends to larger ensemble sizes.

Appendix E. Density based performance

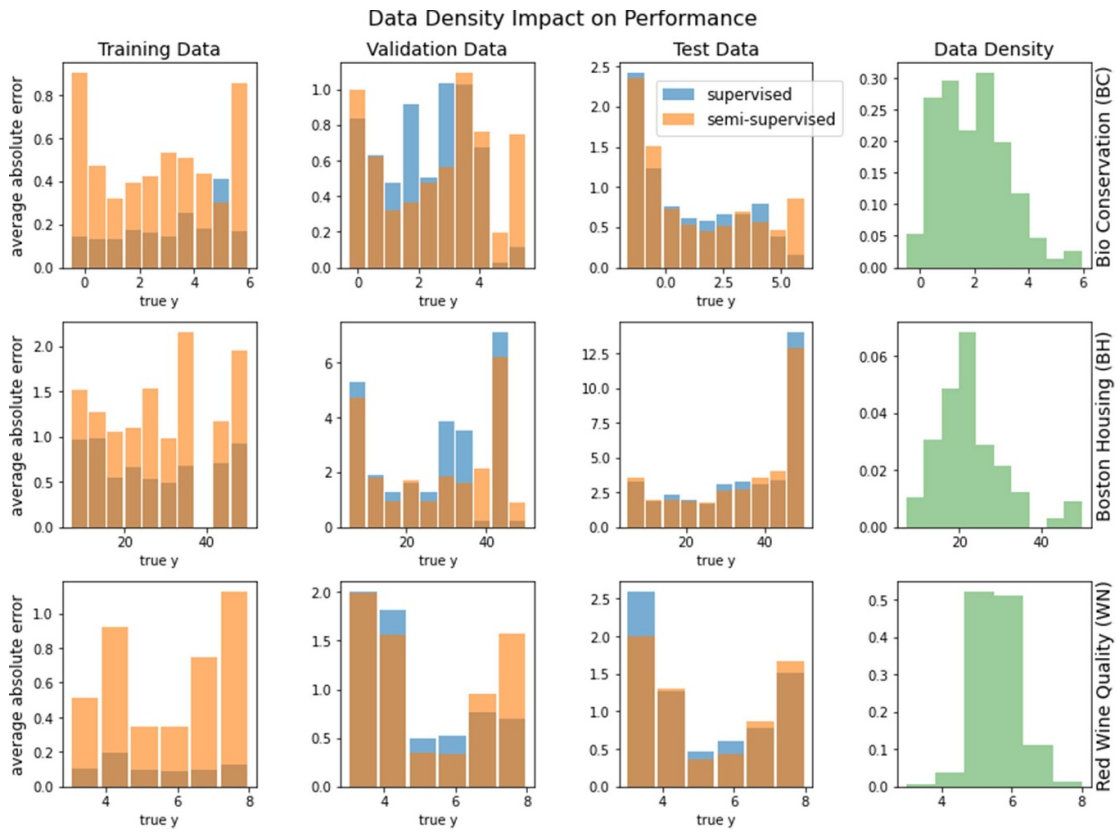

Figure E4. Differential performance of semi-supervised regression. After ordering and binning the true labels y we have averaged the absolute errors in each bin and compared them to the data set density. One can observe the largest performance increases of semi-supervised learning over supervised learning in high-density regions.

ORCID iDs

Sebastian J Wetzel https://orcid.org/0000-0002-2939-9081
Roger G Melko https://orcid.org/0000-0002-5505-8176
Isaac Tamblyn https://orcid.org/0000-0002-8146-6667

References

[1] Zhu X and Goldberg A B 2009 *Synthesis Lectures on Artificial Intelligence and Machine Learning* vol 3 pp 1–130
[2] Chapelle O, Scholkopf B and Zien A 2009 *IEEE Trans. Neural Netw.* **20** 542
[3] Krizhevsky A, Sutskever I and Hinton G E 2012 *Advances in Neural Information Processing Systems* vol 25 pp 1097–105
[4] Jean N, Xie S M and Ermon S 2018 arXiv:1805.10407
[5] Liu C L and Chen Q H 2020 *IEEE Access* **8** 30001–11
[6] Cybenko G 1989 *Math. Control Signals Syst.* **2** 303–14
[7] Hornik K 1991 *Neural Netw.* **4** 251–7
[8] Wetzel S J, Ryczko K, Melko R G and Tamblyn I 2020 (arXiv:2012.14873)
[9] Bromley J, Guyon I, LeCun Y, Säckinger E and Shah R 1993 *Advances in Neural Information Processing Systems* vol 6 pp 737–44
[10] Baldi P and Chauvin Y 1993 *Neural Comput.* **5** 402–18
[11] Kostopoulos G, Karlos S, Kotsiantis S, Ragos O, Tiwari S, Trivedi M and Kohle M L 2018 *J. Intell. Fuzzy Syst.* **35** 1483–500
[12] Blum A and Mitchell T 1998 Combining labeled and unlabeled data with co-training *Proc. 11th Annual Conf. on Computational Learning Theory* pp 92–100
[13] Zhou Z H and Li M 2005 Semi-supervised regression with co-training *Proc. 19th Int. Joint Conf. on Artificial Intelligence* vol 5 pp 908–13
[14] Wang W and Zhou Z H 2010 A new analysis of co-training *ICML'10: Proc. of the 27th Int. Conf. on Machine Learning* (*June 2010*) pp 1135–42
[15] Hady M F A, Schwenker F and Palm G 2009 Semi-supervised learning for regression with co-training by committee *Int. Conf. on Artificial Neural Networks* (Springer) pp 121–30
[16] Bennett K and Demiriz A 1999 *Advances in Neural Information Processing Systems* pp 368–74

[17] Chapelle O, Sindhwani V and Keerthi S S 2008 *J. Mach. Learn. Res.* **9** 203–33
[18] Xu S, An X, Qiao X, Zhu L and Li L 2011 *J. Inf. Comput. Sci.* **8** 885–92
[19] Zhu X, Ghahramani Z and Lafferty J D 2003 Semi-supervised learning using gaussian fields and harmonic functions *Proc. 20th Int. Conf. on Machine Learning (ICML-03)* pp 912–9
[20] Zhu X and Goldberg A 2006 Semi-supervised regression with order preferences *Technical Report* (University of Wisconsin-Madison Department of Computer Sciences)
[21] Timilsina M, Figueroa A, d'Aquin M and Yang H 2021 *Appl. Soft Comput.* **104** 107188
[22] Bachman P, Alsharif O and Precup D 2014 (arXiv:1412.4864)
[23] UCI 2020 Machine learning repository (available at: https://archive.ics.uci.edu/ml/datasets.php) (Accessed 1 May 2020)
[24] Wetzel S 2022 Public semi-supervised tnnr github repository (available at: https://github.com/sjwetzel/PublicTwinNeuralNetworkRegression)