Accounting for Imputation Uncertainty During Neural Network Training

Thomas Ranvier[1,2][0000−0001−9250−9530], Haytham Elghazel[1,2], Emmanuel Coquery[1,2], and Khalid Benabdeslem[1,2]

[1] Univ Lyon, UCBL, CNRS, INSA Lyon, LIRIS, UMR5205
43 bd du 11 Novembre 1918, 69622 Villeurbanne, France
[2] {thomas.ranvier,haytham.elghazel,emmanuel.coquery,khalid.benabdeslem}@univ-lyon1.fr

Abstract. In this paper we are interested in dealing with missing values in a machine learning context, and more especially when training a neural network. We focus on improving neural network training by reducing the potential biases that can occur during the training phase on artificially imputed datasets. We do so by taking into account the between-variance that can be observed between multiple imputations. We propose two new imputation frameworks, *S-HOT* and *M-HOT*, that can be used to train neural networks on completed data in a less biased way, leading to models able of more generalization, and so, to better inference results. We perform extensive comparative experiments and statistically assess the results on both benchmark and real-world datasets. We show that our frameworks compete against and even outperform existing imputation frameworks, while being both useful in different settings. We make our entire code publicly accessible to facilitate reproduction of our experimental results.

Keywords: Data Imputation · Imputation Uncertainty · Between-variance · Optimization · Neural Networks.

1 Introduction

Two types of variances occur when multiply imputing an incomplete dataset: the within-variance and the between-variance. The within-variance corresponds to the variance within each imputed dataset. The between-variance corresponds to the variance in-between every imputed dataset. In real scenarios, it is usually not possible to know the within-variance, since most imputation methods estimate a fixed value in place of missing ones without outputting any probability measure. However, the between-variance can easily be computed between a set of completed datasets. In the following, we refer to the between-variance as imputation uncertainty.

In this paper, we make the difference between imputation methods, which aim to impute missing values in a dataset, and imputation frameworks such as Single-Imputation or Multiple-Imputation, which rely on any imputation method to deal with missing data. Our contributions are imputation frameworks, they aim to train neural networks while taking into account imputation uncertainty and can rely on any imputation method. Note that we are not interested in comparing imputation methods performance.

When dealing with incomplete data, a naive and overly used approach is Single-Imputation [11]. That is, to arbitrarily choose an imputation method, use it to impute the dataset, and treat the completed dataset as the new real dataset to perform any kind of future analysis. When training a neural network or similarly strong learners on completed data, those models are usually able to generalize enough to limit the bias occurring due to imputation uncertainty. This leads to good enough results so that one does not look for better ways to deal with missing values. It has even been shown that, when using strong inference models, almost any imputation asymptotically leads to optimal prediction [6]. This is probably one of the main reasons why accounting for imputation uncertainty has not been widely researched. However, those models might reach good results in such situations but they are biased by imputation uncertainty, we show that accounting for this uncertainty during their training phase helps to reach even better prediction results.

In this paper, we propose two imputation frameworks, *S-HOT* and *M-HOT*, that aim to train neural networks on imputed datasets while accounting for imputation uncertainty to reduce the natural bias occurring when training on completed datasets. Those frameworks are to be used in different situations, *S-HOT* is adapted to train a unique and large neural network, *M-HOT* can be used to train multiple learners in an ensemble way and reach extremely good prediction results at the expense of a higher computational cost. We conduct extensive experiments to compare our two frameworks with the existing Single-Imputation and Multiple-Imputation frameworks. We then perform statistical analysis to assess the obtained results on both benchmark and real-world medical datasets. We show that our proposed frameworks compete against, and even outperform existing imputation frameworks. This paper is a first step towards finding better ways to deal with missing values imputation in machine learning. We hope that it spikes the interest of other machine learning researchers throughout the world on this important and largely overlooked matter in the machine learning literature.

The complete results, supplementary material and source code used to conduct our experiments are available at the following GitHub repository[3].

In the rest of the paper, we first present related works on imputation frameworks and methods. We then present and describe our propositions in section 3. Section 4 shows our experiments and obtained results. Finally, we conclude with a summary of our contributions.

[3] `https://github.com/ThomasRanvier/Accounting_for_Imputation_Uncertainty_During_Neural_Network_Training`

2 Related Works

Single-Imputation (*SI*) is probably the most commonly used framework for handling missing values in practice [11]. *SI* has the advantage of being extremely simple and straightforward: an imputation method is chosen and applied to the incomplete dataset, which yields a completed dataset where missing values have been assigned with new values, which can then be used and exploited as any complete dataset. It is a huge benefit to obtain a completed dataset since it is then possible to integrate *SI* in any existing pipeline or software to make them usable in the presence of missing values [5]. However, *SI* presents several important drawbacks. Once *SI* has been performed, it treats the imputed dataset as the new complete dataset, this is a problem since all methods that exploit the completed dataset will treat missing values as if they were known [11]. The extra variability due to the unknown missing values cannot be taken into account by those methods, thus, their inferences will be too sharp and overstate precision [9]. *SI* results in biased inference models which lack generalization capacity, but it outputs a fixed imputed dataset, which makes it extremely convenient and easy to use in any scenario.

Multiple-Imputation (*MI*) has originally been introduced by Rubin [11]. It consists in replacing each missing value with at least two or more substitution values representing a distribution of possibilities, which represents the uncertainty about the right value to impute [15]. Applying this framework results in several imputed datasets that are then exploited exactly as if the imputed data were the real data [11]. In a machine learning context, one learner is trained on each imputed dataset and the results from all the learners are then pooled in an ensemble manner. An advantage of *MI* over *SI* is that each missing value is represented by a sample of possible imputation values, which results in inferences that reflect the uncertainty level associated with each missing value [15]. Therefore, the results pooled from the ensemble of learners will be less biased compared to those of each learner taken independently. The use of the powerful ensemble paradigm leads to significantly better results when using the *MI* framework compared to *SI*. An obvious disadvantage is the computational cost of such a framework, the ensemble training of the learners multiplies the required amount of calculation time. While being quite an old framework, *MI* is not used in most cases, scientists and users of imputation methods usually still rely on Single-Imputation. This can be partially explained by the computational cost of *MI* which is high because of the ensemble paradigm.

Many methods exist that can be used to deal with missing values by replacing missing values with plausible ones. A very simple method that can be used to deal with missing values is mean substitution, where missing values are replaced with the mean value of the corresponding feature. This method has the advantage of being easy to implement and use, while retaining all non-missing information. More advanced imputation methods can be used to obtain better inference results on the imputed data. A popular method is the SOFTIMPUTE algorithm, introduced by Mazumder et al. [7], which works in an iterative way, at each step missing values are replaced using a single value decomposition. In 2012, Stekhoven and Bühlmann introduced the MISSFOREST algorithm, an iterative imputation method based on random forests [12]. They have shown that MISSFOREST can successfully handle missing values, particularly in datasets including mixed-types of variables. In 2018, Gondara and Wang introduced MIDA: Multiple-Imputation Using Denoising Autoencoders (DAEs) [3]. This method is based on autoencoders, a neural network model used to reconstruct its own input from a reduced latent representation. More recently, Yoon et al. introduced GAIN: Generative Adversarial Imputation Nets [14], which is based on adversarial models to impute missing values. In 2020, Muzellec et al. introduced SINKHORN OT, an optimal transport based method for data imputation [8]. All those imputation methods can be used differently depending on the imputation framework one wishes to apply. We use those popular methods in our experiments to demonstrate the usage and performances of each used imputation framework.

Some imputation methods deal with imputation uncertainty directly within their own algorithm. The MICE algorithm, for Multivariate Imputation by Chained Equations [1], deals with imputation uncertainty by iteratively imputing the dataset within its own algorithm. Missing values are initially imputed by the mean of the corresponding feature, MICE then iteratively trains multiple linear regressions in an ensemble way to impute each missing value by using all other features to fit the regressions until convergence. This is a way of performing *MI* within the algorithm of MICE, in this sense, MICE outputs a final imputed dataset that takes into account the between-imputation uncertainty. MICE can be considered both as an imputation method and an imputation framework as it performs itself Multiple-Imputation. In our experiments we treat MICE as an imputation framework to which we compare our frameworks results. Very recently, Hameed et al. introduced an adversarial imputation method that uses uncertainty-aware predictors [4]. They propose a neural network-based architecture trained using an adversarial strategy to estimate the uncertainty of imputed data. The method estimates the uncertainty of each imputed value and uses this level back as an input in an iterative way to minimize it, leading to better imputed values. We were not able to obtain the source code of the paper from Hameed et al. and could not replicate their method.

3 Contributions

MI is a first step towards taking account of between-imputation uncertainty. It naturally takes into account the uncertainty of imputed values through its ensemble nature, but each inference model is still biased by being trained on an arbitrarily fixed completed dataset [11].

We propose two frameworks that take this between-imputation uncertainty into account and show that neural networks trained using those frameworks have better generalization capacity. Those frameworks are based on the computation of the between-imputation uncertainty, which corresponds to the standard deviation between imputed values of all completed datasets. This uncertainty is then used as a scale to add stochasticity to the imputation of missing values directly on batch extraction during training. It leads to a kind of noise regularization that takes into account the imputation uncertainty, which improves generalization capacity, and thus, prediction results on unseen data.

3.1 Single-Hotpatching

Our Single-Hotpatching (*S-HOT*) framework is similar to *MI*, but has the advantage of training only one strong model. We named this framework Single-Hotpatching since missing values are dynamically imputed on batch extraction in a "hotpatching" manner.

It takes a huge amount of time and resources to train one large neural network, training several such models in an ensemble manner is not always a viable option. *S-HOT* aims to train only one model while relying on multiple imputations such as *MI*. It trains a model that is less biased than if it was trained using *SI* without needing as much computational time as *MI*. We experimentally show that *S-HOT* obtains significantly better results than *SI* with identical training times. Thus, it is interesting to use *S-HOT* in all situations where one aims to train a unique large model on imputed data.

When we train a model on a fixed imputed dataset in which imputed values are most certainly non-optimal, the model repeatedly learns on wrong and imprecise data, leading to a biased model lacking generalization. Instead, *S-HOT* performs multiple imputations and computes the between-imputation standard deviation associated with each imputed value, which we call the uncertainty level. This uncertainty level is used to draw random values from a normal distribution parameterized using the mean and standard deviation computed between the imputations. By training the model with batches in which missing values are dynamically drawn on the imputation distributions, we ensure that the model learns on a multitude of plausible imputations. The trained model has seen a span of possible values in place of each missing value during its training, leading to a less biased model with higher generalization capacity. Figure 1 shows the training and test phases of a neural network when using *S-HOT*. Once the model is trained, prediction results are obtained by randomly patching missing values in the test set using the same process for p iterations, leading to p predictions. The mean of the p model output probabilities is used as the final prediction, which is more robust than a unique iteration of this process.

Mathematically, we note $X \in \mathbb{R}^{n\times d}$ the original incomplete dataset, where each value X_{ij} is either observed or missing, with n and d the number of elements and features in X. We perform m imputations, leading to m different completed datasets $\tilde{X}^{1...m}$, with $\tilde{X}^{k} \in \mathbb{R}^{n\times d}$ the k-th completed dataset. Therefore, a missing value X_{ij} is imputed with m different values $\tilde{X}_{ij}^{1...m}$. Then, we compute the means μ and standard deviations σ of each value of the m completed datasets, with $\mu_{ij} = \frac{1}{m}\sum_{k=1}^{m}\tilde{X}_{ij}^{k}$ (1) and $\sigma_{ij} = \sqrt{\frac{1}{m}\sum_{k=1}^{m}(\tilde{X}_{ij}^{k} - \mu_{ij})^2}$ (2). We note that values in $\tilde{X}^{1...m}$ that are observed in X have a mean of $\mu_{ij} = X_{ij}$ and a standard deviation of $\sigma_{ij} = 0$, only missing values in X have a value $\sigma_{ij} > 0$. Then,

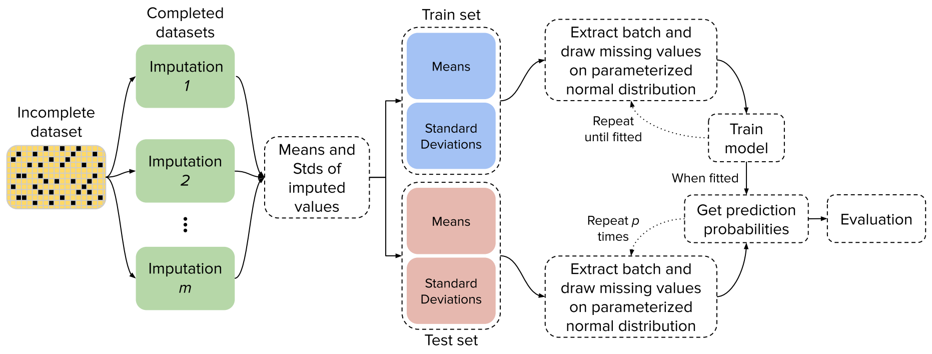

Completed datasets
Imputation 1
Imputation 2
Imputation m
Incomplete dataset
Means and Stds of imputed values
Train set
Means
Standard Deviations
Means
Standard Deviations
Test set
Extract batch and draw missing values on parameterized normal distribution
Repeat until fitted
Train model
When fitted
Get prediction probabilities
Evaluation
Repeat p times
Extract batch and draw missing values on parameterized normal distribution

Fig. 1: Single-Hotpatching. We perform m imputations and compute the means and standard deviations of imputed values. Those are split between the train and test sets. During training, every time a batch is extracted, missing values are drawn from a normal distribution parameterized using previously computed means and stds. Once the model is trained, we extract the prediction probabilities by applying the same process p times for each test instance, which results in p predictions. The final prediction is computed as the mean of the p prediction probabilities, p is set as a few dozens to obtain robust prediction results.

we train a neural network by feeding it batches that are computed from μ and σ. To extract a batch $B \in \mathbb{R}^{b\times d}$, with b the number of elements in the batch, we draw each value of the batch from a normal distribution parameterized with mean μ_{ij} and standard deviation $\alpha \cdot \sigma_{ij}$, such as $B_{ij} \sim \mathcal{N}(\mu_{ij}, \alpha \cdot \sigma_{ij})$. Where α is a scale hyper-parameter that can be set to 1 in most cases and might need to be set lower depending on the average uncertainty level. If the used imputation method outputs imputed values with a wide uncertainty the α scale should be empirically set lower than 1 to limit the stochastic impact induced by the approach. We never found a situation that benefited from increasing the α value. When a data point is presented to the neural network, observed values are set to X_{ij}, and missing values are set to a random value that follows the normal distribution of the m imputations. Thus, the neural network is not repeatedly trained on arbitrarily fixed (and very probably non-optimal) imputations, as it would be using *SI* or *MI* frameworks. Instead, it is trained on values that are randomly drawn from the imputations distribution. This process operates as a noise regularization that takes into account the between-imputation uncertainty, resulting in a less biased and more generalized neural network.

Pseudo-code 1 shows the training phase of the *S-HOT* framework. X is the original incomplete dataset, m the amount of imputations to perform, $impute(\cdot)$ the chosen imputation method, $normal(\mu, \sigma)$ is the method that draws each value x_{ij} on the normal distribution parameterized with μ_{ij} an dσ_{ij}, α is the hyper-parameter used to scale the standard deviation parameter, $n\_batches$ is the amount of batches required to span over the whole dataset.

Algorithm 1: Training phase of the *S-HOT* framework

```
input  : $X$, $m$, $impute(\cdot)$, $normal(\cdot,\cdot)$, $\alpha$, $n\_batches$
output: A trained neural network
for $i = 1$ to $m$ do
    $\tilde{X}^i = impute(X)$;
end
Compute $\mu$ and $\sigma$ from $\tilde{X}$ using equations 1 and 2;
Randomly initialize the neural network;
while Convergence is not reached do
    for $b = 1$ to $n\_batches$ do
        $batch = normal(\mu[batch\_slice], \alpha \cdot \sigma[batch\_slice])$;
        Fit the neural network to extracted $batch$;
    end
end
```

3.2 Multiple-Hotpatching

Multiple-Hotpatching (*M-HOT*) extends *S-HOT* by using the ensemble paradigm. This framework trains as many learners as imputations are performed, those learners are trained while taking into account between-imputation uncertainty, leading to less biased individual learners. We empirically show that *M-HOT* leads to consistently better results than *MI* while not being computationally more expensive. It is beneficial to use *M-HOT* in situations in which one can afford to train several neural networks in an ensemble manner.

The framework is similar to *S-HOT* with the main difference that we use the ensemble paradigm such as in *MI*. We use the previously defined mathematical notations, $X \in \mathbb{R}^{n \times d}$ is the original incomplete dataset, where each value X_{ij} is either observed or missing. We perform m imputations which leads to m different completed datasets $\tilde{X}^{1 \ldots m}$, with $\tilde{X}^k \in \mathbb{R}^{n \times d}$ the k-th completed dataset. We define one learner per completed dataset, leading to m models. We compute the standard deviations σ in the same way as in equation 2, we do not need to compute the means. Then, we train the models. To extract a batch $B^k \in \mathbb{R}^{b \times d}$ that will be fed to model k, we draw each value of the batch from a normal distribution parameterized with mean $\tilde{X}^k_{ij}$ and standard deviation $\alpha \cdot \sigma_{ij}$, such as $B^k_{ij} \sim \mathcal{N}(\tilde{X}^k_{ij}, \alpha \cdot \sigma_{ij})$. Thus, all models are trained in an ensemble manner and are seeing imputed values drawn from a normal distribution centered on the corresponding computed imputation with added diversity during their training. This leads to an ensemble of models that are less biased and capable of more generalization than in *MI*, since they take into account the between-imputation uncertainty. *M-HOT* can be used as a substitute to *MI* in any situation in which *MI* is viable. Figure 2 shows, in a more practical way, how Multiple-Hotpatching takes place during the training phase of the neural networks.

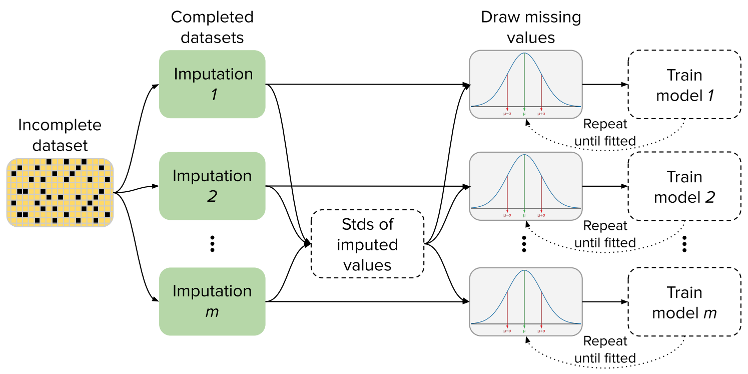


Fig. 2: Multiple-Hotpatching training phase. We perform m imputations and compute the standard deviations of imputed values. We train m models, when a batch is extracted from the k-th completed dataset, missing values are drawn from a normal distribution parameterized using previously computed stds and values from the k-th completed dataset as means. The process is repeated until all models are trained. Those can then be used to obtain predictions in the same way as with *S-HOT* but in an ensemble manner.

4 Experiments

4.1 Experimental Protocol

With our experiments, we show that *S-HOT* obtains significantly better results than *SI* and that *M-HOT* obtains consistently better results than *MI*. In our experiments we are not interested in comparing the performances of the used imputation methods, instead, we are interested in the results we can observe for the same imputation method when using each of the compared imputation frameworks.

We ran part of our experiments on five tabular benchmark classification datasets: the well-known IRIS dataset[4], the STATLOG dataset[5], the WINE Dataset[5], the PIMA dataset[6] and the ABALONE dataset[5].

We also compared the frameworks on three real-world medical datasets containing a certain amount of missing values. In the NHANES dataset, US National Health and Nutrition Examination Surveys, we used data from studies spanning from years 2000 to 2008, with 95 features and about 33% missing values. The COVID-19 dataset was publicly released with the paper [13], it contains medical information collected in early 2020 on pregnant and breastfeeding women. We based our data preprocessing on the one realized in the original paper, it is composed of 361 patients with 76 features, with about 20% missing data. And

[4] `https://scikit-learn.org/stable/modules/generated/sklearets.load_iris.html`
[5] `https://archive.ics.uci.edu`
[6] `https://rioultf.users.greyc.fr/uci/files/pima-indians-diabetes`

the MYOCARDIAL infarction dataset[5], composed of 1700 patients with 107 features, with about 5% missing values.

To run our experiments on the benchmark datasets that do not contain any missing values we artificially add various rates of missing values. We then use various imputation methods to impute the missing values following the tested frameworks and perform neural network training to compare the obtained results. We use three different patterns, MCAR, MAR, and MNAR [10], to introduce missing values to the datasets. We use the term "masked" to refer to the added missing values, the missing rate determines the number of values that will be masked. With the MCAR pattern, we introduce missing values completely at random by randomly masking a certain rate of values. For the MAR pattern, a random subset of features is chosen to remain unmasked, this subset is used as an input to a logistic model, and the output of the model is used to mask values out of the remaining features. We used the public implementation of [8]. With the MNAR pattern, the implementation is close to the one of MAR, but the input of the logistic model is masked using an MCAR pattern. Thus, the output of the model depends on values that are indifferently known or masked.

We compare the frameworks using several commonly used imputation methods: MISSFOREST based on iterative random forests [12], SOFTIMPUTE that uses single value decomposition) to iteratively impute missing values [7], GAIN that imputes using two adversarial models [14], MIDA that uses Denoising Autoencoders to impute missing values [3], and SINKHORN that is based on optimal transport for data imputation [8]. Note that we do not aim at comparing performances of those imputation methods, we apply each imputation framework using each imputation method and are only interested in comparing the imputation framework results.

In our experiments we treat MICE, a method that iteratively improves its last imputation using logistic models [1], as an imputation framework to which we compare our frameworks results. Indeed, as described earlier, MICE perform Multiple-Imputation within its own algorithm, it can be considered both an imputation method and an imputation framework.

Our experiments aim to evaluate neural network performances in a supervised-learning setting to check the bias and generalization capacity of the model. We use a simple neural network under the scikit-learn library[7], parameterized using well performing and identical hyper-parameters for each dataset to ensure a fair and unbiased comparison. We used the Adam optimizer with a default learning rate of 0.001. The used neural network is composed of two fully-connected layers in all our experiments, for datasets IRIS, STATLOG, PIMA and ABALONE both layers contains 32 units, for WINE, MYOCARDIAL and NHANES layers contain 64 and 32 units respectively, for the COVID dataset layers are composed of 128 and 32 units. We evaluate the performances of the models using the AUC metric, the Area Under the Receiver Operating Characteristic (ROC) Curve, the balanced accuracy metric, and the F1-score. The AUC

[7] `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

corresponds to the area under the ROC curve that is obtained by plotting the true positive rate (recall) against the false positive rate (1-specificity), it reflects the capacity of the model to yield pertinent predictions. The balanced accuracy is defined as the average of recall obtained for each class, which is simply the average of the true positive rate between all the classes. The F1-score combines the precision and recall of a classifier into a single metric by taking their harmonic mean.

To better assess the obtained results we base our comparisons on the Friedman and Nemenyi statistical tests such as described in [2]. The Friedman test is first used to check if the null hypothesis that all compared frameworks are statistically equivalent for a given p-value is rejected or not. It ranks the compared frameworks for each dataset from best to worst, ties are assigned an average rank. The average rank for each compared framework is computed over all datasets and the Friedman test checks whether the measured average ranks are significantly different from the mean rank by computing the Friedman statistic: $\chi_F^2 = \frac{12N}{k(k+1)}(\sum_{j=1}^{k} R_j^2 - \frac{k(k+1)^2}{4})$ (3), derived in $F_F = \frac{(N-1)\chi_F^2}{N(k-1)-\chi_F^2}$ (4). The Friedman statistic is distributed according to the F-distribution with $k-1$ and $(k-1)(N-1)$ degrees of freedom, with k the number of frameworks compared and N the number of datasets. If the Friedman statistic is above the critical value of $F(k-1,(k-1)(N-1))$ given the p-value, then the test shows a significant difference between the compared frameworks. If that is the case, a post-hoc test is performed. We use the post-hoc Nemenyi test to compare the frameworks to each other in a pair-wise manner, two frameworks are considered significantly different if their average ranks differ by more than the computed critical distance: $CD = q_\alpha\sqrt{\frac{k(k+1)}{6N}}$, where α is the p-value, q_α comes from [2]. The Nemenyi test can be easily visualized through a simple diagram, which makes its results easy to analyze.

We first perform a comparative study between the four frameworks *SI*, *MI*, *S-HOT*, and *M-HOT* using different imputation methods. We show that no matter the used imputation method and missingness setting, our hotpatching frameworks lead to consistent and good results. We ran our experiments on 9 different missingness settings: MCAR, MAR, and MNAR, and for each pattern missing rates $10\%, 15\%, 25\%$. In each setting, we ran 20 imputations using each of the tested imputation methods and then ran the four frameworks 200 times, each run using a different random seed and train-test set to ensure the results are not biased by the stochastic nature of neural network training. We then compare our results to those of MICE, an imputation method that takes into account imputation uncertainty within its own algorithm, which we consider as a framework. We then experiment on real-world medical datasets containing missing values to evaluate our frameworks results in a real situation. Finally, we compare the required running time for each framework. Since we performed very extensive experiments the complete results can be found in our supplementary material in our GitHub repository.

4.2 Results

Comparative Study Between the Imputation Frameworks. The first step of this experiment is to compute $m = 20$ imputations with each tested imputation method on each dataset and each previously described missingness setting.

Once all imputations have been computed, we run the four compared frameworks *SI*, *MI*, *S-HOT*, and *M-HOT*, following the previously described experimental protocol. Table 1 shows the results of the four compared frameworks when using the MISSFOREST imputation method, the remainder of our results can be found in our supplementary material. We do not observe any influence from the missingness pattern or missing rate on the obtained results, which seems to show that our frameworks are not sensitive to those parameters, and so, can be used in any circumstances. We note that the *M-HOT* framework obtains the best results in the vast majority of cases, while *S-HOT* consistently achieve better results than *SI*. We used Friedman and Nemenyi statistical tests to better assess the obtained results.

Using equations 3 and 4, we compute the Friedman statistic on the average ranks for the results obtained using the MISSFOREST imputation method, we obtain $F_F \approx 344.07$. The amount of compared frameworks is $k = 4$, and $N = 5 \cdot 9 = 45$ for 5 datasets with 9 missingness settings for each. Under a significance level of 0.05 the critical value of $F(3, 132)$ is 2.673, the null-hypothesis is rejected since in our case $F_F > F(3, 132)$. We reiterate the same process to compute the Friedman statistic for each of the other imputation methods used and find that the null hypothesis is rejected in all cases. Thus, we continue with the post-hoc Nemenyi test as described in our protocol. We want to check the significant difference under a p-value of 0.05, given those values we have $q_{0.05} = 2.569$ from table 5 in [2]. We find $CD \approx 0.6992$, two frameworks can be considered as significantly different if their average ranks differ by more than 0.6992.

Figure 3 shows the obtained Nemenyi results that compare the *SI*, *MI*, *S-HOT* and *M-HOT* frameworks using each of the five tested imputation methods. The critical distance is visualized on the top left corner of each diagram, two frameworks can be considered significantly different if they are not linked by the same black bar. We see that in all cases the ordering of the frameworks from worst to best is the same, *SI* performs the worst, then *S-HOT*, followed by *MI*, finally *M-HOT* performs the best. *S-HOT* obtains significantly better results than *SI*, showing that it is a good alternative to *SI* when one want to train a unique large model. We note that *M-HOT* obtains consistently better results than *MI*, which seems to show that it is always a good and viable alternative to *MI*.

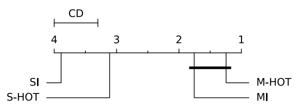
Comparative Study Between MICE and Imputation Frameworks. We compare *S-HOT* and *M-HOT* to MICE, we observe that the *M-HOT* framework obtains the best results in most cases. By reiterating the same calculations as before, we find that the Friedman test rejects the null hypothesis. As before we use the post-hoc Nemenyi test, we compute $CD \approx 0.9093$, Figure 4 shows

Dataset	Pattern		*SI*	*MI*	*S-HOT*	*M-HOT*
		10%	0.9972681 (4)	0.9976699 (2)	0.9973865 (3)	**0.9976913** (1)
	MCAR	15%	0.9942281 (4)	0.9946880 (2)	0.9944422 (3)	**0.9947552** (1)
		25%	0.9835323 (4)	0.9841479 (2)	0.9839692 (3)	**0.9843596** (1)
		10%	0.9975236 (3)	0.9978336 (2)	0.9974886 (4)	**0.9978845** (1)
IRIS	MAR	15%	0.9970188 (3)	**0.9973150** (1)	0.9970006 (4)	0.9973097 (2)
		25%	0.9941960 (4)	0.9943643 (2)	0.9943554 (3)	**0.9943890** (1)
		10%	0.9972209 (4)	**0.9973548** (1)	0.9972343 (3)	0.9973011 (2)
	MNAR	15%	0.9937917 (4)	0.9941570 (2)	0.9938451 (3)	**0.9941622** (1)
		25%	0.9891964 (4)	0.9905233 (2)	0.9902221 (3)	**0.9907251** (1)
		10%	0.9105582 (4)	**0.9132492** (1)	0.9106386 (3)	0.9132475 (2)
	MCAR	15%	0.9060337 (4)	0.9091019 (2)	0.9066862 (3)	**0.9091584** (1)
		25%	0.9047888 (4)	0.9077214 (2)	0.9059597 (3)	**0.9078041** (1)
		10%	0.9127799 (3)	**0.9144097** (1)	0.9127328 (4)	0.9142985 (2)
STAT	MAR	15%	0.9079620 (3)	**0.9093156** (1)	0.9078667 (4)	0.9092031 (2)
		25%	0.8959471 (4)	0.8990894 (2)	0.8968849 (3)	**0.8992881** (1)
		10%	0.9070192 (4)	0.9095510 (2)	0.9071004 (3)	**0.9095576** (1)
	MNAR	15%	0.9040681 (4)	**0.9061872** (1)	0.9042151 (3)	0.9060699 (2)
		25%	0.8951658 (4)	0.8992252 (2)	0.8967363 (3)	**0.8992736** (1)
		10%	0.9987736 (4)	0.9989672 (2)	0.9988008 (3)	**0.9989811** (1)
	MCAR	15%	0.9955454 (4)	0.9960062 (2)	0.9956407 (3)	**0.9960307** (1)
		25%	0.9910498 (4)	0.9919927 (2)	0.9914148 (3)	**0.9923485** (1)
		10%	0.9961058 (4)	0.9965056 (2)	0.9961142 (3)	**0.9965060** (1)
WINE	MAR	15%	0.9977720 (4)	**0.9982010** (1)	0.9978677 (3)	0.9981809 (2)
		25%	0.9952116 (4)	0.9965157 (2)	0.9959576 (3)	**0.9968702** (1)
		10%	0.9987205 (3)	**0.9988244** (1)	0.9987058 (4)	0.9988131 (2)
	MNAR	15%	0.9974746 (4)	0.9976465 (2)	0.9974850 (3)	**0.9976683** (1)
		25%	0.9808498 (4)	0.9841291 (2)	0.9822719 (3)	**0.9844587** (1)
		10%	0.8193054 (4)	**0.8211340** (1)	0.8196586 (3)	0.8211193 (2)
	MCAR	15%	0.8073739 (4)	0.8095505 (2)	0.8078378 (3)	**0.8095824** (1)
		25%	0.8029002 (4)	0.8060367 (2)	0.8043589 (3)	**0.8065611** (1)
		10%	0.8238900 (4)	**0.8257089** (1)	0.8242472 (3)	0.8256414 (2)
PIMA	MAR	15%	0.8045918 (4)	0.8080830 (2)	0.8061503 (3)	**0.8083194** (1)
		25%	0.8017568 (4)	**0.8041203** (1)	0.8025144 (3)	0.8040062 (2)
		10%	0.8280685 (4)	0.8302729 (2)	0.8284115 (3)	**0.8303076** (1)
	MNAR	15%	0.8279577 (4)	0.8298929 (2)	0.8283792 (3)	**0.8300464** (1)
		25%	0.8005008 (4)	0.8043448 (2)	0.8021749 (3)	**0.8047250** (1)
		10%	0.8737739 (4)	0.8748059 (2)	0.8740180 (3)	**0.8749393** (1)
	MCAR	15%	0.8714861 (4)	0.8725539 (2)	0.8717831 (3)	**0.8726250** (1)
		25%	0.8663833 (4)	0.8674332 (2)	0.8666186 (3)	**0.8675645** (1)
		10%	0.8742551 (4)	0.8751972 (2)	0.8743399 (3)	**0.8753671** (1)
ABAL	MAR	15%	0.8720722 (4)	0.8731502 (2)	0.8721856 (3)	**0.8731739** (1)
		25%	0.8697060 (4)	0.8708046 (2)	0.8699619 (3)	**0.8709102** (1)
		10%	0.8760444 (4)	0.8768033 (2)	0.8760447 (3)	**0.8769350** (1)
	MNAR	15%	0.8753217 (4)	0.8763271 (2)	0.8754605 (3)	**0.8764456** (1)
		25%	0.8683507 (4)	0.8696780 (2)	0.8690281 (3)	**0.8697714** (1)
Average rank			3.8889	1.7556	3.1111	1.2444

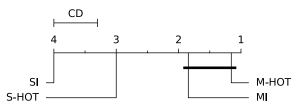
Table 1: Comparison of the results from the four *SI*, *MI*, *S-HOT* and *M-HOT* frameworks using the MISSFOREST imputation method.

the obtained Nemenyi results that compare the frameworks and MICE. We note that MICE performs largely better than *SI*, despite not being significantly better. *S-HOT* obtains significantly better results than *SI* and slightly better results than MICE. Both *MI* and *M-HOT* frameworks are significantly better than the remaining tested frameworks and methods, once again *M-HOT* performs better than *MI*.

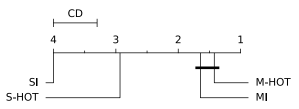
Comparison on Real-World Medical Datasets. We compare our *S-HOT* framework to *SI* and our *M-HOT* framework to *MI* on three real-world medical



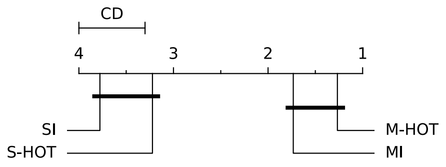
(a) Using MISSFOREST average ranks



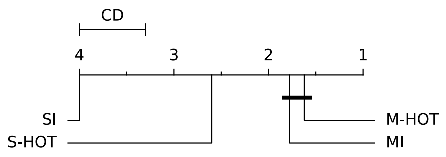
(b) Using SOFTIMPUTE average ranks



(c) Using GAIN average ranks



(d) Using MIDA average ranks



(e) Using SINKHORN average ranks

Fig. 3: Nemenyi tests comparing *SI*, *MI*, *S-HOT* and *M-HOT* frameworks using each tested imputation method, $CD \approx 0.6992$.

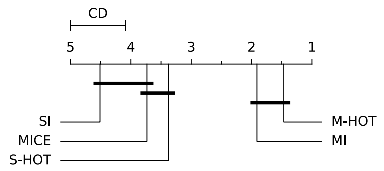


Fig. 4: Nemenyi test comparing the MICE imputation method with the best imputation method for each dataset, $CD \approx 0.9093$.

datasets containing missing values, complete results are in our supplementary material. The results show that our frameworks perform well on real conditions, obtaining better results than their counterpart on the vast majority of cases. *S-HOT* obtains consistently better results than *SI*, which seems to show that *S-HOT* is a good alternative to *SI* in most real-life scenarios. *M-HOT* also performs better than *MI* in those real-world scenarios. Overall, this experiment demonstrates the capacity of our imputation frameworks in real-world situations.

Computational Running Times Comparison. Finally, we compare the computational running time required to execute each framework in each tested scenario, precise results can be found in our supplementary material. In all cases most of the computational time comes from computing the m multiple imputations. *SI* benefits largely from that point, the three remaining frameworks all require the same amount of time to compute the multiple imputations. We observe no difference in the required training time between *SI* and *S-HOT*, only the time required to perform the multiple imputations is different as *SI* needs only one imputation. In the case of *MI* and *M-HOT*, the required time to compute the imputations is the same, there is a small difference in training times. *M-HOT* is a bit slower, of less than a second to a few seconds for larger models and datasets compared to *MI*. The overall difference in total running time between *MI* and *M-HOT* is negligible. Since we have shown that *M-HOT* obtains consistently better results than *MI* most scenarios would benefit from using *M-HOT* over *MI*. When using a fast imputation method, such as SOFTIMPUTE, the imputation computational time is almost negligible in all cases, using *S-HOT* over *SI* in this context leads to better results for the same overall computational cost.

5 Discussion and Conclusion

Taking account of imputation uncertainty while training a neural network is not typically researched. That is because strong learners, such as neural networks, are naturally able of enough generalization to neglect the consequences of the bias induced by imputation uncertainty. In this paper we researched and proposed two imputation frameworks that can be used to train models while taking this uncertainty level into account, leading to models able of more generalization and better inference results on unseen data.

Our two proposed frameworks, *S-HOT* and *M-HOT* aim to be used as substitutes for Single-Imputation (*SI*) and Multiple-Imputation (*MI*) respectively. We perform extensive experiments to compare imputation frameworks on both benchmark and real-world conditions and show that our frameworks compete with and even outperform other imputation frameworks in many situations. We statistically assess the results using Friedman and Nemenyi tests and show that our frameworks lead to less biased neural networks, improving inference results. We also attentively compare required running times for each framework and conclude that the total running-time difference between *MI* and *M-HOT* is negligible, while *SI* is faster than *S-HOT* but obtains significantly worse results than our *S-HOT* framework. We have shown that *S-HOT* obtains significantly better results than *SI* in all tested scenarios, we conclude that it is beneficial to use *S-HOT* when one needs to train a large and unique neural network. Our experiments show that *M-HOT* obtains consistently better results than *MI* in all tested scenarios for a comparable execution time. When one can afford to train multiple models in an ensemble manner, best results can be obtained using our *M-HOT* framework.

In our frameworks we assume a Gaussian distribution of imputed values. It leads to good empirical results but a Gaussian distribution might not always be pertinent depending on the missing feature nature or used imputation method, future works will focus on this matter. In this paper we have shown that our new frameworks improve generalization capacity of neural networks on imputed tabular data, future works could focus on experimenting with similar frameworks on image or sequential data. This work is a first step towards finding better ways to deal with missing values imputation in machine learning. We hope that it spikes the interest of other researchers throughout the world on this important and often overlooked matter in the machine learning literature.

References

1. Stef van Buuren and Karin Groothuis-Oudshoorn. mice: Multivariate Imputation by Chained Equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011.
2. Janez Demsar. Statistical Comparisons of Classifiers over Multiple Data Sets. *The Journal of Machine Learning Research*, pages 1–30, 2006.
3. Lovedeep Gondara and Ke Wang. MIDA: Multiple Imputation Using Denoising Autoencoders. *Pacific-Asia Conference on Knowledge Discovery and Data Mining 2018*, pages 260–272, 2018.
4. Wafaa Mustafa Hameed and Nzar A. Ali. Enhancing imputation techniques performance utilizing uncertainty aware predictors and adversarial learning. *Periodicals of Engineering and Natural Sciences (PEN)*, 10(3):350–367, June 2022.
5. Julie Josse, Nicolas Prost, Erwan Scornet, and Gaël Varoquaux. *On the consistency of supervised learning with missing values.* ArXiv, February 2019.
6. Marine Le Morvan, Julie Josse, Erwan Scornet, and Gael Varoquaux. What's a good imputation to predict with missing values? In *Advances in Neural Information Processing Systems*, volume 34, pages 11530–11540. Curran Associates, Inc., 2021.
7. Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *Journal of machine learning research : JMLR*, 11:2287–2322, March 2010.
8. Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing Data Imputation using Optimal Transport. In *Proceedings of the 37th International Conference on Machine Learning*, pages 7130–7140. PMLR, November 2020. ISSN: 2640-3498.
9. D. B. Rubin and N. Schenker. Multiple imputation in health-care databases: an overview and some applications. *Statistics in Medicine*, 10(4):585–598, April 1991.
10. Donald B. Rubin. Inference and Missing Data. *Biometrika*, 63(3):581–592, 1976. Publisher: [Oxford University Press, Biometrika Trust].
11. Donald B. Rubin. *Multiple Imputation for Nonresponse in Surveys.* John Wiley & Sons, June 2004. Google-Books-ID: bQBtw6rx_mUC.
12. Daniel J. Stekhoven and Peter Bühlmann. MissForest—Non-Parametric Missing Value Imputation for Mixed-Type Data. *Bioinformatics*, 28(1), January 2012.
13. Li Yan, Hai-Tao Zhang, and al. An interpretable mortality prediction model for COVID-19 patients. *Nature Machine Intelligence*, 2(5):283–288, May 2020.
14. Jinsung Yoon, James Jordon, and Mihaela Schaar. GAIN: Missing Data Imputation using Generative Adversarial Nets. In *Proceedings of the 35th International Conference on Machine Learning*, page 5689. PMLR, July 2018. ISSN: 2640-3498.
15. Yang Yuan. Multiple Imputation for Missing Data: Concepts and New Development. *SAS Institute Inc.*, January 2005.

Acknowledgments

This research is supported by the European Union's Horizon 2020 research and innovation program under grant agreement No 875171, project QUALITOP (Monitoring multidimensional aspects of QUAlity of Life after cancer ImmunoTherapy - an Open smart digital Platform for personalized prevention and patient management).