Article

Estimating Neural Network's Performance with Bootstrap: a Tutorial

Umberto Michelucci [1,2,*], **Francesca Venturini** [1,3‡]

[1] TOELT LLC, Birchlenstr. 25, 8600 Dübendorf; umberto.michelucci@toelt.ai
[2] School of Computing, University of Portsmouth, Portsmouth PO1 3HE, UK; umberto.michelucci@port.ac.uk
[3] Institute of Applied Mathematics and Physics, Zurich University of Applied Sciences, Technikumstrasse 9, 8401 Winterthur, Switzerland
* Correspondence: umberto.michelucci@toelt.ai;

Abstract: Neural networks present the characteristics that the results are strongly dependent on the training data, the weight initialisation, and the hyperparameters chosen. The determination of the distribution of a statistical estimator, as the Mean Squared Error (MSE) or the accuracy, is fundamental to evaluate the performance of a neural network model (NNM). For many machine learning models, as linear regression, it is possible to analytically obtain information as variance or confidence intervals on the results. Neural networks present the difficulty of being not analytically tractable due to their complexity. Therefore, it is impossible to easily estimate distributions of statistical estimators. When estimating the global performance of an NNM by estimating the MSE in a regression problem, for example, it is important to know the variance of the MSE. Bootstrap is one of the most important resampling techniques to estimate averages and variances, between other properties, of statistical estimators. In this tutorial, the application of resampling techniques (including bootstrap) to the evaluation of neural networks' performance is explained from both a theoretical and practical point of view. Pseudo-code of the algorithms is provided to facilitate their implementation. Computational aspects, as the training time, are discussed since resampling techniques always require to run simulations many thousands of times and, therefore, are computationally intensive. A specific version of the bootstrap algorithm is presented that allows the estimation of the distribution of a statistical estimator when dealing with an NNM in a computationally effective way. Finally, algorithms are compared on synthetically generated and real data to demonstrate their performance.

Keywords: Neural Networks; Machine Learning; Bootstrap; Resampling; Algorithms

1. Introduction

An essential step in the design of a neural network model (NNM) is the definition of the neural network architecture [1]. In this tutorial, the analysis assumes that the network architecture design phase is completed and the parameters not varied any more. It is assumed that a dataset S is available. For the training and test of an NNM S is split in two parts, called training dataset (S_T) and validation dataset (S_V), with $S = S_T \cup S_V$ and $S_T \cap S_V = \varnothing$. The model is then trained on S_T. Afterwards, a given statistical estimator θ, is evaluated on both S_T and S_V (indicated with $\theta(S_T)$ and $\theta(S_V)$) to check for overfitting [1]. θ can be, for example, the accuracy in a classification problem, or the Mean Square Error (MSE) in a regression one. θ is clearly dependent (sometime strongly) on both S_T and S_V, as the NNM was trained on S_T. The metric evaluated on the validation dataset S_V may be indicated as

$$\theta(S_V) \equiv \theta_{S_T,S_V}(S_V) \tag{1}$$

to make its dependence on the two datasets S_T and S_V transparent. The difficulty in evaluating the distributions of $\theta_{S_T,S_V}(S_V)$ is that is enough to split the data differently (or in other words, to get different S_T and S_V), for the metric $\theta_{S_T,S_V}(S_V)$ to assume a different value. This poses the question on how to evaluate the performance of an NNM.

One of the most important characteristics of an NNM is its ability to generalise to unseen data, which means to maintain its performance when applied to any new dataset. If a model can predict a quantity with an accuracy of, for example, 80%, the accuracy should remain around 80% when the model is applied to new and unseen data. However, changing the training data will change the performance of any given NNM. To give an example of why giving one single number to measure the performance of a NNM can be misleading, let us consider the following example. Suppose we are dealing with a dataset where one of the features is the age. What would happen if, for sheer bad luck, one splits the dataset in one (S_T) with only young people, and one (S_V) with only old people? The trained NNM will, of course, not be able to generalize well to age groups that are different from those present in S_T. Therefore, the model performance, measured by the the statistical estimator θ, will drop significantly. This problem can only be identified by considering multiple splits and by studying the distribution of θ.

The only possibility to estimate the variance of the performance of the model given a dataset S is to split S in many different ways (and, therefore, obtain many different training and validation datasets). Then, the NNM has to be trained on each training datasets. Finally, the chosen statistical estimator θ can be evaluated on the respective validation datasets. This will allow to calculate the average and variance of θ, and use these values as an estimate of the performance of the NNM when applied to different datasets. This technique will be called the "split/train algorithm" in this tutorial. The major disadvantage of this technique is that it requires to repeat the training of the NNM for every split, and is therefore very time-consuming. If the model is large, it will require an enormous amount of computing time, making it not always a practical approach.

This paper shows how this difficulty can be overcome using resampling techniques to give an estimate of the average and the variance of metrics as the MSE or the accuracy, thus avoiding to train hundreds or thousands of models. An alternative to resampling techniques are so called ensemble methods, namely algorithms that train a set of models and then generate a prediction by taking a combination of each single prediction. The interested reader is referred to [2–9].

The goal of this tutorial is to present the main resampling methods, and discuss their applications and limitations when used with NNMs. With the information contained in this tutorial, a reader with some experience in programming should be able to implement them. This tutorial is not meant to be an exhaustive review of the mentioned algorithms. The interested reader is referred to the extensive list of references given in each section for a discussion of the theoretical limitations.

The main contributions of this tutorial are four. Firstly, it highlights the role of the Central Limit Theorem (CLT) in describing the distribution of averaging statistical estimators, like the MSE, in the context of NNMs. In particular in this work it is shown how the distribution of, for example, the MSE will tend to a normal distribution for increasing sample size, thus justifying the use of average and standard deviation to describe it. Secondly, it provides a short review of the main resampling techniques (hold-out set approach, leave-one-out cross validation, k-fold cross validation, jackknife, subsampling, split/train, bootstrap) with emphasis on the challenges when using neural networks. For most of the above-mentioned techniques the steps are described with the help of a pseudo-code to facilitate the implementation. Thirdly, bootstrap, split/train and the mixed approach between bootstrap and split/train are discussed in more depth, again with the help of pseudo-code, including the application to a synthetic and a real datasets. Finally, limitations and promising future research directions in this area are briefly discussed.

This tutorial is structured in the following way. In Section 2 the notation is explained, followed by a discussion of the CLT and its relevance for NNMs in Section 3. A short introduction to the idea behind bootstrap is presented in Section 4, while other resampling algorithms are discussed in Section 5. In Section 6 bootstrap, split/train and the mixed approach between bootstrap and split/train are explained in ore detail and compared. Practical examples with synthetic and with real data are described in Sections 7 and 8,

respectively. Finally an outlook and the conclusions are presented in Sections 9 and 10, respectively.

2. Notation

n independent, identically distributed (iid) observations will be indicated here with $X_n \equiv (x_1, ..., x_n)$. This dataset will come from a population described by a probability density function (PDF) F generally unknown:

$$x_1, ..., x_n \sim F \tag{2}$$

Let's assume that the statistical estimator θ (for example the average or the mean squared error) is a functional. Loosely speaking θ will be a mapping from the space of possible PDFs into real numbers $\mathbb{R}$. To make the concept clear, let's suppose that the estimator is the mean of the observations x_i. In this case

$$\theta(F) = \int_{-\infty}^{\infty} xF(x)dx \tag{3}$$

where it is clear that the right part of Equation (3) is a real number. Unfortunately, in all practical cases the "real" PDF F is unknown. Given a certain dataset X_n, the only possibility is to approximate the estimator θ with $\hat{\theta}_n \equiv \theta(F_n)$, where F_n indicates the empirical distribution obtained from X_n by giving a probability of $1/n$ to each observation x_i. This is the idea at the basis of the bootstrap algorithm, as it will be discussed in detail in Section 4.

3. Central Limit Theorem for an Averaging Estimator θ

A lot of mathematics has been developed to get at least the asymptotic distribution of $\hat{\theta}_n$ for $n \to \infty$ [10–13]. The CLT [14], also known as the Lindeberg–Lévy central limit theorem, enunciates that, considering a sequence of iid observations x_i with $\mu = \mathbb{E}[x_i]$ (the expected value of the inputs), and $\sigma^2 = \mathrm{Var}(x_i) = \mathbb{E}[(x_i - \mu)^2] < \infty$ then

$$\sqrt{n}(\overline{X}_n - \mu) \xrightarrow[n\to\infty]{} \mathcal{N}(0, \sigma^2) \tag{4}$$

where

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{5}$$

For any practical purposes, if n is large enough, the normal distribution $\mathcal{N}$ will give a good approximation of the distribution of the average of a dataset of n iid observations x_i.

Let's consider F to be a chi-squared distribution (notoriously asymmetric) with $k = 10$ [15] normalized to have the average equal to zero (panel(a) in Figure 1). Let's now calculate the average of $\overline{X}_n$ as in Equation (5) 10^6 times in three cases: $n = 2, 10, 200$. The results are shown in Figure 1, panels (b) to (d). When the sample size is small ($n = 2$, panel(b)), the distribution is clearly not symmetrical, but when the sample size grows (panels (c) and (d)), the distribution approximates the normal distribution. Figure 1 is a numerical demonstration of the CLT.

Typically, when dealing with neural networks both in regression and classification problems, one has to deal with complicated functions like the MSE, the cross-entropy, accuracy, or other metrics [1]. Therefore, it may seem that the central limit theorem does not play a role in any practical application involving NNMs. This, however, is not true. Consider, as an example, the MSE function of a given dataset of input observations x_i with average μ

$$\mathrm{MSE} \equiv \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2 \tag{6}$$

It is immediately evident that Equation 6 is nothing else than the average of the transformed inputs $(x_i - \mu)^2$. Note that the CLT does not make any assumption on the distribution of

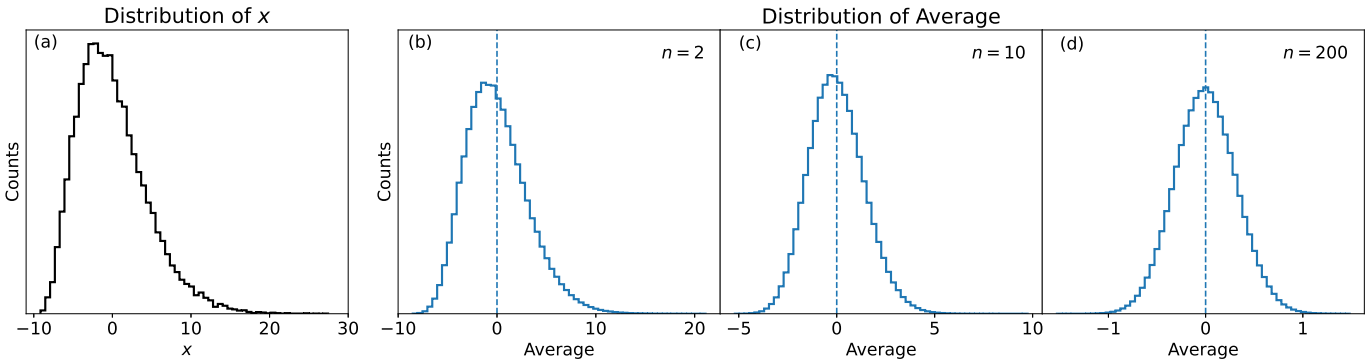

Figure 1. A numerical demonstration of the CLT. Panel (a) shows the asymmetric chi-squared distribution of random values for $k = 10$ [15], normalized to have the average equal to zero; in panel (b), (c) and (d) the distribution of the average of the random values is shown for sample size $n = 2$, $n = 10$ and $n = 200$ respectively.

the observations. Thus, the CLT is valid also for the average of observations that have been transformed (as long as the average and variance of the transformed observations remain finite). In other words, it is valid for both x_i and $(x_i - \mu)^2$. This can be formalized in the following Corollaries.

Corollary 1. *Given a dataset of observations $x_1, ..., x_n$ with a finite mean μ and variance σ^2, define the quantities $\delta_i \equiv (x_i - \mu)^2$. The limiting form of the distributions of the MSE (the average of the δ_i)*

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\delta_i \tag{7}$$

for $n \to \infty$ will be the normal distribution

$$\sqrt{n}(MSE - \mu(\delta_i)) \xrightarrow[n\to\infty]{} \mathcal{N}(0, \sigma(\delta_i)) \tag{8}$$

where with $\mu(\delta_i)$ and $\sigma(\delta_i)$ we have indicated the expected value and the standard deviation of the δ_i respectively.

Proof. The first thing to note is that since the x_i have finite mean and finite variance, it follows that $\delta_i \equiv (x_i - \mu)^2$ will also have finite mean and finite variance, and therefore the CLT can be applied to the average of the δ_i. By applying the CLT to the quantities δ_i Equation (8) is obtained. That concludes the proof. □

A numerical demonstration of this result can be clearly seen in Subsection 7.1. In particular, Figure 2 shows that the distribution of the MSEs approximates $\mathcal{N}$ when the sample size is large enough.

Note that Corollary 1 can be easily generalized to any estimator in the form

$$\theta = \frac{1}{n}\sum_{i=1}^{n} g(x_i) \tag{9}$$

if the quantities $g_i = g(x_i)$ have a finite mean $\overline{g}_i$ and variance $\sigma^2(g_i)$. For completeness, the Corollary 1 can be written in the general form

Corollary 2. *Given a dataset of observations $x_1, ..., x_n$ with a finite mean μ and variance σ^2, define the quantities $g_i \equiv g(x_i)$. It is assumed that the average $\mu(g_i)$ and the variance $\sigma(g_i)^2$ are finite. The limiting form of the distributions of the estimator θ*

$$\theta = \frac{1}{n}\sum_{i=1}^{n} g_i \tag{10}$$

for $n \to \infty$ will be the normal distribution

$$\sqrt{n}(\theta - \mu(g_i)) \xrightarrow[n\to\infty]{} \mathcal{N}(0, \sigma(g_i)) \tag{11}$$

Proof. The proof is trivial, as it is simply necessary to apply the central limit theorem to the quantities g_i since the θ is nothing else than the average of those quantities. □

The previous corollaries play a major role for neural networks. The implications of the final distributions of averaging metrics being Gaussian are

- the distribution is symmetric around the average, with the same number of observations are below and above it;
- the standard deviation of the distribution can be used as a statistical error, knowing that ca. 68% of the results will be in a region of $\pm\sigma$ around the average.

These results justify the use of the average of the statistical estimator, for example the MSE, and of its standard deviation as the only parameters needed to describe the network performance.

4. Bootstrap

Bootstrap is essentially a resampling algorithm. It was introduced by Efron in 1979 [16] and it offers a simulation-based approach for estimating, for example, the variance of statistical estimates of a random variable. It can be used in both parametric and non-parametric settings. The main idea is quite simple and consists of creating new datasets from an existing one by resampling with repetition. The discussion here is limited to the case of n observations that are iid (see Section 2 for more details). The estimator θ calculated on the simulated datasets will then approximate the estimator evaluated on the true population, which is unknown.

There is a huge amount of work done on the use of bootstrap and its theoretical limitations. The interested reader is referred to several general overviews [17–29], to the discussion on the evaluation of the confidence intervals [23,24,26,30,31], to the discussion on how to remove the iid hypothesis [23,32], and the application and limitations in various fields, from medicine to nuclear physics and geochemistry [5,23–26,33–38]. An analysis of the statistical theory on which the boostrap method is based, goes beyond this tutorial and will not be covered here.

The best way to understand the bootstrap technique is to describe it in pseudo-code, as this illustrate its steps. The original bootstrap algorithm proposed by Efron [16], can be seen in pseudo-code in Algorithm 4.1. In the Algorithm, N_s is an integer and indicates the number of new samples generated with the bootstrap algorithm.

As a consequence of Corollary 2 (Section 3), Algorithm 4.1 for a large enough N_s gives an approximation of the average and variance of the statistical estimator θ. Being the distribution Gaussian (albeit for $n \to \infty$), these two parameters describe it completely.

An important question is how big N_s should be. In the original article, Efron [39] suggests that N_s of the order of 100 is already enough to get reasonable estimates. Chernick [23] considers $N_s \approx 500$ already very large and more than enough. In the latter work is however indicated that, above a certain value of N_s, the error is due to the approximation of the true distribution F by the empirical distribution F_n rather than by the the low number of samples. Therefore, particularly given the computational power available today, it is not meaningful to argue if 100 or 500 are enough, as running the algorithm with many thousands samples will take only a few minutes on most modern computers. In many

Algorithm 4.1: Pseudo-code of the bootstrap algorithm.

Result: The estimate $\hat{\theta}_n$ of $\theta(F)$.

for $i = 1, ..., N_s$ **do**
  Generate a new sample $X^*_{n,i}$ selecting n elements from X_n *with repetitions*;
  Calculate $\hat{\theta}_{n,i} \equiv \theta(X^*_{n,i})$;
end
Evaluate $\hat{\theta}_n = \sum_{i=1}^{N_s} \hat{\theta}_{n,i} / N_s$;
Evalute $\sigma^2(\hat{\theta}_n) = \sum_{i=1}^{N_s} (\hat{\theta}_{n,i} - \hat{\theta}_n)^2 / N_s$;

practical applications using N_s between 5000 and 10000 is commonplace [23]. As a general rule of thumb, N_s is large enough when the distribution of the estimator starts to resemble a normal distribution.

The method described here is very advantageous when using neural networks because it allows an estimation of average and variance of quantities as the MSE or the accuracy without training the model hundreds or thousands of times, as will be described in Section 6. Thus, it is extremely attractive from a computational point of view and is a very pragmatic solution to a potentially very time-consuming problem. The specific details and pseudo-code of how to apply this method to NNMs will be discussed at length in Section 6.

5. Other Resampling Techniques

For completeness, in this section additional techniques, namely hold-out set approach, leave-one-out cross validation, k-fold cross Validation, jackknife and subsampling, are briefly discussed including their limitations. For an in-depth analysis the interested reader is referred to [40] and to the given literature.

5.1. Hold-out Set Approach

The simplest approach to estimate a statistical estimator is to divide randomly the dataset in two parts: a training S_T and a validation dataset S_V. The validation dataset is sometimes called *hold-out set* (from which derives the name of this technique). The model is trained on the *training* dataset S_T and then used to evaluate θ on the *validation* dataset S_V. $\theta(S_V)$ is used as an estimate of the expected value of θ. This approach is used also to identify if the model *overfits*, or, in other words, learns to unknowingly extract some of the noise in the data as if that would represent an underlying model structure [1]. The presence of overfitting is checked by comparing $\theta(S_T)$ and $\theta(S_V)$. A large difference is an indication that overfitting is present. The interested reader can find a detailed discussion in [1]. Such an approach is widely used but has two major drawbacks [40]. Firstly, since the split is done only once, it can happen that S_V is not representative of the entire datatset, as described in the age example in Section 1. Using this approach would not allow to identify such a problem, therefore giving the impression that the model has a very bad performance. In other words, this method is highly dependent on the dataset split. The second drawback is that splitting the dataset will reduce the number of observations available in the training dataset, therefore making the training of the model less effective.

The techniques explained in the following sections try to address with different strategies these two drawbacks.

5.2. Leave-One-Out Cross-Validation

Leave-one-out cross-validation (LOOCV) can be well understood with the pseudo-code outlined in Algorithm 5.1. This approach has the clear advantage that the model is trained on almost all observations, therefore address the second drawback of the hold-out

Algorithm 5.1: Leave-one-out cross-validation (LOOCV) Algorithm

Result: Estimate of a statistical estimator: $\hat{\theta}$

Define an NNM by fixing the hyperparameters;
for $i = 1, ..., n$ **do**
    Train the NNM on the dataset S where observation i has been removed. This dataset will have a size of $n - 1$;
    Evaluate the statistical estimator $\hat{\theta}^{(i)}$ by evaluating it on observation i;
end
Evaluate $\hat{\theta} = \sum_{i=1}^{n} \hat{\theta}^{(i)} / n$;

approach, namely that there are less observations for training. This has also the consequence that the LOOCV tends not to overestimate the estimate of the statistical estimator θ [40] as much as the hold-out approach. The second major advantage is that this approach will not present the problem that was described in the age example in the introduction as the training dataset will include almost all observations, and it will vary n times.

The approach, howver, has one major drawbacks: it is very computationally-expensive to implement, if the NNM training is resource-intensive. In all medium to large NNM models, this approach is simply not a realistic possibility.

5.3. k-fold Cross Validation

k-fold Cross Validation (k-fold CV) is similar to LOOCV but tries to address the drawback that the model has to be trained n times. The method involves randomly dividing the dataset in k groups (also called folds) of approximately equal size. The method is outlined in pseudo-code in Algorithm 5.2. Therefore, LOOCV is simply a special case

Algorithm 5.2: k-fold Cross Validation (k-fold CV) Algorithm

Result: Estimate of a statistical estimator: $\hat{\theta}$

Define an NNM by fixing the hyperparameters;
Split the dataset in k groups (*folds*): $S^{(i)}$, with $i = 1, ..., k$;
for $i = 1, ..., k$ **do**
    Train the NNM on the dataset $\bigcup_{i=1, i \neq j}^{k} S^{(i)}$, or in other words the dataset without the i^{th} fold ;
    Evaluate the statistical estimator $\hat{\theta}^{(i)}$ by evaluating it on $S^{(i)}$;
end
Evaluate $\hat{\theta} = \sum_{i=1}^{k} \hat{\theta}^{(i)} / k$;

of k-fold CV for $k = n$. Typical k values are 5 to 10. The main advantage of this method with respect to LOOCV is clearly computational. The model has to be trained only k times instead of n.

When the dataset is not big k-fold CV has the drawback that it reduces the number of observations available for training and for estimating θ, since each fold will be k times smaller than the original dataset. Additionally, it may be argued that using only few values of the statistical estimator (for example 5 or 10) to study its distribution is questionable [40].

5.4. Jackknife

The jackknife algorithm [41,42] is another resampling techniques developed first by Quenouille [41] in 1949. It consists of creating n samples by simply removing each time

one observations from the available $x_1, ..., x_n$. For example, one jackknife sample will be $x_2, ..., x_n$, with $n-1$ elements. To estimate θ, the statistical estimator will be evalauted on each sample (of size $n-1$). Note that jackknife may seem the exact same method as LOOCV but there is one major difference that is important to highlight. While in LOOCV $\hat{t}\theta$ is evaluated on the i^{th} observation held out (in other words on one single observation), in jackknife $\hat{t}\theta$ is evaluated on the remaining $n-1$ observations.

With this method, it is only possible to generate n samples that then can be used to evaluate an approximation of a statistical estimator. This is one significant limitation of the method compared to bootstrap. If the size of the dataset is small, only a limited number of samples will be available. The interested reader is referred to the reviews [25,26,43–47]. A second major limitation is that the jackknife estimation of an averaging estimator θ coincides with the average and standard deviation of the observations [23]. Thus using jackknife is not helpful to approximate $\theta(F)$.

For the limitations discussed above, this technique is not particularly advantageous when dealing with NNMs, and therefore seldom used in such a context, especially when compared with the bootstrap algorithm and its advantages.

5.5. Subsampling

Another technique for resampling is subsampling, achieved by simply choosing from a dataset X_n with n elements, $m < n$ elements without replacement. As a result, the samples generated with this algorithm have a smaller size than the initial dataset X_n. As the bootstrap algorithm, this one has been widely studied and used in the most different fields, from genomics [48,49] to survey science [50,51], finance [52,53] and, of course, statistics [23,54,55]. The two reviews [23] and [56] can be consulted by the interested reader. Precise conditions under which approximating with subsampling lead to a good approximation of the desired estimator can be found in [56–59].

Subsampling presents a fundamental difficulty when dealing with the average as statistical estimator. By its own nature, subsampling requires to consider a sample of smaller size m than the available dataset (of size n). As seen previously, the CLT enunciates that the standard deviation of a sample of size m will tend asymptotically to a normal distribution with a standard deviation that is proportional to the inverse of $\sqrt{m}$. That means that changing the sample size changes the standard deviation of the distribution of θ. Note that this is not a reflection of properties of the MSE, but only of the sample choosen. In the extreme case that $m = n$ (one could argue that this is not subsampling anymore, but let's consider it as extreme case) the average estimator will always have the same value, exactly the average of the inputs, since the subsampling samples are *without* replacement, and therefore the standard deviation will be zero. On the other hand, if $m = 1$, the standard deviation will increase significantly and will coincide with the standard deviation of the observations.

Therefore, the subsampling method presents the fundamental problem of the choice of m. Since there is not a general criterion to choose m, the distribution of θ will reflect the arbitrary choice of m and the properties of the data at the same time. This is why the authors do not think that the subsampling method is well suited to give a reasonable and interpretable estimate of the distribution of a statistical estimator, as the MSE.

6. Algorithms for Performance Estimation

As discussed in Section 1, the performance of an NNM can be assessed by estimating the variance of the statistical estimator θ. The distribution of θ can be evaluated by the split/train algorithm by splitting a given dataset S randomly N_s times in two parts $S_T^{(i)}$ and $S_V^{(i)}$, training each time an NNM on $S_T^{(i)}$ and evaluating the statistical estimator on $S_V^{(i)}$, with $i = 1, ..., N_s$. This algorithm is described with the help of pseudo-code in Subsection 6.1. This approach is unfortunately extremely time-consuming, as the training of the NNM is repeated N_s times.

As an alternative, the approach based on bootstrap is discussed in Subsection 6.2. This algorithm has the advantage over the split/train algorithm, to be very time-efficient, since the NNM is trained only once. After the training the distribution of a statistical estimator is then estimated by using a bootstrap approach on S_V.

6.1. Split/Train Algorithm

The goal of the algorithm is to estimate the distribution of a statistical estimator, like the MSE or accuracy, when considering the many variations of splits, or in other words, the different possible S_T and S_V. To do this, for each split a new model is trained, so to include the effect of the change of the training data.

The algorithm is described in pseudo-code in Algorithm 6.1. Fist, the dataset is randomly split. $S \setminus S_T^{(i)}$ indicates the dataset obtained by removing all $x_i \in S_T^{(i)}$ from S. Then, the training is performed and finally the distribution of a statistical estimator is evaluated.

Algorithm 6.1: Split/Train algorithm applied to the estimation of the distribution of a statistical estimator

Result: Averages and standard deviations of a statistical estimator: $\hat{\theta}_T$, $\hat{\theta}_V$, $\sigma(\hat{\theta}_T)$, $\sigma(\hat{\theta}_V)$

Define an NNM by fixing the hyperparameters;
for $i = 1, ..., N_s$ **do**
  Create $S_T^{(i)}$ from S by choosing m elements from S with $m < n$ without replacement;
  $S_V^{(i)} \leftarrow S \setminus S_T^{(i)}$;
  Train the NNM on $S_T^{(i)}$;
  Evaluate the statistical estimator $\theta_V^{(i)}$ by evaluating it on the results of the NNM when applied to $S_V^{(i)}$;
  Evaluate the statistical estimator $\theta_T^{(i)}$ by evaluating it on the results of the NNM when applied to $S_T^{(i)}$;
end
Evaluate $\hat{\theta}_T = \sum_{i=1}^{N_s} \hat{\theta}_T^{(i)} / N_s$;
Evalute $\sigma^2(\hat{\theta}_T) = \sum_{i=1}^{N_s} (\hat{\theta}_T^{(i)} - \hat{\theta})^2 / N_s$;
Evaluate $\hat{\theta}_V = \sum_{i=1}^{N_s} \hat{\theta}_V^{(i)} / N_s$;
Evalute $\sigma^2(\hat{\theta}_V) = \sum_{i=1}^{N_s} (\hat{\theta}_V^{(i)} - \hat{\theta})^2 / N_s$;

It is important to note that Algorithm 6.1 can be quite time-consuming since the NNM is trained N_s times. Thus, if the training of a single NNM takes a few hours, Algorithm 6.1 can easily take days and therefore may not be of any practical use. Remember that, as discussed previously, N_s should be at least of the order of 500 for the results to be meaningful. Larger values for N_s should be preferred, making this algorithm in many cases of no practical use.

From a practical perspective, besides the issue of the time, care must be taken in the implementation when automatizing Algorithm 6.1. In fact, if a script trains hundreds of models, it may happen that some will not converge. The results of these models will, therefore, be quite different from all the others. This may skew the distribution of the estimator. So, it is necessary to check that all the trained models reach approximately the

same value of the loss function. Models that do not converge should be excluded from the analysis, as they will clearly falsify the results.

It is important to note that the estimate of the distribution of an averaging estimator as the MSE, will always depend on the data used. Therefore the method, allows to assess the performance of an NNMs, measured as its generalisation ability when applied to unseen data.

6.2. Bootstrap

This section describes the application of bootstrap to estimate the distribution of a statistical estimator. Let's suppose one has an NNM trained on a given training dataset S_T and is interested in finding an estimate of the distribution of a statistical estimator, for example, the MSE or the accuracy. In this case, one can apply bootstrap, as described in Algorithm 4.1 to the validation dataset. The steps necessary are highlighted in pseudo-code in Algorithm 6.2.

Algorithm 6.2: Boostrap algorithm applied to the estimation of the distribution of a statistical estimator

Result: Average and standard deviation of a statistical estimator: $\hat{\theta}_n$ and $\sigma^2(\hat{\theta}_n)$

Define an NNM with by fixing the hyperparameters;
Create S_T from S by choosing m elements randomly from S with $m < n$ without replacement;
$S_V \leftarrow S \setminus S_T$;
Train the NNM on S_T;
for $i = 1, ..., N_s$ **do**
    Generate a new validation dataset $S_V^{(i)}$ by choosing $m - n$ elements from S_V with repetitions (create a bootstrap sample);
    Evaluate the statistical estimator $\hat{\theta}^{(i)}$ by evaluating it on the results of the NNM when applied to $S_V^{(i)}$;
end
Evaluate $\hat{\theta}_n = \sum_{i=1}^{N_s} \hat{\theta}^{(i)} / N_s$;
Evaluate $\sigma^2(\hat{\theta}_n) = \sum_{i=1}^{N_s} (\hat{\theta}^{(i)} - \hat{\theta})^2 / N_s$;

Note that Algorithm 6.2 does not require to train an NNM multiple times and is, therefore, quite time-efficient. From a practical perspective, it is important to note that the results of Algorithm 6.2 ($\hat{\theta}_n$ and $\sigma(\hat{\theta}_n)$) approximate the ones from Algorithm 6.1 ($\hat{\theta}_V$ and $\sigma(\hat{\theta}_V)$). In fact, the main difference between the algorithms is that in Algorithm 6.1 an NNM is trained each time on new data, while in Algorithm 6.2 the training is performed only once. Assuming that the dataset is big enough and that the trained NNMs converge to similar minima of the loss functions, it is reasonable to expect that their results will be comparable.

6.3. Mixed Approach between Bootstrap and Split/Train

The bootstrap approach, as described in the previous section, is computationally extremely attractive, but has one major drawback that needs further discussion. Similarly to the hold-out technique, the estimate of the average of the MSE and its variance are strongly influenced by the split: if S_V is not representative of the dataset (see the age example in Section 1), the Algorithm 6.2 will give the impression of a bad performance of the NNM.

A strategy to avoid this drawback is to run Algorithm 6.2 on the data a few times using different splits. As a rule of thumb, when the the average of the MSE and its variance

obtained by the different splits are comparable, these will likely be due to the NNM and not to the splits considered. Normally considering 5 to 10 splits will give an indication if the results can be used as intended. This approach has the advantage of being able to use a large number of samples (the number of bootstrap samples) to estimate a statistical estimator, without being insensitive to possible problematic cases due to splits where training and test parts are not representative of each other and of the entire dataset.

7. Application to Synthetic Data

To illustrate the application of bootstrap and to show its potential compared to the split/train approach, let's consider a regression problem. A synthetic dataset (x_i, y_i) with $i = 1, ..., n$ was generated with Algorithm 7.1. All the simulations in this paper were done with $n = 500$. The data correspond to a quadratic polynomial to which random noise taken from a uniform distribution was added. The goal in this problem is to extract the underlying structure (the polynomial function) from the noise. A simple NNMDOI: 10.1007/978-3-319-63754-9$_2$5*wasusedtopredictthey*$_i$ for each x_i. The NNM consists of a neural network with two layers, each having four neurons with the sigmoid activation functions, trained for 250 epochs, with a mini-batch size of 16 with the Adam optimizer [60]. Classification problems can be treated similarly and will not be discussed explicitly in this tutorial.

Algorithm 7.1: Algorithm for the synthetic data generation.

Result: A dataset (x_i, y_i) for $i = 1, ..., n$.

for $i = 1, ..., n$ **do**
  $x_i \leftarrow -5 + 0.02i$;
  $y_i \leftarrow 1/50 \cdot (2 + 3x_i + 4x_i^2)$;
  $\epsilon_i \leftarrow$ random sample from $U(-0.5, 0.5)$ with $U(-0.5, 0.5)$ the uniform distribution from -0.5 to 0.5;
  $y_i \leftarrow y_i + \epsilon_i$;
end

7.1. Results of Bootstrap

After generating the synthetic data, the dataset was split in 80%, used as training dataset (S_T), and 20% used for validation (S_V). Then, the bootstrap Algorithm 6.2 was applied, training the NNM on S_T and generating 1800 bootstrap samples from the S_V datasets. Finally, the MSE metric $\hat{\theta}_n$ on the bootstrap samples was evaluated and its distribution plotted in Figure 2. The black line is the distributions of the MSE on the 1800 bootstrap samples, while the red line is a Gaussian function with the same mean and standard deviations as the evaluated MSE values. Figure 2 shows that, as expected from Corollary 1, the distribution of the MSE values has a Gaussian shape. This justifies, as discussed, the use of the average and standard deviation of the MSE values to completely describe their distribution.

7.2. Comparison of Split/Train and Bootstrap Algorithms

Now let's compare Algorithms 6.1 and 6.2. The results for the MSE metric $\hat{\theta}_n$ are summarized in Figure 3. The black line shows the distribution obtained with Algorithm 6.1. The gray lines are the distributions obtained with Algorithm 6.2. To illustrate how the distribution depends on the data, $\hat{\theta}_n$ was evaluated for two different splits $(S_T^{(1)}, S_V^{(1)})$ and $(S_T^{(2)}, S_V^{(2)})$. For each of the two cases, a model was trained on the respective training datasets $S_T^{(1)}$ and $S_T^{(2)}$ and then Algorithm 6.2 was applied to $S_V^{(1)}$ and $S_V^{(2)}$. $N_s = 1800$ was used for both Algorithms 6.1 and 6.2. The corresponding averages of the metric distributions (vertical dashed lines) are very similar for the shown cases. Note that, depending on the split, the NNM may learn better or worse and, therefore, the average of

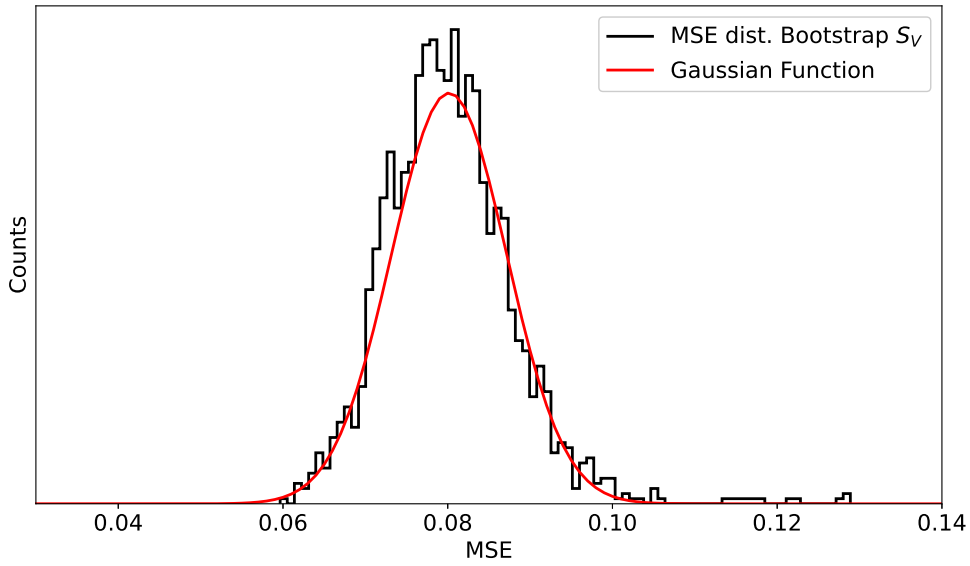

Figure 2. Distribution of the MSE values obtained by evaluating a trained NNM on 1800 bootstrap samples generated from S_V. The NNM used consists of a small neural network with two layers, each having 4 neurons with the sigmoid activation functions, trained for 250 epochs, with a mini-batch size of 16 with the Adam optimizer.

MSE obtained by using Algorithm 6.1 may vary, although in most of the cases will still be between roughly one σ of the average obtained by Algorithm 6.1. The second observation is that, although the average of the MSE may vary a bit, its variance stays quite constant. Finally, the results show that, as expected, the distributions have a Gaussian shape as expected from Corollary 1. As it was mentioned, Algorithm 6.2 is computationally more efficient. For example, on a system with an Intel 2.3 Ghz 8-Core i9 CPU, Algorithm 6.2 took less than a minute, while Algorithm 6.1 took over an hour.

The comparison between the split/train and bootstrap algorithms is summarized in Table 1, where the average of the MSE and its variance are listed. In the table also the result of k-fold cross validation and of the mixed approach (few split/train steps combined with several bootstrap samples) are reported. Note that for the mixed approach the reported average of the MSE and σ are obtained over the several splits.

Table 1. Comparison of the average of the MSE and its variance obtained with selected algorithms applied to the synthetic dataset.

Algorithm	**<MSE>**	σ
Split/Train (100 splits)	0.098	0.01
Simple Bootrap (100 bootstrap samples)	0.097	0.009
k-fold Cross Validation ($k = 5$)	0.106	0.008
Mixed approach (10 splits/100 bootstrap samples)	0.105	0.01

It is important to note that, in the split/train algorithm the NNM was trained on 100 different splits, in the bootstrap algorithm only on one. To avoid the dependence of the average of the MSE and of its variance on the single split, the mixed approach offers the possibility to check for dependencies from the split still remaining computationally efficient. For comparison, the k-fold CV is computationally efficient but the distribution of the MSE is composed of a very limited (here $k = 5$) values. So, even if the results of Table 1 are numerically similar, the difference is in the computation time and robustness of these values.

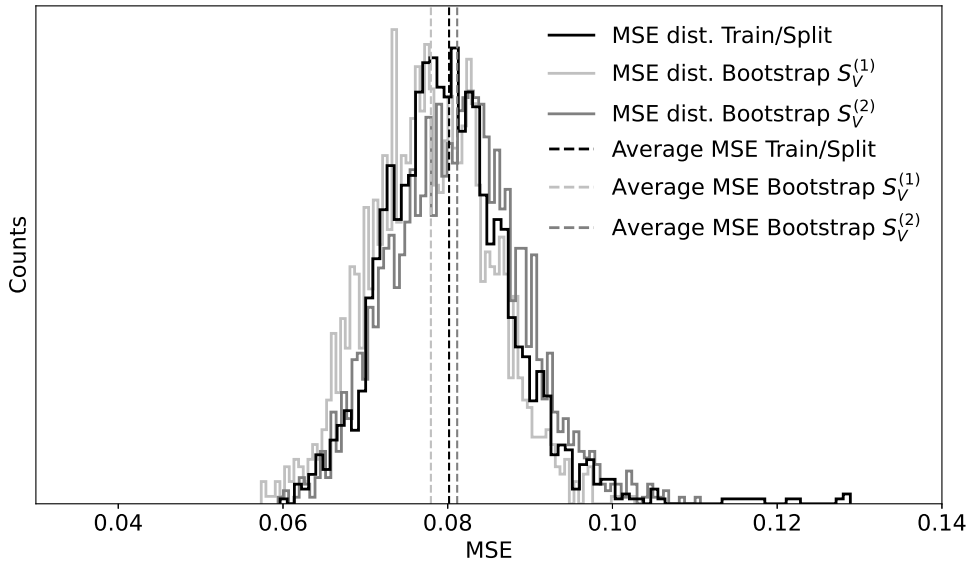

Figure 3. Distribution of the MSE values obtained by using Algorithms 6.1 (black line) and 6.2 (gray lines). The gray lines were obtained by generating 1800 bootstrap samples from two different validation datasets $S_V^{(1)}$ and $S_V^{(2)}$ as described in the text. The vertical dashed lines indicate the average of the respective distributions.

8. Application to Real Data

To test the different approaches on real data, the Boston dataset [61] was chosen. This dataset contains house prices collected by the US Census Service in the area of Boston, US. Each observation has 13 features and the target variable is the average price of a house with those features. The interested reader can found the details in [62].

Table 2. Comparison of the average of the MSE and its variance obtained with selected algorithms applied to the Boston Dataset.

Algorithm	**<MSE>** (1000 USD)	σ (1000 USD)
Split/Train (100 splits)	75.1	18.4
Simple Bootrap (100 bootstrap samples)	74.7	17.8
k-fold Cross Validation ($k = 5$)	77.2	17.2
Mixed Bootrap (10 splits/100 bootstrap samples)	75.0	15.2

The results are summarized in Table 2. As visible from Table 2, the average of the MSE and its variance obtained with the different algorithms are numerically similar, as obtained on the synthetic dataset.

9. Limitations and Promising Research Developments

It is important to note that the algorithms discussed in this work are presented as practical implementation techniques, but are not based on a theoretical mathematical proof, with exception of Section 3. One of the main obstacles in proving such results is the intractability of neural networks (see for example [63]) due to their complexity and non-linearity. Although not yet based on mathematical proofs, those approaches are important tools to be able to estimate the value of a statistical estimator without incurring in problems as described, for example, in the example with the age in the introduction.

The field offers several promising research directions. One interesting question is whether the degradation of the performance of NNMs due to small datasets differs when using the different techniques. This would enable to choose which technique works better with less data. Another promising field is to study different network topologies and the

role of the architecture on the resampling results. It is not obvious at all, that different network architectures behave the same when using different resampling results. Finally, it would be important to support the described observations arising from simulations with a theoretical basis. This, in the opinion of the authors, would be one of the most important research directions to purse in the future.

10. Conclusions

This tutorial shows how the distributions of an average estimator, as the MSE or the accuracy, tends asymptotically to a Gaussian shape. The estimation of the average and variance of such an estimator, the only two parameters needed to describing its distribution, are therefore of great importance when working with NNMs. They allow to assess the performance of an NNM, perceived as its ability to generalise when applied to unseen data.

Classical resampling techniques are explained and discussed, with focus on their application with NNMs: hold-out set approach, leave-one-out cross validation, k-fold cross validation, jackknife, bootstrap and split/train. The pseudo-code included is meant to facilitate the implementation. The relevant practical aspects, as the computation time, are discussed. The application and performance of bootstrap and split/train algorithms are demonstrated with the help of synthetically generated and real data.

The mixed bootstrap algorithm is proposed as a technique to obtain reasonable estimates of the distribution of statistical estimator in a computationally efficient way. The results are comparable with the ones obtained with the much more computationally-intensive algorithms, like the split/train one.

11. Software

The code used in this tutorial for the simulations is available at [64].

Author Contributions: Conceptualization, Umberto Michelucci and Francesca Venturini; methodology, Umberto Michelucci; software, Umberto Michelucci; validation, Umberto Michelucci and Francesca Venturini; original draft preparation, Umberto Michelucci; review and editing, Francesca Venturini and Umberto Michelucci.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MSE	Mean Square Error
NNM	Neural Network Model
CLT	Central Limit Theorem
iid	independent identically distributed
PDF	Probability Density Function
LOOCV	Leave-one-out cross-validation
CV	Cross Validation

References

1. Michelucci, U. *Applied Deep Learning - A Case-Based Approach to Understanding Deep Neural Networks*; APRESS Media, LLC, 2018.
2. Izonin, I.; Tkachenko, R.; Verhun, V.; Zub, K. An approach towards missing data management using improved GRNN-SGTM ensemble method. *Engineering Science and Technology, an International Journal* **2020**. doi:https://doi.org/10.1016/j.jestch.2020.10.005.
3. Tkachenko, R.; Izonin, I.; Kryvinska, N.; Dronyuk, I.; Zub, K. An Approach towards Increasing Prediction Accuracy for the Recovery of Missing IoT Data based on the GRNN-SGTM Ensemble. *Sensors* **2020**, *20*. doi:10.3390/s20092625.
4. Izonin, I.; Tkachenko, R.; Vitynskyi, P.; Zub, K.; Tkachenko, P.; Dronyuk, I. Stacking-based GRNN-SGTM Ensemble Model for Prediction Tasks. 2020 International Conference on Decision Aid Sciences and Application (DASA). IEEE, 2020, pp. 326–330.
5. Alonso-Atienza, F.; Rojo-Álvarez, J.L.; Rosado-Muñoz, A.; Vinagre, J.J.; García-Alberola, A.; Camps-Valls, G. Feature selection using support vector machines and bootstrap methods for ventricular fibrillation detection. *Expert Systems with Applications* **2012**, *39*, 1956–1967.

6. Dietterich, T.G. Ensemble Methods in Machine Learning. Multiple Classifier Systems; Springer Berlin Heidelberg: Berlin, Heidelberg, 2000; pp. 1–15.
7. Perrone, M.P.; Cooper, L.N. When networks disagree: Ensemble methods for hybrid neural networks. Technical report, BROWN UNIV PROVIDENCE RI INST FOR BRAIN AND NEURAL SYSTEMS, 1992.
8. Tkachenko, R.; Tkachenko, P.; Izonin, I.; Vitynskyi, P.; Kryvinska, N.; Tsymbal, Y. Committee of the combined RBF-SGTM neural-like structures for prediction tasks. International Conference on Mobile Web and Intelligent Information Systems. Springer, 2019, pp. 267–277.
9. Sagi, O.; Rokach, L. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2018**, *8*, e1249.
10. Efron, B.; Tibshirani, R.J. *An introduction to the bootstrap*; CRC press, 1994.
11. Good, P.I. *Introduction to statistics through resampling methods and R*; John Wiley & Sons, 2013.
12. Chihara, L.; Hesterberg, T. *Mathematical statistics with resampling and R*; Wiley Online Library, 2011.
13. Williams, J.; MacKinnon, D.P. Resampling and distribution of the product methods for testing indirect effects in complex models. *Structural equation modeling: a multidisciplinary journal* **2008**, *15*, 23–51.
14. Montgomery, D.C.; Runger, G.C. *Applied statistics and probability for engineers*; Wiley, 2014.
15. Johnson, N.; Kotz, S.; Balakrishnan, N. Chi-squared distributions including chi and Rayleigh. *Continuous univariate distributions* **1994**, pp. 415–493.
16. Efron, B. Bootstrap Methods: Another Look at the Jackknife. *Ann. Statist.* **1979**, *7*, 1–26. doi:10.1214/aos/1176344552.
17. Paass, G. Assessing and improving neural network predictions by the bootstrap algorithm. Advances in Neural Information Processing Systems. Citeseer, 1993, pp. 196–203.
18. González-Manteiga, W.; Prada Sánchez, J.M.; Romo, J. The bootstrap-A review **1992**.
19. Lahiri, S. Bootstrap methods: A review. In *Frontiers in statistics*; World Scientific, 2006; pp. 231–255.
20. Swanepoel, J. Invited review paper a review of bootstrap methods. *South African Statistical Journal* **1990**, *24*, 1–34.
21. Hinkley, D.V. Bootstrap methods. *Journal of the Royal Statistical Society: Series B (Methodological)* **1988**, *50*, 321–337.
22. Efron, B.; others. Second thoughts on the bootstrap. *Statistical Science* **2003**, *18*, 135–140.
23. Chernick, M.R. *Bootstrap methods: A guide for practitioners and researchers*; Vol. 619, John Wiley & Sons, 2011.
24. Lahiri, S. Bootstrap methods: a practitioner's guide-MR Chernick, Wiley, New York, 1999, pp. xiv+ 264 , ISBN 0-471-34912-7. *Journal of Statistical Planning and Inference* **2000**, *1*, 171–172.
25. Chernick, M.; Murthy, V.; Nealy, C. Application of bootstrap and other resampling techniques: Evaluation of classifier performance. *Pattern Recognition Letters* **1985**, *3*, 167–178.
26. Efron, B. *The jackknife, the bootstrap and other resampling plans*; SIAM, 1982.
27. Zainuddin, N.H.; Lola, M.S.; Djauhari, M.A.; Yusof, F.; Ramlee, M.N.A.; Deraman, A.; Ibrahim, Y.; Abdullah, M.T. Improvement of time forecasting models using a novel hybridization of bootstrap and double bootstrap artificial neural networks. *Applied Soft Computing* **2019**, *84*, 105676.
28. Li, X.; Deng, S.; Wang, S.; Lv, Z.; Wu, L. Review of small data learning methods. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC). IEEE, 2018, Vol. 2, pp. 106–109.
29. Reed, S.; Lee, H.; Anguelov, D.; Szegedy, C.; Erhan, D.; Rabinovich, A. Training deep neural networks on noisy labels with bootstrapping. *arXiv preprint arXiv:1412.6596* **2014**.
30. Diciccio, T.J.; Romano, J.P. A review of bootstrap confidence intervals. *Journal of the Royal Statistical Society: Series B (Methodological)* **1988**, *50*, 338–354.
31. Khosravi, A.; Nahavandi, S.; Srinivasan, D.; Khosravi, R. Constructing optimal prediction intervals by using neural networks and bootstrap method. *IEEE transactions on neural networks and learning systems* **2014**, *26*, 1810–1815.
32. Gonçalves, S.; Politis, D. Discussion: Bootstrap methods for dependent data: A review. *Journal of the Korean Statistical Society* **2011**, *40*, 383–386.
33. Chernick, M.R. *The essentials of biostatistics for physicians, nurses, and clinicians*; Wiley Online Library, 2011.
34. Pastore, A. An introduction to bootstrap for nuclear physics. *Journal of Physics G: Nuclear and Particle Physics* **2019**, *46*, 052001.
35. Sohn, R.; Menke, W. Application of maximum likelihood and bootstrap methods to nonlinear curve-fit problems in geochemistry. *Geochemistry, Geophysics, Geosystems* **2002**, *3*, 1–17.
36. Anirudh, R.; Thiagarajan, J.J. Bootstrapping graph convolutional neural networks for autism spectrum disorder classification. ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 2019, pp. 3197–3201.
37. Gligic, L.; Kormilitzin, A.; Goldberg, P.; Nevado-Holgado, A. Named entity recognition in electronic health records using transfer learning bootstrapped neural networks. *Neural Networks* **2020**, *121*, 132–139.
38. Ruf, J.; Wang, W. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance, Forthcoming* **2020**.
39. Efron, B. Better bootstrap confidence intervals. *Journal of the American statistical Association* **1987**, *82*, 171–185.
40. Gareth, J.; Daniela, W.; Trevor, H.; Robert, T. *An introduction to statistical learning: with applications in R*; Spinger, 2013.
41. Quenouille, M.H. Approximate tests of correlation in time-series. *Journal of the Royal Statistical Society. Series B (Methodological)* **1949**, *11*, 68–84.
42. Cameron, A.C.; Trivedi, P.K. *Microeconometrics: methods and applications*; Cambridge university press, 2005.

43. Miller, R.G. The jackknife-a review. *Biometrika* **1974**, *61*, 1–15.
44. Efron, B. Nonparametric estimates of standard error: the jackknife, the bootstrap and other methods. *Biometrika* **1981**, *68*, 589–599.
45. Wu, C.F.J.; others. Jackknife, bootstrap and other resampling methods in regression analysis. *the Annals of Statistics* **1986**, *14*, 1261–1295.
46. Efron, B.; Stein, C. The jackknife estimate of variance. *The Annals of Statistics* **1981**, pp. 586–596.
47. Shao, J.; Wu, C.J. A general theory for jackknife variance estimation. *The Annals of Statistics* **1989**, pp. 1176–1197.
48. Bickel, P.J.; Boley, N.; Brown, J.B.; Huang, H.; Zhang, N.R.; others. Subsampling methods for genomic inference. *The Annals of Applied Statistics* **2010**, *4*, 1660–1697.
49. Robinson, D.G.; Storey, J.D. subSeq: determining appropriate sequencing depth through efficient read subsampling. *Bioinformatics* **2014**, *30*, 3424–3426.
50. Quiroz, M.; Villani, M.; Kohn, R.; Tran, M.N.; Dang, K.D. Subsampling MCMC-An introduction for the survey statistician. *Sankhya A* **2018**, *80*, 33–69.
51. Elliott, M.R.; Little, R.J.; Lewitzky, S. Subsampling callbacks to improve survey efficiency. *Journal of the American Statistical Association* **2000**, *95*, 730–738.
52. Paparoditis, E.; Politis, D.N. Resampling and subsampling for financial time series. In *Handbook of financial time series*; Springer, 2009; pp. 983–999.
53. Bertail, P.; Haefke, C.; Politis, D.N.; White Jr, H.L. A subsampling approach to estimating the distribution of diversing statistics with application to assessing financial market risks. *UPF, Economics and Business Working Paper* **2001**.
54. Chernozhukov, V.; Fernández-Val, I. Subsampling inference on quantile regression processes. *Sankhyā: The Indian Journal of Statistics* **2005**, pp. 253–276.
55. Politis, D.N.; Romano, J.P.; Wolf, M. Subsampling for heteroskedastic time series. *Journal of Econometrics* **1997**, *81*, 281–317.
56. Politis, D.N.; Romano, J.P.; Wolf, M. *Subsampling*; Springer Science & Business Media, 1999.
57. Delgado, M.A.; Rodrıguez-Poo, J.M.; Wolf, M. Subsampling inference in cube root asymptotics with an application to Manski's maximum score estimator. *Economics Letters* **2001**, *73*, 241–250.
58. Gonzalo, J.; Wolf, M. Subsampling inference in threshold autoregressive models. *Journal of Econometrics* **2005**, *127*, 201–224.
59. Politis, D.N.; Romano, J.P. Large sample confidence regions based on subsamples under minimal assumptions. *The Annals of Statistics* **1994**, pp. 2031–2050.
60. Kingma, D.P.; Ba, J.A. Adam: A method for stochastic optimization. In Proceedings of 3rd. *In Proceedings of 3rd International Conference on Learning Representations, ICLR 2015* **2015**, pp. 1–15.
61. Harrison Jr, D.; Rubinfeld, D.L. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management* **1978**, *5*, 81–102.
62. Original paper by Harrison, D.; Rubinfeld, D. The Boston Housing Dataset Website. https://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html, 1996.
63. Jones, L.K. The computational intractability of training sigmoidal neural networks. *IEEE Transactions on Information Theory* **1997**, *43*, 167–173. doi:10.1109/18.567673.
64. Michelucci, U. Code for *Estimating Neural Network's Performance with Bootstrap: a Tutorial*. https://github.com/toelt-llc/NN-Performance-Bootstrap-Tutorial, 2021.