

Multiple Additive Neural Networks: A Novel Approach to Continuous Learning in Regression and Classification

Janis Mohr^a, Basile Tousside^b, Marco Schmidt^c and Jörg Frochte^d

Interdisciplinary Institute for Applied Artificial Intelligence and Data Science Ruhr, Bochum University of Applied Sciences, 42579 Heiligenhaus, Germany

Keywords: Gradient Boosting, Machine Learning, Continuous Learning, Neural Networks, Overfitting.

Abstract: Gradient Boosting is one of the leading techniques for the regression and classification of structured data. Recent adaptations and implementations use decision trees as base learners. In this work, a new method based on the original approach of Gradient Boosting was adapted to nearly shallow neural networks as base learners. The proposed method supports a new architecture-based approach for continuous learning and utilises strong heuristics against overfitting. Therefore, the method that we call Multiple Additive Neural Networks (MANN) is robust and achieves high accuracy. As shown by our experiments, MANN obtains more accurate predictions on well-known datasets than Extreme Gradient Boosting (XGB), while also being less prone to overfitting and less dependent on the selection of the hyperparameters learn rate and iterations.

1 INTRODUCTION

Boosting is a technique that combines weak learners to achieve a highly accurate model. Every single learner only needs to be moderately accurate (Shapire, 1990).


Gradient Boosting is a proposed boosting technique that has had great success on data sets with structured data (Friedman, 1999b; Viola and Jones, 2001). Most published papers use different implementations of decision trees as base learners (Chen and Guestrin, 2016; Dorogush et al., 2018). The gradient of a previous learner is used for fitting the next learner in Gradient Boosting. Gradient Boosting can achieve very accurate predictors and is less prone to overfitting than other boosting techniques, but still easily overfits on some datasets if several hundred learners are combined (Bikmukhametov and Jaeschke, 2019). The MANN algorithm is an adaptation and enhancement of the gradient boosting algorithm with nearly shallow neural networks as base learners. Several heuristics and techniques are used to prevent overfitting. Thus, MANN can reach better accuracy than popular implementations of Gradient


Boosting which are based on weak learners like decision trees. We especially emphasise developing an algorithm that is easy to use and supports continuous learning rather than being mainly focused on accuracy. Our approach focuses on using neural networks with a minimum of hidden layers and neurons. We present techniques to automatically stop the training of new neural networks if accuracy is not significantly improved.


Some papers have already considered the possibility of using boosting techniques with neural networks. (Schwenk and Bengio, 2000) used neural networks in combination with the Adaptive Boosting (AdaBoost) method and came to the conclusion that it works as well and sometimes even better than AdaBoost with decision trees. Our experiments rely on the newer, more general, and (in effective implementations) better-performing Gradient Boosting. The results confirm the good quality of predictions of boosting with neural networks. Additionally, advanced heuristics were added and the algorithm was designed to be easy to use and reduce hyperparameter tuning.


(Martinez-Munoz, 2019) proposes to train the neurons of a single neural network with one hidden layer sequentially in a boosting approach. (Shalev-Shwartz, 2014) proposed the SelfieBoost algorithm that uses Stochastic Gradient Descent as a weak learner for boosting a single neural network.

A recently followed path of research is the com-

^a  <https://orcid.org/0000-0001-6450-074X>

^b  <https://orcid.org/0000-0002-9332-5060>

^c  <https://orcid.org/0000-0002-7232-5256>

^d  <https://orcid.org/0000-0002-5908-5649>

bination of trees and neural networks. Papers in this domain use tree-like structures with neural networks. Adaptive Neural Trees (Tanno et al., 2018) adaptively grow tree-like structures with neural networks. (Deboleena et al., 2018) proposes to build a hierarchical model out of several neural networks in a tree-wise manner which can grow to learn new data. MANN differs from these approaches because it is based on an algorithm commonly used with decision trees but does not try to put neural networks into tree-like structures.

This paper will enhance the related work regarding overfitting and adaptivity, showing how to use Gradient Boosting with neural networks and that it achieves even better predictions than popular boosting algorithms. Our approach to achieving highly accurate predictors was extended with continuous learning, which furthermore separates our algorithm from plain implementations of Gradient Boosting. Several real-life use-cases of machine learning benefit from continuous learning (Kaeding et al., 2017). New data is generated over time and while an already trained model is in use. MANN is eminently suitable in these cases to continuously train a model and improve its accuracy on new data. Our algorithm allows two different approaches to support continuous learning, altering the neural networks of an already existing model and calculating residuals with the already trained model to fit a new model and create a combined model. Special attention was given to overfitting as a general problem in machine learning. An implementation of a heuristic that works to prevent overfitting when using Gradient Boosting with neural networks is proposed and it is shown that MANN is less prone to overfitting and easier to use because of less hyper-parameter tuning than other popular boosting techniques.

Our specific contributions in this paper are summarised below:

1. We propose a novel method that incrementally builds deep neural networks out of several neural networks using the Gradient Boosting algorithm. This method is versatile and can easily be adapted for a wide range of tasks and domains while being easier to train and fine-tune than traditional deep neural networks.
2. We develop heuristics against overfitting based on early stopping for this method and propose an architecture-based approach for continuous learning.
3. We demonstrate the usefulness of the developed heuristics and the approach for continuous learning. Furthermore, we show superior results on several regression and classification datasets.

2 AN APPROACH FOR ADDITIVE NEURAL NETWORKS

The accuracy of a predictive model can often be increased by averaging the decisions of an ensemble of predictive models. When the individual predictors are accurate and diverse, significant improvement can be expected. A general idea to use this is to have a base learner and apply it on different training sets for several times.

Gradient Boosting is a boosting algorithm. It sequentially trains predictors to construct an additive model. The gradient of the loss function is used to train the predictor on the next iteration. All fitted predictors have the same influence on the final model, only weighted by a learning rate that is equal for all predictors. Gradient Boosting as proposed by (Friedman, 1999a) is a two-step optimisation method.

A system of output (target) variables y and input variables (features) x can be formed into a dataset $\mathcal{D} = \{y_i, x_i\}$. The goal for our algorithm is then to model a function $F^*(x)$ on the dataset \mathcal{D} of known (y, x) -values. The function $F^*(x)$ maps x to y in a way that the value of a specified differentiable loss function $L(y_j, F(x))$ is minimised. The index j is the ongoing number of the iteration I_j .

The algorithm starts with an initial guess $F_0(x)$, which is a constant value. This initial guess is used as a starting point for the model. At this point, the model would predict $F_0(x)$ for every value of x . In the second step, the (pseudo-)residuals are computed as:

$$r_{i,j} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{j-1}(x)} \quad (1)$$

The term (pseudo-)residuals is used because their nature depends on the loss function. For example the loss function residual sum of squares $L(y_j, F(x)) = \frac{1}{2} \cdot (y_j - F(x))^2$ which is mostly used for regression leads to real residuals (Friedman, 1999a). The base learner is then fit to the (r_i, x_i) -values. Next, the output values γ_j of the fitted base learner for the whole dataset are computed with γ being the predicted value for the current residuum.

$$\gamma_j = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{i,j}} L(y_i, F_{j-1}(x_i) + \gamma) \quad (2)$$

Finally, the predictions $F_j(x)$ of the model at the current iteration I_j are computed. The predictions of the previous iteration are summed up with the predictions of the current iteration and are multiplied by a learning rate v as proposed by (Friedman, 1999a). The learn rate assures that every base learner's influence in the final model is limited. This limitation reduces

```

1: Input: data  $x_i, y_i$ ; Loss function  $L(y_i, F(x))$ 
2: Initialise  $F_0(x) = \operatorname{argmin}_{\gamma} \sum_{i=1}^n L(y_i, \gamma)$ .
3: repeat
4:    $r_{i,j} = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{j-1}(x)}$ 
5:   Fit a neural network to  $r_{i,j}$ 
6:   Early stopping.
7:   for  $j = 1, \dots, J$  do
8:      $\gamma_j = \operatorname{argmin}_{\gamma} \sum_{x_i \in R_{i,j}} L(y_i, F_{j-1}(x_i) + \gamma)$ 
9:   end for
10:   $F_j(x) = F_{j-1}(x) + v \sum_{j=1}^J \gamma_j$ 
11:  Heuristic to prevent overfitting.
12:   $j = j + 1$ 
13: until  $j = J$ 

```

Algorithm 1: Multiple Additive Neural Networks.

the impact of a non-optimal iteration.

$$F_j(x) = F_{j-1}(x) + v \sum_{j=1}^J \gamma_j \quad (3)$$

The process of computing the residuals, fitting the base learner, and updating the predictions is done sequentially for a given number of iterations J . After reaching J iterations the model is considered complete and trained. This trained model is called \mathcal{M} and is ready to be used for predictions.

This description of Gradient Boosting is abstract and not restricted to one specific type of base learner or loss function. Most implementations use decision trees as base learners. MANN uses neural networks as base learners. The networks can be used for regression and classification tasks. The main idea behind using artificial neural networks is that Gradient Boosting might be able to accomplish better accuracy with stronger base learners while allowing continuous learning by directly retraining the neural networks.

The established theory of gradient boosting is used to present a new approach for neural networks offering the previously described advantages. Not quite so deep (no more than three hidden layers) neural networks are used and therefore gradient boosted neural networks for both regression and classification are the core of MANN. The efficiency of neural networks depends on choosing reasonable hyperparameters for their structure and training. The artificial neural networks are kept as small and shallow as possible. The number of neurons on every layer can be adapted. To predict complex functions, higher numbers of neurons are necessary. On every iteration I_j in the algorithm, the (pseudo-)residuals are calculated. A neural network NN_j is fitted on these (pseudo-)residuals. The neural network then predicts the original unaltered training dataset D . These predictions are used to calculate the (pseudo-)residuals

on which a neural network is fitted in the next iteration. The number of epochs that a single neural network is trained in is automatically regulated with an implementation of early stopping. The progress the neural network makes on the training dataset is evaluated after every epoch. If the accuracy of the neural network does not increase further for a given amount of epochs the training is stopped prematurely.

2.1 Heuristic to Prevent Overfitting

```

1: Let  $\mathcal{T}$  be the training set and  $\mathcal{V}$  the validation set.
2: Every time training of a neural network  $NN_i$  is finished:
3: if  $E_{va}(NN_i) \geq E_{va}(NN_{i-1})$  or  $\leq E_t$  then
4:   break
5: end if

```

Algorithm 2: Heuristic to prevent overfitting.

Preventing Overfitting is a central and important part of every algorithm that fits neural networks on data.

Our heuristic takes action after operation 10 on line 10 in algorithm 1. The heuristic is used on every iteration of our algorithm. An amount of data is removed from the training dataset \mathcal{T} and used as the validation dataset \mathcal{V} . The validation dataset remains unaltered during the algorithm and is used on every iteration. Tests on several datasets have shown that taking five percent of the test data as validation data delivers good results. Algorithm 2 shows the used heuristics schematically.

On the current iteration I a model \mathcal{M}_I is built up out of the initial value F_0 and the already fitted neural networks NN_j . This temporary model represents the current state of the training and how the model would perform if the fitting of new neural networks would be stopped now.

\mathcal{M}_I is now fed with the x values of the validation dataset to make predictions on it. The predictions are evaluated and an error E_{va} is calculated. This error is compared to a threshold E_t . If it is below this threshold the model predicts as well as wanted and the fitting of new neural networks is stopped. Otherwise, the training continues. Over n steps the accuracy of predictions is evaluated. If the accuracy is steady or decreases, the fitting of additional neural networks is also stopped because no improvements are to be expected. During experiments, it seemed reasonable to choose $n = 3$.

Our heuristics securely prevent overfitting and furthermore reduce the time needed for computation. The fitting of additional neural networks is stopped when no improvement is to be expected. Therefore,

the algorithm is user-friendly and easy to use. Overfitting is automatically avoided without any parameter tuning. Our algorithm also automatically avoids waste of time and energy from fitting neural networks that do not improve the accuracy of the model further.

In the same mind, an early stopping (Raskutti et al., 2013) was added on the level where the neural networks are trained (algorithm 1 operation 5). For every evaluation, the same evaluation dataset is used. The performance of the single neural networks is constantly evaluated every epoch. If no progress is made for a number of steps the training of this neural network is stopped because it is not expected to achieve better performance.

The proposed algorithm prevents overfitting and training without any positive effect on the performance of the model on two levels. Every neural network is accurately monitored and trained in the optimal amount of epochs. Additionally, the whole model is evaluated at every iteration to keep it as small as possible and make it less prone to overfitting.

2.2 Architecture-Based Approach to Continuous Learning

- 1: Let \mathcal{T}_1 and \mathcal{T}_2 be two datasets with the same number of features and the same target value.
- 2: Model M_1 trained on \mathcal{T}_1 as described in Algorithm 1.
- 3: **if** $E(M_1(\mathcal{T}_1)) - E(M_1(\mathcal{T}_2)) \geq \epsilon$ **then**
- 4: **break**
- 5: **else**
- 6: Retrain Model M_1 on the new training data \mathcal{T}_2 .
- 7: **if** $|E(M_1(\mathcal{T}_1)) - E(M_1(\mathcal{T}_2))| \geq \epsilon$ **then**
- 8: **break**
- 9: **else**
- 10: Algorithm 1 with \mathcal{T}_2 to build Model M_2 .
- 11: Combine M_1 and M_2 to create final Model M_3 .
- 12: **end if**
- 13: **end if**

Algorithm 3: Continuous Learning with MANN.

Humans have the ability to acquire and transfer knowledge throughout their life. This is called life-long learning. Continuous learning is an adaption of this mechanic (Kaeding et al., 2017). In this paper, continuous learning is considered to be a technique that allows a complete model to be able to predict a new dataset with the same features. Therefore, it must be retrained or expanded. An algorithm that supports both the time-efficient retraining and the expansion of the model was developed. This technique is only suitable for continuous learning with a dataset with

the same classes as the original dataset otherwise it would be necessary to alter at least the last layer of the neural networks. The algorithm is not only superior in terms of accuracy as the experiments show but it also supports continuous learning on two levels.

A model M_1 is trained on a dataset \mathcal{T}_1 . This model reaches a certain quality of its predictions until the training is stopped. A second dataset \mathcal{T}_2 exists with new data but the same set of features and targets as the first dataset. For example, this could be data that was gathered after the training on the original data was finished. It would be possible to train a new model on a dataset constructed out of both datasets. But it is much more feasible to use the already fitted model to reduce the amount of time needed for training a new model from scratch, especially on very large datasets. It will require some retraining.

First, the algorithm checks if new training is required. The already trained model is fed with the new data and the predictions of the model are evaluated by comparing the error E of the prediction of \mathcal{T}_1 with the error E of the prediction of \mathcal{T}_2 . If the results in a specific metric are close to the original dataset, apparently retraining is not necessary. It is very likely that better results will not be achieved. The two datasets seem to be quite similar.

$$|E(M_1(\mathcal{T}_1)) - E(M_1(\mathcal{T}_2))| \geq \epsilon \quad (4)$$

Given the case that there is a difference in the accuracy of predictions, the new dataset is used for training. At first, the already existing neural networks in the model are retrained on the new dataset. Therefore, the weights are adapted to fit the new data better. After this assimilation, the new dataset is again predicted with the model. If the performance is similar to the original one there is no need to continue the training. Equation (4) is again used for the rating of the performance. E can be an arbitrary error metric for example mean-square-error. If there is still a significant difference the model is extended. Algorithm 1 is used again but with the new dataset. Generally speaking, new neural networks that are fitted to the new dataset are added to the already existing model M_1 .

The algorithm starts off with the initial guess $F_0(x)$. When not training continuously the target values y_i are used for initialisation. Taking advantage of already having a model, the loss of Model M_1 on training data \mathcal{T}_2 is used as the initial value. Therefore, model M_1 is used to start the fitting of new neural networks. From this point on, model M_2 is built with neural networks that are fit to the data for a given amount of iterations I . The final model is M_3 and is then used for predictions. Continuous learning can be repeated as new training data becomes available. This

approach to continuous learning is especially suitable for regression tasks. The complete process of continuous learning as pseudo-code is given in Algorithm 3.

3 EXPERIMENTS

In the following section, different experiments are described. The experiments show the accuracy of the proposed method and how it works including demonstrations of the effectiveness of the heuristics against overfitting. An academic implementation of MANN is used for the experiments. The loss function $L(y_j, F(x)) = \frac{1}{2} \cdot (y_j - F(x))^2$ is used for regression tasks and the loss function $L(y_j, p) = y_i \cdot \log(p) + (1 - y_i) \cdot \log(1 - p)$, with the predicted probability p , is used for classification. The implemented artificial neural networks have three hidden layers with 8 neurons each. We decided to use this fixed set of layers and neurons to reduce the variation of parameters during the following experiments. This strongly enhances the validity of the experiments and aligns the number of hyperparameters that are available throughout the compared learners. Extensive repetitions of the experiments were done to ensure that the parameters are chosen in a way that the neural networks achieve good performance. XGB is used to compare and evaluate the performance. The freely available implementation of XGB was used. In the regression experiments, a multi-layer perceptron (MLP) with 5 layers was used as an example of the performance of deep neural networks without any boosting.

3.1 Results on Regression Benchmarks

The following sections contain a study on the performance on several regression benchmarks to show off the accuracy of MANN compared to other popular methods.

3.1.1 Results on the Bike Sharing Dataset

In this experiment, a dataset known as the Bike Sharing dataset is used. It is a popular dataset that was released in 2013 and since then used to test algorithms and for educational purposes (Fanaee-T and Gama, 2014).

Data for every rented bike is logged. The information consists out of duration, start date, end date, start station, end station, bike number, and member type. This data is enhanced with general information about the environment like weather, humidity, temperature, wind speed, and weekday. The dataset has data for

every hour for two years which leads to 17,379 rows of data. Each with 15 features.

First, training is started on the whole dataset. Data augmentation or elimination of outliers was not done. The unaltered dataset is used for our experiment to have a good comparison to the accuracy of XGB on the same dataset. A parameter grid search was used to find the best combination of learn rate and iterations with a maximum of 500 epochs. Stochastic gradient descent was used as optimiser and the fitting of new networks was halted when the mean-absolute error on the validation dataset falls below 1. The learning rate was set to 0.3. With these parameters, an RMSE of 56 was achieved in predicting the test data. The data from the last days of every month, starting with the 20th was used as test data to have a use case that is close to real life. For comparison, a XGB model is trained on the same dataset with a learning rate of 0.2 and a maximum tree depth of 6. This XGB model acquired an RMSE of 62 on the test data. These results lead to two insights. Firstly XGB overfits the dataset with reasonably picked parameters while MANN does not. Secondly, the accuracy of the prediction is better with MANN.

The aforementioned parameters for both methods were chosen with a parameter grid search and are the parameters that had the best results. Figure 1 is a plot of the RMSE over the iterations and learn rate for MANN and XGB.

MANN has a peak for a combination of a small learn rate and few iterations. This suggests avoiding choosing small learn rates and few iterations, which is consistent. XGB shows overfitting on this dataset strongly depending on the learn rate. With a higher learn rate the overfitting increases. This experiment demonstrates that the heuristic proposed in section 2.1 is working. To furthermore prove the claim that the proposed heuristics work, MANN was used to train a model on the bike-sharing dataset with and without the heuristic. The accuracy on the training data increases progressively for the model without the heuristics and decreases on the test data. The other model keeps improving until no more improvement can be made and then stops.

3.1.2 Million Song Dataset & CT Scan Slize Localization Dataset

The CT Scan Slize Localization Dataset is a dataset with medicinal data proposed by (Graf et al., 2011). It consists of 384 features extracted from a set of 53500 CT images from 74 different patients (43 male, 31 female). The final feature vector is a concatenation of two histograms, one about the location of bone structures and one about air inclusions. The target values

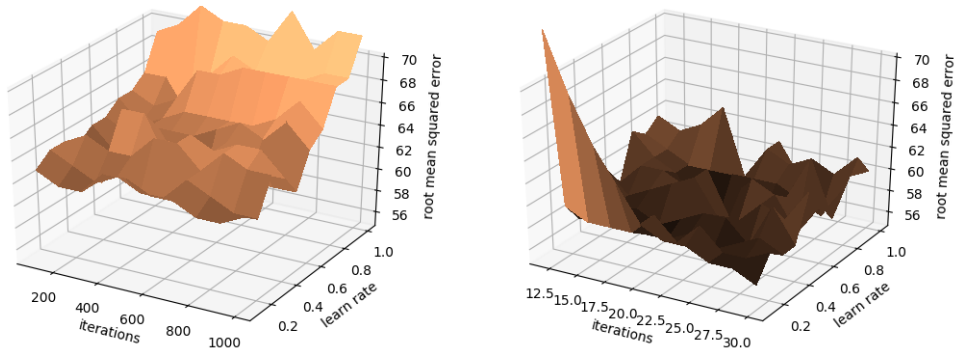


Figure 1: RMSE plotted depending on the number of iterations and learn rate training the bike sharing dataset with XGB (left) and MANN (right). This plot represents the parameter grid search that was used to find the parameters for MANN and XGB that lead to the best accuracy. The root mean squared error is used as a metric. Furthermore, this plot shows that MANN is less dependent on its parameters than XGB.

Table 1: Regression Datasets (CT Scan and MSD in RMSE).

Dataset	MANN	XGB	ANT	MLP
CT Scan	5.34	6.67	-	8.49
MSD	8.57	9.38	-	12.73

are in a range from 0 to 180.

The Million Song Dataset (MSD) is a set of features for songs from 1922 to 2011 published by (Bertin-Mahieux et al., 2011). Predicting a track’s year of origin based only on audio features and without any metadata is the task. The split into test and training data recommended by the authors of the dataset is used to avoid having artists appear in both training and test data. The data per year is highly non-uniformly distributed. All results can be seen in table 1.

3.2 Results on Binary Classification Datasets

As described in Section 2 MANN does not only support regression but also classification tasks. Three binary classification datasets were selected to present a variety of complexity and dataset sizes and the accuracy of MANN, XGB, and ANT are compared. Results can be seen in table 2.

3.2.1 UCI Heart Disease & Rain in Australia Dataset

The UCI heart disease dataset was published by (De-trano et al., 1989). This dataset consists out of 4 subsets from different hospitals with data on patients with and without heart disease. The goal is to predict whether a patient has heart disease and if so, which type. It has 14 features and 303 instances that rep-

resent medical data about patients and it reduces the task to find out if a patient has heart disease or not.

While some papers (Dangare and Apte, 2012; Zriqat et al., 2016) reach an accuracy of nearly 100 percent on this dataset with the use of heavy data augmentation most papers (Chen et al., 2011; Sabarinathan and Sugumaran, 2014; Sabay et al., 2018) reach an accuracy of around 85 percent without any data augmentation. No data augmentation was used in this experiment for better comparability.

A learning rate of 0.6 and 18 neural networks were used. Every neural network was trained in a maximum of 400 epochs.

With these parameters, MANN reaches an accuracy of 90 percent and XGB of 85 percent. MANN is more accurate than XGB and is also slightly above what most published papers (Aljanabi et al., 2018) accomplish on this dataset showing that MANN works well on very small datasets.

The rain in Australia dataset contains daily weather information from different locations all over Australia. The target is to predict whether it is going to rain the next day. The dataset consists of 23 features in 142,193 rows. Data was collected from different weather stations in Australia over a time of 10 years. The data origins from the Bureau of Meteorology of the Australian government. Predicting rainfall is of immense interest because it can be an early warning sign for natural disasters and is important for agriculture (Parmar et al., 2017).

The feature RISK_MM was excluded from the training dataset. It is the amount of rainfall that occurred on the next day and was used as a source to ascertain the value for the target variable. Several features are categorical, for example, the wind direction. Label Encoding was used to deal with all categorical

Table 2: Binary Classification Datasets Accuracy.

Dataset	MANN	XGB	ANT
UCI heart disease	0.90	0.85	0.88
Rain in Australia	0.89	0.87	0.89
Higgs Boson	0.85	0.83	0.82

features present in the Rain in Australia dataset.

The best results were found with a parameter grid search with a learn rate $M_v = \{v \in \mathcal{N} | 2 \leq v \leq 20 \wedge v = 2 \times k\}$ for both learners. 600 iterations were used for XGB and 18 for MANN. The development of the accuracy is steady for XGB with one peak and a mean of 85.04 percent accuracy. The best accuracy for XGB is 87 percent with a learn rate of 0.4 and 600 iterations. MANN reaches a better accuracy of 89 percent with a learn rate of 0.5 and 18 iterations. It has a mean accuracy of 84.87 percent. With an accuracy of 89 percent ANT is as good as MANN on this task.

3.2.2 Higgs Boson Dataset

The Higgs Boson dataset is a classification problem to distinguish between a signal process that produces Higgs bosons and a process that does not produce Higgs bosons. Monte Carlo simulations were used to create the data. The features in this dataset represent kinematic properties and some derived functions, usually used by physicists. The full dataset consists out of 1 million entries. (Baldi et al., 2014) This dataset was used to evaluate the accuracy of MANN on a big and imbalanced (there are many more data for not producing a Higgs boson) dataset. Again results can be seen in table 2.

3.3 Continuous Learning Benchmark

The Bike Sharing dataset will be revisited to demonstrate the strength of our continuous learning. The dataset was divided into two independent pieces of similar size and with the same features because the data's structure already clearly indicates to do so. The split is done into two years 2011 and 2012. On average the number of rented bikes in 2012 is higher than in 2011. A model M_1 was trained with MANN with the same parameters mentioned above on the first part of the dataset. The model predicts the 2011 dataset with a root-mean-squared error of 57. Predicting the 2012 dataset with this model though leads to a root-mean-squared error of 128. The already trained model is used and continuously fit on the second dataset that relates to the log of 2012 in the manner that is explained in section 2.2. The neural networks from model M_1 are retrained with the new data. The RMSE improves to 106 and therefore the second

Table 3: Bike Sharing Comparison RMSE.

ALGORITHM	2011	2012	BOTH
MANN FROZEN	57	128	56
MANN CL 1ST LEVEL	57	106	56
MANN CL ALL LEVELS	57	79	56
XGB FROZEN	58	130	62
LEARN++.MT	60	87	63
ANN FROZEN	69	155	67
ANN CL	69	92	67

level of continuous learning takes effect. Neural networks are trained on the new data and then added to the already existing model. An RMSE of 79 is the result of the new model combining both levels of continuous learning. The neural networks of the original model were modified and new neural networks trained on the 2012 dataset were added. XGB trained on the 2011 dataset achieves a root-mean-squared error of 58. Trained on the 2011 dataset it reaches an RMSE of 130 on the 2012 dataset. An artificial neural network with 5 hidden layers and 20, 15, 10, 5, and 1 neurons trained in 700 epochs achieves an RMSE of 69 and 155. This ANN was retrained on the 2012 dataset. After retraining, the RMSE on the 2012 dataset is 92. Learn++.MT, an algorithm that is based on AdaBoost with decision trees and specifically proposed for incremental learning by (Muhlbaier et al., 2004) was also tested. All results can be seen in table 3.

4 CONCLUSION

This paper proposed a novel approach for regression and classification machine learning tasks based on the well-known and highly popular Gradient Boosting framework. The original Gradient Boosting algorithm was altered to use artificial neural networks as base learners. MANN makes heavy use of early stopping and a heuristic to prevent overfitting. Both make our algorithm effective and easy to use with a minimum of hyper-parameter tuning. We demonstrated that our algorithm achieves better results on datasets than popular implementations of the Gradient Boosting algorithm for example XGB. Our algorithm makes significant improvements in prediction accuracy. Our algorithm also has a built-in method for continuous learning. MANN can change the weights of neural networks in an already trained model or train new neural networks on a new dataset and add them to an already trained model. It is sensible to do more experiments with continuous learning and try to overcome catastrophic forgetting with even the most elaborate learning tasks. Furthermore, it makes sense to

look into possibilities to not only add more models or retrain existing models but also to fine-tune the existing neural networks or maybe even specific layers to use this algorithm in incremental learning tasks. In future research, it would make sense to add support for unstructured data.

ACKNOWLEDGEMENTS

This work was funded by the Federal Ministry of Education and Research under 16-DHB-4021.

REFERENCES

- Aljanabi, M., Qutqut, M. H., and Hijjawi, M. (2018). Machine learning classification techniques for heart disease prediction: A review. *International Journal of Enigneering and Technology*.
- Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature Communications*.
- Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.
- Bikmukhametov, T. and Jaeschke, J. (2019). Oil production monitoring using gradient boosting machine learning algorithm. *12th IFAC Symposium on Dynamics and Control of Process Systems*.
- Chen, A. S., Huang, S.-W., Hong, P. S., Cheng, C., and Lin, E. J. (2011). Hdps: Heart disease predictions system. *Computing in Cardiology*.
- Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *KDD 16*.
- Dangare, C. S. and Apte, S. S. (2012). A data mining approach for predictions of heart disease using neural networks. *International Journal of Computer Engineering and Technology*, pages 30–40.
- Deboleena, R., Priyadarshini, P., and Kaushik, R. (2018). Tree-cnn: A hierarchical deep convolutional neural network for incremental learning.
- Detrano, R., Janosi, A., Steinbrunn, W., Pfisterer, M., Schmid, J.-J., Sandhu, S., Kern H. Guppy, S. L., and Froehlicher, V. (1989). International application of a new probability algorithm for the diagnosis of coronary artery diseases. *The American Journal of Cardiology*.
- Dorogush, A. V., Ershov, V., and Gulin, A. (2018). Catboost: gradient boosting with categorical features support. *Conference on Neural Information Processing Systems (NeurIPS 2018)*.
- Fanaee-T, H. and Gama, J. (2014). Event labeling combining ensemble detectors and background knowledge. *Progress in Artificial Intelligence*, pages 113–127.
- Friedman, J. H. (1999a). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*.
- Friedman, J. H. (1999b). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38.
- Graf, F., Kriegel, H.-P., Schubert, M., Pölsterl, S., and Cavallaro, A. (2011). 2d image registration in ct images using radial image descriptors. In Fichtinger, G., Martel, A., and Peters, T., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2011*, pages 607–614, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Kaeding, C., Rodner, E., Freytag, A., and Denzler, J. (2017). Fine-tuning deep neural networks in continuous learning scenarios. In *Computer Vision - ACCV 2016 Workshops*, pages 588–605.
- Martinez-Munoz, G. (2019). Sequential training of neural networks with gradient boosting.
- Muhlbaier, M., Topalis, A., and Polikar, R. (2004). Learn++.mt: A new approach to incremental learning. volume 3077, pages 52–61.
- Parmar, A., Mistree, K., and Sompura, M. (2017). Machine learning techniques for rainfall prediction: A review. In *International Conference on Innovation in Information Embedded and Communication Systems*.
- Raskutti, G., Wainwright, M. J., and Yu, B. (2013). Early stopping and non-parametric regression: An optimal data-dependent stopping rule.
- Sabarinathan, V. and Sugumaran, V. (2014). Diagnosis of heart disease using decision trees. *International Journal of research in Computer Applications and Information Technology*, pages 74–79.
- Sabay, A., Harris, L., Bejugama, V., and Jaceldo-Siegl, K. (2018). Overcoming small data limitations in heart disease prediction by using surrogate data. *SMU Data Science Review*, 1(3).
- Schwenk, H. and Bengio, Y. (2000). Boosting neural networks. *Neural Computation*.
- Shalev-Shwartz, S. (2014). Selfieboost: A boosting algorithm for deep learning.
- Shapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, pages 197–227.
- Tanno, R., Arulkumaran, K., Alexander, D., Criminisi, A., and Nori, A. (2018). Adaptive neural trees. *7th International Conference on Advanced Technologies*.
- Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Conference on Computer Vision and Pattern Recognition*.
- Zriqat, E., Altamimi, A., and Azzeh, M. (2016). A comparative study for predicting heart diseases using data mining classification methods. *International Journal of Computer Science and Information Security*, 12.