DeepSample: DNN sampling-based testing for operational accuracy assessment

Anonymous Author(s)

ABSTRACT

Deep Neural Networks (DNN) are core components for classification and regression tasks of many software systems. Companies incur in high costs for testing DNN before release with datasets representative of the inputs they are expected to receive in the operational phase, as these need to be manually labelled.

The challenge is to select a representative set of test inputs as small as possible to reduce the labelling and testing cost, while sufficing to yield unbiased high-confidence estimates of the expected DNN accuracy. At the same time, testers are interested in exposing as many DNN mispredictions as possible to improve the DNN, ending up in the need for techniques pursuing a threefold aim: small dataset size, trustworthy estimates, mispredictions exposure.

This study presents `DeepSample`, a family of DNN testing techniques for cost-effective accuracy assessment based on probabilistic sampling. We investigate whether, to what extent, and under which conditions probabilistic sampling can help to tackle the outlined challenge. We implement five new sampling-based testing techniques, and perform a comprehensive comparison of such techniques and of three further state-of-the-art techniques for both DNN classification and regression tasks. Results serve as guidance for best use of sampling-based testing for faithful and high-confidence estimates of DNN accuracy in operation at low cost.

CCS CONCEPTS

• **Software and its engineering → Soft. testing and debugging**.

KEYWORDS

Software testing, Deep Neural Networks, Sampling

ACM Reference Format:

Anonymous Author(s). 2023. DeepSample: DNN sampling-based testing for operational accuracy assessment. In *Proceedings of 46th International Conference on Software Engineering (ICSE 2024).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

A countless number of software systems today rely on Deep Neural Networks (DNN) predictions. Before release, engineers need to test the DNN to estimate their accuracy (i.e., probability of not having mispredictions). This enables to establish a release criterion and to correct or tune the DNN until the criterion is met.

The reference scenario is the following: a DNN model meant to operate in a target context is trained with a *training dataset*. The goal of the tester is to select a small yet representative subset of inputs from an arbitrary large *operational dataset*, to use as test cases to estimate the DNN accuracy [1]. The label of the selected inputs is unknown; their manual labelling is a high cost for companies. The challenge is to build a small test set able to provide a good (i.e., unbiased, high-confidence) estimate of the DNN accuracy. At the same time, testers are interested in exposing DNN mispredictions, since they are input to DNN debugging and re-training [2]. The goal thus becomes threefold: build a *small dataset*, able to faithfully *estimate DNN accuracy*, and with a good ability to *expose mispredictions*.

Inspired by *operational testing*, a known practice in software reliability engineering [3–7], researchers proposed probabilistic sampling to test DNN. The basic scheme is simple random sampling (SRS). Li *et al.* proposed the first sampling-based scheme, aimed at minimizing the cross-entropy between the selected tests and the operational dataset [1]; Guerriero *et al.* [2] leveraged adaptive sampling [8] to propose DeepEST, an algorithm aiming to expose many DNN mispredictions while preserving the ability to provide good accuracy estimates, hence targeting the threefold challenge outlined. However, the full potential of statistical sampling theory is still largely unexploited, and better trade-offs can be achieved by properly using the information available to drive sampling.

In this work, we propose `DeepSample`, a family of sampling-based DNN testing techniques, differing from each other in the sampling strategy, in the auxiliary information used for sample selection as well as for partitioning, and in the estimation process. The framework includes five new testing techniques, each implemented in three variants depending on the auxiliary information used to drive sampling. The idea is to give multiple choices based on the prior knowledge available about the operational dataset, which can be exploited to improve the efficiency while preserving the unbiasedness of the estimation. We present a comprehensive comparison of the new techniques and of three existing ones, SRS, CES, DeepEST, to assess their ability to closely and efficiently assess DNN accuracy as well as to select failing examples. Overall, the evaluation includes 20 variants, and is conducted on both classification and regression tasks, under 5 testing budgets, 3 datasets, with 3 models per dataset for classification, and 1 dataset, with 2 models for regression.[1] The goal is to establish whether, to what extent, and under which conditions probabilistic sampling tackles the outlined challenge. The results highlight the pros and cons of key choices in the design of a sampling strategy, such as whether and what auxiliary information is useful and how to use it. They serve as guidance for the best use of sampling-based testing for high-accuracy high-confidence estimates at low cost, traded off with mispredictions exposure ability.

ICSE 2024, April 2024, Lisbon, Portugal

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1]The replication package is at: https://anonymous.4open.science/r/DeepSample-DBE9

2 RELATED WORK

Probabilistic sampling is used in *operational testing* (OT) to estimate the expected reliability of a software system after release. In OT, test suites are built by selecting or generating tests according to the expected operational profile, which is a probabilistic characterization of the expected usage. OT was the core technique of Cleanroom software engineering [3–6] and of SRET, the Software Reliability Engineering Test process developed by Musa at AT&T [7].

Over the years, researchers adopted better sampling strategies to improve the estimate at a lower cost. Cai *et al.* developed *Adaptive Testing* (AT), still based on the operational profile, but with an adaptive assignment of test cases to partitions [9–11]. AT with gradient descent proposed in [12] is the basis for one of the techniques considered in this study. Stratified sampling has also been used for reliability assessment [13], [14]. Later, Pietrantuono *et al.* stressed the use of (unequal probability) sampling to improve the efficiency of the estimates [15][16], formalizing several sampling schemes to this aim [17]. The same team employed *adaptive sampling* to select tests for the operational reliability assessment of Microservice Architectures [18, 19], then exploited to implement DeepEST [2], one of the strategies we assess in this work.

Researchers have recently proposed sampling strategies for the operational accuracy assessment of DNN. Li *et al.* [1] presented CES (Cross-Entropy Sampling), the first technique adopting sampling in operational testing for DNN accuracy estimation. Like conventional OT, CES aims to select a small sample that accurately represents the population by minimizing the cross-entropy between the selected and the operational dataset. A representative sample is expected to contain the same proportion of failure-causing examples as the operational dataset. Guerriero *et al.* [2] observed that the mere imitation of the expected input can be inefficient, especially with very accurate DNN, because much effort is wasted to manually label correctly classified images; hence they proposed DeepEST, using adaptive sampling, to improve the detection of failing examples while preserving the accuracy estimation power.

A further technique for accuracy estimation is PACE (Practical accuracy estimation) [20]. PACE is a heuristic-based method that exploits clustering to partition tests into groups, and then adopt adaptive random selection of test inputs representative of the cluster. Zhou *et al.* proposed DeepReduce [21], a two-stage heuristic method exploiting neuron coverage to first select a subset of inputs, then using the KL divergence to drive the selection in the second stage.

The latter two techniques are however not based on probabilistic sampling like the techniques compared in this work, and they do not provide a guarantee of unbiasedness and convergence.

3 SAMPLING-BASED TESTING

3.1 Formulation

- M is the DNN model under test;
- $D = \{d_1, d_2, \ldots, d_N\}$ is the *operational dataset*, namely an arbitrarily large set of examples with unknown labels, which are possibly given as input to the model M during the operational phase. Its size is $N = |D|$;
- Tr is the *training dataset* used to train the model M;
- $T \in D = \{t_1, \ldots, t_n\}$ is the subset of examples to select from D and to be manually labelled. This set is used for estimating the expected accuracy, and can also be used to enlarge the training set and improve the DNN performance in successive releases. Its size is $n = |T|$. When an example t_i is submitted to the DNN, a human oracle assigns the true expected output to t_i, and then compares it with the actual DNN output. In classification tasks, this comparison gives a binary outcome z_i (i.e., 0 if the obtained and expected labels match, 1 otherwise). In regression tasks, the comparison gives an offset δ_i, which is the absolute difference between the true (r_i) and predicted ($\hat{r}_i$) value – considering this a failure or not depends on the tolerable thresholds, which in turn depend on the problem. For our purpose, we just need to focus on its value regardless of any specific threshold.
- $\theta = Pr(z_i = 1)$, with $i = 1, \ldots |D|$, is, in classification tasks, the true failure probability on a randomly selected example from the entire operational dataset, and corresponds to the true (unknown) proportion $\theta = \frac{1}{N}\sum_{i=1}^{N} z_i$. Accuracy is defined as: $\xi = 1-\theta$. In the case of regression, we look at the mean squared error between the true (r_i) and predicted ($\hat{r}_i$) value over the entire operational dataset: $\Delta = \frac{1}{N}\sum_{i=1}^{N} \delta_i^2$, and $\xi = 1 - \Delta$. Its estimate is $\hat{\xi}$.

Given a sample size budget n, the goal of `DeepSample` is to select a subset T able of giving an *unbiased* (i.e.: such that $\mathbb{E}[\hat{\xi}] = \xi$) estimate of ξ while maximizing the *efficiency* of the estimator (i.e., minimizing the variance of the estimate).[2] In addition, the set T is wanted to expose as many failing examples as possible.

3.2 Overview of DeepSample

`DeepSample` is a family of techniques, which try to differently exploit the prior knowledge available about the operational dataset to pursue the stated goal. Specifically, sampling aims to exploit any known information supposed to be correlated to the variable to estimate (namely, accuracy). This prior information is encoded through what are called *auxiliary variables* [22], which we denote as χ; for instance, the confidence value provided by classifiers when predicting a label can be assumed to be (negatively) correlated with the failure probability θ. Clearly, the accuracy and efficiency of the estimate will depend on the extent to which this assumption is true.

The `DeepSample` techniques are characterized by two dimensions: *i)* the **sampling algorithm**, and *ii)* the **auxiliary variable**.

The former specifies a *sampling scheme*, namely the sequence of steps required to select the tests t_i. The latter specifies what is the *auxiliary variable* χ, if used by the sampling scheme (not all auxiliary variables can be used in all the schemes).

There are two ways of exploiting the auxiliary variables. The first is to partition the dataset into classes that are homogeneous with respect to the auxiliary variable (e.g., similar confidence), similarly to stratification in sampling theory [22]. If the variable is well correlated with the failure probability θ (or Δ for regression), the partitions should be homogeneous also from the failure probability point of view. This allows to wisely allocate the number of examples to draw from each partition with the aim of reducing the overall variance of the estimation. The second way to exploit the auxiliary variable is in the examples selection; the sampling scheme selects the examples proportionally to the auxiliary variable's value, so as to get the ones with higher expected failure probability. Since this

[2] Minimizing the variance is equivalent to minimize the MSE since the estimators are required to be unbiased. Low variance (or MSE) implies maximizing the confidence

would bias the estimation, a proper estimator needs to be adopted to account for this *unequal* selection probability.

Techniques can be *with* or *without replacement*. The former ones (allowing an example to be selected more times) are associated with simpler estimators - a common choice in literature [1] [9] [10] [11] [12]; the latter ones are expected to give higher efficiency, though the gain in large populations can be marginal (with-replacement schemes will unlikely select twice the same example).

The **estimator** takes the result of submitting the selected sample T to the DNN M (which we denoted as z_i and δ_i for classification and regression, respectively) and yields an unbiased estimate of ξ.

3.3 Auxiliary variables

We consider three auxiliary variables for classification problems, and for regression as well. For classification, they are: Confidence, Distance-based Surprise Adequacy (DSA), and Likelihood-based Surprise Adequacy (LSA). For regression, they are LSA, and two variables based on the reconstruction error of a simple autoencoder (SAE) and of a variational autoencoder (VAE).

Confidence C_{d_i} of an input d_i is the maximum value in the probability vector obtained from the last layer's output of the DNN; it is for classification problems only. DSA and LSA are defined by Kim *et al.* [23]. They exploit Activation Traces (AT), which are vectors of neurons' activation values belonging to a certain layer. DSA is defined as: $DSA_{d_i} = \frac{\sigma_A}{\sigma_B}$, where σ_A is the Euclidean distance between the ATs of the input d_i (whose predicted class is A) and its nearest neighbour belonging to the same class A, σ_B is the distance between the ATs of d_i and its nearest neighbour belonging to a different class B. It makes sense for classification models only. LSA uses Kernel Density Estimation (KDE) [24] to estimate the probability density of each activation value, obtaining the surprise of a new input with respect to the estimated density. LSA is a measure of rareness computed as: $LSA_{d_i} = -log(\hat{f}(d_i))$, where $\hat{f}(d_i)$ is the KDE applied to the new input d_i. LSA can be used both in the case of classification and regression models.

For SAE/VAE-based variables, we leverage the reconstruction error ϵ. We used the two best-performing autoencoders implemented by Stocco *et al.* [25], SAE (Simple Autoencoder) with a single hidden layer, and VAE (Variational Autoencoder). We consider autoencoders as single-image reconstructors, computing their outputs for all the operational examples, and then calculating the reconstruction error as: $\epsilon_{d_i} = \frac{1}{WHC}\sum_{k=1,j=1,c=1}^{W,H,C}(d_i[c][k,j] - d_i'[c][k,j])^2$, where d_i is the original image, d_i' is the reconstructed image, W, H, and C are width, height, and channels respectively. The corresponding auxiliary variables are synthetically called SAE and VAE, meaning the ϵ_{d_i} value obtained by SAE and VAE.

All the variables are assumed to be correlated to accuracy: lower confidence, higher surprise (DSA, LSA), and higher reconstruction error (SAE, VAE) are expected to be related to higher failure probability. To have all positive variables (from which selection probabilities need to be derived), DSA and LSA for classification are *min-max* normalized. For regression, as the min-max normalization affects the distribution of test data, we just shift the values to avoid null/negative values: $DSA_{d_i} = DSA_{d_i} + \lceil |min(DSA_{d_i})| \rceil$ (the same for *LSA*). All the above variables are denoted as χ_i in the following, when there is no need to distinguish them.

3.4 Testing techniques

The characteristics of the eight compared testing techniques are summarized in Table 1; their description follows.

3.4.1 *Without-partitioning techniques.*

*Simple Random Sampling (**SRS**).* SRS with replacement, where all examples have the same probability to be selected, is the simplest technique for the assessment, typically used as baseline [17][1]. For SRS; unbiased estimators of θ (for classification) and Δ (regression) are, respectively, the observed proportion and mean squared error over the subset of selected tests:

$$\hat{\theta} = \frac{1}{n}\sum_{i=1}^{n} z_i \quad (1) \qquad \hat{\Delta} = \frac{1}{n}\sum_{i=1}^{n} \delta_i^2 \quad (2)$$

*Simple Unequal Probability Sampling (**SUPS**).* This scheme leverages auxiliary variables χ for selecting the examples. The selection probability π_i for the i-th example t_i is obtained by normalizing the auxiliary variable $\pi_i = \chi_i / \sum_{i=1}^{N} \chi_i$; this is known as probability-proportional-to-size (PPS) sampling [22]. The selection is with replacement. An unbiased estimator is the sample mean of the observed values re-scaled by the inverse of their selection probability π_i and by N, known as Hansen-Hurwitz estimator [26]:

$$\hat{\theta} = \frac{1}{nN}\sum_{i=1}^{n} \frac{z_i}{\pi_i} \quad (3) \qquad \hat{\Delta} = \frac{1}{nN}\sum_{i=1}^{n} \frac{\delta_i^2}{\pi_i} \quad (4)$$

Note that this is a generalization of SRS, wherein the selection probability is $\pi_i = 1/N$ for all the examples.

*RHC-Sampling (**RHC-S**).* This is another unequal probability selection scheme, but without replacement, and uses the Rao, Hartley, and Cochran (RHC) estimator [27]. The scheme is as follows:

(1) Given the budget of $n = |T|$ test cases, divide randomly the $N = |D|$ units of the operational dataset into n groups, by selecting G_1 inputs with SRS *without replacement* for the first group, then G_2 inputs out of the remaining $(N - G_1)$ for the second, and so on. This will lead to n groups of size $G_1, \ldots, G_n$ with $\sum_{r=1}^{n} G_r = N$. The group size is arbitrary, but we select $G_1 = G_2 = \cdots = G_n = N/n$, as this minimizes the variance.
(2) One test case is then drawn by taking an input t_i in each of these n groups independently and with a PPS sampling according to the above-defined π variable.
(3) Denote with $\pi_{i,r}$ the probability associated with the t_i-th unit in the r-th group, and with $q_r = \sum_{i \in G_r} \pi_{i,r}$ the sum in the r-th group. The unbiased estimators are:

$$\hat{\theta} = \frac{1}{N}\sum_{r=1}^{n} \frac{z_r}{\pi_r / q_r} \quad (5) \qquad \hat{\Delta} = \frac{1}{N}\sum_{r=1}^{n} \frac{\delta_r^2}{\pi_r / q_r} \quad (6)$$

Table 1: `DeepSample` techniques

Technique	Partitioning	Unequal selection	Without replacement
SRS	✗	✗	✗
SUPS	✗	✓	✗
RHC-S	✗	✓	✓
CES	✗	✓	✗
DeepEST	✗	✓	✓
SSRS	✓	✗	✓
GBS	✓	✗	✗
2-UPS	✓	✓	✓

*Cross-entropy Sampling (**CES**).* Cross Entropy-based Sampling (CES) was proposed by Li *et al.* [1]. The CES algorithm builds the sample first selecting randomly an initial set of examples, and then selecting the remaining examples trying to minimize the average cross-entropy between the probability distribution of the m-dimensional representation of neurons output computed on the operational dataset and the selected images. The objective is to sample a set of examples as much as possible representative of the operational dataset, namely if it contains the same proportion of mispredictions as the operational dataset. For CES, the authors demonstrate that the estimator is the same as SRS (Eq. 1 and 2).

*Deep neural networks Enhanced Sampler for operational Testing (**DeepEST**).* Guerriero *et al.* presented DeepEST [2], a technique for DNN operational testing with the twofold objective of accuracy estimation and accuracy improvement. DeepEST exploits adaptive sampling [8] to select a sample providing a close and efficient estimate *and*, at the same time, including a high number of failing examples. The original version of DeepEST works only for classification tasks. We hereafter extend it for regression too, defining the corresponding estimator. The auxiliary variable, χ, is used by DeepEST to define a *weight* $w_{i,j}$ between any pair of examples d_i and d_j of the operational dataset, used to explore the example space adaptively. The weight $w_{i,j}$ is the value of χ_{d_j} if χ_{d_i} exceeds a threshold (i.e., it means that t_i is in an interesting cluster to explore), 0 otherwise. The thresholds are those of the original paper. The strategy acts as follows: the first input is selected via SRS, then a *weight-based sampling* (WBS) is used with probability r to sample the next example (or SRS with probability *1-r*). The example d_i is selected at step k with probability q_{k,t_i}:

$$q_{k,i} = r \cdot \frac{\sum_{j \in s_k} w_{i,j}}{\sum_{h \notin s_k, t_j \in s_k} w_{h,j}} + (1-r) \cdot \frac{1}{N - n_{s_k}} \tag{7}$$

where:

- r: probability of using WBS;
- s_k: current sample (all examples selected up to step k);
- $w_{i,j}$: weight relating example d_j in s_k to example d_i;
- n_{s_k}: the size of the current sample s_k;
- N: the size of the operational dataset.

WBS selects an example d_i proportionally to the sum of weights $w_{i,j}$ of already selected examples toward d_i. We compute the following step-by-step estimators to balance for the adaptive sampling:

$$\hat{\theta} = \frac{1}{n}(z_1 + \frac{1}{N}\sum_{k=2}^{n}\tilde{\theta}_k) \tag{8} \qquad \hat{\Delta} = \frac{1}{n}(\delta_1^2 + \frac{1}{N}\sum_{k=2}^{n}\tilde{\Delta}_k) \tag{9}$$

where z_1 and δ_1^2 are the estimates obtained at step $k = 1$ (hence when $n = 1$), $\tilde{\theta}_k$ and $\tilde{\Delta}_k$ are the Hansen-Hurwitz estimates at step $k > 1$ for the total failures and for the mean-squared error:

$$\tilde{\theta}_k = \sum_{j \in s_k} z_j + \frac{z_i}{q_{k,i}} \tag{10} \qquad \tilde{\Delta}_k = \sum_{j \in s_k} \frac{\delta_j^2}{k-1} + \frac{\delta_i^2/k}{q_{k,i}} \tag{11}$$

The final estimators (Eq. 8, 9) are the sample mean of the step-by-step estimators. For regression, the k-th MSE estimate is $\tilde{\Delta}_k$.

3.4.2 Partition-based techniques. Partition-based techniques split the operational dataset into classes to improve sampling. Stratification in sampling theory aims to split the population to have a small expected intra-stratum variance of the variable to estimate ξ and a large inter-strata variance, so as to sample more from partitions with higher variance. Since the true variance of ξ is unknown, stratification can be done on an estimate of such variance (e.g., computed from a preliminary sample) [17]. However, this would require labeling a subset only just for the purpose of estimating the variance and then applying stratification. Another common solution, that we adopt, is to stratify based on auxiliary variables. Although risky (performance depends on the extent to which they are correlated to ξ), this requires no prior knowledge about ξ. We used $k-$means clustering [28] on χ, with k set to 10 after a preliminary tuning on 30 random samples from MNIST, with $k = 6, 8, 10, 12$.

*Stratified Simple Random Sampling (**SSRS**).* In this scheme, the number of examples to draw from each partition p is computed by the Neyman allocation [22] applied to χ, namely proportionally to the standard deviation of the (normalized) χ values for that partition, and to the size of the partition, N_p. Selection within the partition is without-replacement. The estimators are the weighted sum of the SRS estimates for partitions:

$$\hat{\theta} = \frac{1}{N}(\sum_{p=1}^{P} N_p \hat{\theta}_p) \tag{12} \qquad \hat{\Delta} = \frac{1}{N}(\sum_{p=1}^{P} N_p \hat{\Delta}_p) \tag{13}$$

where $\hat{\theta}_p$ and $\hat{\Delta}_p$ are the within-partition SRS estimators (Section 3.4.1), $P = k = 10$ is the number of partitions.

*Gradient-Based Sampling (**GBS**).* Unlike SSRS, this technique does not initially allocate a sample size for each stratum, but it decides step by step which partition the next example will be drawn from. Inspired by adaptive testing with gradient descent [12], at each step the partition is chosen so as to maximize the reduction of the variance $Var(\hat{\xi})$ of the ξ estimator, by taking the partition with the largest negative gradient: $-\partial Var(\hat{\xi})/\partial n_p$ (ties broken randomly), n_p being the number of examples selected from partition p up to the current step. The selection within the partition is then with replacement. The estimators are the same as SSRS (Eq. 12, 13). Note that the with-replacement SRS, used in GBS, and without-replacement SRS, used in SSRS, have the same mean estimators – they differ for the variance of these estimators.

*Two-stage Unequal Probability Sampling (**2-UPS**).* This technique implements a two-stage sampling scheme, where unequal probability sampling is adopted to select the partition (first stage), and SRS without replacement is adopted to select the example from the chosen partition (second stage). The selection probability for partition p is proportional to the sum of (normalized) χ values (denoted as π_i as in SUPS and RHC-S) within that partition:

$$\psi_p = \frac{\sum_{i=1}^{N_p} \pi_i}{\sum_{p=1}^{P}\sum_{i=1}^{N_p} \pi_i}. \tag{14}$$

Selection of partitions is, clearly, with replacement; Q_p is the number of times partition p is selected. The estimator for this technique is the average over n estimates:

$$\hat{\theta} = \frac{1}{Nn}(\sum_{p=1}^{P}\sum_{i=1}^{Q_p} \frac{z_i N_p}{\psi_p}) \tag{15} \qquad \hat{\Delta} = \frac{1}{Nn}(\sum_{p=1}^{P}\sum_{i=1}^{Q_p} \frac{\delta_i^2 N_p}{\psi_p}) \tag{16}$$

Inner terms $(z_i N_p)/(\psi_p)$ and $(\delta_i^2 N_p)/(\psi_p)$ are Hansen-Hurwitz estimates for the total number of failures and squared errors in partition p, respectively. These estimates are summed up over all partitions and divided by the sample size n to get an average total estimate. The division by N gives $\hat{\theta}$ and $\hat{\Delta}$.

4 EVALUATION

4.1 Research questions and metrics

RQ1: *How do the sampling techniques perform in assessing the operational accuracy of DNN models?*

- **RQ1.1**: *How do the techniques perform for classification?*
- **RQ1.2**: *How do the techniques perform for regression?*

Over $R = 30$ repetitions, we measure the root mean squared error (RMSE) between the accuracy estimates $\hat{\xi}$ and the true accuracy ξ computed on the operational datasets by labeling all the images:

$$RMSE = \sqrt{\frac{\sum_{r=1}^{R}(\xi - \hat{\xi}^2)}{R}} \tag{17}$$

where $\hat{\xi}$ is computed using $\hat{\theta}$ and $\hat{\Delta}$ for classification and regression, respectively. Lower RMSE means higher confidence in the estimate.
RQ2: *How do the sampling techniques perform in detecting failing examples?* One issue with some sampling-based techniques (e.g., CES) is that they, with reason, try to have in the sample the same proportion of failures as in the operational dataset to faithfully estimate the accuracy (what is called the *imitation bias* [2]); but with highly-accurate DNNs, this entails very few failures exposed, which requires engineers to run further testing sessions to expose failures – an issue addressed by DeepEST [2]. Thus a desirable property is to expose a high number of failures, besides the ability to provide unbiased high-confidence estimates.

- **RQ2.1**: Classification task. *How many failures (namely, misclassifications) are exposed by the techniques?*
- **RQ2.2**: Regression task. *How many examples with an inaccurate prediction are selected by the techniques?* Since in regression we have continuous outputs, we measure the number of examples having a difference between true and predicted output (i.e., the offset: $\delta_i = |r - \hat{r}_i|$) greater than or equal to a given value y: $N_{\delta \geq y}$ with y ranging from 0° to 25°, with a step of 2.5°.[3]
- **RQ3**: *How does the budgeted sample size affect performance?* The sample size is directly related to the cost of labelling, as it determines the number of examples to be manually labelled.
  - **RQ3.1**: *How does the size affect the accuracy estimate?*
  - **RQ3.2**: *How does the size affect the failing examples detection?*

To answer RQ1 and RQ2, we consider a budget size of 200, as in [1, 2]. Runs are [(6 techniques ×3 auxiliary variables + 2 techniques not using auxiliary variables, i.e. CES and SRS) ×11 models ×30 repetitions] = 6, 600. For RQ3, we have 5 sample size values: 50, 100, 200, 400, and 800, hence we add further $6,600 \times 4 = 26,400$ runs.

4.2 Subjects

The evaluation is on 11 DNN models on commonly used datasets. Table 2 lists the number of layers and parameters per subject. We consider 3 models for each of the following 3 classification datasets: MNIST [29], CIFAR10 and CIFAR100 [30]. MNIST has 70, 000 entries; CIFAR10 has 60, 000 entries; both have 10 classes. CIFAR100 also has 60, 000 entries, with 100 classes. For regression, we consider 2 models for the Udacity dataset[4] (101, 396 entries for training and 5, 614 for test) for steering angle prediction in Autonomous Driving Systems: *Dave_orig* (DO) and *Dave_dropout* (DD) [1][31].

Table 2: List of experimental subjects

Model	Dataset	Layers	Parameters	Accuracy
A	MNIST	7	6,237	0.903
B		6	97,114	0.948
C		8	545,546	0.933
D	CIFAR10	13	1,084,234	0.715
E		10	258,762	0.790
F		12	550,570	0.651
G	CIFAR100	16	15,047,588	0.663
H		9	564,484	0.574
I		13	1,465,220	0.588
DO	Udacity	13	2,116,983	0.904
DD		15	3,276,225	0.918

Strategies like *transfer learning* are often used to achieve high accuracy [32]. Recht *et al.* [33] showed that if the accuracy is computed on previously unseen data, the claimed accuracy (computed on the test set) of the DNN drops from 3% to 15% on CIFAR10 and from 11% to 14% on ImageNet. Therefore, to have a more realistic accuracy, each DNN is trained "from scratch" by separating training, verification, and operational sets, like in [34]. The verification set is the set of examples used to evaluate the DNN and to compute the generalization error. The operational set is the unlabelled images.

The three datasets are split as follows. For MNIST, 7, 000 images are for training and 2, 500 for verification; the remaining 60, 500 entries are the operational dataset. The models trained with this configuration are all able to achieve accuracy greater than 0.9. For CIFAR10, we use 24, 000 images for training and 2, 500 for verification; the remaining 33, 500 entries are the operational dataset. For CIFAR100, 40, 000 entries are for training and 5, 000 for verification; thus, the operational dataset has 15, 000 images. The greater sizes of the training sets for CIFAR10 and CIFAR100 are due to the higher complexity of the images, to guarantee an acceptable accuracy. For regression models, we use as operational dataset the entire test dataset, as all its examples are unseen during training.

5 RESULTS

5.1 RQ1: operational accuracy assessment

5.1.1 RQ1.1: Classification. To check if there is at least one pair of techniques with a statistically significant difference, we run the Friedman test ($\alpha = .05$) [35] on all subjects. The p-value is lower than $\alpha = .05$ in all the cases, hence the null hypothesis of no difference among the techniques is always rejected. For pairwise comparison, we run the non-parametric post hoc Dunn test [36]. The results are in Figure 1, where gray squares mean *no significant difference* for the pair, white squares mean the technique on the row is statistically better than the one on the column, and black squares the opposite. All exact p-values are in the replication package[1].

On MNIST, DeepEST and 2-UPS significantly differ from the other techniques (which perform approximately in the same way). To better explore the difference, we report three examples in Figure 2a-2c. The first example (Fig. 2a) is on Model A (corresponding to the top-left box in Fig. 1), with *confidence* as auxiliary variable. Here, the RMSE of 2-UPS is by far the worst one; however, it is affected by few outliers due to the inability of the estimator to properly balance, within the available budget, the examples whose auxiliary

[3]The output of the DNN for regression is a steering angle degree; a difference greater than 25° is unrealistic, and never occurred in our experiments.
[4]https://github.com/udacity/self-driving-car.

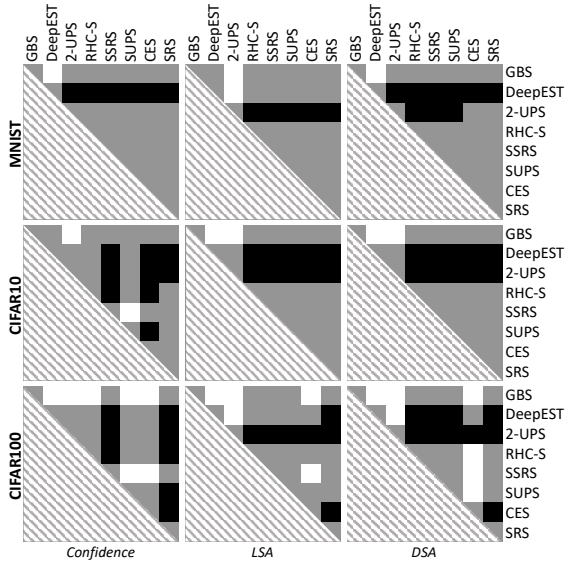


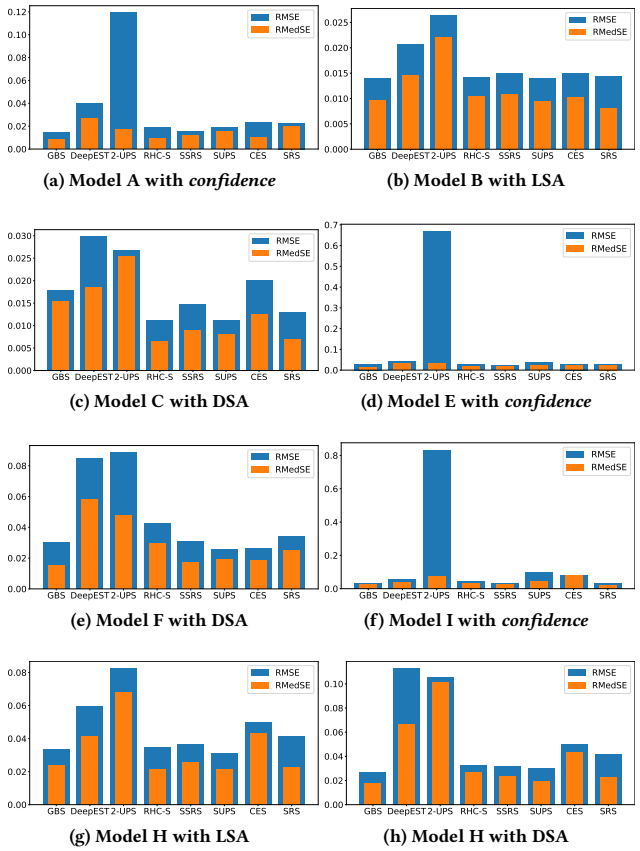
Figure 1: RQ1.1 Dunn test on the classification task

information is incoherent with the result (a.g. failures with high confidence). If we take the root square of the *median* of squared errors, called RMedSE, we see 2-UPS is in line with the others. This causes 2-UPS to be not reported as significantly different by the Dunn test (which is non-parametric, hence robust to outliers). DeepEST, instead, is reported to be significantly worse.

The second example (Fig. 2b) is on Model B with LSA as auxiliary variable (corresponding to the top-middle box in Fig. 1). In this case, 2-UPS performs worse than the others, the second one is DeepEST although the difference is not detected by the Dunn test. The third example (Fig. 2c) is on Model C with DSA auxiliary variable (corresponding to the top-right box in Fig. 1). In this case, both DeepEST and 2-UPS perform worse. In the second and third example, the values of the RMSE and RMedSE for 2-UPS are close (hence, no outliers); this is attributable to the higher representativeness of LSA and DSA, which was more robust than *confidence* to misclassification on those inputs closer to training set.

On CIFAR10, the outliers when using *confidence* in 2-UPS are confirmed and are even more pronounced (e.g. in Fig. 2d). DeepEST and 2-UPS are confirmed to give the worst estimates, consistently for DSA and LSA. The other algorithms are similar (e.g. in Fig. 2e).

On CIFAR100, as for the *confidence*, we have that GBS, SSRS, and SRS differ significantly from the other techniques. As an example, consider Figure 2f (Model I with *confidence*). GBS, SSRS, and SRS exhibit the best values among compared algorithms. Outliers in 2-UPS outliers are confirmed. These are more frequent when 2-UPS is used with *confidence* as auxiliary variable, especially on low-accurate models (the behaviour is more evident with CIFAR10 and CIFAR100, less accurate than MNIST). On the other hand, it is also worth to stress that not all the algorithms relying on the auxiliary variable suffer from unstable results; RHC-S and SUPS are more stable. With LSA (an example in Fig. 2g) and DSA (an example in Fig. 2h), the previous results are confirmed; besides DeepEST and 2-UPS, CES turned out to be the third worst one.



(a) Model A with *confidence*
(b) Model B with LSA
(c) Model C with DSA
(d) Model E with *confidence*
(e) Model F with DSA
(f) Model I with *confidence*
(g) Model H with LSA
(h) Model H with DSA

Figure 2: RQ1.1: Examples

Table 3: RQ1: Overall best and worst RMedSE

	Classification			*Regression*	
Tech.	*Best/Worst*	*Best/Worst*	*Tech.*	*Best/Worst*	*Best/Worst*
DeepEST	0.74%/7.37%	0.69%/3.92%	RHC-S	0.43%/7.54%	0.36%/0.77%
2-UPS	0.65%/14.30%	0.58%/4.25%	SRS	0.69%/2.95%	0.40%/0.53%
CES	1.02%/7.94%	0.69%/0.78%	SSRS	0.58%/3.29%	0.38%/0.77%
GBS	0.61%/3.19%	0.55%/1.71%	SUPS	0.47%/8.77%	0.38%/0.82%

If we look at the RMedSE % values (to be robust to outliers), Table 3 reports the min-max RMedSE by technique over all the dataset, confirming the discussed differences. Looking at the datasets, we have: for MNIST, DeepEST shows the worst value (**3.3**%), while RHC-S has the best values (**1.0**%) (model A, with *DSA*); for CIFAR10, 2-UPS and DeepEST achieve, respectively, **10.7**% and **5.8**% while CES is the best one (**1.8**%) (model F, with *confidence*); for CIFAR100, the 2-UPS RMedSE is **14.3**%, while for GBS it is **2.2**% (model G, *LSA*).

5.1.2 RQ1.2: Regression. The Friedman test gives a p-value lower than $\alpha = .05$ in all the cases, except for DO with the SAE auxiliary variable. Figure 3 shows the results of the Dunn test for pairwise comparison. When using LSA, DeepEST and 2-UPS are significantly worse. Figures 4 and 5 (and Table 3) confirm their higher values of RMSE and RMedSE for both DO and DD models.

When using VAE as auxiliary variable, GBS differs from SRS, but it is almost equivalent to the other algorithms in the DO model.

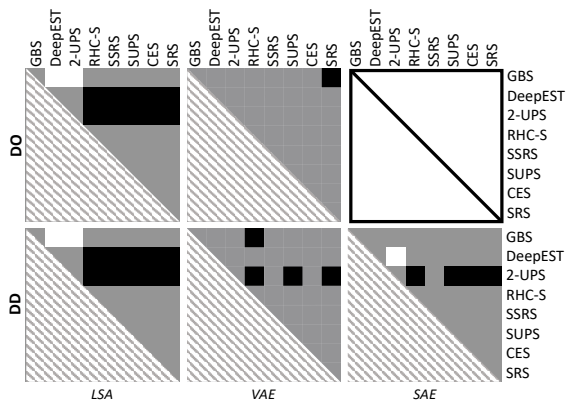
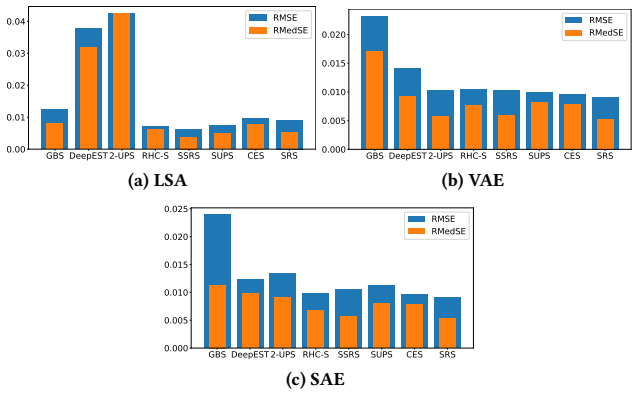


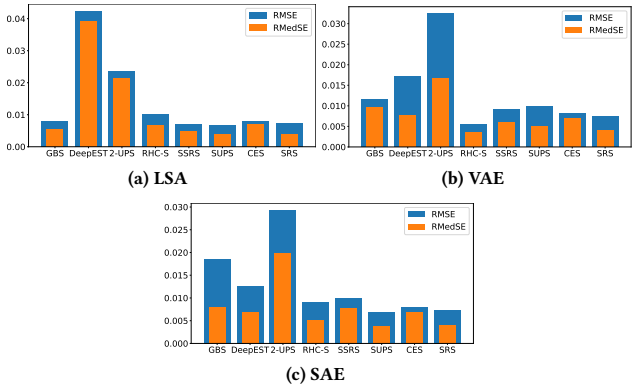
Figure 3: RQ1.2: Dunn test on the regression task



(a) LSA **(b) VAE**

(c) SAE

Figure 4: RQ1.2: DO - Bar charts



(a) LSA **(b) VAE**

(c) SAE

Figure 5: RQ1.2: DD - Bar charts

Figure 4 confirms that GBS has a higher RMSE than the others. For the DD model, 2-UPS differs from SUPS and RHC-S. 2-UPS has the highest RMSE values, while RHC-S has the lowest ones (Figure 5).

With *SAE*, the Friedman test did not detect any difference for DO, while, for DD, 2-UPS is still the worst technique, although it is closer to GBS and SSRS than the previous cases.

Looking at the RMedSE % values, the worst values are always with LSA auxiliary variable: for DO, 2-UPS and DeepEST show the worst values (**4.2**% and **3.2**%, respectively); SSRS has the best value (**0.4**%); for DD, DeepEST and 2-UPS show **3.9**% and **2.1**%, respectively, against SUPS with **0.4**%. The worst case with the autoencoders is in the case of DD with SAE auxiliary variable, with 2-UPS (**2.0**%), while the best one is SRS (**0.4**%). These results are in line with the classification ones.

5.2 RQ2: failing examples detection

For RQ2, we treat classification and regression differently. For the former, we count the number of misclassifications. For the latter, we count the number of examples whose offset δ (predicted *vs* actual value) is greater than a threshold y, with $y \in [0°, 2.5°, 5°, \ldots, 25°]$ – the higher the difference, the more severe the misprediction.

5.2.1 RQ2.1: Classification. Table 4 reports the number of misclassifications broken down by dataset and auxiliary variable – the best mean values are in bold. DeepEST exposes more failures than the others in 7 out of 9 times. 2-UPS has the highest value only for CIFAR10 with confidence, and RHC-S for CIFAR100 with confidence. 2-UPS, RHC-S and SUPS almost equivalently follow DeepEST.

These results counterbalance the DeepEST (as well as 2-UPS) results on the accuracy estimates, which were worse than the others (RQ1.1). The DeepEST adaptive sampling algorithm assumes that failures belong to a rare population, and is conceived to actively spot them. The greater ability to find misclassifications causes a greater variability of the estimates, and more time (i.e., more budget) is needed to converge. A similar problem is observed with 2-UPS; we hypothesize that partitioning combined with unequal sampling (both based on the auxiliary variable χ) can push toward failing examples, but the estimator needs more time to converge. Also, 2-UPS, unlike DeepEST, showed many spikes in the accuracy estimation; the estimator generates spikes every time a failure is

Table 4: RQ2.1: Number of failures in the classification task

		MNIST		CIFAR10		CIFAR100	
χ	Technique	*mean*	*std*	*mean*	*std*	*mean*	*std*
confidence	GBS	28.20	7.23	68.42	8.04	86.17	10.59
	DeepEST	**80.50**	10.33	108.70	9.38	136.44	6.56
	2-UPS	69.50	17.40	**108.92**	11.67	141.48	6.15
	RHC-S	70.56	16.47	106.00	12.83	142.28	6.40
	SSRS	38.44	10.24	78.70	13.52	109.24	6.19
	SUPS	69.82	16.93	106.90	12.31	**143.56**	5.47
	CES	15.64	5.78	55.76	12.63	70.42	7.48
	SRS	14.58	5.32	57.34	13.12	77.99	12.82
LSA	GBS	21.00	5.24	58.19	10.59	84.62	7.37
	DeepEST	**35.89**	10.65	**69.23**	10.35	**119.89**	12.36
	2-UPS	25.04	7.45	61.80	11.26	110.46	20.11
	RHC-S	25.51	7.13	62.94	11.41	110.30	19.33
	SSRS	27.30	6.40	60.09	10.68	93.16	10.73
	SUPS	25.39	6.80	63.46	11.24	110.17	21.65
	CES	15.64	5.78	55.76	12.63	70.42	7.48
	SRS	14.58	5.32	57.34	13.12	77.99	12.82
DSA	GBS	20.27	6.00	63.57	10.97	88.50	7.48
	DeepEST	**72.98**	17.91	**102.37**	8.87	**136.70**	4.93
	2-UPS	25.18	7.19	65.16	11.98	96.31	8.66
	RHC-S	23.94	7.02	65.48	13.79	96.88	9.64
	SSRS	21.69	5.57	58.30	11.96	82.38	6.40
	SUPS	24.72	6.87	65.71	12.09	97.28	10.67
	CES	15.64	5.78	55.76	12.63	70.42	7.48
	SRS	14.58	5.32	57.34	13.12	77.99	12.82

Table 5: RQ2.2: Histogram of offsets $12.5° < \delta < 25°$, with step $2.5°$. DO model

Technique	Auxiliary variable χ: LSA		Auxiliary variable χ: VAE		Auxiliary variable χ: SAE	
GBS	4.2 2.8 1.6 0.7 0.2 0.6	10.2	12.9 6.0 3.0 0.6 0.2 0.5	**23.1**	8.9 4.4 2.3 0.4 0.1 0.5	16.6
DeepEST	12.8 7.0 4.0 1.5 1.3 2.2	28.9	10.3 4.6 2.1 1.2 0.5 0.9	19.5	9.7 4.9 2.2 0.8 0.6 1.0	**19.2**
2-UPS	13.7 7.2 4.5 2.0 1.3 2.4	31.0	10.5 4.8 2.3 0.7 0.4 0.9	19.6	9.0 4.5 2.6 0.9 0.4 0.8	18.2
RHC-S	14.4 8.4 4.7 2.2 1.3 1.9	32.8	8.4 5.0 2.9 1.0 0.7 0.8	18.7	9.8 4.3 2.4 0.8 0.6 0.7	18.7
SSRS	11.2 9.3 4.6 2.3 1.6 2.4	31.4	9.5 4.9 2.3 1.1 0.7 0.6	19.1	9.3 4.4 1.5 0.7 0.5 0.7	17.1
SUPS	13.4 8.2 4.7 2.1 1.2 2.9	**32.7**	8.8 5.0 2.0 0.8 0.8 0.7	18.1	10.2 5.2 1.9 0.9 0.5 0.6	19.1
CES	8.5 5.1 2.9 0.7 0.5 1.2	18.9	8.5 5.1 2.9 0.7 0.5 1.2	18.9	8.5 5.1 2.9 0.7 0.5 1.2	18.9
SRS	9.4 4.0 1.9 0.8 0.4 1.1	17.7	9.4 4.0 1.9 0.8 0.4 1.1	17.7	9.4 4.0 1.9 0.8 0.4 1.1	17.7

Table 6: RQ2.2: Histogram of offsets $12.5° < \delta < 25°$, with step $2.5°$. DD model

Technique	auxiliary variable χ: LSA		auxiliary variable χ: VAE		auxiliary variable χ: SAE	
GBS	3.7 1.9 1.4 0.8 0.8 0.8	9.4	8.3 4.9 1.7 2.0 0.9 1.9	**19.7**	5.7 4.9 1.7 0.9 0.6 1.0	14.7
DeepEST	8.8 6.9 4.2 2.8 1.2 2.2	26.1	4.7 3.0 1.9 1.1 0.5 0.8	12.0	4.8 2.9 2.1 1.1 0.5 0.7	12.1
2-UPS	10.5 7.9 5.9 3.0 1.5 2.3	31.1	4.2 3.0 1.7 1.0 0.4 0.7	11.0	4.8 3.1 1.7 1.1 0.6 0.8	12.0
RHC-S	11.1 7.2 4.7 2.7 1.9 2.3	29.9	4.7 2.5 1.3 0.8 0.5 0.9	10.6	5.3 2.5 1.9 0.9 0.5 0.6	11.8
SSRS	9.1 5.9 4.3 3.3 2.1 2.5	27.3	5.7 2.7 1.8 0.9 0.6 0.7	12.4	5.0 3.5 2.9 1.5 0.9 1.6	**15.4**
SUPS	11.1 8.1 5.2 2.6 2.1 2.8	**31.9**	4.6 3.1 1.7 1.2 0.7 0.7	12.1	4.4 2.9 1.5 1.0 0.7 0.8	11.4
CES	3.9 2.6 1.5 1.1 0.8 0.5	10.3	3.9 2.6 1.5 1.1 0.8 0.5	10.3	3.9 2.6 1.5 1.1 0.8 0.5	10.3
SRS	4.6 2.3 2.1 0.3 0.4 0.1	9.8	4.6 2.3 2.1 0.3 0.4 0.1	9.8	4.6 2.3 2.1 0.3 0.4 0.1	9.8

detected with "misleading" values of χ, namely misclassified examples with values of χ that would indicate a correct classification. For instance, failures with high *confidence*, or with low LSA/DSA. SUPS and RHC-S seem very good compromises between the two – more details in the final discussion. GBS, CES, and SRS detect fewer failures. For SRS and CES this is likely because the former does not use any auxiliary variable, the latter uses cross-entropy, not supposed to be related to failures. GBS and SSRS both use χ only for partitioning; but GBS detects fewer failures likely because the algorithm is thought to minimize the variance of the estimate.

5.2.2 RQ2.2: Regression. Tables 5 and 6 report the histograms of the offset, starting from 12.5° to 25°. Compared to the classification case, the differences here are less pronounced. Looking at the sum of the bins, we notice that CES and SRS select less examples with higher offset with respect to the others under the LSA case, while all the techniques are roughly equivalent with VAE and SAE[5]. GBS has similar poor performance, but it performs much better when used with VAE (consistently with the more unstable RMSE (Fig. 4). SUPS is the best one with LSA. The good performance of partitioning-based techniques (which achieve or even outperform DeepEST) is attributable to a better effect of partitioning when applied to regression compared to classification (since the auxiliary variable, used for partitioning, and the offset are more correlated).

5.3 RQ3: sensitivity analysis

5.3.1 RQ3.1. Accuracy assessment. We synthesize in Tables 7 and 8 the results for classification and regression. Besides the RMSE value at each point,[6] we are interested in figuring out if the techniques smoothly converge as the sample size increase. First, we report for each dataset, technique, auxiliary variable, and model, how many times the minimum RMSE is reached under the given sample size. For instance, 3/3/3 of GBS for sample size 800 in MNIST, means that the minimum RMSE was reached for all the 3 models used with MNIST, using respectively *confidence/LSA/DSA* as auxiliary variable. This is marked as green, and is the expected behaviour.

[5]Note that the histograms for CES and SRS are the same along the three columns of the Table since they do not use LSA/VAE/SAE.

[6]The full set of graphs for each dataset-auxiliary variable-model combination over the sample size are in the replication package.

Table 7: RQ3.1: RMSE sensitivity analysis (conf./LSA/DSA)

	Technique	Minimum RMSE 50	100	200	400	800	Inversions 50>800
MNIST	GBS	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	DeepEST	0/0/0	0/0/0	0/0/0	0/0/1	3/3/2	3/3/3
	2-UPS	1/0/0	1/0/0	0/0/0	0/1/0	1/2/3	1/3/3
	RHC-S	0/0/0	0/0/0	0/0/0	0/1/1	3/2/2	3/3/3
	SSRS	0/0/0	0/0/0	0/0/0	0/0/1	3/3/2	3/3/3
	SUPS	0/0/0	0/0/0	1/0/0	0/0/0	3/3/3	
	CES	0	0	0	0	3	3
	SRS	0	0	0	0	3	3
CIFAR10	GBS	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	DeepEST	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	2-UPS	0/0/0	0/0/1	0/0/0	1/0/0	2/3/2	3/3/3
	RHC-S	0/0/0	0/0/0	0/0/0	1/0/0	2/3/3	3/3/3
	SSRS	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	SUPS	0/0/0	0/0/0	0/0/0	1/0/0	2/3/3	3/3/3
	CES	0	0	0	0	3	3
	SRS	0	0	0	0	3	3
CIFAR100	GBS	0/0/0	0/0/0	0/0/0	0/0/1	3/3/2	3/3/3
	DeepEST	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	2-UPS	0/0/1	0/0/0	0/0/0	0/0/0	3/3/2	3/3/2
	RHC-S	0/0/0	0/0/0	0/0/0	0/1/0	3/2/3	3/3/3
	SSRS	0/0/0	0/0/0	0/0/0	0/0/0	3/3/3	3/3/3
	SUPS	0/0/0	0/0/0	0/0/0	2/1/0	1/2/3	2/3/3
	CES	0	1	1	0	1	2
	SRS	0	0	0	0	3	3

Table 8: RQ3.1: RMSE sensitivity analysis (LSA/VAE/SAE)

	Technique	Minimum RMSE 50	100	200	400	800	Inversions 50>800
Udacity	GBS	0/0/0	0/0/0	0/2/0	0/0/1	2/0/1	2/2/2
	DeepEST	0/0/0	0/0/0	0/0/0	0/0/0	2/2/2	2/2/2
	2-UPS	0/0/0	0/0/0	0/0/0	1/1/0	1/1/2	2/2/2
	RHC-S	0/0/0	0/0/0	0/0/0	0/0/0	2/2/2	2/2/2
	SSRS	0/0/0	0/0/0	0/0/0	0/0/0	2/2/2	2/2/2
	SUPS	0/0/0	0/0/0	0/0/0	1/0/0	1/2/2	2/2/2
	CES	0	0	1	0	1	2
	SRS	0	0	0	0	2	2

When this is not true for at least one case, we mark it as red, and correspondingly mark as yellow those cells in the same row (with sample size smaller than 800) where the minimum was reached.

There are many cases where the minimum is not achieved with the largest sample size (red cells). For instance, the instability of 2-UPS makes it even reach the best values with a sample size 50 (MNIST and CIFAR100) and sample size 100 (MNIST and CIFAR10). CES with CIFAR100 also has the same convergence problem, while it is stable in MNIST and CIFAR10. In the other red cases, the minimum is at 400. SRS is the most stable technique due to its simplicity and independence from any auxiliary variable. GBS and DeepEST are also stable for 2 out of 3 datasets; in the single bad cases, they converge at sample size 400. For regression, performance is better; GBS is more unstable, while the others converge at 800 with few exceptions at 400 and one (CES) at 200.

Tables 7 and 8 report also in how many cases the RMSE with budget 50 is smaller than the one obtained with budget 800 (we call this an *inversion*. This denotes convergence problems, and the cell is marked red. There are 5 cases in which this happens: 3 for 2-UPS (2 with MNIST and 1 with CIFAR100), 1 for SUPS and 1 for CES (both with CIFAR100). This case never occurs with the regression dataset for which techniques converge more stably. SRS is still the most stable technique for both classification and regression, as there is no violation of convergence. Again, 2-UPS is the most affected one.

Table 9: RQ3.2: Failures sensitivity analysis (conf./LSA/DSA)

	Technique	*mean(min)*	$F_{800/50}$	*mean(max)*
MNIST	GBS	5.3/5.8/4.5	26.3/15.7/21.9	136.5/91.9/95.9
	DeepEST	19.3/8.0/17.9	16.7/17.7/16.5	321.6/140.0/295.9
	2-UPS	17.4/5.8/5.9	15.9/17.1/16.9	277.2/100.5/98.0
	RHC-S	17.1/6.7/6.2	16.1/15.1/16.0	274.2/101.8/100.3
	SSRS	10.0/6.9/5.5	15.5/15.8/16.6	153.3/107.4/90.9
	SUPS	17.7/6.2/5.8	16.0/16.4/17.3	282.6/102.6/100.9
	CES	3.8	16.1	61.5
	SRS	3.8	15.9	58.6
CIFAR10	GBS	15.4/14.9/14.4	17.9/15.9/18.3	272.5/238.7/261.1
	DeepEST	26.9/17.1/25.3	16.1/16.2/16.2	432.7/277.7/408.6
	2-UPS	26.7/15.8/16.8	16.1/16.1/15.8	431.8/252.7/264.2
	RHC-S	27.2/15.9/16.1	15.7/16.0/16.3	427.7/252.6/262.4
	SSRS	19.7/15.3/14.4	16.0/15.7/16.0	315.1/239.4/231.1
	SUPS	26.7/15.1/16.4	16.2/16.6/16.2	433.8/250.6/263.3
	CES	14.1	15.2	216.5
	SRS	13.8	16.4	227.6
CIFAR100	GBS	21.5/21.3/21.7	15.9/15.9/16.5	341.0/339.2/357.8
	DeepEST	33.7/29.8/34.9	16.2/16.0/11.3	546.6/475.3/393.3
	2-UPS	35.7/27.2/24.5	15.9/16.0/15.7	565.9/434.7/385.8
	RHC-S	35.3/27.6/24.1	15.9/15.7/15.9	560.9/432.6/383.3
	SSRS	28.2/22.7/20.8	15.5/16.5/15.9	436.5/374.5/329.8
	SUPS	35.2/27.4/23.8	16.2/16.2/16.4	571.2/441.1/389.7
	CES	19.7	13.8	270.5
	SRS	20.8	15.2	315.0

Table 10: RQ3.2: Failures sensitivity analysis (LSA/VAE/SAE)

	Technique	*mean(min)*	$F_{800/50}$	*mean(max)*
Udacity	GBS	4.2/3.7/4.1	3.0/34.1/14.9	12.4/122.8/57.8
	DeepEST	6.8/3.6/3.6	16.2/16.6/18.0	109.3/60.3/61.2
	2-UPS	7.4/3.8/3.4	16.6/15.5/18.9	123.1/59.1/61.2
	RHC-S	8.1/3.5/3.7	13.8/17.3/16.4	112.1/60.8/61.5
	SSRS	7.5/4.2/4.2	15.6/15.1/15.2	117.5/64.2/64.9
	SUPS	8.1/3.7/3.8	15.9/16.5/16.2	127.3/60.6/61.1
	CES	3.1	20.9	65.3
	SRS	3.3	17.9	58.3

5.3.2 RQ3.2: failing examples detection. Results for this question are summarized in Tables 9. To count the failures in the case of regression, we consider as failures all the predictions with an error on the steering angle greater than 12.5°. The Table reports the mean of the minimum and maximum number of failures (over the sample sizes), and the ratio between the number of failures detected with samples size 800 and with 50 ($F_{800/50}$). In all the cases, there is no *inversion*; the number of failures constantly increases with the budget size – we noticed it approximately doubles as the sample size doubles (detailed ratios are in the replication package) for both classification and regression. The expectation in this case is fully matched by all the techniques. Looking at $F_{800/50}$, all the results of RQ2 are confirmed for all the evaluated budget sizes.

6 DISCUSSION

Sampling technique. Table 11 summarizes performance of the techniques. We count the number of times a technique is among the top-3 ones in terms of RMSE (RMedSE) and failures, considering all *<model, auxiliary variable, sample size, dataset>* configurations. For classification, on the RMSE (RMedSE), the techniques occurring more often are GBS, SSRS, SUPS, SRS, RHC-S. Considering the Dunn's test results (Figure 1), the differences among these techniques is slight, as they are statistically equivalent in almost all the cases. DeepEST, 2-UPS and, to a less extent, CES have statistically worse RMSE, with 2-UPS also having problems of convergence (Sec.

Table 11: Number of occurrences in the top-3 ranking out of 135 (classification) and 30 (regression) configurations

	Classification			Regression		
Technique	RMSE	RMedSE	Failures	RMSE	RMedSE	Failures
GBS	95	79	12	0	0	16
DeepEST	2	3	106	0	1	7
2-UPS	5	10	85	0	3	13
RHC-S	58	51	91	18	15	13
SSRS	88	86	15	19	24	19
SUPS	64	58	94	16	18	16
CES	36	51	0	20	14	6
SRS	57	67	2	17	15	0

Table 12: Number of best-performing occurrences out of 270 (classification) and 60 (regression) configurations

	Classification				Regression		
aux.	RMSE	RMedSE	Failures	aux.	RMSE	RMedSE	Failures
conf.	73	82	243	LSA	35	30	52
LSA	85	82	8	VAE	12	17	7
DSA	112	106	19	SAE	13	13	1

5.3). On the other hand, these are confirmed to be better at spotting failures, as they heavily rely on the auxiliary variable. Looking also at the best/worst RMSE (and RMedSE) values (e.g. in Figure 2). DeepEST should be chosen when the tolerable estimate RMSE is up to approximately 10% and a high number of failures is desired.

Considering both the RMSE (RMedSE) and failures together, techniques like SUPS and RHC are the only ones having good performance for both metrics, hence are suitable choices if we look for good average performance on both accuracy estimation and failure exposure. Regarding regression, CES is the first technique for RMSE, followed by SSRS; also in this case, significant differences (Figure 3) mostly regard DeepEST and 2-UPS. For failures, SSRS is the best technique, followed by SUPS and GBS. SSRS is the best one for regression if we consider both metrics together, followed by SUPS and RHC-S. If we take the sum over the entire row (classification and regression), SUPS is the one occurring most often, followed by RHC-S and SSRS. Hence these are the best choices for good average performance for all the needs; the others work particularly well for one of the two metrics and/or task (classification/regression).

Partitioning. Partitioning (based on auxiliary variables) works well for accuracy estimate, except when combined with unequal selection probability (2-UPS). SSRS is also good at exposing failures, but just for regression, especially when the LSA is used for the partitioning, as this is better correlated to (in)accuracy.

With/out replacement. We found no remarkable and consistent advantage of without-replacement sampling; techniques like SUPS (that is with replacement) works well for both RMSE and failure exposure and, for its simplicity, works well in all scenarios.

Auxiliary variable. For classification, auxiliary variable are especially beneficial for failures exposure: DeepEST, 2-UPS, RHC-S and SUPS all use the auxiliary variable for selection and are the best ones in detecting failures. This is approximately confirmed for regression, where however SSRS is the best one.

Table 12 reports the number of times that each auxiliary variable yields the best RMSE (RMedSE) and the number of failures. For classification, DSA and *confidence* are the best variables for the RMSE (RMedSE) and number of failures, respectively. It is important to highlight that *confidence* is cheaper to collect, as it comes with the output of the classification. For regression, LSA achieves the best results. The variables derived by SAE/VAE perform poorly.

We have looked, for each model, at the correlation between the auxiliary variables and failures (for classification) and the offset between predicted and actual values (for regression). For classification, all the three variables are significantly correlated (p-value $<< 0.05$) with failures in all (9 models $\times$ 3 variables = 27) cases[7], justifying the high failure exposure ability of techniques relying on them. For regression, the Spearman correlations are: for LSA, $\rho = 0.238$ and $\rho = 0.187$ (for DO and DD model, respectively), both statistically significant; for VAE, $\rho = 0.010$ and $\rho = -0.005$, both not statistically significant; for SAE, $\rho = -0.001$ and $\rho = 0.007$, both not significant. This explains the large difference between variants using LSA and those using VAE/SAE in all RQs, especially for techniques using the auxiliary variable (such as DeepEST, 2-UPS, RHC), and confirms that the selection of such a variable is crucial.

7 THREATS TO VALIDITY

As for the selection of the experimental subjects, we have considered publicly available DNNs [34]; we have however re-trained them from scratch to have realistic accuracy and avoid the mentioned inflated accuracy issue described in [33]. The choice of the sample size affects the results. We ran a sensitivity analysis under five (from 50 to 800) values of the sample size. Different values could yield different results. The evaluation does not include an extensive analysis of partitioning. We ran k-means, a widely-known algorithm, with $k = 10$ partitions, after a preliminary tuning on 30 random samples from MNIST and $k = 6, 8, 10, 12$. Extending the fine-tuning of k to all cases would allow improving performance. Despite code inspection by all co-authors to ensure that the prototypes are correct, the presence of defects cannot be excluded. External validity is undermined by the number of models and datasets; we considered state-of-the-art DNNs and widely-used datasets for classification and regression. The replicability of the experiments with artefacts available mitigates this threat.

8 CONCLUSIONS

We presented `DeepSample`, a framework encompassing a set of sampling-based techniques for DNN operational accuracy assessment. We implemented techniques with and without partitioning, with and without replacement, with and without auxiliary variables to drive the selection, and empirically evaluated them on both classification and regression, in terms of accuracy estimation and number of failures. The findings pertaining to the individual techniques, as well as to the key factors impacting the sampling algorithms, serve as guidance for testers to select the technique depending on the need (e.g., accurate estimate and/or many failing examples) and on the auxiliary information available to expedite sampling, and serve researchers to define new, possibly better, techniques exploiting the advances in the theory of statistical sampling.

9 DATA AVAILABILITY

All results and the artefacts for replication are available at: https://anonymous.4open.science/r/DeepSample-DBE9.

[7]The detailed results are reported in the online repository.

REFERENCES

[1] Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and J. Lü. Boosting Operational DNN Testing Efficiency through Conditioning. In *Proc. 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, pages 499–509. ACM, 2019.

[2] A. Guerriero, R. Pietrantuono, and S. Russo. Operation is the Hardest Teacher: Estimating DNN Accuracy Looking for Mispredictions. In *Proceedings of the IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 348–358. IEEE, 2021.

[3] P. A. Currit, M. Dyer, and H. D. Mills. Certifying the reliability of software. *IEEE Transactions on Software Engineering*, SE-12(1):3–11, 1986.

[4] H. D. Mills, M. Dyer, and R. C. Linger. Cleanroom software engineering. *IEEE Software*, 4(55):19–24, 1987.

[5] R. C. Linger and H. D. Mills. A case study in cleanroom software engineering: the IBM COBOL Structuring Facility. In *12th International Computer Software and Applications Conference*, COMPSAC, pages 10–17. IEEE, 1988.

[6] R. H. Cobb and H. D. Mills. Engineering software under statistical quality control. *IEEE Software*, 7(6):45–54, 1990.

[7] J. D. Musa. Software reliability-engineered testing. *Computer*, 29(11):61–68, 1996.

[8] D. G. Horvitz and D. J. Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American Statistical Association*, 47(260):pp. 663–685, 1952.

[9] J. Lv, B. Yin, and K. Cai. Estimating confidence interval of software reliability with adaptive testing strategy. *Journal of Systems and Software*, 97:192–206, 2014.

[10] K. Cai, C. Jiang, H. Hu, and C. Bai. An experimental study of adaptive testing for software reliability assessment. *Journal of Systems and Software*, 81(8):1406–1429, 2008.

[11] K. Cai, Y. Li, and K. Liu. Optimal and adaptive testing for software reliability assessment. *Information and Software Technology*, 46(15):989–1000, 2004.

[12] J. Lv, B. Yin, and K. Cai. On the asymptotic behavior of adaptive testing strategy for software reliability assessment. *IEEE Transactions on Software Engineering*, 40(4):396–412, 2014.

[13] A. Podgurski, W. Masri, Y. McCleese, F.G. Wolff, and C. Yang. Estimation of software reliability by stratified sampling, 1999.

[14] F.b.N. Omri. Weighted statistical white-box testing with proportional-optimal stratification. In *Proc. 19th International Doctoral Symposium on Components and Architecture*, WCOP'14, pages 19–24. ACM, 2014.

[15] D. Cotroneo, R. Pietrantuono, and S. Russo. Relai testing: A technique to assess and improve software reliability. *IEEE Trans. on Software Engineering*, 42(5):452–475, 2016.

[16] R. Pietrantuono and S. Russo. Probabilistic sampling-based testing for accelerated reliability assessment. In *IEEE International Conference on Software Quality, Reliability and Security*, QRS, pages 35–46. IEEE, 2018.

[17] R. Pietrantuono and S. Russo. On adaptive sampling-based testing for software reliability assessment. In *27th International Symposium on Software Reliability Engineering*, ISSRE, pages 1–11. IEEE, 2016.

[18] R. Pietrantuono, S. Russo, and A. Guerriero. Run-time reliability estimation of microservice architectures. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 25–35, 2018.

[19] R. Pietrantuono, S. Russo, and A. Guerriero. Testing microservice architectures for operational reliability. *Software Testing Verification and Reliability*, 30(2), 2020.

[20] J. Chen, Z. Wu, Z. Wang, H. You, L. Zhang, and M. Yan. Practical accuracy estimation for efficient deep neural network testing. *ACM Trans. Softw. Eng. Methodol.*, 29(4), oct 2020.

[21] J. Zhou, F. Li, J. Dong, H. Zhang, and D. Hao. Cost-effective testing of a deep learning model through input reduction. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pages 289–300, 2020.

[22] S. L. Lohr. *Sampling: Design and Analysis*. Duxbury Press, 2nd edition, 2009.

[23] J. Kim, R. Feldt, and S. Yoo. Guiding Deep Learning System Testing Using Surprise Adequacy. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

[24] M. P. Wand and M. C. Jones. *Kernel smoothing*. CRC press, 1994.

[25] A. Stocco, M. Weiss, M. Calzana, and P. Tonella. Misbehaviour prediction for autonomous driving systems. In *Proc. of the IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 359–371. ACM, 2020.

[26] H. H. Morris and N. H. William. On the Theory of Sampling from Finite Populations. *The Annals of Mathematical Statistics*, 14(4):333–362, 1943.

[27] J.N.K. Rao, H.O. Hartley, and W.G. Cochran. On a simple procedure of unequal probability sampling without replacement. *Journal of the Royal Statistical Society. Series B (Methodological)*, 24(2):482–491, 1962.

[28] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[29] Y. LeCun and C. Cortes. MNIST handwritten digit database. http://yann.lecun.com/exdb/mnist/, 2010.

[30] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical Report TR-2009, University of Toronto, 2009.

[31] K. Pei, Y. Cao, J. Yang, and S. Jana. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. *Communications of the ACM*, 62(11):137–145, 2019.

[32] M. Shaha and M. Pawar. Transfer Learning for Image Classification. In *Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 656–660, 2018.

[33] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do ImageNet Classifiers Generalize to ImageNet? In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of Machine Learning Research (PMLR)*, volume 97, pages 5389–5400, 2019.

[34] A. Guerriero, M. R. Lyu, R. Pietrantuono, and S. Russo. Assessing operational accuracy of cnn-based image classifiers using an oracle surrogate. *Intelligent Systems with Applications*, 17:200172, 2023.

[35] R. L. Iman and J. M. Davenport. Approximations of the critical region of the fbietkan statistic. *Communications in Statistics - Theory and Methods*, 9(6):571–595, 1980.

[36] A. Dinno. Nonparametric pairwise multiple comparisons in independent groups using dunn's test. *The Stata Journal*, 15(1):292–300, 2015.