LU-TP 22-26
May 2022

Combining Cross-Validation and Ensemble Creation for Artificial Neural Networks

Anna Lena Hölldobler

Department of Astronomy and Theoretical Physics, Lund University

Bachelor thesis supervised by Patrik Edén

LUND
UNIVERSITY

Abstract

Artificial neural networks (ANNs) are widely used nowadays, and the research into improving their performances is continually ongoing. One main goal of ANNs is to have a high generalization performance, which can be estimated through validation. Ensembles can be useful to raise the generalization performance, but the validation of ensembles is often computationally costly if the size of the training data set is limited. Therefore, this thesis introduces shortcut ensembles during cross-validation, where several validation outputs get averaged to estimate the generalization performance of an ensemble. To evaluate this method the validation performance of the shortcut ensemble was compared to validation and test performances of a single model and an actual ensemble, using two different data sets for classification problems. The results show that the shortcut ensemble gives better estimates for the generalization performance of an ensemble than a single model during validation and it can approximate the validation performance of an actual ensemble. Hence, the shortcut ensemble can provide a less costly way of validating ensembles during cross-validation.

Popular Abstract

Robots are already playing a big role in our world, and if you dive into the world of science fiction, you will very quickly come across stories of robots with artificial intelligence taking over the world. Is this a realistic scenario? In reality, artificial intelligence is far away from human thinking. But research is constantly aiming to improve artificial neural networks - networks that partly mimic functions of human brains. This thesis investigates a method, so-called shortcut ensembles, that might have the potential to improve their performance at least a little bit.

A central function of our brain is to make conscious decisions, and everyone knows that making important decisions is hard and can take a bit of time. We all experienced the situation where we had a decision that we liked to discuss with other people to collect their opinions before choosing ourselves. Usually, we feel that our decision has been approved if many people agree with it, and it is more likely that we made a good decision. Similarly, if you want an artificial neural network to make a decision it can be helpful to also ask more than one network for an opinion and then choose the decision that agrees with the most networks. This process is called ensemble learning and is already widely used.

In general, artificial brains need to be trained, similarly as we humans have to study if we want to increase our ability to make decisions. While training, there are several commonly used possibilities to check the performance of an artificial neural network. However, some of them are very costly in combination with ensemble learning, since then tedious computations need to be performed and lots of time is consumed. Here, the new method of creating shortcut ensembles could be able to help. The method suggests asking for opinions from other networks directly during the process of checking the performance.

After studying the shortcut ensembles, it was possible to conclude that they are indeed useful to approximate performances that can be reached when using ensemble learning with artificial neural networks. There is still more research needed, testing the new method on more complex systems, but it can make ensemble learning easier and more efficient. Furthermore, ensemble learning might become an even more used learning method, helping artificial brains to make better decisions. With this improvement, robots including artificial intelligence might not be able to rule the world, but they could give us greater help in everyday life.

Contents

List of Figures

1 Introduction

Artificial neural networks (ANNs) denote a collection of different algorithms, inspired by some functions of the human brain. They have the ability to learn during a training process, meaning they can adapt their weights, mimicking synapses, while processing given training data [1]. With this ability, ANNs play a key role in fields such as machine learning and artificial intelligence.

During the past decades, the significance of machine learning including artificial intelligence increased drastically. With the availability of large amounts of data, increasing computing powers, and improved algorithms, artificial neural networks grew and were capable to solve larger and larger problems. Their high flexibility and large scope lead to a constant search for improvement of their performances.

A main goal of ANNs is to generalize well, e.g., having a high generalization performance, which means that the network performs a mainly correct input-output mapping also for previously unseen data [1]. The unseen data can be called test set and contains data that was not used during creation or training of the ANN [1, 2]. However, it is usually assumed that the test data is generated with the same process as the training data [2].

To estimate the generalization performance of an ANN, different validation techniques can be used [2]. Similar to test data sets, validation data sets should be obtained in the same way as the training data [3]. Often, the available training data set is simply divided into training and validation data. The network is only trained on the training data and the validation set is used to approximate its generalization performance [2]. Validation techniques can further be used to decide on a suitable set of hyperparameters for the network [3]. Some common methods are: the holdout method for large data sets, and cross-validation or bootstrap for smaller data sets [2].

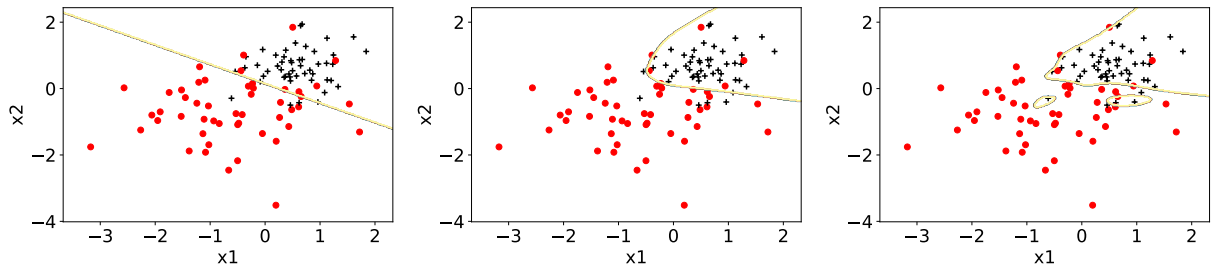
A common and powerful method to improve performances of ANNs is the creation of ensembles. Ensembles, sometimes also called committees or committee machines, are denoting a combination of several artificial neural networks, designed to improve the overall output [4]. The general idea is that the combination of several networks is leading to a better performance than even the best single member of the ensemble can provide [1]. In practice, ensembles often lead to high time consumption during validation if the data set is not large enough to allow for the holdout method. Hence, ensembles are often avoided when performing cross-validation.

To improve the estimate of the generalization performance given by the validation performance, this thesis investigates the process of building averages of the validation outputs during cross-validation, here referred to as shortcut ensemble. Later these average outputs are applied to a performance measure to obtain the validation performance of the shortcut ensemble. In order to evaluate this validation performance, it will be compared to validation and test performances from a single model as well as an actual ensemble.

2 Theory

2.1 Overfitting

Overfitting or overtraining is a process that can lead to a lower ability of a network to generalize well [1]. During this process, a network gets trained too well on the training data, thus it might also fit to noise of the training set [2]. An example of overfitting for a two-class classification problem is shown in Figure 1(c), where the decision boundary is also curved around single data points, that can be seen as noise. As opposed to overfitting, an ANN is said to be underfitting if it fits poorly to both training and test data [2]. Here, an example is illustrated in Figure 1(a), where the decision boundary partitions the classes badly in the training set. Figure 1(b) shows a well fitting model, where the decision boundary separates the classes well, without fitting to noise. Both overfitting and underfitting can be detected using validation methods during training.



(a) *(b)* *(c)*

Figure 1: Plots showing how well a model can fit to data. Plot (a) shows an underfitting model, plot (b) a well fitting model, and plot (c) an overfitting model.

2.2 Performance Measures

There exist several methods to measure the performance of an ANN. One common measure is the loss function, which quantifies the difference between an output y_n computed by an ANN and the corresponding target value d_n [5]. Here, $n = 1, 2, ..., N$ denotes a pattern of the data set. The output y_n is dependent on the input vector $\boldsymbol{x_n}$ and the weights vector $\boldsymbol{\omega}$ of the ANN, $y_n = y(\boldsymbol{x_n}, \boldsymbol{\omega})$. Therefore, the loss represents a function of the weights $\boldsymbol{\omega}$. In this project the cross-entropy error (CEE) for binary classification problems was used as loss function $E(\boldsymbol{\omega})$ [2]:

$$E(\boldsymbol{\omega}) = -\frac{1}{N} \sum_n \left[d_n \ln(y_n(\boldsymbol{x_n}, \boldsymbol{\omega})) + (1 - d_n) \ln(1 - y_n(\boldsymbol{x_n}, \boldsymbol{\omega})) \right], \tag{1}$$

where the factor $1/N$ is a scaling factor [2].

For multi-class classification problems, so-called one-hot encoding is introduced to avoid ordering of the classes. Hence, instead of assigning consecutive integers to the classes, the

target values are given by [2]:

$$d_{ni} = \begin{cases} 1, & \text{if } n \in I_i \\ 0, & \text{if } n \notin I_i \end{cases}, \tag{2}$$

where I_i stands for the ith class. Then, the categorical cross-entropy error for multiple classes can be defined as [2]:

$$E(\boldsymbol{\omega}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{I} d_{ni} \ln y_i(\boldsymbol{x_n}, \boldsymbol{\omega}), \tag{3}$$

where $y_i(\boldsymbol{x_n}, \boldsymbol{\omega})$ denotes the ith value of the output vector of pattern n. In general, the goal is to minimize the loss for an ANN [5].

Using classification problems for ANNs gives rise to several new performance measures in addition to the loss function. In this project, the accuracy is used, which stands for the fraction of correctly classified inputs compared to the total amount of inputs [2].

2.3 K-fold Cross-Validation

K-fold cross-validation is a commonly used validation method when working with smaller data sets, since statistical uncertainties lead to unreliable results when either training or validation sets are too small [3]. Figure 2 presents the method schematically.

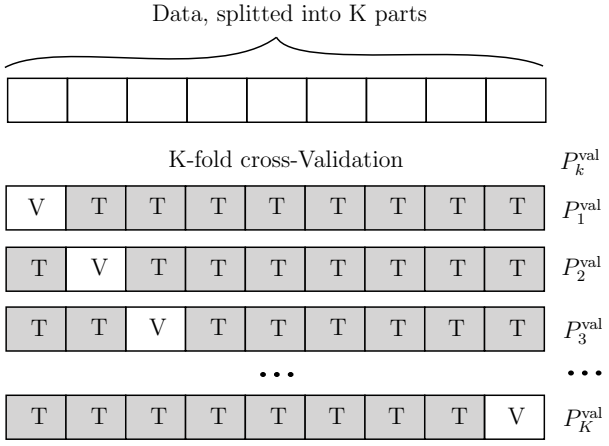


Figure 2: Schematic showing K-fold cross-validation, adapted from [2]. The data is split into K parts, then validation is performed in K folds, so that each partition of the data serves as validation set (V) once. The partitions that are not used for validation (T) are combined into the training set. Each validation leads to a validation performance P_k^{val} of the kth fold.

Following the description presented in [2], the data set is first split into K different parts of approximately equal size, where the split is usually performed randomly. Then K different folds are achieved, where in each fold one of the K data parts is declared as validation

set (V). The remaining parts (T) are joined together and form the training set for each fold, respectively. Now K models get trained, one for each fold, using the respective training and validation set. Last, the validation performance for the whole data set P^{val} is given by the average of validation performances P_k^{val}, $k = 1, 2, ..., K$, obtained from the K folds. Thus,

$$P^{\text{val}} = \frac{1}{K} \sum_{k=1}^{K} P_k^{\text{val}}. \tag{4}$$

Further, it is possible to repeat the procedure in C cycles using a different random split every time. In this case the validation performance is given by

$$P^{\text{val}} = \frac{1}{CK} \sum_{c=1}^{C} \sum_{k=1}^{K} P_{k,c}^{\text{val}}. \tag{5}$$

When performing cross-validation, only the validation performances are stored, whereas the trained models are discarded [2].

Cross-validation is used to estimate the generalization performance of an ANN with a specific set of hyperparameters. The idea is that a model trained on all data should perform similarly to the models trained on training data excluding partition K. Hence, for future data a network is trained with the whole training data set and the same hyperparameters as during cross-validation. In this research, the performance of such a single model on future data is denoted as P^{one}.

2.4 Ensembles

One method to improve generalization is the creation of ensembles. A simple possibility to combine the individual networks is by averaging their output [6], which is also the method used in this thesis. Considering the output y_m of the individual network m ($m = 1, 2, ..., M$), the output y_{ens} of an ensemble build by M members and using ensemble averaging, is obtained by [6]:

$$y_{\text{ens}} = \frac{1}{M} \sum_{m=1}^{M} y_m. \tag{6}$$

Using ensembles can improve the generalization performance, hence they can be used as regularization tool [2]. This can be illustrated by looking at the squared error of the ensemble, $(y_{\text{ens}}(\boldsymbol{x}) - \langle d|\boldsymbol{x}\rangle)^2$, where $\langle d|\boldsymbol{x}\rangle$ is the conditional expectation value of the target [2]. Then, it can be observed that the ensemble error is decomposed as [2]:

$$(y_{\text{ens}}(\boldsymbol{x}) - \langle d|\boldsymbol{x}\rangle)^2 = \underbrace{\frac{1}{M} \sum_{m} (y_m(\boldsymbol{x}) - \langle d|\boldsymbol{x}\rangle)^2}_{(\star)} - \underbrace{\frac{1}{M} \sum_{m} (y_m(\boldsymbol{x}) - y_{\text{ens}}(\boldsymbol{x}))^2}_{(\star\star)}. \tag{7}$$

In this decomposition, the first term ($\star$) is solely depending on the errors made by the individual members of the ensemble. Meanwhile, the second term ($\star\star$), which is subtracted,

depends on the difference among the ensemble members [7]. The second term is also being referred to as diversity term [2], and it shows that with increasing diversity among the networks, it is possible to decrease the total ensemble error in equation (7) [7]. An enlargement of variance among the members can be realised by allowing overfitting of the individual ANNs [8].

There exist several approaches how different ensemble members can be created. The strategy this project focuses on is K-fold splitting, which shows similarities to K-fold cross-validation. The process is sketched in Figure 3.

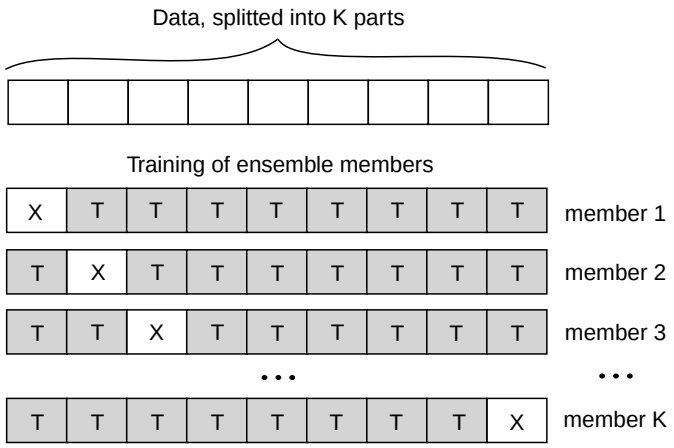


Figure 3: Schematic showing ensemble creation using K-fold splitting, adapted from [2]. Each member is trained on a slightly different data set, where one part (X) from the total data is left out.

The data set is again randomly split into K approximately equally large divisions. Then, K ensemble members are trained disregarding a different data division (X) for each member. If the process is repeated in C cycles, each starting with a new random split, the ensemble obtains a size of $C \cdot K$ members. Due to training the members on slightly different data sets, variance is introduced in between the members. It becomes clear that the diversity of the networks increases with smaller values of K. In contrast to cross-validation, the obtained models are kept, whereas the validation performances are usually not obtained [2].

3 Methods

3.1 Shortcut Ensemble

The shortcut ensemble aims to combine the two previously explained methods of K-fold cross-validation and ensemble creation using K-fold splitting. In general, K-fold cross-validation and K-fold splitting can have different K values independently from each other.

However, to be able to combine these methods, it needs to be required that K is identical for both. The goal of the shortcut ensemble is to find a simpler way of estimating the generalization performance of an ensemble. Figure 4 illustrates the construction of a shortcut ensemble.

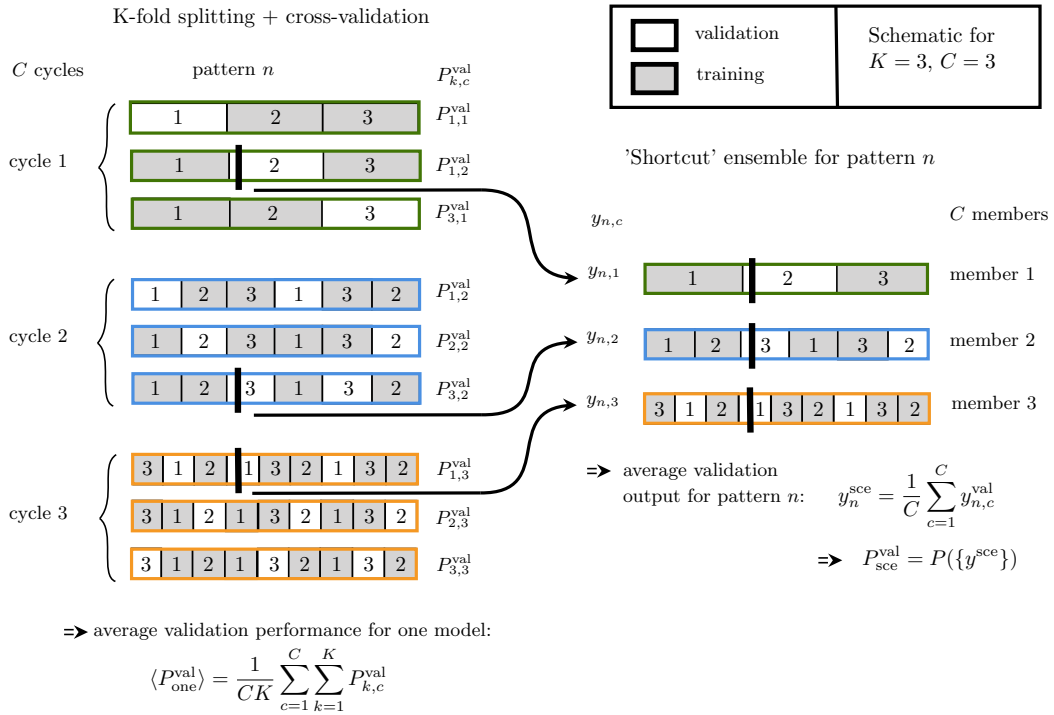


Figure 4: Schematic of how to obtain a shortcut ensemble for pattern n. First, the average of the validation outputs y_n^{sce} for pattern n from different cycles is build. Then the validation performance of the shortcut ensemble P_{sce}^{val} is determined by applying the averaged outputs from all patterns y^{sce} to a performance measure.

In the beginning, K-fold cross-validation is performed in C cycles, but this time, the trained models are kept. After C cycles, every pattern n of the data set has been validated C times, every time by a different trained model. Thus, for each pattern n, there exist C validation outputs $y_{n,c}^{\text{val}}$. Therefore, a shortcut ensemble can be created for each pattern consisting of C members. The output from the shortcut ensemble for pattern n is equal to the average of the outputs obtained during cross-validation:

$$y_n^{\text{sce}} = \frac{1}{C} \sum_{c=1}^{C} y_{n,c}^{\text{val}}. \tag{8}$$

The average outputs can then be used to acquire the validation performance $P_{\text{sce}}^{\text{val}}$ of the shortcut ensemble:

$$P_{\text{sce}}^{\text{val}} = P(\{y^{\text{sce}}\}). \tag{9}$$

The project compares three different estimates of the generalization performance obtained during cross-validation: $\langle P_{\text{one}}^{\text{val}} \rangle$ from one single model, $P_{\text{sce}}^{\text{val}}$ from the shortcut ensemble, and $\langle P_{\text{ens}}^{\text{val}} \rangle$ where an actual ensemble is created with K-fold splitting in each fold during validation. To enable comparison the same amount of folds and cycles are used as for cross-validation. Moreover, the generalization performance is tested on new unseen data, by both a single model giving P^{one} and an ensemble leading to P^{ens}. In total, this offers five different values, three validation performances and two performances on new data, that can be compared to each other in order to investigate the usefulness of a shortcut ensemble.

3.2 Software and Algorithms

The investigation of the shortcut ensemble is performed numerically using the machine learning platform TensorFlow [9] for python. For creating the ANNs the functions from Keras [10], a deep learning API (Application Programming Interface) of TensorFlow, are used. This library provides simple possibilities to construct ANNs with adjustable hyperparameters. For this project, only multi-layer perceptrons (MLPs) with a single hidden layer are used.

An outline of the algorithm used for this project is illustrated in Figure 5. As already sketched in Section 3.1, cross-validation is performed in C cycles and K folds. During validation, the validation performance of a single model and an actual ensemble is collected. Further, all validation outputs are saved and combined to shortcut ensemble outputs. In addition, the trained models from cross-validating the single model are kept. After the cross-validation process, it is left to find the generalization performances P^{one} and P^{ens}. Hence, a single model is trained on all training data and the already saved models from cross-validation are combined into one ensemble. Both the single model and the ensemble are applied on the test set to obtain their generalization performances.

In general, it was not the goal to optimize hyperparameters, but to concentrate on the comparison of different validation and test performances, when using one reasonable set of hyperparameters. The sets of hyperparameters that were specialized in the MLP models for different data sets are shown in Appendix A. The only hyperparameter that was not fixed in the beginning of the study is the number of hidden nodes, since that one has a high influence on overfitting and it needs to be found during training. Therefore, the obtainment of all validation and performance measures was embedded in a loop through different numbers of hidden nodes in the range of 5 to 100. Further, it was chosen to use $C = 3$ for the number of cycles and $K = 5$ for the number of folds. The number of cycles must be larger than 1 for a shortcut ensemble to be formed, but should not be too large in order to keep the running time of the program reasonable.

1. Load training and test data.
2. Define set of hyperparameters.

3. Loop through different numbers of hidden nodes or fix number of hidden nodes.

Cross-validation (C cycles, K folds)

3.1 Loop through cycles.

(a) Split training data in K folds.

(b) Loop through folds.

i. Train model.

ii. Save trained model.

iii. Obtain output of validation data.

iv. Calculate loss and accuracy.

v. Save output for shortcut ensemble.

vi. Build actual ensemble (with same K and C) and obtain validation performances.

3.2 Build average validation performance $\langle P^{\text{val}}_{\text{one}} \rangle$ for one model.

3.3 Build average output y^{sce} for shortcut ensemble.

3.4 Validate shortcut ensemble to obtain $P^{\text{val}}_{\text{sce}}$.

3.5 Build average validation performance $\langle P^{\text{val}}_{\text{ens}} \rangle$ of actual ensemble.

Test set

3.6 Use the same set of hyperparameters as before, train one model on all training data.

3.7 Validate the model using the test set to obtain P^{one}.

3.8 Combine saved, trained models from cross-validation to a test ensemble.

3.9 Run test ensemble on test data, obtain P^{ens}.

Figure 5: Sketch of the algorithm used to obtain the five performances that are compared to each other.

To approximate the errors on the performances, standard error estimates were used. The validation performances of a single model and an ensemble were build by creating averages from C cycles and K folds. Thus the standard error of the mean (SEM) is used:

$$\text{SEM} = \frac{\sigma}{\sqrt{C \cdot K}}, \tag{10}$$

where σ is the standard deviation of the validation performances $P_{k,c}^{\text{val}}$ of the different folds. For the test performances and the validation performance of the shortcut ensemble, the standard error (SE) was used:

$$\text{SE} = \frac{\sigma}{\sqrt{N}}. \tag{11}$$

Here, N is the total number of patterns and σ denotes the standard deviation of the validation performances from the different patterns.

3.3 Data Sets

Two different data sets were used in this research. The first one is a 6-dimensional synthetic data set for binary classification. Secondly, the MNIST database is used, providing a 10-class classification problem.

The use of synthetic data rises the possibility to control the exactness of the performance on test data, since the test set can be created arbitrarily large. With a test set that is large enough, the error on the generalization performance estimates P^{one} and P^{ens} will have a negligible size. This simplifies the comparison between the different validation performances.

The synthetic data is obtained using random data points from two six-dimensional normal distributions [11]. The two classes are chosen to have different width and to be overlapping. Furthermore, one distribution is assigned to the target value 0, whereas the other one is assigned to the target value 1. A similar two-dimensional data set is shown in Figure 6.

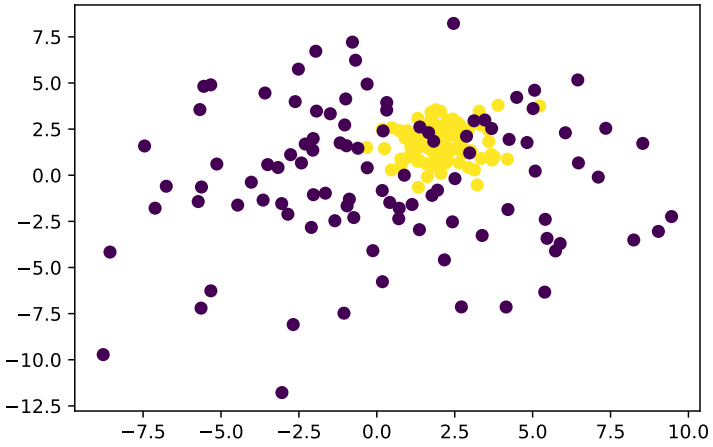


Figure 6: Plot of a 2D synthetic data set similar to the one used in this project.

For this project a training set of 400 input points with corresponding target values was used together with a test set consisting of 15000 data points. Before applying the data to the algorithm, it got preprocessed, such that each input in the training data has a variance of 1 and a mean of 0. The same preprocessing was used for both training and test data.

The MNIST database (Modified National Institute of Standards and Technology database) introduced by LeCun et al. [12] consists of greyscale images of handwritten digits. The database originates from the larger NIST database (National Institute of Standards and Technology database). The training set of MNIST consists of 60000 samples, whereas the training set includes 10000 images [13]. Some examples of images from the training set are shown in Figure 7.

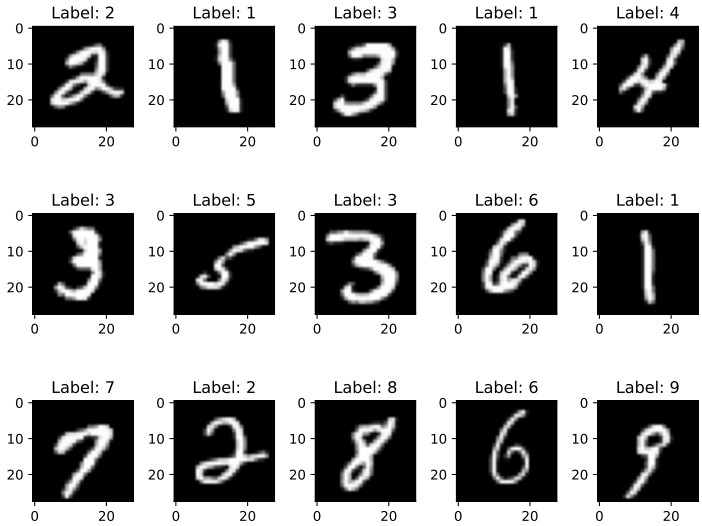
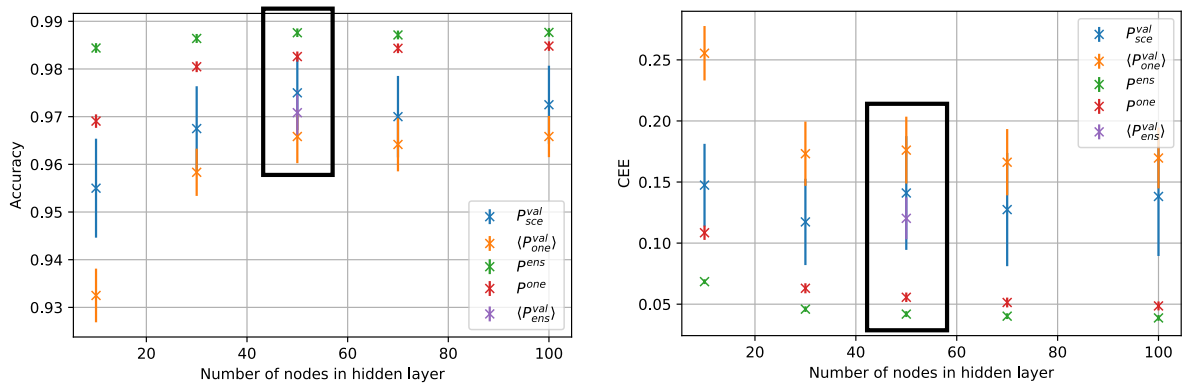


Figure 7: Plot of examples from the training set of the MNIST database.

4 Results

4.1 Six-Dimensional Synthetic Data

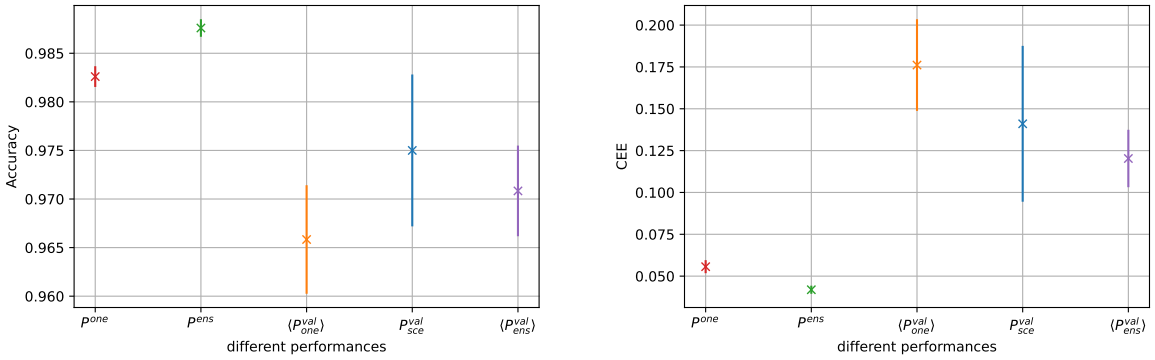
For a first comparison of the different performances introduced in Section 3.1, all performances except the validation performance of the actual ensemble $\langle P_{\text{ens}}^{\text{val}} \rangle$ are plotted for different numbers of hidden nodes. These plots are shown in Figure 8, where the accuracy (Figure 8(a)) and the cross-entropy error (Figure 8(b)) are used as performance measures. The plots are showing that 50 hidden nodes are an acceptable size of the network, which will be used from now on for further comparison of the estimates. Therefore, $\langle P_{\text{ens}}^{\text{val}} \rangle$ was also plotted for 50 nodes.



(a) *(b)*

Figure 8: Plot of the validation performance of one model and the shortcut ensemble, as well as the performance of one model and the ensemble for different numbers of hidden nodes. Plot (a) shows the accuracy, and plot (b) the cross-entropy error. As data set, the 6-dimensional synthetic data was used. The results for 50 nodes, highlighted by a box, are shown in greater detail in Figure 9.

Figure 9 shows the different performance estimates spread out using 50 hidden nodes. First, there are the two generalization performances P^{one} and P^{ens}. As expected, the ensemble gives a better performance with a lower CEE and higher accuracy, even though the differences are relatively small, approximately $5 \cdot 10^{-3}$ for the accuracy and $1.5 \cdot 10^{-2}$ for the CEE. This result is consistent with the theory. Thus, it is desirable to find a good approximation for P^{ens}. Furthermore, the standard errors on the test performances are indeed small enough to be neglected, which was desired by choosing a large test set.



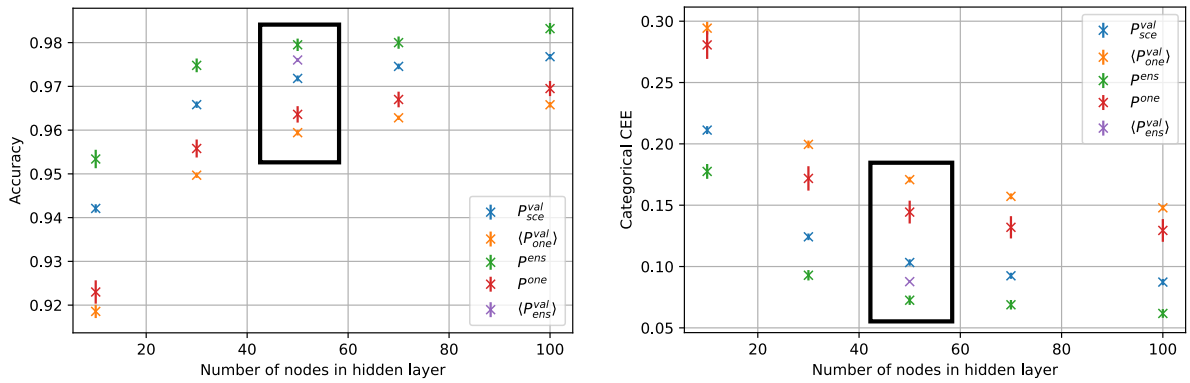
(a) *(b)*

Figure 9: Different performances for accuracy (a) and CEE (b) when using 50 hidden nodes are separated for better visualization of the error bars. As data set the 6-dimensional synthetic data was used.

Next, the three validation performances are compared to P^{ens}. First of all, the shortcut ensemble seems to offer a better estimate for the performance on future data then a single model. The standard error of the mean for $\langle P^{\text{val}}_{\text{one}} \rangle$ has the order of $5 \cdot 10^{-3}$ for the accuracy and $3 \cdot 10^{-2}$ for the loss. For $P^{\text{val}}_{\text{sce}}$ there is a standard error of approximately $8 \cdot 10^{-3}$ for accuracy and $5 \cdot 10^{-2}$ for the cross-entropy error. Considering the actual ensemble, the shortcut ensemble leads to a similar validation performance. In this case, the shortcut ensemble performs slightly better when looking at the accuracy, but slightly worse when considering the cross-entropy error. The standard errors of the mean for $\langle P^{\text{val}}_{\text{ens}} \rangle$ are approximately $5 \cdot 10^{-3}$ for accuracy and $2 \cdot 10^{-2}$ for the CEE. Although the goal was to estimate P^{ens}, it is still interesting to also look at the comparison to P^{one}. It can be noticed that the shortcut ensemble also estimates P^{one} better during validation than a single model.

4.2 MNIST Data Set

Similarly as for the synthetic data set, both accuracy and loss performances are first plotted for several numbers of hidden nodes. This can be seen in Figure 10, which shows the plots for the performances on the test set of one model and the ensemble as well as the validation performances of one model and the shortcut ensemble. A network with 50 hidden nodes seems to give acceptable results, even though the performance is further improved using a larger network with 70 or 100 hidden nodes. Considering that the computing time for the validation of an actual ensemble should stay reasonable, a network size of 50 nodes is chosen for further comparison. Thus, for 50 nodes also the validation performance of the ensemble is plotted.

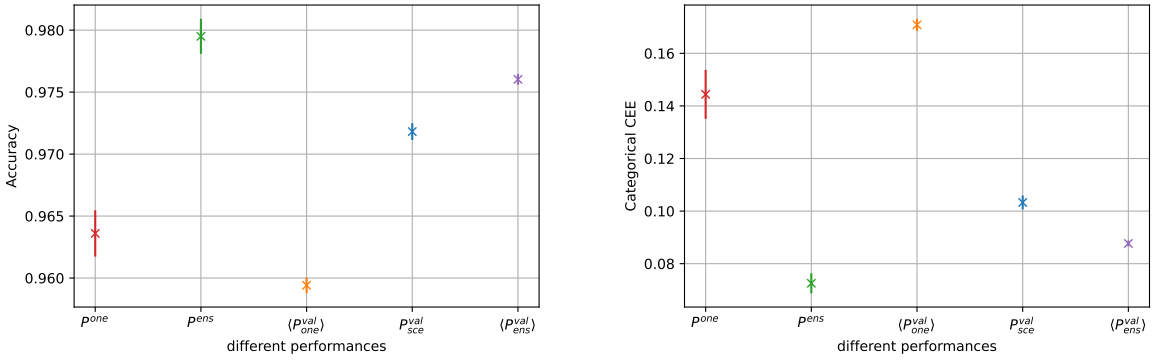


(a) *(b)*

Figure 10: Plot of the validation performance of one model and the shortcut ensemble, as well as the performance of one model and the ensemble for different numbers of hidden nodes. Plot (a) shows the accuracy, and plot (b) the categorical cross-entropy error, as data set, the MNIST database was used. The results for 50 nodes, highlighted by a box, are shown in greater detail in Figure 11.

In general, almost all error bars are of negligible sizes. Solely the standard errors for the test performance for one model are slightly larger, they are of the order of 10^{-3} for the accuracy and 10^{-2} for the loss.

Focusing on a network with 50 nodes (Figure 11), it appears that the generalization performance of the ensemble P^{ens} is clearly better than the generalization performance of a single model P^{one}. The performance of the ensemble is predicted fairly well by both $P^{\text{val}}_{\text{sce}}$ and $\langle P^{\text{val}}_{\text{ens}} \rangle$, but $\langle P^{\text{val}}_{\text{ens}} \rangle$ remains a better estimate. Compared to the validation performance of a single model $\langle P^{\text{val}}_{\text{one}} \rangle$, the ensemble and the shortcut ensemble approximate P^{ens} clearly better. However, $\langle P^{\text{val}}_{\text{one}} \rangle$ gives a good approximation of the generalization performance of one model.



(a) *(b)*

Figure 11: Different performances for accuracy (a) and categorical CEE (b) when using 50 hidden nodes are separated for better visualization of the error bars. As data set, the MNIST database was used.

5 Discussion

The results obtained in this research indicate that a shortcut ensemble is an appropriate tool to estimate the generalization performance P^{ens} of an ensemble. For both data sets, the 6-dimensional synthetic data and the MNIST database, the validation performance of the shortcut ensemble gave a better approximation of P^{ens} than one single model during cross-validation. This holds true for both performance measures that were used, accuracy and cross-entropy error.

Considering the 6-dimensional synthetic data set, the shortcut ensemble gave consistently a better validation result compared to the validation of one model. This holds true regarding the test performances of both one single model and the ensemble. With the use of synthetic data, it was possible to reduce the standard errors of the generalization performance to be of negligible sizes. In contrast to that, the standard error of the mean of $\langle P^{\text{val}}_{\text{one}} \rangle$ and $\langle P^{\text{val}}_{\text{ens}} \rangle$, as well as the standard error for the shortcut ensemble, are significant

and cannot be neglected.

One possibility to further reduce these uncertainties could be to use a larger number of cycles, although this can lead to longer computation times. Additionally, an increase of the number of folds could limit the error on the validation performances, but this possibility is often limited by the amount of available training data. Also, the number of folds influence the diversity of the ensemble. An increased number of folds leads to less variance among the training data of the ensemble members and, therefore, to more similar ensemble members. This can weaken the performance of an ensemble.

The comparison of $P_{\text{sce}}^{\text{val}}$ with $\langle P_{\text{ens}}^{\text{val}} \rangle$ does not lead to a clear conclusion in the case of the 6-dimensional synthetic data. The shortcut ensemble gave a better accuracy, whereas the actual ensemble led to a loss value closer to the loss of the generalization performance P^{ens}. However, due to the error bars on both quantities $P_{\text{sce}}^{\text{val}}$ and $\langle P_{\text{ens}}^{\text{val}} \rangle$ it is not possible to be certain about the tendency which of the validation performances leads to a better estimate. Nonetheless, it is reasonable to say that the shortcut ensemble can be used to approximate the validation result of the ensemble. Moreover, it can save time during the validation process compared to an actual ensemble. Hence, it can serve as en estimate of P^{ens}.

In contrast to the synthetic data set, the MNIST data set leads to a very clear result. Especially on the validation performances the errors are small enough to be neglected, due to the large data set provided by MNIST. Looking at how well the different estimates approximate P^{ens}, the use of the MNIST data set confirms that the shortcut ensemble is a closer approximation than the single model. Contrarily, the actual ensemble clearly yields an even better estimate of the generalization performance in this case, considering both accuracy and loss. For the use of a single model on future data, the single model provides the best estimate during validation.

Concentrating on the generalization performances on MNIST, the study clearly showed that ensembles should be created, when approaching MNIST with a simple MLP. The generalization performance of the ensemble was clearly better than of a single model. This trend was also observed among the validation performance. Both $P_{\text{sce}}^{\text{val}}$ and $\langle P_{\text{ens}}^{\text{val}} \rangle$ were visibly better than $\langle P_{\text{one}}^{\text{val}} \rangle$. Therefore, the shortcut ensemble indicated successfully that the use of an ensemble improves the generalization performance significantly.

A simple MLP might not be as good for approaching MNIST as, e.g., convolutional neural networks (CNNs), which make it possible to achieve very high accuracies on the test set [14]. However, this study focused on comparing different validation results. Hence, the performance of the ANN only had to be reasonably good but not optimized.

For further research it would be interesting to try out the method of a shortcut ensemble on a broader spectrum of network structures, tasks, and data sets. This research only applied MLPs with one hidden layer, whereas nowadays often different or more complex network structures are used, e.g., CNNs. The study of shortcut ensembles can also be extended by putting a higher focus on optimizing hyperparameters. Furthermore, this study focused solely on two classification problems, one binary and one multi-class problem. However, ANNs are often used for regression tasks as well. Therefore, it remains open to study the shortcut ensemble applied on various other settings.

6 Conclusion

The possibility of building a shortcut ensemble to improve the validation performance of ANNs was investigated by comparing it to a single model and an actual ensemble. Using the shortcut ensemble for validation gave better estimates of the generalization performance of an ensemble than a single model during validation. Further, it was observed that the shortcut ensemble leads to an approximation of the validation performance of an actual ensemble. Using the MNIST database, the shortcut ensemble managed to show that an ensemble leads to a clearly better performance on the data set compared to a single model. Overall, these findings show that a shortcut ensemble can replace an actual ensemble during validation, which yields a quicker possibility to approximate the validation performance of an ensemble during cross-validation.

Acknowledgments

First, I would like to thank my supervisor Patrik Edén for his great support and guidance during this project. In addition, I would like to express my appreciation to my friends and family, for their mental support and their feedback on this thesis.

References

[1] Haykin S. Neural Networks - a comprehensive foundation. 2nd ed. Prentice Hall; 1999.

[2] Ohlsson M, Edén P. Lecture notes on Introduction to Artificial Neural Networks and Deep Learning (FYTN14/EXTQ40/NTF005F). Lund University; 2021.

[3] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016. [Accessed 19 May 2022]. Available from: `http://www.deeplearningbook.org`.

[4] Re M, Valentini G. In: Ensemble methods: A review; 2012. p. 563-94.

[5] Jung A. Machine Learning: The Basics. Springer Nature Singapore Pte Ltd; 2022.

[6] Bishop CM. Pattern Recognition and Machine Learning. Springer Science+Business Media; 2006.

[7] Bishop CM. Neural Networks for Pattern Recognition. Oxford University Press, Inc.; 1995.

[8] Sollich P, Krogh A. Learning with ensembles: How overfitting can be useful. vol. 8; 1995. p. 190-6.

[9] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al.. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems; 2015. [Accessed 19 May 2022]. Available from: `https://www.tensorflow.org/`.

[10] Chollet F, et al.. Keras; 2015. [Accessed 19 May 2022]. Available from: `https://keras.io`.

[11] Developers N. NumPy Reference; 2022. [Accessed 19 May 2022]. Available from: `https://numpy.org/doc/stable/reference/index.html`.

[12] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278-324.

[13] LeCun Y, Cortes C, Burges CJC. THE MNIST DATABASE of handwritten digits;. [Accessed 19 May 2022]. Available from: `http://yann.lecun.com/exdb/mnist/`.

[14] An S, Lee M, Park S, Yang H, So J. An Ensemble of Simple Convolutional Neural Network Models for MNIST Digit Recognition; 2020.

A Hyperparameters

Table 1: Table presenting the hyperparameters used for the synthetic data set.

Hyperparameter	Value
number of hidden layers	1
activation function hidden layer	tanh
activation function output layer	sigmoidal
minimization method	Adam
learning rate	0.02
epochs	300
batch size	30

Table 2: Table presenting the hyperparameters used for the MNIST data set.

Hyperparameter	Value
number of hidden layers	1
activation function hidden layer	tanh
activation function output layer	softmax
minimization method	Adam
learning rate	0.01
epochs	20
batch size	200