IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 April 2023, accepted 26 April 2023, date of publication 5 May 2023, date of current version 22 May 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3273530

RESEARCH ARTICLE

Metamodelling of Noise to Image Classification Performance

JENS DE HOOG[1], ALI ANWAR[1], (Member, IEEE), PHILIPPE REITER[1], SIEGFRIED MERCELIS[1], AND PETER HELLINCKX[2]
[1]Faculty of Applied Engineering, IDLab, University of Antwerp-imec, 2000 Antwerp, Belgium
[2]Faculty of Applied Engineering, University of Antwerp, 2020 Antwerp, Belgium

Corresponding author: Jens de Hoog (jens.dehoog@uantwerpen.be)

This work was supported in part by the Flemish Government under the "Onderzoeksprogramma Artificifile Intelligentie (AI) Vlaanderen" Program, in part by the Shared Situational Awareness for Vessels Project (SSAVE), and in part by the Flanders Innovation and Entrepreneurship (VLAIO) and the Blue Cluster.

ABSTRACT Machine Learning (ML) has made its way into a wide variety of advanced applications, where high accuracies can be achieved when these ML models are evaluated in the same context as they were trained and validated on. However, when these high-accuracy models are exposed to out-of-distribution points such as noisy inputs, their performance could potentially degrade significantly. Recommending the most suitable ML model that retains a higher accuracy when exposed to these noisy inputs can overcome this performance degradation. For this, a mapping between the noise distribution at the input and the resulting accuracy needs to be obtained. Though, this relationship is costly to evaluate as this is a computationally intensive task. To minimize this computational cost, we employ metalearning to predict this mapping; that is, the performance of different ML models is predicted given the distribution parameters of the input noise. Although metalearning is an established research field, performance predictions based on noise distribution parameters have not been accomplished before. Hence, this research focuses on predicting the per-class classification performance based on the distribution parameters of the input noise. For this, our approach is twofold. First, in order to gain insights in this noise-to-performance relationship, we analyse the per-class performance of well-established convolutional neural networks through our multi-level Monte Carlo simulation. Second, we employ metalearning to learn this relationship between the input noise distribution and the resulting per-class performance in a sample-efficient way by incorporating Latin Hypercube Sampling. The noise performance analyses present novel insights about the per-class performance degradation when gradually increasing noise is augmented on the input. Additionally, we show that metalearning is capable of accurately predicting the per-class performance based on the noise distribution parameters. We also show the relationship between the number of metasamples and the metaprediction accuracy. Consequently, this research enables future work to make accurate classifier recommendations in noisy environments.

INDEX TERMS Noise propagation, image classifiers, classification performance, monte carlo simulation, metalearning.

I. INTRODUCTION

In recent years, Machine Learning (ML) has made strides in advanced autonomous applications, such as robotics [1], autonomous driving vehicles [2] and inland vessels [3]. The tasks of ML within these applications range from lane detection and follower algorithms, to sophisticated object-detection systems on a wide range of objects [4].

Traditionally, these ML systems have been trained on a set of data points in a certain context and are validated within that same context. This way of training and validation is proven to be very effective for operating and reasoning in many different use case domains [5]. However, problems arise when the ML model is operating in a context that was

The associate editor coordinating the review of this manuscript and approving it for publication was Jeon Gwanggil.

not represented in the dataset, such as noise on the input. The model then would make predictions that could be ambiguous and result in a cascade of problems [5].

One potential solution is to include these discrepancies into the learning process, so that the ML algorithm learns to generalise over them. This has been done in previous research, where the datasets have been augmented with certain noises; the algorithm reasoned over what it needs to do in case such noises appear [6]. Although this is a common technique in research regarding robustness of ML models, this does not fully solve the problem. Data points that are still outside of the learned context, also called *out-of-distribution*, would still result in ambiguous outputs [7]. An additional complexity is the fact that it is nearly impossible to take all discrepancies into account, as this implies the need for datasets that cover this wide range of noises and faults.

Another potential solution is to tackle the problem through a divide-and-conquer paradigm, since no single ML algorithm can achieve full coverage over all possible contexts, discrepancies and noises. This is also called the '*no free lunch*' theorem [8], [9]. Instead, one could design different ML models where each performs sufficient enough in one context, but lacks performance in others. Therefore, given a certain input with a particular noise characteristic, the model with the best performance on this input could be chosen in order to provide the best output. Differences in performance between ML models originate from different aspects, such as hyperparameters of the ML model or the dataset used for training.

Such a decision-making problem for similar algorithms in different contexts has already been addressed by Garcia et al. [10]. The authors analyse the performance of different noise filters in certain contexts where noise is present. As expected, these filters all perform differently in terms of denoising their input. Hence, by recommending algorithms through metamodelling, one could pick the filter with the best expected performance given the characteristics of the present dataset.

In this research, we aim to leverage this metamodelling-based classifier recommendation approach for predicting the per-class image classification performance given the parameters of the noise distribution on the input. In this way, selecting the best performing classifier can then be based on the per-class performance, which has not been done before. Therefore, in this paper, we propose a twofold methodology for image classifying artificial neural networks (ANNs). First, their per-class classification performance is analysed by propagating a prescribed range of noise distributions through the models. Second, for each ANN, this analysis is then captured in a corresponding metaset and learned by a metamodel, which covers the relation between the applied noise distribution characteristics and the resulting classification performance.

The remainder of this paper is as follows. Section II elaborates on related work in the field of noise propagation analyses and metalearning. In Section III, we elaborate on the proposed methodology. Section IV discusses the experimental setup used to validate our approach. Section V and VI elaborate and form a discussion on the results, respectively. Finally, Sections VII and VIII present a conclusion and possible tracks for future work, respectively.

II. RELATED WORK

In this research, we focus on analysing the relationship between noise at the input of an image classifier and the resulting per-class performance, as well as learning this relationship with metalearning to predict the per-class performance based on the noise distribution parameters. Therefore, this section first elaborates on related work regarding noise propagation methods, after which related research in metalearning is discussed.

A. NOISE PROPAGATION ANALYSIS

Several related works have carried out noise propagation analyses on image classifiers. Nemcovsky et al. [11] and Liu et al. [12] have examined the performance of CIFAR-10 classifiers when exposed to adversarial noise. Hendrycks and Dietterich [13] have compared the mean Corruption Errors of different classifiers to adversarial and common noises on the ImageNet dataset. Liang et al. [14] and Berend et al. [7] compared the classification performance of classifiers when exposed to out-of-distribution samples from CIFAR-10, which includes the augmentation of common noises such as Gaussian and Uniform noise.

To perform noise propagation analyses of neural networks with noise distributions, different methods exist to acquire performance insights. Abdelaziz et al. [15] and Nathwani et al. [16] compared different techniques to perform uncertainty propagation through ANN-based automated speech recognition systems. Reference [15] performed uncertainty propagation with a multivariate Gaussian distribution to approximate uncertainty of the acoustic score at the output. They compared the usage of layer-wise Unscented Transform and piece-wise exponential approximations, along with network-wise Unscented Transform and Monte Carlo approximations. They concluded that Monte Carlo performed the best approximation in all circumstances, with the network-wise Unscented Transform as second best approximator. Reference [16] performed similar experiments and also showed that Monte Carlo consistently outperforms the Unscented Transform approximation. Nemcovsky et al. [11] use the Monte Carlo simulation for improving adversarial robustness of CIFAR-10 image classifiers via randomized smoothing of the classifier. Liu et al. [12] employed an adaptation of Monte Carlo simulation where N clean images are perturbed with corruptions by perturbing each image M times. With this, they analysed the robustness of models via the ε-Empirical Noise Insensitivity metric.

Considerable work has already been carried out regarding robustness of image classifiers and noise propagation methods. However, these works have only considered the global robustness and classification performance of the classifier.

To our knowledge, no work has been done regarding the relation between input noise and the robustness and classification performance of the individual classes.

B. METALEARNING

Three major categories exist within metalearning over Artificial Intelligence (AI) models [17]: (i) regression-based prediction of the performance of a certain AI model, (ii) single-label classification-based prediction of the most suitable AI model for a given task, and (iii) multi-label classification-based prediction of a ranking of the most suitable AI models for a given task.

Reif et al. [17] researched the possibility of creating a regression-based metamodel for non-expert end-users. They considered that predicting performance values of a classifier based on dataset characteristics was most suitable, as this would provide the most valuable insights for the end-users. Garcia et al. [10] tackled the problem of estimating the most suitable filter against noise in classification datasets. They do so by creating a regression-based metamodel which maps the relation between the characteristics of noise datasets and the induced performance of a classifier. In the work of Muñoz et al. [18], an objective way of assessing the performance of supervised learning methods is proposed. Additionally, they designed a methodology where the relation between dataset characteristics and objective performance measures is captured in a regression-based metamodel. Kotlar et al. [19] elaborate on the performance of anomaly detectors on different datasets. They capture the relationship between characteristics of the anomaly-containing datasets and the performance of detectors into a regression-based metamodel. Eggensperger et al. employ metaregressors to benchmark hyperparameter optimization systems [20] and algorithm configuration procedures [21].

In the field of using metamodels for predicting the best AI model for a given task, several works have been proposed. Lorena et al. [22] proposed a methodology for predicting the best classifier given the complexities of increasingly complex synthetic datasets. Additionally, they also propose metalearning techniques for recommending parameters of a certain AI model. Next to that, they also designed a metaregressor which predicts the expected Normalized Mean Squared Error of a regression-based AI model. Ler et al. [23] elaborate on the selection of algorithms via metamodelling by focusing on the complexity of the dataset itself, without having prior knowledge about the performance of a classifier on it. With this, they create a set of metafeatures which is less model-centric and focuses more on the general complexity. The metaset labels consist of the predictive accuracy of the underlying AI model. Aguiar et al. [8] focuses on recommending the most performant multi-target regressor for a given set of dataset characteristics.

In terms of metamodels used in research, different approaches exist. The survey of Hutter et al. [24] notes that different metaregressors have been used for similar performance prediction applications, and that the choice of metamodel depends on the metasamples. Garcia et al. [10] employed metamodels based on the Random Forest (RF) and k-Nearest Neighbours (k-NN) techniques. They concluded that the RF implementation performed best. Eggensperger et al. [20], [21] also selected RF and k-NN, along with Gaussian Processes (GP), Gradient Boosting (GB), Support Vector Regression (SVR) and Ridge Regression (RR). They concluded that GB and RF performed best, with RF performing better on metasets with a larger amount of samples and parameters. Horváth et al. [25] used metaregressors based on RF and k-NN to predict hyperparameters of Decision Tree (DT) and Support Vector Machines (SVM) algorithms. Roy et al. [26] discussed the usage of linear metaregressors, along with DTs and non-linear models such as SVRs and Multi-Layer Perceptrons (MLPs).

In order to generate the metasets on which the metamodels are trained on, metasamples need to be gathered that originate from the underlying ML model. Jin et al. [27] focus on creating surrogate models of more complex and expensive AI models. As the samples are of key importance in the performance of surrogate models, the authors propose an adaptive sampling strategy based on Latin Hypercube Sampling (LHS) where the performance of the surrogate model guides the process of gathering extra samples. Metta et al. [28] research the generation of surrogate models for feasibility region analysis of complex AI models. For this, they also make use of LHS when building the initial surrogate model. In the research of Duchanoy et al. [29], the authors conducted a compact survey for which sampling technique to use for their metamodelling use case. Out of the possible options, candidates were Monte Carlo simulation, LHS, Uniform Design and Voronoi sampling. The authors chose the latter, as this was most suitable for their research project. Muñoz et al. [18] used LHS to sample instances from a 2D design space.

III. TWOFOLD METHODOLOGY

The proposed methodology is designed in a twofold way. Firstly, we elaborate on a methodology to propagate noise through an image classifier, such that the relation between the input noise characteristic and the resulting per-class performance can be evaluated. Secondly, we aim to capture this noise-to-performance relation of the classifier into a metamodel. Given the characteristic of the input noise, this metamodel is then able to predict the per-class performance of a certain image classifier.

A. NOISE PROPAGATION THROUGH IMAGE CLASSIFIERS

In this research, we aim to observe the performance of an image classification ANN when exposed to noise on the input. We carry out this observation through our multi-level Monte Carlo simulation, using the whole validation set of the data the network has been trained on. The technique is shown in Figure 1 via steps A to D.

First, in Eq. (1), the original input image $\mathbf{X} \in \mathbb{R}^p$ is augmented with a noise sample $\mathbf{E} \in \mathbb{R}^p$, sampled from

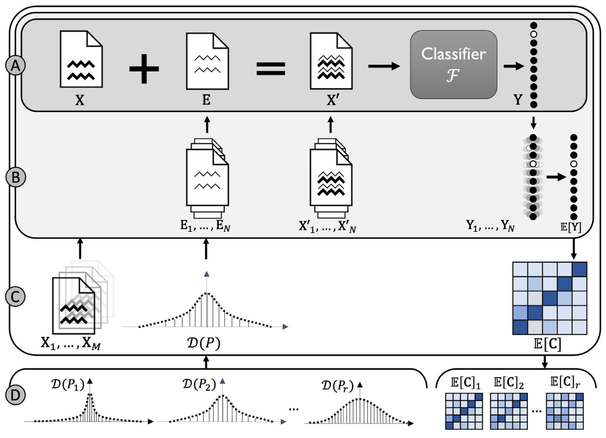


FIGURE 1. The multi-level Monte Carlo technique is visually shown via steps *A* through *D*. In step *A*, a noisy image X′ is propagated through classifier $\mathcal{F}$. In step *B*, step *A* is carried out for multiple noise samples drawn from the same noise distribution $\mathcal{D}(P)$. The resulting outputs are aggregated into expected output $\mathbb{E}$[Y]. Step *C* carries out the previous steps for all images in the dataset, resulting in an expected confusion matrix $\mathbb{E}$[C]. Finally, in step *D*, all previous steps are carried out for multiple noise distributions to be applied on the input images, resulting in a corresponding set of expected confusion matrices.

distribution $\mathcal{D}$ with parameter tuple $P \in \mathbb{R}^s$. The noisy input is then propagated through a classifying ANN $\mathcal{F} : \mathbb{R}^p \rightarrow \mathbb{R}^q$, which results in a classification output $\mathbf{Y} \in \mathbb{R}^q$. This is shown in Figure 1A. Note that this research primarily focuses on additive noise. However, this methodology can also be applied to other types of noise, such as multiplicative noise models.

$$\mathbf{Y}(P) = \mathcal{F}\left(\mathbf{X} + \mathbf{E} \sim \mathcal{D}\left(P\right)\right) \tag{1}$$

The process of propagating a noisy image in Eq. (1) is now performed N times in Eq. (2), with N as a hyperparameter. Each time, a new noise sample is augmented onto $\mathbf{X}$, after which each noisy image is propagated through $\mathcal{F}$. This results in N corresponding classification outputs $\mathbf{Y}$. By averaging all outputs, we get the expected value $\mathbb{E}[\mathbf{Y}]$, which represents the expected output of $\mathcal{F}$ when noise distribution $\mathcal{D}\left(P\right)$ has been added on image $\mathbf{X}$. By adjusting hyperparameter N, we can adjust the accuracy of the result $\mathbb{E}[\mathbf{Y}]$. This process is presented in Figure 1B.

$$\mathbb{E}\left[\mathbf{Y}(P)\right] = \frac{1}{N}\sum_{n=1}^{N}\mathcal{F}\left(\mathbf{X} + \mathbf{E}_n \sim \mathcal{D}\left(P\right)\right) \tag{2}$$

In Eq. (3) and Figure 1C, the aforementioned process is leveraged over the whole validation set $\mathbf{X}_{1\ldots M}$ the model $\mathcal{F}$ was validated on, with M as the total amount of images in the set. All expected values $\mathbb{E}[\mathbf{Y}]_{1\ldots M}$ are combined together into an expected value $\mathbb{E}[\mathbf{C}]$, which represents the expected confusion matrix of $\mathcal{F}$ when exposed to noise distribution $\mathcal{D}\left(P\right)$.

$$\mathbb{E}\left[\mathbf{C}\right] = \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left[\mathbf{Y}_m(P)\right] \tag{3}$$

Based on this expected confusion matrix, various performance metrics can be derived. For example, the classification accuracy can be derived from the ratio of correct predictions to the total number of predictions. Other examples of metrics include precision, recall and F1-score. With these metrics, the performance of the ANN can be judged for a particular input space along with a certain noise distribution.

The next analysis step comprises of the performance evaluation for a range of parameter sets of the noise distribution. For example, the degradation of performance for each class can be observed for increasing noise on the input space. This gives a clear sense about the classes of the classifier that start to degrade the quickest, and which classes retain a reasonable performance.

For a particular noise distribution $\mathcal{D}$, a parameter space $\mathbf{P} \in \mathbb{R}^{r \times s}$ is defined, as shown in Eq. (4). The first dimension denotes the actual parameter tuple P of the distribution, whereas the second dimension denotes the set of different parameter tuples. For example, a Gaussian distribution is characterised by a parameter tuple with two parameters: *mean* μ and *variance* σ. The second dimension then denotes different settings for μ and σ.

$$\begin{aligned} \mathbf{P} &= \{P_1, \ldots, P_r\} \\ \forall P \in \mathbf{P} : P &= (p_1, \ldots, p_s) \end{aligned} \tag{4}$$

Eqs. (1 - 3) are evaluated for all elements of vector $\mathbf{P}$, ranging from P_1 to P_r. In this way, we obtain a corresponding set $\mathbf{S}$ consisting of expected confusion matrices $\mathbb{E}[\mathbf{C}]_1$ to $\mathbb{E}[\mathbf{C}]_r$, as defined in Eq. (5), where each of the elements represents the performance of $\mathcal{F}$ when subjected to $\mathcal{D}(P_i)$.

$$\mathbf{S} = \left\{ \frac{1}{M}\sum_{m=1}^{M}\mathbb{E}\left[\mathbf{Y}_m(P_i)\right] \mid P_i \in \mathbf{P},\, 1 \leq i \leq r \right\} \tag{5}$$

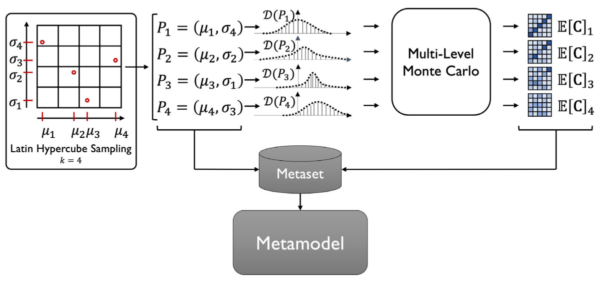
B. METAMODELLING NOISE-TO-PERFORMANCE RELATION

As the goal is to capture the relation between the noise characteristic on the input and the performance on the output, this defines the input and outputs of the metamodel. More specifically, as presented in Eq. (6), the metamodel $\mathcal{G}$ serves as a regression function that learns the relationship between the parameters of the input noise distribution $P \in \mathbf{P}$ and the expected performance $\mathbb{E}[\mathbf{C}]$ of the classifier $\mathcal{F}$.

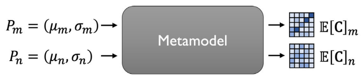
$$\mathcal{G} : (P \in \mathbf{P}) \mapsto \mathbb{E}[\mathbf{C}]_P \tag{6}$$

This immediately implies the advantage of using metamodels: the metamodel only needs to be created once, so the analysis on the underlying classifier is also carried out once. Inferring this relation now needs only one evaluation of the regression function. Without metamodels, the noise-to-performance relation could only be inferred by carrying out the noise propagation ad-hoc, which is a costly process.

To create such a metamodel, a metaset is needed on which it is fitted. Figure 2a shows the technique visually. We use a variant of Latin Hypercube Sampling (LHS) that operates within a defined range of parameters for a particular noise distribution. This sampling method provides metasamples



(a) Training process of the metamodel



(b) Evaluation process of the metamodel

FIGURE 2. The process for training and evaluating metamodels is shown in Figures (b) and (b), respectively.

where each one represents a set of parameters of a noise distribution. By using LHS, each dimension of the noise distribution parameter space is divided into k equal parts, after which a random sample is drawn from each equal part, resulting in k different samples. For example, as shown in Figure 2, LHS could operate over a range of means and standard deviations for Gaussian noise. Hence, each sample represents a Gaussian distribution with a particular mean and standard deviation.

For each metasample obtained via LHS, we carry out the Monte Carlo simulation described in Section III-A. Hence, this yields a metaset where the inputs are the parameters for this noise distribution and the outputs being the corresponding performance measures for all classes of the classifier.

Next, we fit a regression model on this metaset, which then learns the function between the parameters of the noise distribution to the neural network's performance.

Finally, when the metamodel has been trained on the generated metaset, the model has learned the relationship between the noise distribution parameters and the expected performance of the ANN. Therefore, as shown in Figure 2b, given a particular set of parameters of the noise distribution, the expected performance is predicted.

IV. EXPERIMENTAL SETUP

As our methodology consists of two major parts (i.e. the noise propagation analysis and generation of metamodels), the experiments are set up in these two categories as well. First, we compare and elaborate on the performance of image classifiers when analysed by the process mentioned in Section III-A. Second, we present the performance of the created metamodels and how the different models hold up to each other. Finally, the influence of the collected metaset on the prediction performance of the metamodels is also being investigated.

The experiments are executed on Tesla V100 graphical processing units within an NVIDIA DGX-2.

TABLE 1. For each architectural design of image classifiers, different pre-trained versions are employed.

Architecture	Versions
VGG [33]	11, 13, 16, 19
ResNet [34]	18, 34, 50
DenseNet [35]	121, 161, 169

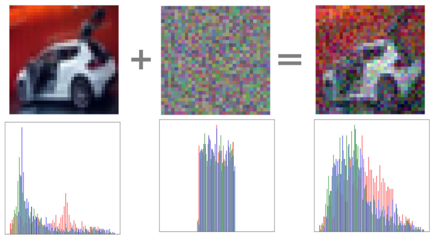


FIGURE 3. The addition of uniform noise to a sample image from CIFAR-10 is shown. Underneath are the histograms, showing the distribution of the three RGB channels.

A. DATASET & MODELS

Regarding the dataset for these experiments, we selected the CIFAR-10 dataset. As this dataset is significantly complex [30] and widely used in other research tracks [31], [32], it implies the generalisability and extensive usability of this dataset. Given the CIFAR-10 dataset, we opted for different pre-trained convolutional neural networks (CNNs) that already underwent the hyperparameter optimization process and were trained in the same conditions. As shown in Table 1, we opted for three architecturally different CNNs, each with three or four distinct versions.

As mentioned in Section II-B, different metaregressors exist that have been employed in various application fields. As the choice of metaregressor depends on the application and metasamples at hand [24], there is no straightforward choice of metaregressor for our application. Therefore, we opted to implement and compare the regression models that have been discussed in Section II-B, i.e.: (i) Random Forest, (ii) k-Nearest Neighbours, (iii) Decision Trees, (iv) Gradient Boosting, (v) Gaussian Processes, (vi) Support Vector Regression, and (vii) Ridge Regression. As the performance of these metaregressors depends on their hyperparameter configuration, we conducted a randomized hyperparameter search and implemented the most optimal parameters accordingly. These hyperparameter search spaces and best-performing values are shown in Table 2.

B. NOISE DISTRIBUTION

For this research, we make use of the frequently used Uniform Distribution [36], [37]. This distribution has two parameters: the start and end points of the range of this distribution. In our research, however, we use the distribution in a symmetric fashion; that is, it is centred around zero and the width is parametrised by only one parameter r, shown in Eq. (7).

$$\mathbf{X}' = \mathbf{X} + \mathbf{u} \sim \mathcal{D}_{\mathcal{U}}(-r, r) \tag{7}$$

TABLE 2. Hyperparameter search spaces and best-performing values of the metaregressors used in our experiments.

Metaregressor	Hyperparameters	Range	Values
Random Forest	Number of estimators	$\{2^x \mid x \in \mathbb{N}, 1 \leq x \leq 10\}$	512
	Minimum samples to split	$\{x \in \mathbb{N} \mid 2 \leq x \leq 20\}$	2
	Minimum samples per leaf	$\{x \in \mathbb{N} \mid 1 \leq x \leq 20\}$	2
	Loss criterion	MSE, MAE, Poisson, Friedman's MSE	MSE
k-Nearest Neighbours	Kernel	Uniform, Gaussian, Distance-based	Distance-based
	Distance metric	Manhattan, Euclidian, Minkowski	Euclidian
	Number of neighbours	$\{x \in \mathbb{N} \mid 1 \leq x \leq 11\}$	2
Decision Tree	Loss criterion	MSE, MAE, Poisson, Friedman's MSE	MSE
	Minimum samples to split	$\{x \in \mathbb{N} \mid 2 \leq x \leq 20\}$	5
	Minimum samples per leaf	$\{x \in \mathbb{N} \mid 1 \leq x \leq 20\}$	1
Gradient Boost	Loss criterion	Least Squares, Least Absolute Deviation, Huber, Quantile	Huber
	Number of estimators	$\{2^x \mid x \in \mathbb{N}, 1 \leq x \leq 8\}$	8
	Minimum samples to split	$\{x \in \mathbb{N} \mid 2 \leq x \leq 20\}$	2
	Minimum samples per leaf	$\{x \in \mathbb{N} \mid 1 \leq x \leq 20\}$	1
	Learning rate	$\{x \in \mathbb{R} \mid 0.01 \leq x \leq 1.0\}$	0.8727
	Max depth	$\{x \in \mathbb{N} \mid 1 \leq x \leq 32\}$	27
Gaussian Process	Kernel	Radial Basis Function, White, Matérn, Rational Quadratic, Contant, Exp-Sine-Squared, Dot-Product, Product(Dot-Product, RBF)	Rational Quadratic
	Added kernel noise	$\{x \in \mathbb{R} \mid 1e-10 \leq x \leq 1.0\}$	1.617e-6
	Number of optimizer restarts	$\{x \in \mathbb{N} \mid 0 \leq x \leq 10\}$	4
Support Vector Regression	Kernel	Radial Basis Function, Polynomial	Radial Basis Function
	Epsilon	$\{x \in \mathbb{R} \mid 1e-6 \leq x \leq 1.0\}$	0.0001317
	Regularisation parameter	$\{x \in \mathbb{N} \mid 1 \leq x \leq 100\}$	67
Ridge Regression	Regularisation parameter	$\{x \in \mathbb{R} \mid 1e-3 \leq x \leq 100\}$	0.001025

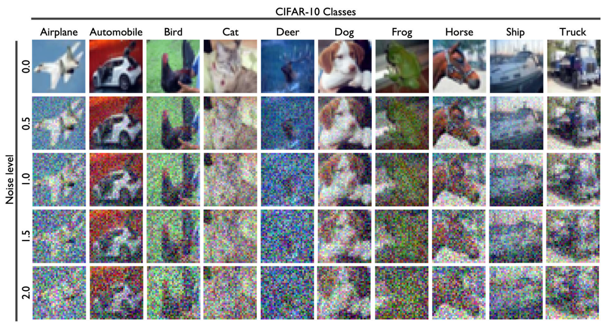


FIGURE 4. The result of adding different levels of noise distribution $\mathcal{D}_{\mathcal{U}}$ to the CIFAR-10 dataset is shown.

The value $\mathbf{u}$ in Eq. (7) represents the sample from the uniform noise distribution $\mathcal{D}_{\mathcal{U}}$. This sample $\mathbf{u}$ is a three-dimensional matrix with the same shape as the image $\mathbf{X}$ to which the noise is added. For each pixel (i.e., each tuple of RGB channels), three samples are drawn from $\mathcal{D}_{\mathcal{U}}$. As shown in Figure 3, the image of the dataset and the equally-shaped noise sample are added together, resulting in a perturbed image $\mathbf{X}'$. Figure 4 shows how different values for r affect an image sample from each class in the CIFAR-10 dataset. We chose value 2.0 as a maximum as this already heavily distorts the image; values higher than 2.0 would have a low research contribution. This is also shown experimentally in the results of Section V-A.

C. SAMPLING STRATEGY

In order to fit a metamodel on the metasamples that capture the performance-to-noise relationship, these samples need to be generated via the multi-level Monte Carlo simulation presented in Section III-A. As parameter N is a hyperparameter of the Monte Carlo simulation, this parameter is open for adjustments. According to our tests, $N = 10$ is an acceptable compromise between computational complexity and accuracy. Hence, 10 corresponding classification results of the ANN are collected, which are aggregated together into the expected value $\mathbb{E}[\mathbf{Y}]$. This process is carried out for the whole validation set of CIFAR-10, which contains 10,000 different images. This in turn adds up to 100,000 forward propagations through the ANN to represent its classification performance for one parameter setting of the noise distribution.

For the analysis process presented in the first part of the Results section below, we gathered 200 equal-distanced samples, ranging from noise values 0.0 to 2.0. Hence, each noise sample represents an increase of 0.01, which yields a granular and detailed analysis.

As the aforementioned analysis process is computationally expensive (i.e. 20,000,000 forward propagations), a more intelligent way of generating metasamples for the metamodel needs to be devised. Therefore, the methodology presented in Section III-B is designed to cover the distribution parameter space more intelligently. By employing LHS, the number of metasamples s determines the number of equal divisions k of the parameter space. For all k selected noise distribution settings with LHS, we apply the aforementioned multi-level Monte Carlo simulation. In the second part of the results, we alter the amount of metasamples s to train the metamodels on, which therefore also changes the number of equal divisions k of the parameter space. In this way, we are able to examine the relation between the number of metasamples and the performance of the different metamodels.

V. RESULTS

In this section, the results are presented. The first subsection elaborates on the analysis of noise-to-classification

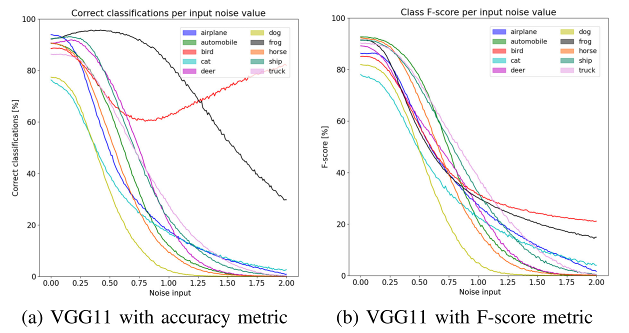

(a) VGG11 with accuracy metric (b) VGG11 with F-score metric

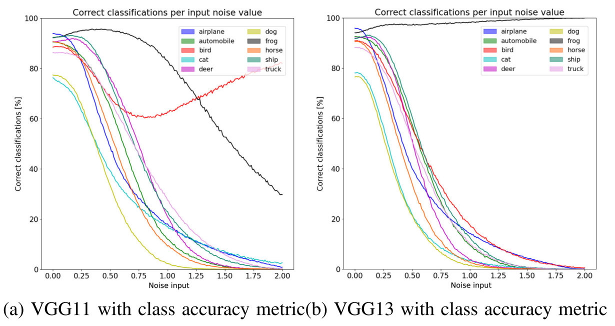
FIGURE 5. The classification performance of the VGG11 network architecture is shown. As classification accuracy alone is not a sufficient metric for judging performance, the F-score is included as a second, complimentary metric.

behaviour. In the second subsection, the results of the trained metamodels are presented.

A. ANALYSIS OF IMAGE CLASSIFIERS

The noise-to-classification behaviour analysis is carried out for all ANNs shown in Table 1. For legibility, the results of all analyses are shown on different abstraction levels. First, the results of one particular ANN implementation are thoroughly discussed. Next, the analyses of different implementations of one architecture are compared. Finally, we compare different architectures through aggregations of their different implementations.

Figure 5 shows the performance of VGG11 in terms of its classification accuracy and corresponding F-score. In Figure 5a we immediately notice that the global trend of the classification accuracy decreases when the augmented input noise increases. However, some classes deviate from this global trend. First, when no noise is present on the input, all classes but '*cat*' and '*dog*' are high in accuracy. This classification confusion between '*cat*' and '*dog*' has also been demonstrated by Yan et al. [38], where they show a higher corresponding misclassification rate of said classes. They attribute this confusion to an ambiguous boundary of classification, due to similar features in both classes. Second, the progress of classes '*bird*' and '*frog*' is remarkable: at medium and high noise levels the network seems to bias towards frogs and birds respectively. This would seem to indicate that, when a high level of noise is present on the input, these classes are still detected with a high accuracy; the class accuracy for birds has only dropped 6%, as it was 88% when no noise was present. However, only using the accuracy metric to judge the performance of an ANN is not good practice. Figure 5b shows the F-score performance metric of the network. As this metric combines the precision and recall performance values, it provides context around the observed classification accuracies. We notice that for all classes, the F-score monotonically decreases in value, which indicates a high number of misclassifications. Hence, the accuracy for class '*bird*' is high due to the fact that all images in the dataset are being detected as a bird. This means the image classifier is biased towards the '*bird*' class when the input noise level


(a) VGG11 with class accuracy metric(b) VGG13 with class accuracy metric

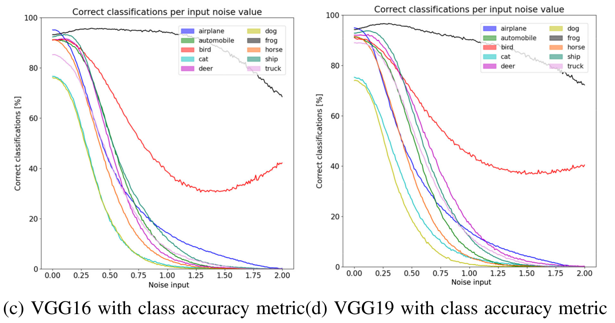

(c) VGG16 with class accuracy metric(d) VGG19 with class accuracy metric

FIGURE 6. The classification accuracies for the four different versions of the VGG architecture are shown.

is high and no features can be detected. This could be due to the hyperparameters regarding architecture, initialisation or training procedure, which results in the current local minimum of the neural loss function [39]. Interesting to see is that the F-score for '*cat*' and '*dog*' are lower than the rest, similar to the graphs from Figure 5a. The progress of the other classes shows that the number of misclassifications is high.

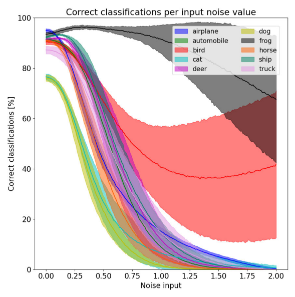
As mentioned in the introduction of this paper, we aim to use different classifiers that all solve the same classification problem. As VGG11, shown in Figure 5a, is only one ML model in a range of models, we aim to compare different versions of the same architecture, in order to examine differences between them. Figure 6 shows a collection of versions of the VGG architecture. Figures 6a and 6b show VGG11 and VGG13, while VGG16 and VGG19 are shown in Figures 6c and 6d, respectively. From the start, we notice that the four figures are quite similar. When no noise is present on the input, the classes '*cat*' and '*dog*' share the same characteristic. They are consistently lower than the other classes from the dataset, albeit small differences are still present. Next, small differences in accuracy are noticeable when a relatively low noise level is present. For example, the class '*horse*' is more stable regarding uniform noise in VGG11 than in VGG13. At a noise level of 0.4, the accuracy is 67% in VGG11, whereas VGG13 reaches a performance of only 47%. Both VGG16 and VGG19 achieve an accuracy of approximately 51%. At first sight, the lower complexity of VGG11 suggests a higher robustness to uniform noise than the higher-complexity implementations of VGG. Hence, these differences already show that it is feasible to recommend suitable architectures for appropriate classes of interest

TABLE 3. The relation between noise and class accuracies in percentages is shown. Each cell shows the performance of four VGG networks (from left to right, top to bottom: VGG11, VGG13, VGG16, VGG19).

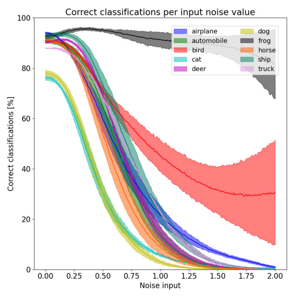
	Airplane		Auto		Bird		Cat		Deer		Dog		Frog		Horse		Ship		Truck	
0.0	93	96	90	93	89	91	76	78	91	92	77	77	92	94	91	91	92	92	86	88
	95	95	91	91	91	91	77	75	91	92	76	74	93	94	91	92	92	93	85	89
0.1	93	93	89	92	88	90	73	75	91	92	76	73	93	95	89	88	93	93	86	87
	92	92	91	90	91	90	74	72	92	92	73	70	94	93	88	89	93	93	83	89
0.2	87	82	87	88	85	85	68	65	92	90	70	61	95	97	86	79	93	92	85	82
	83	81	88	86	89	87	64	63	89	90	63	60	95	96	81	81	91	92	78	85
0.3	75	68	82	82	80	76	57	47	89	82	59	44	95	97	79	64	91	86	81	75
	69	66	81	79	84	82	48	51	81	84	47	42	96	97	67	67	82	87	68	78
0.4	62	51	75	71	74	67	48	33	83	68	46	31	96	97	67	48	85	74	75	63
	54	52	68	67	79	77	32	38	66	76	31	28	96	96	51	52	68	77	55	68
0.5	48	40	64	57	69	56	39	22	76	51	34	21	95	97	54	34	74	61	68	51
	42	41	54	53	72	70	19	27	49	66	20	19	95	95	37	38	53	64	43	56
0.6	38	33	51	44	64	46	32	15	67	35	22	14	95	97	42	22	63	46	59	39
	33	33	40	39	64	64	11	18	35	55	12	12	95	94	25	26	40	49	32	44
0.7	31	26	39	32	61	38	27	09	55	22	15	08	93	98	31	14	51	34	50	30
	27	27	27	26	55	57	07	12	22	42	07	07	95	93	15	17	31	36	24	33
0.8	26	22	27	22	60	30	23	07	43	13	09	05	91	98	21	08	40	25	42	22
	22	21	18	17	48	52	04	09	14	33	04	04	95	92	09	11	23	26	17	24
0.9	21	17	18	15	62	23	20	05	31	07	05	03	88	98	14	05	30	18	35	15
	18	17	12	11	44	48	03	05	09	24	02	02	94	91	06	07	17	18	12	17

based on class-specific accuracy for a given measure of the input noise. When the noise level continues to increase, more noticeable differences start to appear. As mentioned in the previous paragraph, the characteristics of classes '*bird*' and '*frog*' are remarkable when in the presence of high-noise levels. As shown in Figure 5a and 6a, the classification accuracies of both classes start to cross at a noise level of 1.27. However, for VGG13 shown in Figure 6b, this is not the case. The accuracy for '*frog*' keeps increasing, while the class '*bird*' is monotonically decreasing along with the other classes in the dataset. It is confirmed that the F-score for class '*frog*' in VGG13 is also consistently lower than in VGG11. The reason for this behaviour is unclear and could be due to several reasons. For instance, the random weight initialisation or optimization during training could lead to another local minimum of the loss function. This, in turn, leads to different outcomes the network defaults to when a high level of noise is present.

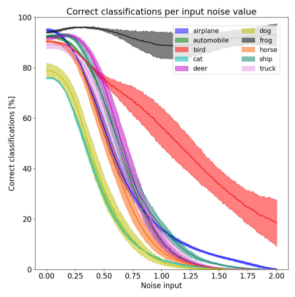
In Table 3, we present a more in-depth numeric representation of the results for the four versions of the VGG architecture. First, we look at the performance of the ANN models regarding increasing noise. When no noise is present on the input, VGG13 is mainly better than the other classifiers for seven out of 10 classes. In the other three classes (i.e. '*horse*', '*ship*' and '*truck*'), VGG19 is the best image classifier. However, these differences are only marginal and negligible. When the input noise is increased to a low level ($0.1 \leq r \leq 0.3$), there is no single classifier that performs best at all classes. Hence, depending on the class of interest, another image classifier would be recommended. However, when the noise further increases ($r \geq 0.5$), we notice VGG11 being better than the other classifiers for most of the classes, with the exceptions of the class '*frog*'. This is an interesting insight, as the complexity of the classifier is smaller than the other three networks. This could be attributed to the problem of overfitting. This is a common issue in training deep ANNs, for which several works have proposed methods for preventing overfitting in such complex networks [40], [41].



(a) Avg. accuracy of VGG11, 13, 16 and 19



(b) Avg. accuracy of ResNet 18, 34 and 50



(c) Avg. accuracy of DenseNet 121, 161 and 169

FIGURE 7. The average class accuracies with standard deviations for each set of architectures are shown.

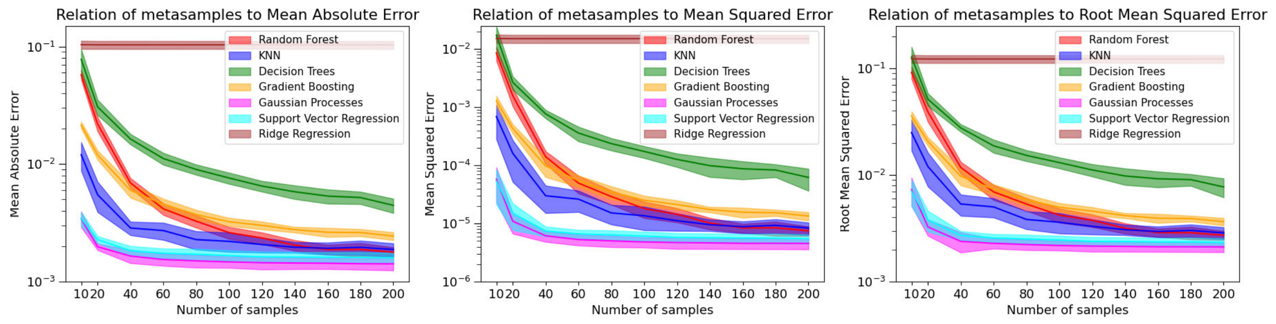
As the previous results only compared different implementations of VGG, said results only give insights into the VGG architecture itself. Therefore, in this final paragraph, we compare different architectures relative to VGG. Since the different implementations of a particular architecture have mutual similarities, for legibility these implementations have been aggregated into one single graph showing mean and standard deviation. Hence, the four different implementations shown in Figure 6 are aggregated in Figure 7a. Similarly, Figure 7b shows the aggregation of ResNet with ResNet18, 34 and 50. Figure 7c shows the DenseNet architecture with DenseNet121, 161 and 169 aggregated. We notice that the distribution over the different versions is more concentrated for both ResNet and DenseNet than it is for VGG. However, except for the difference in distributions, the overall characteristics are similar. Again, classes '*cat*' and '*dog*' are consistently lower than other classes, and classes '*bird*' and '*frog*' have similar outlier characteristics. Minor differences are present in classification accuracy of other classes. For example, with VGG and ResNet, class '*ship*' is performing better than '*deer*' for low noise levels, but the opposite is true for DenseNet. This figure shows that the difference in architecture does not result in a significant difference in performance. On the one hand, we notice that the robustness to uniform noise behaves in a very similar way, but small differences are present. On the other hand, we notice that the characteristics of these behaviours are very similar, too. Classes '*bird*' and '*frog*' are outliers and the networks bias towards '*frog*', while classes '*cat*' and '*dog*' are consistently lower than the others. This sheds light on the importance of the dataset on which the networks were trained towards each network's respective performance to uniform noise. This importance of the CIFAR-10 dataset has also been shown in other researches: different CNNs experience similar performance drops in case of benign data distribution shifts [42] or learned similar non-sensical statistical patterns of the dataset [43].

B. EVALUATION OF METAMODELS

To assess the predictive performance of the regressive metamodel, we assess the errors between the actual values from the multi-level Monte Carlo analysis, mentioned in Section III-A, and the predicted values by the metamodels. These errors are presented in the graphs in Figure 8. Each graph represents an average error with standard deviation over four metamodels, which have been trained on metasets of VGG11 through VGG19. As the ResNet and DenseNet architectures were determined to have a similar noise-to-classification performance relationship, and therefore have similar performing metamodels, we omitted these architectures for legibility. The X-axis denotes the number of metasamples in each set; these samples are collected with LHS in the way as described in Sections III-B and IV-C. This results in a noise distribution parameter space divided into as many parts as samples taken, with a random sample drawn in each part. After training, each metamodel has been evaluated with 8 different validation metasets, where each set consists of 40 metasamples, sampled with LHS in the same way as described before. For each metaset, the Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) are calculated. For all validation sets applied on the 4 metamodels, these error measurements are aggregated into a corresponding average and standard deviation. These aggregated MAEs, MSEs and RMSEs are shown in Figures 8a, 8b and 8c, respectively. We use these three different metrics to show the errors with three different nuances. The MAE shows the distribution over the average error in the metamodel predictions, whereas the MSE indicates the presence of outliers as it gives more weight to outlier values. The RMSE gives an indication to the distribution of the standard deviations over the prediction errors.

First, the MAE graphs are presented in Figure 8a. Note that the vertical axis denotes the differences in classification accuracy percentages. Hence, a value of 10^{-2} denotes an absolute difference of one percent between the predicted classification accuracy and the actual accuracy coming from the noise propagation analysis. We immediately notice the inaccuracy of the RR metamodel. As this metaregressor is a linear approximator, we show that our problem is rather non-linear. Next, we notice the inaccuracies of the other metamodels when using a low amount of samples. That is, when using 10 LHS-generated samples, each metaregressor performs worse than when using a larger amount of samples. Though, in absolute values, the MAE for the GP, SVR and k-NN-based metaregressors is still low. The GP metaregressor has an average MAE of approximately 0.00342 or 0.342%, while SVR and k-NN have MAEs of 0.345% and 1.204% respectively. Next to that, it can be seen that the MAE of the tree-based metaregressors (i.e. RF & DT) are considerably higher than the other metaregressors, aside from RR. This could be due to different factors related to the low size and dimensionality of the metaset, such as overfitting of the regressors or inconsistent trees due the limited subsampling of data or features [44]. When increasing the amount of samples to 40 or more, we notice that the performance of GP and SVR converges to an average MAE of approximately 0.14% and 0.16%, respectively. As the size of the metaset increases, the performance of k-NN slightly improves, but the performance of RF increases rather drastically. RF performs even better than k-NN when employing a metaset size of 160 samples or more. This trend is in line with the research of Noi and Kappas [45]; they show that the performance of SVM is superior to RF and k-NN when using a low amount of samples, but that the performance of the last two improves as the amount of samples increases. It should be noted that all metaregressors aside from RR and DT perform similarly when using a large amount of samples; the MAE ranges between 0.24% and 0.14% for GB and GP as worst and best regressor, respectively.

Figure 8b presents the MSE graphs. Again, we notice that RR is the worst predictor due to the non-linear characteristic of our use case. For all other metamodels, we notice that the MSE is increased when using a limited metaset (i.e. 10 to 40 samples). This shows that all metamodels reveal inconsistent prediction accuracies when fitted on a limited metaset. However, when increasing the metaset size to 40 samples or more, all MSEs start to converge. Similarly to the previous figure, GP and SVR show the best performance with the least outlier consistencies. RF, k-NN and GB show slightly worse MSE values, although the difference is small. The DT metamodel shows worse MSE values, implying that the prediction accuracies are more inconsistent than the other metamodels. We should note that the MAE for the classification accuracy is



(a) Mean Absolute Error (MAE) (b) Mean Squared Error (MSE) (c) Root Mean Squared Error (RMSE)

FIGURE 8. Relation of number of metasamples to (a) MAE, (b) MSE, and (c) RMSE. The error is shown in class accuracy percentages. Every graph shows the average and standard deviation over the four versions of the VGG architecture.

less than one percent, so although outliers are clearly present, they are still within limited ranges.

Finally, the RMSE measurements are shown in the graphs of Figure 8c. Similar to the previous figures, the error range is increased when using a low amount of samples, while it starts to converge when using 40 samples or more. The same relation between the metamodels is present as well: RR shows the highest RMSE due to its linear approach, the DT model is second-to-last, and the other metamodels are showing small differences in RMSE, bounded between GB and GP as highest and lowest RMSE respectively.

Finally, Figure 9 presents the class accuracy predictions of the different metamodels when trained on 10 samples of the VGG11 network. In this way, it shows how the increased errors of Figure 8 manifest in the actual predictions. As the other versions of VGG and network architectures have similar noise-to-performance relationships, and therefore similar performing metamodels, these other network configurations are omitted for legibility. It is clear that the GP and SVR models already show accurate predictions when trained on 10 metasamples: the global trend of the individual classification accuracies is clearly shown. However, small aberrations are still present, such as the classification accuracy of most classes at very low noise inputs. Both GP and SVR show that cat has a higher classification accuracy than dogs, but this is the other way around in the actual analysis of VGG11. Additionally, the predicted class accuracies tend to have a positive slope towards a maximum, such as the classes '*deer*' and '*truck*'. However, in the actual analysis, this slope is flatter, if not negative for some classes. This shows that GP and SVR can accurately predict classification performance with only 10 metasamples, but that small errors are still present. Regarding the other metamodels, it is shown that 10 metasamples do not contain enough information yet for accurate predictions. The interpolation process of k-NN is clearly struggling with the limited information, while the other metamodels are showing even larger errors with their inaccurate step-wise decreases. The RR model is clearly unable to model our non-linear application.

In our use case, it is clear that GP and SVR are sample-efficient metamodels. Using 10 metasamples already results in accurate predictions, but 20 or 40 metasamples are required for minimizing the small aberrations, yielding high-accuracy predictions. k-NN and RF do not achieve the same sample-efficiency, but they start to achieve high prediction accuracies when trained on 40 samples or more. The other metamodels require more metasamples for high prediction accuracies, making them less sample-efficient.

VI. DISCUSSION

When looking at the analysis results of the image classifiers and their performances on CIFAR-10 with uniform noise, an interesting insight arises. Their respective performances for the same classes are very similar: (i) the networks appear to bias towards '*bird*' and '*frog*' when the noise level increases, and (ii) classes '*cat*' and '*dog*' all have consistently lower accuracies than the other classes, even when no noise is added. This implies that the observed differences are more likely due to the characteristics of the dataset on which the classifiers are trained, than the classifier architectures themselves. It is indeed shown that '*cat*' and '*dog*' have a more difficult feature set [38], while other classes could be trained on rather non-sensical statistical patterns of the dataset [43]. This shows that our noise propagation analysis is in synergy with prior findings regarding the effect of the CIFAR-10 dataset on image classifiers. It can be noted, however, that slight differences are present in the different classifier architectures. For example, as presented in Figure 6 and Table 3, the four versions of VGG-based architectures have small differences. VGG11 appears to be consistently better than the other versions for a certain set of classes, especially for the class '*horse*'; the difference between VGG11 and the other variants increases considerably when the noise increases too. Dodge and Karam [46] also have a similar finding: noise on the input is being amplified through the convolutional layers. Less layers therefore results in less noise amplification. In the case of the DenseNet-based architectures, it is the only set of architectures where the class '*deer*' has a better performance than '*ship*', which is reversed for VGG- and ResNet-based architectures. Hence, as each classifier has its own advantages and disadvantages, specific classifiers could be recommended based on the task at hand.

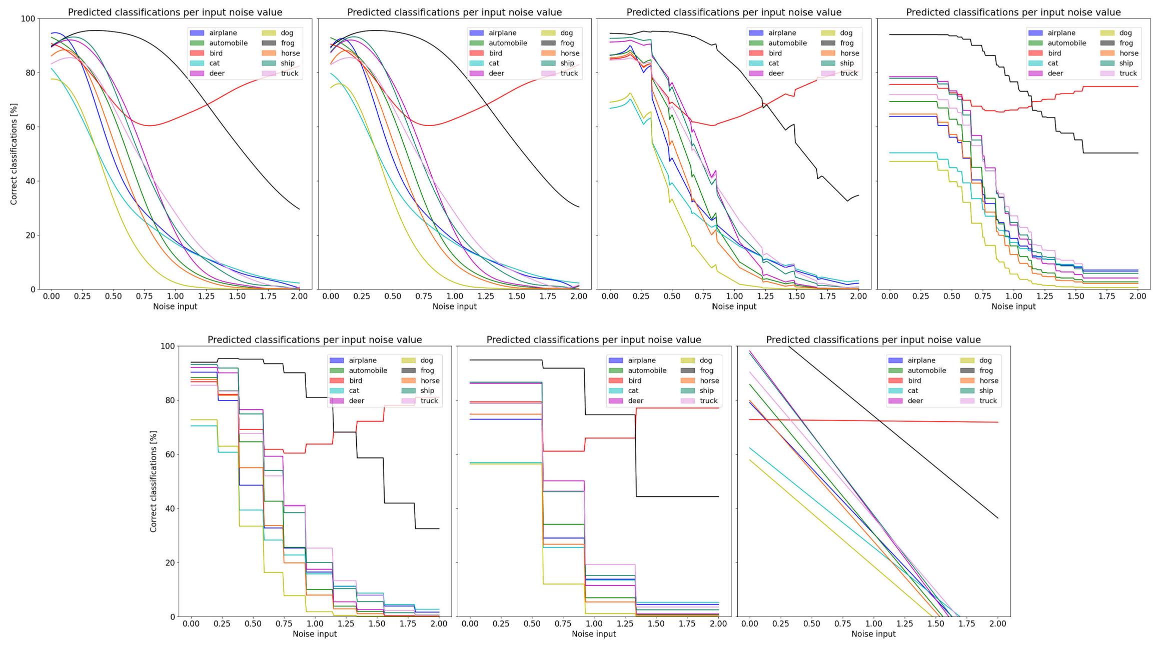


FIGURE 9. **The predicted class accuracies of the metamodels are shown when trained on 10 metasamples. Metamodels are shown from left to right, top to bottom: Gaussian Processes, Support Vector Regression, k-Nearest Neighbours, Random Forest, Gradient Boosting, Decision Tree, and Ridge Regression.**

As stated before, metamodels noticeably facilitate the recommendation of classifiers [10]. In our research, we focused on regressive metamodels predicting the accuracy of widely used and researched classifiers, instead of directly recommending one or more networks. We conducted experiments over the number of samples needed for an accurate regressive prediction and noticed that, with a limited number of samples (i.e. 10 - 40 samples), a wider distribution of errors occurred when predicting the classification accuracy. Although this is true for all tested metamodels, the GP and SVR regressors still performed very accurate, achieving prediction errors of only 0.342% and 0.345% respectively. This could be attributed to the fact that both are kernel-based metaregressors, which makes them accurate function approximators with only a low amount of samples [47], [48]. However, more research is needed to validate this statement. The other metamodels suffer from interpolation aberrations due to the lack of metasamples, resulting in the distribution of the prediction errors being higher and wider. The worst performing metamodel is the RR model, which is unable to model our classification accuracy problem in a linear fashion. When employing a medium amount of metasamples (i.e. 40 - 100 samples), all metamodels start to converge towards their best prediction accuracy. GP and SVR reach convergence at 40 samples, at which point most aberrations are minimized. k-NN, GB and RF reach convergence at around 100 samples. When using 100 metasamples or more, their MAE values are very similar: their MAEs range between 0.245% and 0.142% for GB and GP respectively. Only the DT and RR metamodels do not reach prediction accuracy convergence before 200 metasamples. RR fails to model our non-linear problem, while DT only achieves a prediction accuracy that is similar to other metamodels that were trained on 60 metasamples or less. This shows the sample-efficiency of the other, more accurate metamodels, with GP and SVR as best performing metaregressors in our use case.

VII. CONCLUSION

In the field of algorithm recommendation, the most suitable algorithm is selected based on a set of features of interest. In this work, we present a twofold methodology for predicting the performance of classifiers based on the characteristics of the input noise distribution. This, in turn, could be used for algorithm recommendation systems. First, we propose a multi-level Monte Carlo simulation on an image classifier, which yields an in-depth analysis of the classifier performance. Afterwards, we make use of a combination of LHS and the aforementioned multi-level Monte Carlo simulation to generate metasamples for the metamodel to fit on. Carrying out the noise-to-performance analyses on pre-trained state-of-the-art image classifiers, we notice that they have small differences between them, while showing largely similar behaviour.

Regarding the metamodel fitting, we studied the relation between the amount of metasamples and the resulting predictive performance of classification accuracies. A low to medium amount of metasamples in combination with the use of sample-efficient metamodels such as GP or SVR is recommended to accurately predict classification accuracies. Increasing the amount of metasamples would only result in a marginally better prediction performance as nearly all metamodels converge towards similar performance with low absolute errors.

By using our metamodelling strategy, algorithm recommendation systems can efficiently select the most suitable algorithm given the parameters of a noise distribution on the input. Without using these metamodels, this would result in a very computational complex analysis each time a performance prediction needs to be made.

VIII. FUTURE WORK

To improve this research, several tracks of future work follow. First, a more in-depth analysis of per-class performance could be carried out. In this way, more detailed insights can be gathered on class accuracy fluctuations and biases. Secondly, this methodology could be carried out on a more diverse set of noise distributions. This could include a synthetic distribution with a larger amount of parameters, as well as real-life noise distributions, such as rain interference, overexposure or blur. Finally, our method could be validated on larger and more complex image datasets, such as CIFAR-100 or ImageNet. It could also be of interest to extend this methodology towards non-image-based tasks, such as object recognition algorithms on point clouds [49].

REFERENCES

[1] D. Kim, S.-H. Kim, T. Kim, B. B. Kang, M. Lee, W. Park, S. Ku, D. Kim, J. Kwon, H. Lee, J. Bae, Y.-L. Park, K.-J. Cho, and S. Jo, "Review of machine learning methods in soft robotics," *PLoS ONE*, vol. 16, no. 2, Feb. 2021, Art. no. e0246102.

[2] M. Rabe, S. Milz, and P. Mader, "Development methodologies for safety critical machine learning applications in the automotive domain: A survey," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2021, pp. 129–141.

[3] Z. Yuan, J. Liu, Q. Zhang, Y. Liu, Y. Yuan, and Z. Li, "A practical estimation method of inland ship speed under complex and changeful navigation environment," *IEEE Access*, vol. 9, pp. 15643–15658, 2021.

[4] S. M. Reza, S. Choudhury, J. K. Dash, and D. S. Roy, "An AI-based real-time roadway-environment perception for autonomous driving," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE-Taiwan)*, Sep. 2020, pp. 1–2.

[5] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: A systematic mapping," *Empirical Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, Nov. 2020.

[6] L. Deng, B. Yang, Z. Kang, S. Yang, and S. Wu, "A noisy label and negative sample robust loss function for DNN-based distant supervised relation extraction," *Neural Netw.*, vol. 139, pp. 358–370, Jul. 2021.

[7] D. Berend, X. Xie, L. Ma, L. Zhou, Y. Liu, C. Xu, and J. Zhao, "Cats are not fish: Deep learning testing calls for out-of-distribution awareness," in *Proc. 35th IEEE/ACM Int. Conf. Automated Softw. Eng.*, Dec. 2020, pp. 1041–1052.

[8] G. J. Aguiar, E. José Santana, S. M. Mastelini, R. G. Mantovani, and S. Barbon, "Towards meta-learning for multi-target regression problems," in *Proc. 8th Brazilian Conf. Intell. Syst. (BRACIS)*, Oct. 2019, pp. 377–382.

[9] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural Comput.*, vol. 8, no. 7, pp. 1341–1390, Oct. 1996.

[10] L. P. F. Garcia, A. C. P. L. F. D. Carvalho, and A. C. Lorena, "Noise detection in the meta-learning level," *Neurocomputing*, vol. 176, pp. 14–25, Feb. 2016.

[11] Y. Nemcovsky, E. Zheltonozhskii, C. Baskin, B. Chmiel, A. M. Bronstein, and A. Mendelson, "Adversarial robustness via noise injection in smoothed models," *Int. J. Speech Technol.*, vol. 53, no. 8, pp. 9483–9498, Aug. 2022.

[12] A. Liu, X. Liu, H. Yu, C. Zhang, Q. Liu, and D. Tao, "Training robust deep neural networks via adversarial noise propagation," *IEEE Trans. Image Process.*, vol. 30, pp. 5769–5781, 2021.

[13] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *Proc. 7th Int. Conf. Learn. Represent. (ICLR)*, New Orleans, LA, USA, 2019, pp. 1–16.

[14] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," in *Proc. 6th Int. Conf. Learn. Represent. (ICLR)*, Vancouver, BC, Canada, 2018, pp. 1–27.

[15] A. H. Abdelaziz, S. Watanabe, J. R. Hershey, E. Vincent, and D. Kolossa, "Uncertainty propagation through deep neural networks," in *Proc. Interspeech*, Dresden, Germany, Sep. 2015, pp. 1–9.

[16] K. Nathwani, E. Vincent, and I. Illina, "DNN uncertainty propagation using GMM-derived uncertainty features for noise robust ASR," *IEEE Signal Process. Lett.*, vol. 25, no. 3, pp. 338–342, Mar. 2018.

[17] M. Reif, F. Shafait, M. Goldstein, T. Breuel, and A. Dengel, "Automatic classifier selection for non-experts," *Pattern Anal. Appl.*, vol. 17, no. 1, pp. 83–96, Feb. 2014.

[18] M. A. Muñoz, L. Villanova, D. Baatar, and K. Smith-Miles, "Instance spaces for machine learning classification," *Mach. Learn.*, vol. 107, no. 1, pp. 109–147, Jan. 2018.

[19] M. Kotlar, M. Punt, Z. Radivojevic, M. Cvetanovic, and V. Milutinovic, "Novel meta-features for automated machine learning model selection in anomaly detection," *IEEE Access*, vol. 9, pp. 89675–89687, 2021.

[20] K. Eggensperger, F. Hutter, H. Hoos, and K. Leyton-Brown, "Efficient benchmarking of hyperparameter optimizers via surrogates," *Proc. AAAI Conf. Artif. Intell.*, vol. 29, no. 1, Feb. 2015, pp. 1–7.

[21] K. Eggensperger, M. Lindauer, H. H. Hoos, F. Hutter, and K. Leyton-Brown, "Efficient benchmarking of algorithm configurators via model-based surrogates," *Mach. Learn.*, vol. 107, no. 1, pp. 15–41, Jan. 2018.

[22] A. C. Lorena, A. I. Maciel, P. B. C. de Miranda, I. G. Costa, and R. B. C. Prudencio, "Data complexity meta-features for regression problems," *Mach. Learn.*, vol. 107, no. 1, pp. 209–246, Jan. 2018.

[23] D. Ler, H. Teng, Y. He, and R. Gidijala, "Algorithm selection for classification problems via cluster-based meta-features," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 4952–4960.

[24] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated Machine Learning: Methods, Systems, Challenges* (The Springer Series on Challenges in Machine Learning). Cham, Switzerland: Springer, 2019, pp. 35–61.

[25] T. Horváth, R. G. Mantovani, and A. C. P. L. F. de Carvalho, "Hyperparameter initialization of classification algorithms using dynamic time warping: A perspective on PCA meta-features," *Appl. Soft Comput.*, vol. 134, Feb. 2023, Art. no. 109969.

[26] A. Roy, R. M. O. Cruz, R. Sabourin, and G. D. C. Cavalcanti, "Meta-regression based pool size prediction scheme for dynamic selection of classifiers," in *Proc. 23rd Int. Conf. Pattern Recognit. (ICPR)*, Cancun, Dec. 2016, pp. 216–221.

[27] Y. Jin, J. Li, W. Du, and F. Qian, "Adaptive sampling for surrogate modelling with artificial neural network and its application in an industrial cracking furnace," *Can. J. Chem. Eng.*, vol. 94, no. 2, pp. 262–272, Feb. 2016.

[28] N. Metta, R. Ramachandran, and M. Ierapetritou, "A novel adaptive sampling based methodology for feasible region identification of compute intensive models using artificial neural network," *AIChE J.*, vol. 67, no. 2, pp. 1–17, Feb. 2021.

[29] A. C. Duchanoy, H. Calvo, and A. M. Moreno-Armendáriz, "ASAMS: An adaptive sequential sampling and automatic model selection for artificial intelligence surrogate modeling," *Sensors*, vol. 20, no. 18, pp. 1–26, 2020.

[30] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, ON, Canada, Tech. Rep. TR-2009, 2009. [Online]. Available: https://www.scirp.org/(S(lz5mqp453ed%20snp55rrgjct55))/reference/referencespapers.aspx?referenceid=2208762

[31] K. Zhang, Y. Guo, X. Wang, D. Chang, Z. Zhao, Z. Ma, and T. X. Han, "Competing ratio loss for discriminative multi-class image classification," *Neurocomputing*, vol. 464, pp. 473–484, Nov. 2021.

[32] H. Kim, C. Wang, H. Byun, W. Hu, S. Kim, Q. Jiao, and T. H. Lee, "Variable three-term conjugate gradient method for training artificial neural networks," *Neural Netw.*, vol. 159, pp. 125–136, Feb. 2023.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, San Diego, CA, USA, 2015, pp. 1–12.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[35] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2261–2269.

[36] B. R. Barmish and C. M. Lagoa, "The uniform distribution: A rigorous justification for its use in robustness analysis," *Math. Control, Signals, Syst.*, vol. 10, no. 3, pp. 203–222, Sep. 1997.

[37] D. Rajan, "Probability, random variables, and stochastic processes," in *Mathematical Foundations for Signal Processing, Communications, and Networking*. Boca Raton, FL, USA: CRC Press, 2017.

[38] L. Yan, B. Zhong, and K.-K. Ma, "Confusion-aware convolutional neural network for image classification," in *Proc. ICONIP*, Sydney, NSW, Australia, in Lecture Notes in Computer Science: Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 11953. Cham, Switzerland: Springer, 2019, pp. 151–161.

[39] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Advances Neural Information Processing Systems*, vol. 31, S. Bengio, H. M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett, Eds. Montreal, QC, Canada: Curran Associates, Dec. 2018, pp. 6391–6401.

[40] L. Qian, L. Hu, L. Zhao, T. Wang, and R. Jiang, "Sequence-dropout block for reducing overfitting problem in image classification," *IEEE Access*, vol. 8, pp. 62830–62840, 2020.

[41] A. Mahdavi-Hormat, M. B. Menhaj, and A. Shakarami, "An effective reinforcement learning method for preventing the overfitting of convolutional neural networks," *Adv. Comput. Intell.*, vol. 2, no. 5, p. 34, Oct. 2022.

[42] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, "Do CIFAR-10 classifiers generalize to CIFAR-10?" 2018, *arXiv:1806.00451*.

[43] B. Carter, S. Jain, J. W. Mueller, and D. Gifford, "Overinterpretation reveals image classification model pathologies," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 15395–15407.

[44] C. Tang, D. Garreau, and U. V. Luxburg, "When do random forests fail?" in *Advances in Neural Information Processing Systems*, vol. 31, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Montreal, QC, Canada: Curran Associates, Dec. 2018, pp. 2987–2997.

[45] P. Thanh Noi and M. Kappas, "Comparison of random forest, K-nearest neighbor, and support vector machine classifiers for land cover classification using Sentinel-2 imagery," *Sensors*, vol. 18, no. 2, p. 18, Dec. 2017.

[46] S. Dodge and L. Karam, "Understanding how image quality affects deep neural networks," in *Proc. 8th Int. Conf. Quality Multimedia Exper. (QoMEX)*, Jun. 2016, pp. 1–6.

[47] E. Schulz, M. Speekenbrink, and A. Krause, "A tutorial on Gaussian process regression: Modelling, exploring, and exploiting functions," *J. Math. Psychol.*, vol. 85, pp. 1–16, Aug. 2018.

[48] B. Schölkopf and A. J. Smola, *Learning With Kernels: Support Vector Machines, Regularization, Optimization, and Beyond* (Adaptive Computation and Machine Learning Series). Cambridge, U.K.: MIT Press, 2002.

[49] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12689–12697.

JENS DE HOOG received the bachelor's and master's degrees in electronics-ICT from the Faculty of Applied Engineering, University of Antwerp, in 2016 and 2017, respectively. He is currently pursuing the Ph.D. degree with IDLab, a research group of the University of Antwerp and imec. His research interests include data quality in machine learning applications, generative models in computer vision and autonomous driving and shipping.

ALI ANWAR (Member, IEEE) received the Ph.D. degree in control science and engineering from the Harbin Institute of Technology, Harbin, China, in 2019. Since 2020, he has been a Principle Research Fellow with IDLab, an imec research group with the University of Antwerp, where he currently leads a team on context aware control systems. His research interests include autonomous vessel navigation, safe reinforcement learning, cooperative perception, and generative modeling in computer vision. He serves in the IEEE Industrial Electronics and Systems, Man and Cybernetics Society, where he is the part of the technical committees on motion control, and control, robotics and mechatronics. He serves regularly as a Reviewer in journals including IEEE Transactions on Industrial Electronics, IEEE Transactions on Industrial Informatics, IEEE/ASME Transactions on Mechatronics, and IEEE Access.

PHILIPPE REITER received the B.Eng. degree in computer engineering with specialization in biotechnology from McGill University, Montreal, Canada, in 2006, and the M.Sc. degree in machine learning and deep learning from the University of Strathclyde, Glasgow, U.K., in 2020. He has over a decade of industry experience as a Technical Lead in semiconductor design verification and business communications with Advanced Micro Devices, Inc. (AMD), Toronto, Canada. His primary research interests include biologically-inspired and very low-power AI systems.

SIEGFRIED MERCELIS received the master's degree in music production and engineering (electronics and ICT), and the Ph.D. degree in applied engineering from the University of Antwerp, in December 2016. From 2012 to 2016, he was employed with Van den Berghe Research and Development under a Baekeland Ph.D. mandate on the subject of optimizing and parallelizing real time media applications. He is currently an Assistant Professor with the University of Antwerp, where he is also an Assistant Professor and a Program Manager of the AI Applications Team. His team of more than 30 researchers is committed to bridging the gap between academic AI research and industry in domains, such as chemical process control, autonomous shipping, smart buildings, logistics, and mobility. He received the VIK Award for his master's thesis on parallel data structures.

PETER HELLINCKX received the master's degree in computer science and the Ph.D. degree in science from the University of Antwerp, in 2002 and 2008, respectively. He is currently a Professor with the University of Antwerp. He is also the Head of the Department of Electronics-ICT. He supervises more than 20 Ph.D. students in the field of distributed artificial intelligence. He is also teaching third year bachelor's courses advanced programming techniques and artificial intelligence, and the master's courses distributed AI and computer graphics. He is the Co-Founder of the spin-offs Hysopt, Hi10, and Digitrans. His research interests include distributed artificial intelligence for the IoT and cyber physical systems with as main application domains: autonomous driving/shipping, logistics, mobility, Industry 4.0, and smart cities. In this field, he is a reviewer in many scientific project evaluation commissions, both on a national and an international level.