IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2023.0322000

Deep Learning for Multi-Output Regression using Gradient Boosting

SEYEDSAMAN EMAMI, GONZALO MARTíNEZ-MUÑOZ
Escuela Politécnica Superior, Universidad Autónoma de Madrid, Francisco Tomás y Valiente, 11, 28049 Madrid, Spain.
Corresponding author: Seyedsaman Emami (e-mail: emami.seyedsaman@uam.es).

This work was supported by project PID2022-139856NB-I00 funded by MCIN/ AEI / 10.13039/501100011033 / FEDER, UE and project PID2019-106827GB-I00 / AEI / 10.13039/501100011033 and from the Autonomous Community of Madrid (ELLIS Unit Madrid).

ABSTRACT

This paper presents a novel methodology to address multi-output regression problems through the incorporation of deep-neural networks and gradient boosting. The proposed approach involves the use of dense layers as additive models within the Gradient Boosting framework using an auto transfer learning technique. At each boosting iteration, the deep model is cloned with the already trained layers frozen, and a new dense layer is concatenated to the frozen ones. Subsequently, only the weights of the newly added layer are trained in order to reduce the complexity of the learning task. Each layer is trained on the residuals of the squared loss function from previous iterations, resulting in the creation of a robust sequentially deep-trained neural network ensemble. Our experimental results demonstrate that the proposed approach leads to a significant improvement in the performance of the deep framework, resulting in more accurate predictions and improved model interpretability.

INDEX TERMS Deep Neural Network, Gradient Boosting, Multi-Output Regression

I. INTRODUCTION

Multi-output regression is a machine learning problem where the goal is to predict multiple outputs (or targets) based on the given input space. In contrast, in standard or single-output regression, only a single output is learnt. The history of multi-output regression models can be traced back to the field of statistical regression, where the goal was to model the relationship between a set of inputs and a set of outputs [1]–[3]. Multi-output regression can be used in various applications where there are multiple relevant outputs to be predicted from a set of inputs. For instance, in education area to predict student performance in multiple subjects [4]. Another recent application of multi-output regression is a study that discusses the use of soft sensors for predicting quality-related variables in wastewater [5].

Before the creation of multi-output models, the emphasis was on single-output models, in which one output variable was modeled using a set of input variables. However, as the field of regression evolved, the need for models that can handle multiple outputs or dependent variables, became increasingly interesting. This led to the development of multi-output regression models. Studies conducted by [6], [7] focused on enhancing Support Vector Regression (SVR) for handling multiple outputs. In [6], they extended traditional SVR, by incorporating ideas from the Cokriging method [8] (multi-output edition of Kriging). The authors utilize the equivalent covariance of multi-output to single-output in the Cokriging approach as the kernel for SVR, thereby expanding the feature space. In [7], a locally linear transformation (LLT) approach was developed to define loss functions on the multi-output space. The authors obtain local coordinate systems for each output point through singular value decomposition, and their SVR model is trained by solving a convex quadratic programming problem. Applying a rule-based methodology, a research investigation by [9] introduces the FIRE (Fitted Rule Ensembles) algorithm, designed for multi-output regression. FIRE utilizes a diverse ensemble of regression trees transformed into rule sets, with inclusion of linear terms to address complexities associated with approximating linear dependencies. The optimization of weights for both rules and linear terms is accomplished through a gradient-directed optimization algorithm. In a different approach, two methods proposed by [10] treat the outputs as auxiliary inputs, expanding the input space and thereby qualifying for more effective modeling of the relationships between the outputs. The authors evaluate their approaches using various datasets, demonstrating the significance of their technique in solving

multi-output regression problems. The authors introduce two methods: Stacked Single-Target (SST) and Ensemble of Regressor Chains (ERC). SST is a stacked ensemble that trains numerous single-target regression models and integrates their predictions to produce the final multi-target prediction. On the other hand, ERC is an ensemble method that renders multiple regressor chains and combines their predictions to produce the final multi-output prediction. Each regressor chain is trained on the extended input space, which includes the original features and the target variables from the previously trained regressor chains. This approach qualifies for the modeling of the relationships between the targets. They demonstrate that SST and ERC outperform models such as, Ensembles of Multi-Objective Decision Trees [11], convex multi-task feature learning [12], dirty model for multi-task learning [13], and Random Linear target [14], in terms of prediction accuracy. Furthermore, they prove the robustness of their methods concerning datasets with high dimensionality and target correlation. Yet the SST and ERC models adopt a phased approach to training, wherein each individual output is trained separately rather than training on all targets simultaneously. In other recent multi-output method, a novel approach for training Neural Network (NN) for multi-output regression tasks is proposed [15]. The authors propose using Gradient Boosting (GB) [16] to train a shallow NN in phases, with each phase focusing on a small portion of the network. The final result is obtained by combining the different portions of each phase to conform a single multi-output shallow NN. The authors report that the proposed method outperforms traditional NN training techniques on ten multi-output regression tasks.

The present study introduces a novel methodology termed Gradient Boosted - Deep Neural Network Regression (GB-DNNR). The proposed approach involves the creation of an ensemble of Deep Neural Networks (DNNs) iteratively where each new DNN embeds the previous models. To do so, the previous network is cloned and a newly dense layer is added. In addition, and in order to reduce the complexity of the models, only the parameters of the newly added layer are trained, freezing the weights of the previous layers. This new DNN is then trained on the residuals of previous iterations following the GB framework. This allows the model to account for the complex relationships among outputs. The proposed GB-DNNR method diverges from previously developed models in several ways. Specifically, the GB-DNNR model distinguishes itself from the original GB model [16] mainly in two aspects. First, our proposal can handle multiple regression outputs where the original GB would use a separate training for each individual target. Also, GB was designed to work mainly with shallow decision trees to avoid over-fitting. We show that the proposed auto transfer approach allows GB to use deep NN architectures to extract information from non-linear and intricately complex datasets avoiding over-fitting. The approach presented in [6] integrates Cokriging concepts to leverage output space correlations for addressing multiple outputs in SVR. However, Cokriging works by considering how outputs and input variables are correlated. It assumes that the relationships and connections between different outputs can be represented using spatial closeness and covariances. Unlike multi-output SVR using Cokriging, our approach does not rely on prior assumptions about correlation structures between input and output variables. Likewise, in the study presented by [7], the focus lies on treating the output space as a sub-manifold to integrate geometric structure into the regression process. The authors formulate a convex optimization problem to train the model. In contrast, our proposed model takes a different approach by not explicitly representing the geometric structure of the output space as a sub-manifold. Instead, we define loss functions in the global coordinate of the output space using backpropagation and gradient descent for optimization. On the other hand, SST and ERC methods [10] re-frame the multi-output problem as a collection of single-target problems, with each individual problem predicting one of the target variables. Their technique involves training a multitude of single-output regression models and combining their outputs to produce the final multi-output prediction. Contrariwise, our proposed method simultaneously learns all output variables without modifying the input or output space. In contrast to the work presented in [15], which trains a single shallow NN on width by adding one (or few) hidden neuron(s) at each iterations, the present study proposes a model that is an ensemble of deep models that are trained sequentially on depth by adding one layer per iteration. This allows us to build more complex models that can better adapt to the complexities of multi-output problems. This present investigation also distinguishes itself from prior multi-task inquiries undertaken by [11]–[13]. In the study presented by [11], which involve a comparative analysis of two ensemble models, namely Random Forest and Bagging, specifically designed for multi-output applications, they applied predictive clustering trees (PCTs), where they center on clustering data into hierarchies based on similarity. Furthermore, convex multi-task feature learning, as outlined in [12], emphasizes employing a combination of $L1$ and $L2$ regularizations in multi-task learning. This approach aims to reconstruct the Hilbert space, treating non-convex problems as convex optimizations. The convex multi-task method focuses on identifying shared features, regression parameters, and feature count. The dirty multi-task learning model [13], focuses on explicitly capturing simultaneous-sparsity structures in multi-task regression. It involves estimating block-sparse and element-wise sparse components using two distinct matrices, achieved through solving a convex optimization problem. This approach aims to accommodate various levels of sharedness among tasks. Unlike these multi-task learning studies, our GB-DNNR model takes a fundamentally different approach. Rather than relying on specific orthogonality or linearity constraints, or pre-defined assumptions, our model aims to uncover relationships between input and output spaces, where the neural network base learners have the capability to adapt without any assumption.

This study has two primary objectives. Firstly, it proposes a method that improves the performance of multi-output regression tasks with respect to deep and shallow multi-output NNs. Secondly, this research aims to extend the scope of the GBNN study [15] by conducting a comprehensive exploration and comparison of multi-output regression problems. The proposed approach integrates the combined strengths of GB, NN, and deep networks to tackle multi-output regression tasks. To achieve these objectives, an extensive collection of multi-output datasets will be employed, ensuring a thorough investigation and analysis of the matter.

The present study provides a comprehensive analysis of the proposed methodology. Section II described the proposed method in detail providing its theoretical background. Section III provides a thorough overview of the utilized dataset, conducts exploratory analysis, outlines the model structure, details the experimental setup (including models' configurations), and presents the results. Additionally, it includes a comparative analysis between the proposed model and existing state-of-the-art approaches. And finally, the concluding remarks are presented in the final section (section IV).

II. PROPOSED METHOD

This section presents in detail the Gradient Boosted - Deep Neural Network Regression (GB-DNNR) approach for solving multi-output regression problems simultaneously. The proposed method builds upon the ideas of the GB-DNN approach, conducted by [17] designed for image and tabular classification. The GB-DNNR model creates an ensemble of deep neural network models iteratively, by adding one deep model at each iteration. Each new deep model is trained on the residuals of the previous models and employs a within transfer learning strategy at each epoch to transfer learnt parameter information and architecture from previous models.

Considering a dataset composed of N instances, $D = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{N}$, drawn from a distribution $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ where $\mathcal{X} \in \mathbb{R}^M$ is a M-dimension feature space and distribution $\mathcal{Y} \in \mathbb{R}^K$ is in the dependent output space with K outputs, the primary goal of the optimization process is to accurately map the input space to the output space distribution through the use of a DNNs ensemble and its associated trainable parameters, denoted by $\mathbf{\Omega}$. To achieve this goal, the coefficients of the computational models are optimized with the aim of simultaneously predicting multiple continuous outputs. The loss function, which is defined as the mean squared error, is minimized in order to reduce the magnitude of the error and to improve the accuracy of the mapping

$$\ell(\mathbf{y}, \hat{\mathbf{F}}(\mathbf{x})) = \frac{1}{N}\sum_{i=1}^{N}\left(\mathbf{y}_i - \hat{\mathbf{F}}(\mathbf{x}_i)\right)^2, \tag{1}$$

where $\mathbf{y}_i$ is a vector representing the true values for the K outputs of instance $\mathbf{x}_i$. The objective is to create incrementally an approximated function, $\hat{\mathbf{F}}(\mathbf{x})$ as a linear combination of non-linear additive models. Specifically, at each boosting iteration t, the function is updated as $\hat{\mathbf{F}}_t(\mathbf{x}) = \hat{\mathbf{F}}_{t-1}(\mathbf{x}) + \nu\mathbf{\Omega}_t(\mathbf{x})$, where, $\hat{\mathbf{F}}_{t-1}(\mathbf{x})$ is the mapping obtained at the previous boosting iteration $t-1$, and $\mathbf{\Omega}_t$ is the current trained deep neural network trained on the residuals of the previous iterations. As a regularization term, each model is multiplied by a shrinkage rate $\nu \in (0, 1]$, which determines the contribution of the additive models to the final GB ensemble. The main theoretical difference with respect to the original GB [16] is that this model handles multi-output responses.

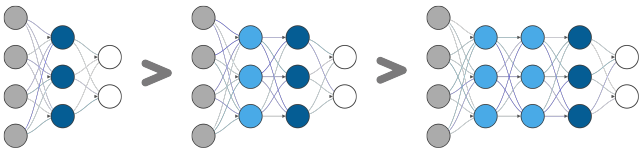

FIGURE 1: A snippet of the GB-DNNR training protocol. The neurons depicted in light blue denote layers that are held fixed, whereas the neurons in dark blue represent layers undergoing training. The white neurons correspond to the output layer, which is also undergoing training, and the gray units are the input

An important aspect of the proposed method is that the architecture of the ensemble is created using an auto transfer technique. Specifically, the architecture of the model of iteration t, $\mathbf{\Omega}_t$, is built from model at iteration $t-1$. To do so, the input and hidden layers of model $\mathbf{\Omega}_{t-1}$ are cloned, including its learnt parameters (auto transfer), and then frozen. Then a new hidden layer composed of l neurons and an output layer is added to the network. Finally, the parameters of the newly added layers (last hidden and output layers) are fit to the residuals of the previous models. This methodological approach allows us to combine complex deep models in contrast to the original GB [16] that works by combining rather simple decision trees. Fig. 1 presents a visual representation of the learning process of the proposed method. The figure showcases three successive boosting iterations (from left to right), where each iteration involves the training of a new dense layer (indicated by dark blue nodes) and the transfer of prior knowledge using frozen layers (represented by light blue nodes). It is important to note that the figure does not provide information about the specific number of neurons or the presence of batch normalization and dropout layers.

The aim of the t-th boosting iteration is to optimize the objective function, which involves updating the trainable parameters of $\mathbf{\Omega}_t$ with the goal of minimizing the loss function, Eq.1. The $\mathbf{\Omega}_t$ Deep Network utilizes the $\sigma = ReLU$ activation function and frozen, updated coefficients matrices from prior iterations, denoted as $\omega_j \mid j \in [1, t-1]$, as the hidden dense layers, which collectively incorporate a pre-trained, non-trainable set of coefficients. Additionally, an unexplored dense layer with l trainable neurons is added.

$$\mathbf{\Omega}_t = \sigma_t\big(\omega_t * \big[\sigma_j\omega_j | j \in [1, t-1]\big] + \mathbf{b}_t + \big[\mathbf{b}_j | j \in [1, t-1]\big]\big) \tag{2}$$

The optimization process whereby the value of $\mathbf{\Omega}_t$ is fitted

Algorithm 1 Training procedure of GB-DNNR

Input:
- A multi-output dataset $D = \{\mathbf{x}_i, \mathbf{y}_i\}_1^N$.
- Number of boosting iterations T.
- Number of training epoch on mini-batches E.
- GB loss function ℓ (Eq.1).

Output: Deep trained ensemble multi-output regressor

Training the model:

$\hat{\mathbf{F}}_0 = average(\mathbf{y})$.
Update residual Eq. 4.
for $e = 0$ to E **do**
  Train the GB-DNNR with one dense layer on residual.
end for
Update the trainable parameters ω_0.
Freeze the added dense layer's parameters ω_0.
for $j \in range(1, T)$ **do**
  Update residual Eq. 4.
  Auto transfer pre-trained frozen layer with $\omega_{j-1} | j \in [0, T-1]$.
  Add a new dense layer with uniform randomized coefficients.
  for $e = 0$ to E **do**
    Train the GB-DNNR on the residual.
    if If the training additive loss converges **then**
      break
    end if
  end for
  Update the trainable coefficients ω_j.
  Freeze the trainable coefficients ω_j
  if If the square loss (Eq. 1) converges **then**
    break
  end if
end for

based on the residual obtained from the previous boosting epoch $(t-1)$ using standard back propagation and square loss is

$$\mathbf{\Omega}_t = \underset{\omega_t}{\operatorname{argmin}}\, \ell\big(\mathbf{y}, \hat{\mathbf{F}}_{t-1}(\mathbf{x}) + \nu\mathbf{\Omega}_t\big) \tag{3}$$

This fitting procedure is performed concurrently with the calculation of the gradient of the loss function given in Eq. 1 as

$$\mathbf{r}(\mathbf{y}, \hat{\mathbf{F}}) = -\frac{\partial \ell(\mathbf{y}, \hat{\mathbf{F}}(\mathbf{x}))}{\partial \hat{\mathbf{F}}(\mathbf{x})} = \mathbf{y} - \hat{\mathbf{F}}_{t-1}(\mathbf{x}). \tag{4}$$

The t deep networks are trained sequentially on the residual $\mathbf{r}(\mathbf{y}, \hat{\mathbf{F}})$ to obtain the final trained network at the t boosting epoch, which incorporates all the frozen and trained weights from the prior boosting iterations, in addition to the final trained dense layers ω_t. This ultimate model can then be utilized as a regressor to determine the connection between the input and output space of the unobserved data. Algorithm. 1 exemplifies a schematic overview of the proposed Gradient Boosted - Deep Neural Network Regression for the multi-output tasks approach.

III. EXPERIMENTS AND RESULTS

This section includes the following subsections. First, the details of the multi-output regression datasets utilized in the experiment are given, offering a description of each dataset along with its unique characteristics (refer to subsection III-A). In addition, a comprehensive exploratory analysis, as detailed in subsection III-B, is conducted to assess the performance of the proposed model under two distinct conditions: one involving the freezing of dense layers and another employing a no-freezing approach. Furthermore, a detailed overview of both the architecture of the GB-DNNR model and the state-of-the-art deep model employed as a benchmark is provided in subsection III-C. The next subsection presents a detailed exposition of the hyper-parameter configuration and tested values for the studied models, which will undergo optimization of the models with grid search cross validation (subsection III-D). Finally, a comparative of the performance of the different tested models is provided using different metrics (subsection III-E). The source code for the proposed models is accessible on GitHub under the name `GB-DNNR`[1], which is based on `TensorFlow 2.13.0`[2].

A. DATASET DESCRIPTION

In order to assess the efficacy of the developed Deep multi-output model, a comprehensive mixture of 17 multi-output regression datasets has been utilized for experimentation purposes. These datasets originating from multiple subject domains exhibit heterogeneous characteristics, encompassing variations in the quantity of instances, attributes, and target variables. A detailed description of the utilized datasets is shown in Table 1.

The *andro* dataset is specifically designed to facilitate the prognostication of six distinct water quality parameters within the Thermaikos Gulf in Greece. Regarding the datasets *atp1d* and *atp7d* the dependent variables refer to the flight ticket price of the subsequent day and the minimum price observed over the following seven days, respectively. As for the *edm*, primary aim is to expedite the machining process by emulating the actions of a human operator who precisely manipulates the values of two variables. The *enb* pertains to the estimation of the heating and cooling energy

[1] github.com/GAA-UAM/GB-DNNR
[2] github.com/tensorflow/tensorflow

Dataset	Subject Area	# N	# M	# K
andro [18]	Environmental Studies	49	30	6
atp1d [10]	Air Travel Pricing	337	411	6
atp7d [10]	Air Travel Pricing	296	411	6
edm [19]	electrical engineering	154	16	2
enb [20]	Building Energy Efficiency	768	8	2
jura [21]	Environmental Studies	359	15	3
oes10 [10]	Labor Economics	403	298	16
oes97 [10]	Labor Economics	334	263	16
rf1 [10]	River Flow	9,125	64	8
rf2 [10]	River Flow	9,125	576	8
scm1d [10]	Supply Chain	9,803	280	16
scm20d [10]	Supply Chain	8,966	61	16
scpf [22]	Data Science	1,137	23	3
sf1 [23]	Space Science	323	10	3
sf2 [23]	Space Science	1,066	10	3
slump [24]	Civil Engineering	103	7	3
wq [9]	Ecology	1,060	16	14

TABLE 1: The datasets utilized in the experiments

load requirements of buildings. The *jura* dataset is centered around metallic elements with higher measurement costs. The *oes10* and *oes97* designate the datasets derived from the Occupational Employment Survey (OES) conducted in the years 1997 (oes97) and 2010 (oes10) respectively. Regarding the topic of River Flow (RF) analysis, *rf2* constitutes an extension of the preexisting *rf1* dataset, encompassing supplementary data concerning precipitation forecasts at eight distinct geographical locations. The supply chain dataset contains targets representing two distinct metrics: *scm1d*, denoting the mean price of a product for the following day, and *scm20d*, symbolizing the mean cost of the same product over a 20-day horizon in the future, as observed within the simulation. The *scpf* encompasses the prognostication of three distinct dependent variables, representing the quantitative assessment of user engagement metrics, namely, the number of views, clicks, and comments for each user interaction. In the domain of space science, we analyze data concerning discrete solar flare events during two distinct time intervals referred to as *sf1* (spanning from 1969) and *sf2* (spanning from 1978). The *slump* comprises three properties of concrete, namely slump, flow, and compressive strength, each of which is dependent on the composition of seven distinct concrete ingredients. The *wq* comprises 14 output attributes that quantitatively indicate the occurrence and distribution of diverse plant and animal species in the rivers of Slovenia.

B. EXPLORATORY ANALYSIS

Firstly, an exploratory experiment was carried out to ascertain if freezing previous layers is a good strategy with respect to not freezing them. In addition, in this experiment we test the convergence of the models with respect to the number of boosting iterations. The same model hyper-parameters were used for both freezing and non-freezing approaches. These hyper-parameters include a learning rate of 0.01, a batch size of 128, an $L2$ regularizer value of 0.1, a dropout layer rate of 0.1, and dense layers with a size of 100 units. The number of boosting iterations, corresponding to deep models, was fixed at six. This exploratory experiment is conducted on the *atp1d* dataset. The dataset is partitioned using a shuffled stratified approach, allocating 20% of its size to the test partition and the remaining portion for training.

The results of the experiment are shown in Figure 2. The plots illustrate the performance in RMSE of the proposed multi-output GB-DNNR for the train (blue curves) and test (green curves) sets when freezing previous layers (left subplot) and not freezing them (right subplot). The analysis of the plots indicates that, in the absence of freezing previous trained dense layers, the model exhibits a tendency to overfit. On the other hand, the left subplot underscores the observation that freezing previous layers removes the overfitting showing a more consistent learning process. Finally, we can observe that starting from three models the model generalization accuracy has converged. For subsequent experiments, we will use freezing and three boosting iterations.

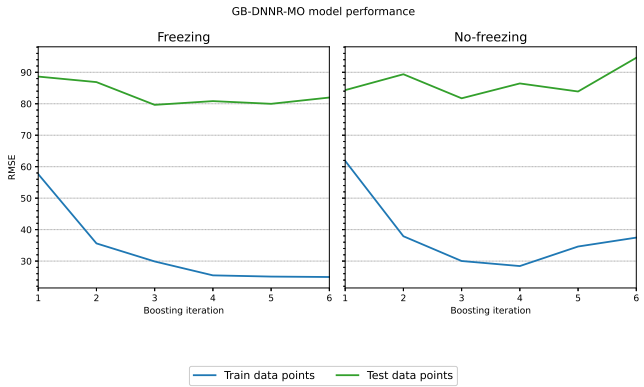

FIGURE 2: Train and test performance of GB-DNNR-MO in *atp1d* dataset with respect to the number of boosting iterations when freezing the previous layers (left plot) and not freezing (right plot)

C. DEEP STRUCTURE OF THE PROPOSED MODEL

For subsequent experiments, ensembles of three deep models (iterations) are used with dense hidden layers comprising $l = 100$ neurons. For each boosting iteration the following steps are carried out: clone the previous model, freeze weights, remove output layer, concatenate a new trainable dense layer and an output layer, and, finally, train it on the residuals of previous iterations. Hence, the free parameters of each model are only the ones of the last hidden and outputs layers. Note that, ReLU activation function is applied to all dense layers, and that all dense layers are followed by batch normalization and dropout layers. To mitigate overfitting, the model is subjected to $L2$ regularization in the dense layers. Ultimately, the output layer is configured to match the size of the regression outputs. In Fig. 1 a sketch of the proposed architecture is shown.

D. EXPERIMENTAL SETTING

In order to assess the performance of the proposed model, a comparison was carried out with respect to shallow and deep neural networks (labeled as NN and DNN respectively) as well as with respect to the shallow multi-output architecture GBNN, which is also based on GB [15]. All methods are tested in the multi-output setting (labeled with `MO`) as well as in the single output approach (labeled with `SO`). We compared the performance of these methods against the proposed GB-DNNR. The `SO` technique entails an independent training for each output, whereas in the `MO` procedures, a single model is trained simultaneously for all outputs.

The experimental training procedure involves the use of a three-fold cross-validation practice to partition the normalized dataset, where all attributes display a zero mean and one variance. To determine the optimal hyper-parameter combination for each model, grid search methodology is used (scikit-learn package[3]) using three-fold within-train cross-

[3] scikit-learn.org

Model	Hyper-parameters	Values
GB-DNNR	Learning rate	[0.001, 0.01, 0.1]
	$L2$ regularization	[0.001, 0.01, 0.1, 1, 10]
	Drop out	[0.1, 0.3]
	Shrinkage	[0.025, 0.5, 0.1, 0.75, 1]
	Boosting iterations	3
	Training epochs	200
GBNN	Shrinkage	[0.025, 0.1, 0.5, 1]
	Subsample	[0.5, 0.75, 1]
	Neurons per step	[1, 2, 3, 5]
	Boosting iterations	200/(Neurons per step)
	Training epochs	200
DNN	Learning rate	[0.001, 0.01, 0.1]
	$L2$ regularization	[0.001, 0.01, 0.1, 1, 10]
	Drop out	[0.1, 0.3]
	Training epochs	200
NN	hidden layer	range(1, 201, 2)
	Training epochs	200

TABLE 2: Grid of hyper-parameters and fixed hyper-parameter values for the various models

validation during the training process. The process of hyper-parameter optimization utilized in the analyzed models involved the consideration of different grids for each model. These hyper-parameter grids are shown in the Table 2. The architecture of the deep neural network (DNN) is built with three dense hidden layers, each including 100 neurons in order to obtain a model with the same number of parameters as GB-DNNR. In order to perform a fair comparison, the DNN model was configured using the same: architecture, batch normalization and dropout layers, $L2$ regularization in the dense layers, and activation function, as the proposed model. Their training process involves 200 epochs, with a batch size of 128. Notably, the GB-DNNR implements the hidden layer in a sequential and additive manner during the three boosting epochs. Furthermore, the deep models are enhanced with an early stopping criterion, which actively monitors the model's progression on the training data by evaluating the value of the loss function. The primary objective of this criterion is to mitigate overfitting. Concerning the GBNN model, the number of boosting epochs is determined by dividing 200 by the number of neurons trained at each iteration, so that at the end 200 neurons are included in the model (a shallow network with one hidden layer).

In the conducted experiments, the evaluation of various studied models in diverse aspects was carried out utilizing the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) as regression metrics for each target within the distinct datasets

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}, \tag{5}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|, \tag{6}$$

where N is the number of instances, y_i is the true value of one target, and $\hat{y}_i$ is the predicted value. As a result of a large number of models and targets that were incorporated into the experiments, several methodologies were utilized to further investigate the differences in performance. These methodologies included the Demsâr diagram [25], which visually presents the average ranking of the analyzed models along a horizontal axis, and the Nemenyi test, which facilitates the statistical comparison of pairwise models by demonstrating their differences. As well as the demonstration of the density and distribution estimate of achieved RMSE values for each model.

E. RESULTS

The average performance of each method and target, quantified in terms of RMSE is shown in Tables 3–5. In these tables, the targets are arranged in columns, and the datasets and methods are listed in rows. The best-performing method result for each target is highlighted with a light blue background. The results are organized by number of outputs. Table 3 includes the seven datasets with the lowest number of targets. Subsequently, Tables 4 and 5 show the results for the datasets with the highest number of output dimensions.

As observed from Tables 3–5, the GB-DNNR achieved the best average RMSE in both approaches. The SO approach of the proposed GB-DNNR method outperformed others by achieving the best results in 40 out of 131 tested targets. The second and third-best performances were observed in the SO approach of GBNN, with 23 targets, and GB-DNNR-MO approach, with 19 targets having the lowest RMSE scores, respectively. Following closely behind, GBNN-MO and DNN-SO, which both achieve the lowest RMSE in 17 targets. The good performance of the proposed model can also be observed in Fig. 3. This figure shows the median rank of the eight studied models across the 131 targets utilizing RMSE values. In this plot, GB-DNNR-SO shows the lowest median ranking with a value of two, which indicates that in at least half of the tested targets, the proposed model was ranked as first or second best. This method is followed the MO approach of GB-DNNR with a median ranking of three. The results for the shallow networks (NN-SO and MO) exhibit the lowest average median ranking.

To visualize more easily all 131 results from all eight models, a series of box plots have been generated (Fig. 4) for all MO and SO approaches. These box plots are depicted in separate subplots within Fig. 4, with the top subplot representing the MO models and the bottom subplot the SO models. In addition, in order to have comparable measures, instead of plotting the RMSE values directly, normalized RMSE are used. The values are normalized by dividing, for each target, the RMSE results by the best outcome of the eight scores. Hence, the values depicted indicate the performance factor with respect to the best one (i.e. a value of 5 indicates that the method performed 5 times worse in RMSE than the best method).

The top subplot illustrated in Fig. 4 demonstrates notable distinctions among the normalized RMSE scores of the various models when utilizing the MO approach. Notably,

Dataset	Method	Targets			Dataset	Method	Targets		
		1	2	3			1	2	3
edm	NN-MO	0.292	0.512		enb	NN-MO	3.666	4.218	
	NN-SO	0.310	0.492			NN-SO	3.561	4.130	
	GBNN-MO	0.361	0.524			GBNN-MO	2.437	2.764	
	GBNN-SO	0.365	0.521			GBNN-SO	2.338	2.748	
	DNN-MO	0.323	0.491			DNN-MO	0.681	1.016	
	DNN-SO	0.315	0.521			DNN-SO	0.679	1.027	
	GB-DNNR-MO	0.293	0.484			GB-DNNR-MO	0.423	0.744	
	GB-DNNR-SO	0.304	0.482			GB-DNNR-SO	0.411	0.803	
jura	NN-MO	0.636	2.710	11.894	scpf	NN-MO	31.940	0.968	0.703
	NN-SO	0.603	2.266	11.817		NN-SO	30.825	0.911	0.710
	GBNN-MO	1.473	2.185	11.378		GBNN-MO	30.396	0.946	0.743
	GBNN-SO	0.586	2.129	11.446		GBNN-SO	31.388	0.895	0.677
	DNN-MO	0.689	2.297	15.294		DNN-MO	33.589	1.162	0.822
	DNN-SO	0.675	2.088	14.695		DNN-SO	35.360	1.139	0.682
	GB-DNNR-MO	0.643	1.939	13.086		GB-DNNR-MO	34.085	0.957	0.737
	GB-DNNR-SO	0.589	1.945	12.720		GB-DNNR-SO	31.176	0.925	0.671
sf1	NN-MO	0.415	0.483	0.164	sf2	NN-MO	0.799	0.304	0.100
	NN-SO	0.413	0.500	0.159		NN-SO	0.795	0.302	0.097
	GBNN-MO	0.393	0.457	0.152		GBNN-MO	0.790	0.287	0.092
	GBNN-SO	0.393	0.450	0.144		GBNN-SO	0.789	0.288	0.092
	DNN-MO	0.401	0.485	0.131		DNN-MO	0.795	0.288	0.074
	DNN-SO	0.401	0.466	0.138		DNN-SO	0.828	0.293	0.113
	GB-DNNR-MO	0.394	0.455	0.128		GB-DNNR-MO	0.783	0.287	0.072
	GB-DNNR-SO	0.410	0.461	0.124		GB-DNNR-SO	0.778	0.290	0.074
slump	NN-MO	9.473	30.458	16.336					
	NN-SO	9.368	31.154	16.099					
	GBNN-MO	7.469	14.994	2.407					
	GBNN-SO	7.450	15.166	2.031					
	DNN-MO	7.352	14.410	3.227					
	DNN-SO	6.559	13.764	1.822					
	GB-DNNR-MO	6.809	13.132	2.063					
	GB-DNNR-SO	6.788	12.924	1.001					

TABLE 3: Average generalization RMSE performance for the different SO and MO approaches

the GB-DNNR model exhibits the narrowest interquartile range (IQR) followed by DNN and GBNN. The majority of data points in the proposed model are concentrated in the range below 2.5, a pattern that is not the case with the rest of the models except maybe for DNN-MO. The bottom subplot in Fig. 4 shows the results for the SO approaches. As it can be observed, GB-DNNR stands out as the method smallest interquartile range and more concentrated values close to 1. The next methods are DNN and GBNN, which demonstrate comparable levels of performance. In summary, Fig. 4 illustrates the superior performance of the proposed GB-DNNR model for both MO and SO approaches. It exhibits the narrowest IQR and integrated data points, free of outliers in contrast to other models such as NN and GBNN.

To evaluate the performance of the analyzed models in

Dataset	Method	Targets							
		1	2	3	4	5	6	7	8
andro	NN-MO	8.035	1.899	12.203	7.740	37.636	2.460		
	NN-SO	6.354	1.225	11.832	7.435	34.372	1.654		
	GBNN-MO	2.231	3.402	2.689	2.409	19.591	3.361		
	GBNN-SO	1.689	0.409	1.962	1.495	14.612	1.049		
	DNN-MO	2.186	0.466	2.336	1.644	14.843	1.052		
	DNN-SO	1.707	0.454	1.820	1.760	15.607	0.882		
	GB-DNNR-MO	1.255	0.770	1.809	1.242	13.350	1.255		
	GB-DNNR-SO	1.783	0.300	1.595	1.171	12.710	0.900		
atp1d	NN-MO	99.531	155.915	151.829	144.501	106.504	146.042		
	NN-SO	92.853	157.947	161.039	151.330	100.946	150.399		
	GBNN-MO	52.371	98.349	80.785	60.836	55.780	65.200		
	GBNN-SO	46.823	107.103	84.931	60.733	52.478	64.969		
	DNN-MO	47.853	111.477	80.121	60.536	54.110	61.415		
	DNN-SO	50.568	149.577	83.662	57.012	59.613	69.753		
	GB-DNNR-MO	46.469	102.247	77.367	52.813	54.635	56.453		
	GB-DNNR-SO	47.228	98.337	75.048	51.655	47.918	58.773		
atp7d	NN-MO	119.231	166.242	174.407	170.832	125.904	171.063		
	NN-SO	115.587	171.005	181.852	179.135	126.350	179.582		
	GBNN-MO	34.464	79.827	67.161	62.185	39.213	64.223		
	GBNN-SO	33.429	78.209	66.498	56.085	44.058	62.888		
	DNN-MO	36.302	83.851	74.166	71.904	47.817	74.885		
	DNN-SO	44.291	81.221	73.848	63.301	43.578	74.566		
	GB-DNNR-MO	30.390	90.556	60.528	59.655	36.562	61.585		
	GB-DNNR-SO	30.159	89.482	60.777	51.632	39.148	54.222		
rf1	NN-MO	34.057	0.846	35.072	21.682	15.494	3.999	9.643	9.285
	NN-SO	22.675	0.927	28.554	18.515	9.216	2.035	4.239	5.591
	GBNN-MO	18.465	7.792	20.891	13.638	7.163	8.455	6.938	7.074
	GBNN-SO	19.217	0.783	20.282	14.741	7.516	2.221	5.853	6.375
	DNN-MO	5.552	0.642	5.563	3.595	4.387	1.558	2.446	3.284
	DNN-SO	5.348	0.654	5.490	3.053	2.489	0.611	1.787	1.271
	GB-DNNR-MO	8.838	0.683	9.772	7.503	6.615	2.524	4.350	5.209
	GB-DNNR-SO	7.991	0.654	9.017	7.265	2.977	0.734	3.393	3.082
rf2	NN-MO	8.526	0.750	8.144	5.317	6.449	2.421	4.889	7.278
	NN-SO	8.662	0.903	8.165	5.305	4.752	1.222	3.710	3.617
	GBNN-MO	11.231	5.241	9.607	6.685	3.863	5.000	2.857	4.463
	GBNN-SO	6.288	0.750	8.313	5.024	5.944	0.863	2.159	4.639
	DNN-MO	4.375	0.647	4.415	3.049	3.488	1.549	2.255	2.444
	DNN-SO	5.299	0.685	4.269	2.765	2.475	0.543	1.824	1.408
	GB-DNNR-MO	10.184	0.779	10.245	8.069	7.381	3.315	5.725	6.321
	GB-DNNR-SO	4.473	0.658	4.388	6.470	2.171	0.736	1.366	1.271

TABLE 4: Average generalization RMSE performance for the different SO and MO approaches

a statistical manner, a comparative analysis was undertaken utilizing Demšar plots [26] and the Nemenyi test. The models were positioned along a horizontal x-axis based on their average rank in relation to 131 regression targets. Subsequently, the Nemenyi test was utilized to conduct a statistical comparison between the models, with the objective of identifying any significant differences that may exist among them. Models linked with a solid black line do not show statistically significant differences. The *subplots a, b and c* of Fig. 5 present the comparison between the MO models only, SO models only and all models respectively. The critical differences for these plots are CD=0.41, CD=0.41 and CD=0.91 respectively.

In *subplot a*, the proposed GB-DNNR model exhibits the

Dataset	Method	Targets															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
wq	NN-MO	1.402	1.500	0.786	1.118	1.582	1.441	1.286	0.782	1.328	1.378	0.887	0.934	0.947	1.399		
	NN-SO	1.415	1.506	0.886	0.962	0.993	1.478	0.798	1.149	1.585	1.451	1.295	0.800	1.347	1.342		
	GBNN-MO	1.370	1.450	0.760	1.082	1.610	1.451	1.267	0.776	1.310	1.296	0.885	0.907	0.948	1.389		
	GBNN-SO	1.368	1.451	0.873	0.896	0.946	1.407	0.762	1.114	1.549	1.440	1.261	0.776	1.309	1.297		
	DNN-MO	1.361	1.460	0.877	0.899	0.942	1.349	0.761	1.072	1.533	1.425	1.248	0.766	1.288	1.302		
	DNN-SO	1.376	1.470	0.902	0.956	0.985	1.440	0.780	1.098	1.580	1.491	1.339	0.761	1.335	1.285		
	GB-DNNR-MO	1.383	1.463	0.879	0.902	0.937	1.361	0.760	1.099	1.517	1.422	1.257	0.761	1.301	1.269		
	GB-DNNR-SO	1.385	1.447	0.868	0.909	0.937	1.372	0.779	1.063	1.527	1.433	1.254	0.766	1.323	1.290		
oes10	NN-MO	208.817	290.690	509.238	366.414	837.491	380.828	840.715	2323.656	237.935	934.639	288.382	780.317	272.805	977.464	296.641	223.826
	NN-SO	198.527	266.679	276.830	761.960	243.170	885.421	274.631	221.972	441.606	330.983	730.818	318.889	781.448	1980.932	220.021	866.897
	GBNN-MO	243.137	234.438	495.565	230.757	703.914	239.548	746.571	3078.710	252.242	788.519	251.058	698.753	282.263	852.244	295.973	217.172
	GBNN-SO	155.351	208.831	188.882	617.081	171.313	795.678	223.885	164.424	417.252	202.619	398.374	159.435	589.949	1326.219	163.228	618.972
	DNN-MO	184.301	229.966	232.312	906.921	273.482	1039.132	364.685	193.496	522.078	247.833	736.020	302.424	766.451	2285.978	203.362	855.895
	DNN-SO	184.495	239.520	206.190	848.154	215.344	753.425	403.279	167.821	386.145	302.232	654.845	341.184	616.179	1688.824	192.446	664.844
	GB-DNNR-MO	158.486	188.646	214.147	625.848	186.686	889.510	265.933	169.791	353.366	223.991	443.882	223.915	580.209	1264.136	185.190	577.269
	GB-DNNR-SO	184.341	218.820	245.556	681.822	187.094	868.289	509.855	182.201	369.068	206.931	391.450	181.779	609.867	1120.587	191.283	611.061
oes97	NN-MO	2153.893	478.112	1875.517	665.708	623.971	2240.808	1153.199	1338.325	180.413	1033.544	163.993	568.089	438.044	333.109	225.025	261.547
	NN-SO	2033.927	444.165	156.898	520.115	426.381	321.883	231.353	236.531	1852.002	616.937	567.487	2172.508	1125.196	1309.412	167.503	1010.937
	GBNN-MO	2450.146	448.977	2348.745	558.353	618.314	2371.982	943.759	1244.351	489.023	919.549	412.510	455.716	438.223	372.689	433.455	440.147
	GBNN-SO	1087.574	215.474	152.000	266.562	325.649	243.658	237.448	165.087	1035.044	408.320	481.059	1737.799	799.072	1084.949	143.329	896.417
	DNN-MO	2315.065	564.292	190.549	431.456	470.726	309.450	211.452	516.073	2193.812	756.709	875.806	2074.543	1018.993	1649.162	239.252	1472.958
	DNN-SO	3219.207	512.120	149.473	406.290	397.583	273.780	258.936	372.912	2097.866	669.912	840.426	2777.252	1007.785	1522.176	177.773	1553.220
	GB-DNNR-MO	1338.183	299.309	140.712	321.642	338.745	249.978	187.344	218.459	1197.660	475.936	545.627	1742.494	850.802	1075.533	148.606	956.144
	GB-DNNR-SO	1281.912	257.218	154.932	275.111	355.402	268.446	216.105	175.370	1017.134	454.576	499.323	1770.435	840.768	1192.113	162.555	1013.488
scm1d	NN-MO	112.108	123.576	116.143	125.986	121.804	134.170	150.503	164.428	153.536	160.977	121.852	133.316	154.269	167.533	164.744	177.067
	NN-SO	108.250	121.523	117.262	131.295	156.216	173.972	170.525	183.445	107.976	122.144	119.743	129.858	147.200	165.797	157.096	167.388
	GBNN-MO	72.556	78.822	75.643	83.297	78.067	88.700	93.684	107.252	99.293	105.392	80.720	88.740	100.497	112.155	113.287	119.374
	GBNN-SO	62.080	73.417	72.709	78.893	88.346	101.803	102.106	108.338	67.524	76.561	70.714	81.646	90.649	98.652	93.351	98.514
	DNN-MO	71.136	77.651	76.066	84.214	94.065	102.868	103.200	108.828	72.585	81.280	75.605	88.007	94.994	105.216	97.927	104.115
	DNN-SO	62.544	73.640	72.525	79.763	86.198	98.207	96.186	105.469	67.274	76.291	81.290	83.452	88.138	98.291	93.106	100.678
	GB-DNNR-MO	62.219	71.880	71.092	77.940	86.368	96.878	97.843	102.814	67.317	76.573	70.239	80.314	87.720	96.751	91.773	97.585
	GB-DNNR-SO	64.232	71.257	71.546	76.400	83.017	95.600	96.771	102.608	65.115	74.491	69.584	80.164	88.131	94.502	91.665	96.937
scm20d	NN-MO	156.666	173.363	164.395	185.166	177.277	195.353	218.484	234.959	229.787	243.717	174.883	194.367	223.698	241.776	244.523	261.530
	NN-SO	135.471	151.046	156.429	173.668	203.820	221.215	227.182	245.452	141.068	162.717	155.360	175.074	197.453	215.520	212.490	229.109
	GBNN-MO	105.756	113.873	113.169	127.234	118.112	129.708	147.890	161.128	157.179	165.939	115.131	129.746	152.513	162.263	166.543	179.086
	GBNN-SO	87.121	97.439	99.639	111.780	133.908	142.798	148.865	158.207	96.190	110.180	101.020	112.826	128.632	141.138	138.661	146.680
	DNN-MO	74.886	81.062	79.940	89.007	96.802	100.875	99.671	108.413	77.664	89.819	82.407	92.643	97.815	111.525	100.906	107.459
	DNN-SO	69.235	78.333	78.494	85.178	85.554	96.354	96.757	109.409	70.000	82.822	77.149	88.355	94.346	106.787	98.156	108.178
	GB-DNNR-MO	71.006	78.503	77.607	86.198	92.610	99.755	100.688	109.142	73.509	85.441	78.829	88.718	96.162	106.772	98.969	104.897
	GB-DNNR-SO	66.094	76.918	77.096	85.132	87.332	98.499	98.885	109.904	70.596	83.709	75.396	86.794	93.960	104.394	97.950	105.241

TABLE 5: Average generalization RMSE performance for the different SO and MO approaches

best result, which is statistically significant in relation with all the other MO models. Likewise, the *subplot b* (referred to as SO strategy) demonstrates that the proposed model is statistically better than all the rest SO models. The overall analysis shown in *subplot c* reveals that the proposed model with MO and SO obtain the best average ranks with statistically better results than most of the other approaches. Notably, the NN models and MO approaches of GBNN and DNN exhibit the lowest rankings.

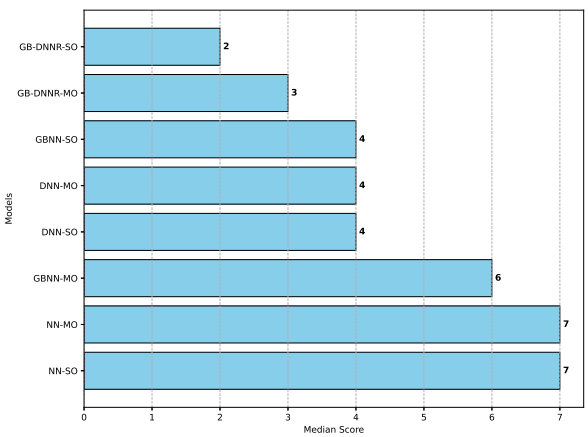

FIGURE 3: Visual ranking using ranking median

To further assess the robustness of the proposed GB-DNNR model, we conducted an additional performance evaluation, comparing its outcomes to those of state-of-the-art models using Mean Absolute Error (MAE). The MAE results are presented in Tables 6–8, wherein best scores are highlighted using a light blue shading. Notably, the proposed model employing the MO approach demonstrated superior performance, achieving the lowest MSE values across 72 targetsi out of 131, followed by its SO approach which exhibited the lowest MSE in 31 targets. In contrast, the DNN model, in both its MO and SO approaches, achieved the lowest MSE in only 12 targets. Furthermore, the NN and GBNN models collectively achieved the lowest MSE in four targets.

Dataset	Method	Targets			Dataset	Method	Targets		
		1	2	3			1	2	3
edm	NN-MO	0.209	0.433		enb	NN-MO	2.580	2.984	
	NN-SO	0.203	0.401			NN-SO	2.504	2.936	
	GBNN-MO	0.267	0.453			GBNN-MO	1.742	1.966	
	GBNN-SO	0.281	0.439			GBNN-SO	1.651	1.944	
	DNN-MO	0.206	0.411			DNN-MO	0.518	0.667	
	DNN-SO	0.200	0.437			DNN-SO	0.514	0.717	
	GB-DNNR-MO	0.195	0.401			GB-DNNR-MO	0.306	0.516	
	GB-DNNR-SO	0.201	0.404			GB-DNNR-SO	0.300	0.536	
jura	NN-MO	0.466	2.033	6.600	scpf	NN-MO	9.816	0.432	0.195
	NN-SO	0.426	1.732	6.616		NN-SO	10.031	0.338	0.196
	GBNN-MO	1.096	1.670	6.161		GBNN-MO	9.949	0.423	0.285
	GBNN-SO	0.420	1.611	6.578		GBNN-SO	10.309	0.339	0.192
	DNN-MO	0.504	1.743	7.314		DNN-MO	11.933	0.595	0.266
	DNN-SO	0.448	1.607	6.985		DNN-SO	12.106	0.463	0.144
	GB-DNNR-MO	0.487	1.504	6.221		GB-DNNR-MO	10.209	0.381	0.212
	GB-DNNR-SO	0.410	1.492	6.293		GB-DNNR-SO	10.262	0.340	0.179
sf1	NN-MO	0.236	0.248	0.088	sf2	NN-MO	0.429	0.106	0.035
	NN-SO	0.242	0.258	0.069		NN-SO	0.431	0.108	0.024
	GBNN-MO	0.204	0.227	0.075		GBNN-MO	0.413	0.097	0.041
	GBNN-SO	0.243	0.224	0.067		GBNN-SO	0.449	0.090	0.027
	DNN-MO	0.227	0.237	0.043		DNN-MO	0.389	0.094	0.018
	DNN-SO	0.193	0.213	0.054		DNN-SO	0.411	0.069	0.022
	GB-DNNR-MO	0.210	0.217	0.042		GB-DNNR-MO	0.428	0.100	0.016
	GB-DNNR-SO	0.243	0.222	0.042		GB-DNNR-SO	0.433	0.084	0.014
slump	NN-MO	8.046	26.224	13.946					
	NN-SO	8.208	27.113	14.023					
	GBNN-MO	6.107	12.519	1.813					
	GBNN-SO	5.841	12.677	1.474					
	DNN-MO	5.430	11.185	2.537					
	DNN-SO	4.788	10.184	1.572					
	GB-DNNR-MO	4.887	10.218	1.685					
	GB-DNNR-SO	4.962	10.009	0.789					

TABLE 6: Average generalization MAE performance for the different SO and MO approaches

IV. CONCLUSION

The study introduced a novel methodology termed Gradient Boosted - Deep Neural Network Regression (GB-DNNR) for tackling multi-output regression tasks using an auto transfer learning technique. The proposed model constructs an ensemble of deep networks through sequential incorporation of dense layers at each boosting iteration. At each of these iterations, the previous network is frozen, cloned and a newly dense layer is concatenated. Then the training involves minimizing the loss function on the residuals of previous iterations by optimizing only the free parameter weights introduced in the last layer. We have shown that these auto transfer and freezing techniques reduce the complexity of the models and serve as a regularization for the final model that mitigates overfitting.

Through extensive experimentation on multiple multi-

Dataset	Method	Targets							
		1	2	3	4	5	6	7	8
andro	NN-MO	6.443	1.540	9.594	6.288	29.318	1.846		
	NN-SO	5.135	0.980	9.508	6.071	25.844	1.223		
	GBNN-MO	1.830	2.687	2.266	1.884	15.526	2.821		
	GBNN-SO	1.393	0.325	1.684	1.260	11.045	0.805		
	DNN-MO	1.807	0.360	1.923	1.375	12.116	0.872		
	DNN-SO	1.336	0.289	1.478	1.499	12.034	0.680		
	GB-DNNR-MO	0.979	0.585	1.541	1.012	9.518	0.962		
	GB-DNNR-SO	1.450	0.209	1.292	0.947	9.938	0.676		
atp1d	NN-MO	79.382	117.064	114.350	112.318	84.483	112.984		
	NN-SO	72.405	116.628	117.661	116.939	78.069	115.480		
	GBNN-MO	35.148	58.506	52.372	41.896	36.967	43.777		
	GBNN-SO	28.285	63.734	54.829	41.211	33.802	42.805		
	DNN-MO	30.533	63.092	52.602	42.108	37.030	43.300		
	DNN-SO	30.329	82.555	52.648	38.222	35.299	47.767		
	GB-DNNR-MO	27.527	55.396	48.918	36.819	33.838	37.993		
	GB-DNNR-SO	27.521	54.993	48.217	33.972	30.944	36.026		
atp7d	NN-MO	96.017	123.929	138.210	130.062	100.477	129.448		
	NN-SO	91.714	128.386	142.332	134.884	99.349	134.211		
	GBNN-MO	21.983	43.620	44.465	36.182	23.098	37.247		
	GBNN-SO	19.837	40.429	44.812	32.362	21.413	35.924		
	DNN-MO	23.343	47.165	53.193	46.782	28.845	48.811		
	DNN-SO	24.703	40.872	44.782	35.653	24.393	49.924		
	GB-DNNR-MO	17.629	40.733	39.593	35.168	20.154	36.341		
	GB-DNNR-SO	18.599	39.851	41.120	30.377	20.298	30.203		
rf1	NN-MO	8.339	0.212	9.411	6.082	5.536	1.798	3.689	4.157
	NN-SO	5.031	0.152	6.959	5.385	3.778	0.996	1.395	2.055
	GBNN-MO	5.063	1.843	6.254	4.358	4.082	2.296	2.448	3.119
	GBNN-SO	3.500	0.140	3.521	3.115	2.454	0.813	0.996	1.519
	DNN-MO	3.574	0.179	3.669	2.023	3.276	1.230	1.493	2.412
	DNN-SO	2.471	0.144	2.943	1.164	1.633	0.378	0.733	0.864
	GB-DNNR-MO	5.178	0.241	6.324	4.193	4.769	1.911	2.873	3.565
	GB-DNNR-SO	2.103	0.113	3.798	1.493	1.634	0.524	0.758	1.017
rf2	NN-MO	3.895	0.257	4.193	3.104	3.724	1.740	2.597	3.180
	NN-SO	3.698	0.173	3.086	1.933	2.399	0.512	1.418	1.374
	GBNN-MO	2.047	0.956	2.321	1.374	1.768	1.091	1.051	1.424
	GBNN-SO	1.515	0.112	1.947	1.227	1.409	0.395	0.487	0.812
	DNN-MO	2.866	0.190	3.009	1.942	2.553	1.223	1.493	1.834
	DNN-SO	3.119	0.150	2.919	1.249	1.590	0.376	0.848	0.872
	GB-DNNR-MO	6.195	0.320	6.811	5.015	5.348	2.456	4.053	4.638
	GB-DNNR-SO	1.171	0.123	1.495	0.963	1.188	0.346	0.650	0.595

TABLE 7: Average generalization MAE performance for the different SO and MO approaches

output regression datasets employing both single and multi-output (SO and MO) approaches, we have demonstrated the statistically significant better performance of the proposed GB-DNNR model over deep neural networks and other state-of-the-art methods. The results indicate its effectiveness in tackling multi-output problems and highlight its potential for advancing the field of multi-output regression.

ACKNOWLEDGMENT

The authors acknowledge the Centro de Computación Científica CCC-UAM.

REFERENCES

[1] AJ Baranchik, "Inadmissibility of maximum likelihood estimators in some multiple regression problems with three or more independent variables," The Annals of Statistics, vol. 1, no. 2, pp. 312–321, 1973.

Dataset	Method	Targets															
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
wq	NN-MO	1.080	1.101	0.579	0.555	0.678	1.106	0.520	0.737	1.279	1.094	0.986	0.535	0.971	0.907		
	NN-SO	1.074	1.207	0.745	0.983	0.975	1.324	0.618	0.745	1.437	1.679	1.029	0.804	1.630	1.375		
	GBNN-MO	1.072	1.100	0.586	0.547	0.676	1.143	0.517	0.733	1.354	1.142	0.981	0.545	0.976	0.903		
	GBNN-SO	1.063	1.177	0.704	0.943	0.892	1.272	0.595	0.673	1.416	1.645	0.998	0.774	1.552	1.376		
	DNN-MO	1.033	1.048	0.551	0.495	0.677	1.094	0.487	0.669	1.219	1.032	0.970	0.520	0.920	0.852		
	DNN-SO	1.084	1.156	0.683	1.012	0.931	1.302	0.643	0.683	1.414	1.629	1.000	0.786	1.642	1.260		
	GB-DNNR-MO	1.061	1.084	0.559	0.520	0.662	1.093	0.497	0.701	1.211	1.054	0.961	0.517	0.961	0.858		
	GB-DNNR-SO	1.107	1.181	0.708	0.965	0.949	1.294	0.584	0.684	1.420	1.629	0.996	0.793	1.569	1.329		
oes10	NN-MO	133.678	179.843	177.293	508.650	213.891	495.120	204.770	150.820	300.880	256.381	460.516	230.478	376.552	1289.600	145.569	440.461
	NN-SO	124.261	415.131	254.823	625.973	1798.113	608.929	396.600	160.653	293.828	494.714	618.402	349.995	349.472	2722.292	244.730	542.850
	GBNN-MO	168.757	168.804	186.024	436.854	212.621	427.422	198.287	164.585	306.859	166.879	386.803	154.182	322.205	1548.464	183.943	361.804
	GBNN-SO	100.415	476.415	299.667	539.237	2713.180	582.851	447.451	139.192	278.349	464.508	611.109	271.495	307.027	2696.924	241.055	478.361
	DNN-MO	130.222	163.834	161.633	678.170	221.716	583.096	250.962	153.974	344.920	196.452	490.814	252.146	367.282	1201.987	121.729	544.274
	DNN-SO	127.026	498.907	375.466	547.448	3032.169	572.226	656.696	142.147	282.850	682.946	627.371	374.121	368.205	2662.458	272.678	471.798
	GB-DNNR-MO	104.028	119.654	133.958	394.064	130.749	398.763	145.223	114.124	225.548	165.654	279.740	108.545	265.036	464.186	107.912	254.532
	GB-DNNR-SO	118.165	492.764	290.895	579.686	2566.521	582.370	463.241	147.188	277.524	510.118	614.603	301.770	523.422	2689.705	205.841	479.749
oes97	NN-MO	1220.796	270.987	116.919	291.896	302.549	228.049	145.999	165.899	1075.765	389.733	366.975	1319.714	750.572	850.931	134.436	729.186
	NN-SO	1174.478	306.213	1476.404	655.591	710.298	242.096	1006.593	173.585	2188.332	371.067	397.293	2054.977	1131.273	1066.235	1358.977	969.366
	GBNN-MO	1198.340	314.552	341.648	289.791	308.225	283.483	381.311	385.333	1149.514	330.106	389.613	1265.585	654.094	763.456	452.824	637.112
	GBNN-SO	601.402	244.410	2313.905	805.210	856.464	211.688	1249.724	160.222	2163.480	331.581	343.221	2005.468	1091.409	1010.947	2212.166	885.450
	DNN-MO	1265.347	354.176	125.201	308.875	321.293	213.145	127.893	340.085	1388.474	461.500	499.539	1201.385	730.737	975.923	145.216	909.114
	DNN-SO	2582.459	649.053	3233.924	1220.438	1794.632	202.382	1916.714	520.086	2159.598	405.864	412.240	1974.262	1043.426	1021.211	2277.248	814.848
	GB-DNNR-MO	544.276	184.520	93.364	162.618	261.724	184.185	109.878	105.982	512.682	280.081	302.924	934.348	581.885	673.092	109.441	623.719
	GB-DNNR-SO	588.659	278.320	2174.859	817.618	808.817	225.377	1260.606	179.917	2179.074	357.191	363.700	2010.531	1102.115	1042.384	2057.024	920.293
scm1d	NN-MO	78.572	86.540	86.413	95.087	109.143	119.623	116.754	125.966	80.983	88.839	85.653	94.071	105.408	113.703	109.366	115.341
	NN-SO	74.076	163.983	159.327	251.385	211.804	232.113	212.468	477.886	177.125	190.259	324.656	337.578	232.734	243.876	484.692	485.208
	GBNN-MO	48.432	51.902	53.661	60.343	66.771	74.419	74.350	80.481	51.058	55.112	52.894	59.483	60.597	68.427	65.284	69.930
	GBNN-SO	39.069	146.488	142.998	245.546	193.711	206.162	187.395	469.442	167.470	177.117	322.163	336.288	216.531	225.183	477.932	476.809
	DNN-MO	46.685	50.053	50.505	56.695	62.071	66.607	65.629	70.408	48.301	52.819	50.671	59.144	60.325	67.256	64.880	69.099
	DNN-SO	37.991	146.757	142.691	244.324	191.660	205.677	183.662	472.157	166.702	174.495	316.466	334.631	215.599	223.480	482.371	480.151
	GB-DNNR-MO	40.138	44.576	45.386	51.172	54.881	60.946	61.031	65.093	42.834	48.125	45.214	51.542	54.288	59.877	58.106	61.826
	GB-DNNR-SO	40.461	145.237	141.910	242.512	189.256	202.508	184.875	468.579	165.797	175.223	320.742	333.888	213.979	222.184	477.296	476.214
scm20d	NN-MO	120.144	134.510	133.808	150.201	173.188	187.260	188.049	199.198	125.034	141.372	135.280	149.965	167.128	179.582	174.818	185.332
	NN-SO	102.529	174.487	176.777	262.443	223.657	242.821	234.862	470.937	183.303	197.817	320.341	336.109	235.456	248.614	477.145	478.346
	GBNN-MO	78.364	84.330	84.572	95.694	112.162	119.419	122.674	130.214	83.063	92.210	86.798	95.457	108.842	117.978	116.585	122.854
	GBNN-SO	62.303	150.201	146.968	245.162	194.545	209.353	197.041	461.461	168.556	179.448	318.971	331.900	214.708	222.010	468.662	467.362
	DNN-MO	50.222	54.862	53.420	60.788	64.916	67.518	66.372	70.779	53.192	58.704	55.390	63.186	65.659	72.702	69.003	72.542
	DNN-SO	43.380	152.134	145.417	241.830	198.515	213.606	196.388	474.090	170.086	182.480	318.959	332.845	222.463	235.699	471.146	471.846
	GB-DNNR-MO	47.027	52.062	51.547	57.945	61.769	66.786	67.327	71.264	49.760	56.553	52.707	60.074	64.361	69.906	66.982	71.084
	GB-DNNR-SO	41.800	148.738	142.128	245.890	192.546	205.485	188.258	464.956	167.034	174.280	316.336	330.710	215.450	225.281	471.746	469.411

TABLE 8: Average generalization MAE performance for the different SO and MO approaches

[2] Philip J Brown and James V Zidek, "Adaptive multivariate ridge regression," The Annals of Statistics, vol. 8, no. 1, pp. 64–74, 1980.

[3] Leo Breiman and Jerome H. Friedman, "Predicting Multivariate Responses in Multiple Linear Regression," Journal of the Royal Statistical Society: Series B (Methodological), vol. 59, no. 1, pp. 3–54, 01 2002.

[4] Amita Dhankhar, Kamna Solanki, Sandeep Dalal, and Omdev, "Predicting students performance using educational data mining and learning analytics: A systematic literature review," in Innovative Data Communication Technologies and Application, Jennifer S. Raj, Abdullah M. Iliyasu, Robert Bestak, and Zubair A. Baig, Eds., Singapore, 2021, pp. 127–140, Springer Singapore.

[5] Jing Wu, Hongchao Cheng, Yiqi Liu, Bin Liu, and Daoping Huang, "Modeling of adaptive multi-output soft-sensors with applications in wastewater treatments," IEEE Access, vol. 7, pp. 161887–161898, 2019.

[6] Emmanuel Vazquez and Eric Walter, "Multi-output suppport vector regression," IFAC Proceedings Volumes, vol. 36, no. 16, pp. 1783–1788, 2003, 13th IFAC Symposium on System Identification (SYSID 2003), Rotterdam, The Netherlands, 27-29 August, 2003.

[7] Guangcan Liu, Zhouchen Lin, and Yong Yu, "Multi-output regression on the output manifold," Pattern Recognition, vol. 42, no. 11, pp. 2737–2743, 2009.

[8] Jean-Paul Chiles and Pierre Delfiner, Geostatistics: modeling spatial uncertainty, John Wiley & Sons, 1999.

[9] Timo Aho, Bernard Zenko, Saso Dzeroski, and Tapio Elomaa, "Multi-target regression with rule ensembles," J. Mach. Learn. Res., vol. 13, pp. 2367–2407, 2012.

[10] Eleftherios Spyromitros Xioufis, Grigorios Tsoumakas, William Groves, and Ioannis P. Vlahavas, "Multi-target regression via input space expansion: treating targets as inputs," Machine Learning, vol. 104, no. 1, pp. 55–98, 2016.

[11] Dragi Kocev, Celine Vens, Jan Struyf, and Sašo Džeroski, "Ensembles of multi-objective decision trees," in Machine Learning: ECML 2007, Joost N. Kok, Jacek Koronacki, Raomon Lopez de Mantaras, Stan Matwin, Dunja Mladenič, and Andrzej Skowron, Eds., Berlin, Heidelberg, 2007, pp. 624–631, Springer Berlin Heidelberg.

[12] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil, "Convex multi-task feature learning," Machine Learning, vol. 73, no. 3, pp. 243–272, December 1 2008.

[13] Ali Jalali, Sujay Sanghavi, Chao Ruan, and Pradeep Ravikumar, "A dirty model for multi-task learning," in Advances in Neural Information Processing Systems, J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, Eds. 2010, vol. 23, Curran Associates, Inc.

[14] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Aikaterini Vrekou, and Ioannis Vlahavas, "Multi-target regression via random linear target combinations," in Machine Learning and Knowledge Discovery in Databases, Toon Calders, Floriana Esposito, Eyke Hüllermeier, and Rosa Meo, Eds., Berlin, Heidelberg, 2014, pp. 225–240, Springer Berlin Heidelberg.

[15] Seyedsaman Emami and Gonzalo Martínez-Muñoz, "Multioutput regression neural network training via gradient boosting," in 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2022, Bruges, Belgium, October 5-7, 2022, 2022.

[16] Jerome H. Friedman, "Greedy function approximation: A gradient boost-

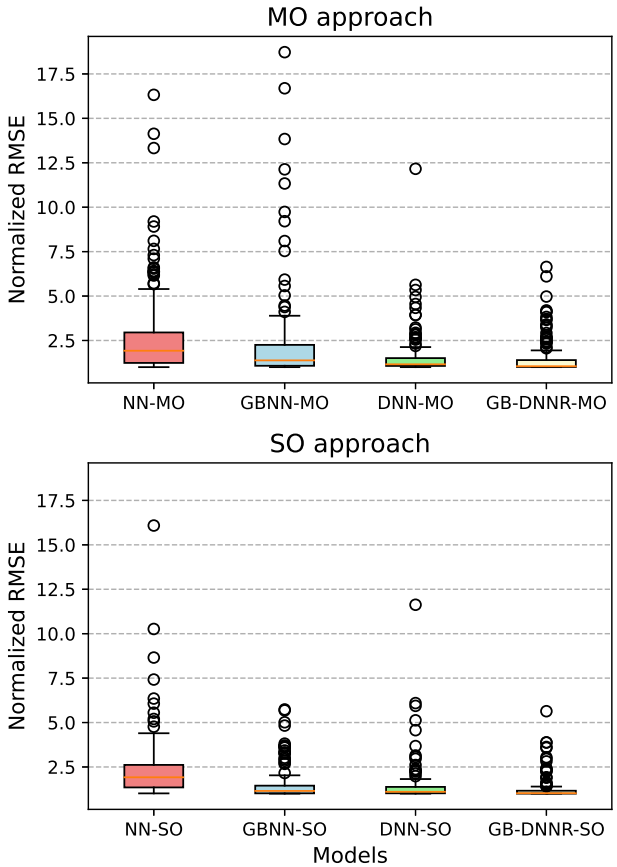

FIGURE 4: Box plot of normalized RMSE scores for different models, with MO and SO approaches

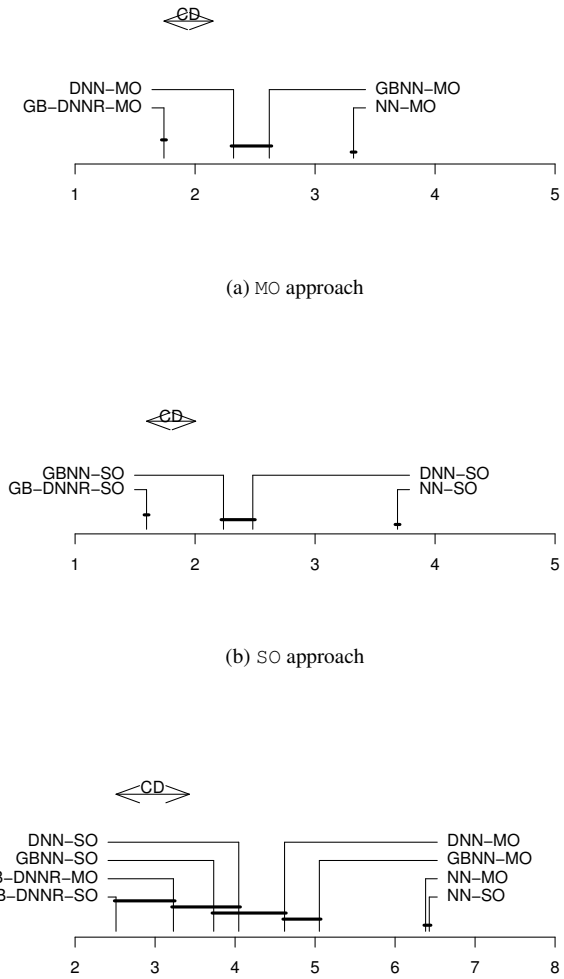

FIGURE 5: Demšar plot illustrating the comparative performance of the studied models on 131 regression targets for MO approach (subplot a), SO approach (subplot b) and both approaches (subplot c)

ing machine," The Annals of Statistics, vol. 29, no. 5, pp. 1189–1232, 2001.

[17] Seyedsaman Emami and Gonzalo Martínez-Muñoz, "A gradient boosting approach for training convolutional and deep neural networks," IEEE Open Journal of Signal Processing, vol. 4, pp. 313–321, 2023.

[18] Evaggelos V. Hatzikos, Grigorios Tsoumakas, George Tzanis, Nick Bassiliades, and Ioannis Vlahavas, "An empirical study on sea water quality prediction," Knowledge-Based Systems, vol. 21, no. 6, pp. 471–478, 2008.

[19] Aram Karalic and Ivan Bratko, "First order regression," Mach. Learn., vol. 26, no. 2-3, pp. 147–176, 1997.

[20] Athanasios Tsanas and Angeliki Xifara, "Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools," Energy and Buildings, vol. 49, pp. 560–567, 2012.

[21] Pierre Goovaerts et al., Geostatistics for natural resources evaluation, Oxford University Press on Demand, 1997.

[22] Kaggle, "Kaggle competition: Online product sales," https://www.kaggle.com/c/online-sales, 2012.

[23] Dheeru Dua and Casey Graff, "UCI machine learning repository," https://archive.ics.uci.edu/ml/index.php, 2009, visited on 2023-08-01.

[24] I-Cheng Yeh, "Modeling slump flow of concrete using second-order regressions and artificial neural networks," Cement and Concrete Composites, vol. 29, no. 6, pp. 474–480, 2007.

[25] Janez Demsar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. Res., vol. 7, pp. 1–30, 2006.

[26] Janez Demsar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. Res., vol. 7, pp. 1–30, 2006.

SEYEDSAMAN EMAMI holds a Bachelor of Science degree in Industrial Engineering (2012) and a Master of Science degree in Financial Engineering (2018). He is currently pursuing a Ph.D. degree in Computer and Telecommunications Engineering with a specialization in Information Engineering at Universidad Autónoma de Madrid (UAM). The author's research interests are focused on machine learning topics, such as ensemble learning, deep learning frameworks, and the application of machine learning in the domains of science, media, and education. Mr. Emami also gained valuable practical experience through collaborations with industry partners and participation in research projects. They have demonstrated strong problem-solving skills, attention to detail, and the ability to work collaboratively with diverse teams. Overall, the author is a dedicated and accomplished researcher with a strong track record in

machine learning, and their work promises to make significant contributions to the field.

GONZALO MARTíNEZ-MUÑOZ received the university degree in Physics (1995) and Ph.D. degree in Computer Science (2006) from the Universidad Autónoma de Madrid Madrid (UAM). From 1996 to 2002, he worked in industry. Until 2008 he was an interim assistant professor in the Computer Science Department of the UAM. During 2008/09, he worked as a Fulbright post- doc researcher at Oregon State University in the group of Professor Thomas G. Dietterich, world reference in the field of Machine Learning. He is currently Assistant Professor at Computer Science Department at UAM as well as coordinator of the master's program on Bioinformatics and Computational Biology. In addition, he is the IP of several research projects related to AI. His research interests include machine learning, decision trees, and ensemble learning, and applications of machine learning to science, sports and education. Mr. Martínez-Muñoz has published his research in some of the journals with highest impact in machine learning.