DEEPNAQQAL: Human-Aligned Automated Validation of Test Inputs for Deep Learning

Maryam Maryam
University of Udine
Udine, Italy
maryam@spes.uniud.it

Matteo Biagiola
Università della Svizzera italiana
Lugano, Switzerland
University of St. Gallen
St. Gallen, Switzerland
matteo.biagiola@{usi,unisg}.ch

Paolo Tonella
Università della Svizzera italiana
Lugano, Switzerland
paolo.tonella@usi.ch

Vincenzo Riccio
University of Udine
Udine, Italy
vincenzo.riccio@uniud.it

***Abstract*—Test input generators (TIGs) are widely used to assess the robustness of Deep Learning (DL) image classifiers, yet they often produce invalid inputs that fall outside the semantic domain of the task, misleading quality assessment. While several automated validators have been proposed, there is a critical mismatch between automated and human validation criteria and, thus, automated validators are merely a proxy of domain validity, as perceived by human testers.**

We introduce DEEPNAQQAL, a supervised test input validator that learns validity directly from human-annotated labels using transfer learning on deep vision models. Our empirical study on automated validation of misclassification-inducing inputs compares DEEPNAQQAL against six state-of-the-art validators across three image classification tasks and multiple TIG families, using independent human assessment as ground truth. Our results show that DEEPNAQQAL consistently achieves the highest agreement with human judgments, while generalizing to unseen TIGs and remaining effective with substantially reduced labeled data.

***Index Terms*—Test Validity, Generative AI, Deep Learning**

I. INTRODUCTION

DL has reshaped image processing, with modern image classifiers often outperforming traditional vision techniques and even human experts in accuracy and efficiency [1]. Ensuring the quality of DL image classifiers is crucial, since they are increasingly deployed in safety-critical domains where failures can have severe consequences [2].

Beyond the accuracy measured during development, assessing a model's ability to generalize to unseen inputs is particularly challenging. Training datasets inevitably provide only a partial representation of the real-world conditions that can be encountered after deployment [3], [4]. Thus, engineers gather test images that accurately reflect real-world operating conditions and possibly trigger *misclassifications*, i.e., unexpected behaviors where predicted labels deviate from the expected ones. To this aim, software testing research has proposed test input generators (TIGs), which automatically synthesize inputs images to assess the quality of DL classifiers [5]–[7].

A key challenge in this setting is the absence of ground-truth labels for automatically generated test inputs. Most TIGs address this issue by applying small perturbations to inputs with known labels or sampling inputs from specific data distributions. However, prior work has shown that these approaches can easily produce *invalid* inputs, i.e., samples that fall outside the semantic domain of the classification task [7]–[9]. Misclassifications triggered by invalid inputs can mislead quality assessment, inflating testing metrics while providing little actionable insight to testers.

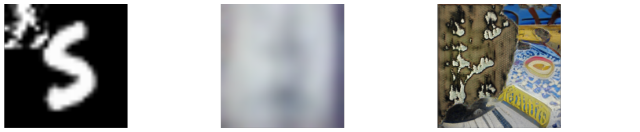

Fig. 1. Examples of misclassification-inducing inputs generated by three TIGs for MNIST, SVHN, and ImageNet. For these images, human validity judgments disagree with existing automated validators.

Manually inspecting the validity of test inputs is costly and does not scale. To mitigate this burden, several studies proposed automated input validators, typically based on distributional similarity (e.g., reconstruction error of Variational Auto-Encoders) or distance computed on a learned feature space [8], [10]. Recent advances in Large Language Models have enabled multimodal reasoning, allowing them to process information across different input types, and thus motivating their use for image validity assessment. Despite these advances, it remains unclear which validation approaches most closely align with human judgments of validity.

Prior work compared automated validators against independent human assessments across multiple TIGs and datasets [7]. It revealed a fundamental mismatch between human notions of validity and automated distribution-based criteria, showing that existing validators often act as proxies for in-distribution detection rather than semantic validity, as shown in Figure 1.

In this paper, we propose DEEPNAQQAL, a *supervised* input validator that learns validity directly from human-annotated labels. Unlike prior approaches that rely on proxy

This work was supported by the project PNRR M4 C2 I1.3 "SEcurity and RIghts in the CyberSpace (SERICS)" (PE00000014 - CUP H73C2200089001, D33C22001300002) PE7, funded by Next-Generation EU.

signals such as reconstruction error or feature distance, DEEPNAQQAL is trained to explicitly distinguish valid from invalid misclassification-inducing inputs. It builds on Deep Neural Networks (DNNs) originally designed for image classification and uses transfer learning to internalize the semantic features humans use when judging validity. In particular, we train DEEPNAQQAL using synthetic misclassification-inducing inputs generated by multiple TIGs and annotated for validity from existing benchmarks [7].

We conduct a comprehensive empirical evaluation comparing DEEPNAQQAL against state-of-the-art automated validators, including reconstruction-, distance-, and LLM-based approaches. Our study addresses: (i) how accurately DEEPNAQQAL aligns with human judgments; (ii) how it compares to alternative automated validators; and (iii) how well it generalizes across TIGs and under reduced training data. Our results show that DEEPNAQQAL consistently achieves the highest agreement with human assessors across all considered datasets, improving accuracy by up to 14% over the strongest baseline. We further show that supervised validators benefit substantially from transfer learning, remain effective even when trained on reduced labeled datasets, and generalize well to previously unseen TIGs. Thus, we offer a practical and robust solution for filtering invalid test inputs in DL testing pipelines. Our key contributions are the following:

- DEEPNAQQAL, a novel *supervised* automated validator that learns to distinguish valid from invalid misclassification-inducing inputs directly from humans.
- The largest empirical study to date on automated validation of misclassification-inducing inputs for DL image classifiers generated by 5 TIGs, comparing 7 validation approaches across 3 image classification tasks of increasing complexity, with ground-truth validity established through independent human assessment.

To foster open and reproducible research, we make our benchmark and experimental artifacts publicly available [11].

II. VALIDITY ASSESSMENT FOR DL IMAGE CLASSIFIERS

TIGs operate in a vast input space where only a small, task-specific subset corresponds to semantically valid images [12]. However, TIGs may drift outside this subset, producing inputs that are syntactically plausible but *invalid* for the target classification task. As manual inspection does not scale [13], recent work has proposed automated validators to assess whether generated inputs remain within the intended domain.

A. Reconstruction-Based Validators

Reconstruction-based validators assess input validity using Variational Autoencoders (VAEs). They model the input domain through a probabilistic representation of the training data distribution and flag inputs that deviate from it as out-of-distribution (OOD) [14]. A VAE encodes an input into a low-dimensional latent distribution and reconstructs it through a decoder; inputs similar to the training data are reconstructed with low error, while OOD inputs typically yield higher reconstruction error [15], [16]. Thus, reconstruction error serves as a proxy for validity. This principle is adopted by two representative techniques: Distribution-Aware Input Validation (DAIV) [8] and SelfOracle [10].

DAIV trains a VAE on nominal data and selects a reconstruction-fidelity threshold using an auxiliary anomaly set, optimizing the separation between nominal and anomalous samples. SelfOracle instead models the reconstruction-error distribution of nominal inputs only and determines a threshold based on a target false-alarm rate, without requiring anomalous examples.

Empirical evidence [7] shows that reconstruction-based validators struggle on feature-rich datasets (e.g., ImageNet). In such domains, invalid inputs often share low-level visual features with nominal samples, leading VAEs to reconstruct both with comparable fidelity and making effective validity threshold selection difficult. This reveals a key limitation of reconstruction-based validation: it uses low-level distributional similarity, whereas humans actually assess whether inputs belong semantically to the given domain.

B. Distance-Based Validators

Distance-based validators assess input validity by measuring proximity to the distribution of nominal training data in a learned feature space. The core assumption is that valid inputs cluster near dense regions of nominal examples, while invalid inputs lie farther away. Unlike reconstruction-based approaches, these validators do not rely on generative models. Instead, they operate directly on feature representations extracted from a single trained network (e.g., penultimate-layer embeddings of a DL classifier) and apply geometric or density-based criteria to detect outliers.

Deep Support Vector Data Description (DeepSVDD) [17] is a reconstruction-based approach that has shown strong performance in recent benchmarks [18]. It learns a transformation that maps nominal samples into a feature space where they are tightly enclosed by a minimum-volume hypersphere. During training, DeepSVDD optimizes the network parameters so that the feature representations of nominal samples cluster around a learned center. The method leverages two alternative objectives: (i) a *one-class objective* that constrains all nominal samples to lie near the center, and (ii) a *soft-boundary objective* that introduces a learnable radius and allows a small fraction of samples to fall outside the hypersphere, accounting natural variability in the data. Inputs whose representations fall outside this hypersphere are considered invalid.

DeepSVDD benefits from classifier-derived embeddings that better capture high-level semantics. However, it can still misalign with human judgments when invalid inputs remain close to nominal data in feature space.

C. Large Language Models

Multimodal LLMs enable automated input validation by performing high-level semantic reasoning over images conditioned on natural-language instructions [19]–[21]. By jointly processing visual inputs and textual prompts, LLMs can judge

whether an image semantically belongs to a target domain, framing validity assessment as an instruction-following task.

LLM-based validators take as input (i) a textual definition of validity and (ii) the image to be assessed, and produce a reasoning-based judgment based on its semantic interpretation, which is then parsed into a binary valid/invalid decision. This allows them to detect statistically plausible but semantically invalid inputs, such as images with implausible object structure or unnatural shapes or textures.

Closed-source models such as GPT-4o [22], [23] and GPT-5.1 [24], [25]) offer strong vision–language capabilities but limit reproducibility due to opacity and evolving APIs. Open-source models such as LLaVA offer weaker performance but support fine-grained customization and reproducibility.

LLM validators can operate in a *zero-shot* setting, relying on the prompt instructions and models' semantic generalization, or in a *few-shot* setting, where examples of valid and invalid inputs are provided to improve consistency (at the cost of longer prompts and over-fitting to the provided examples). Decoding parameters, such as *temperature*, affect output determinism: lower values reduce variability across runs but do not eliminate the risk of *hallucinations* [26], i.e., reasoning steps or visual interpretations not grounded in the input.

III. The DeepNaqqal approach

In this section, we introduce DeepNaqqal, a supervised learning validator used across diverse image classification tasks and compared with state-of-the-art input validators. In validators based on Supervised Learning, a model (typically a Deep Neural Network (DNN)) is trained to distinguish between valid and invalid inputs using examples mostly labeled by human annotators. Supervised validators provide a flexible and semantically expressive mechanism for assessing validity, particularly in domains where human interpretation plays a central role. This approach is particularly useful when validity depends on semantic properties that are difficult to encode through fixed rules or similarity metrics.

Our approach builds upon the benchmark by Riccio and Tonella [7], which provides misclassification-inducing inputs generated by multiple TIGs along with human-annotated validity labels. Following their definition of input validity, an input is deemed *valid* if it belongs to the input domain and domain experts can assign it a label from the task's label set.

A key challenge with supervised validators is data scarcity, especially due to the prevalence of valid inputs over invalid ones. To address this problem, standard data augmentation techniques can be applied to enrich the training set and improve generalization.

Supervised validators can be efficiently instantiated through transfer learning, i.e., by using pre-trained and robust deep architectures that are fine-tuned on the validity task. Such models benefit from the rich feature representations learned on large generic datasets and can achieve good performance even when only a modest number of labeled examples is available, mitigating the high cost of labeling.

Algorithm 1: DeepNaqqal's training process

Input: X: TIG-generated images;
Y: human-annotated validity labels;
Y_{exp}: intended class labels;
T: TIG identifiers;
M: validator's architecture;
s: random seed;
SUT: classifier under test;
n_{split}: percentage for stratified partitioning;
$tshd_{aug}$: augmentation threshold;
Output: $\hat{M}$: trained model
X_{test}: test inputs
Y_{test}: test labels

seed $\leftarrow$ s
C $\leftarrow$ loadModel(SUT)
$Y_{\text{enc}} \leftarrow$ labelEncoding(Y)
$X_{\text{train}}, Y_{\text{train}}, X_{\text{test}}, Y_{\text{test}} \leftarrow$ StratifiedSampling($X, Y_{\text{enc}}, T, n_{\text{split}}$)
$X_{\text{aug}} \leftarrow \emptyset$; $Y_{\text{aug}} \leftarrow \emptyset$
foreach unique (c, t) subgroup in (Y_{train}, T) **do**
  $I \leftarrow \{i \mid Y_{\text{train}}[i] = c \ \wedge \ T[i] = t\}$; *// Indices of samples in subgroup (c, t)*
  $N \leftarrow |I|$; *// Current size of the subgroup*
  if $N < \mathbf{tshd_{aug}}$ **then**
    $Z \leftarrow \mathbf{tshd_{aug}} - N$; *// Needed augmentation*
    for $z = 1$ **to** Z **do**
      $i \leftarrow$ sample(I)
      $\tilde{x} \leftarrow$ augment($X_{\text{train}}[i]$)
      if $SUT(\tilde{x}) \neq Y_{exp}[i]$ **then**
        $X_{\text{aug}} \leftarrow X_{\text{aug}} \cup \{\tilde{x}\}$
        $Y_{\text{aug}} \leftarrow Y_{\text{aug}} \cup \{c\}$
$X_{\text{train}} \leftarrow X_{\text{train}} \cup X_{\text{aug}}$
$Y_{\text{train}} \leftarrow Y_{\text{train}} \cup Y_{\text{aug}}$
$\hat{M} \leftarrow$ train($M, X_{\text{train}}, Y_{\text{train}}$)
return $\hat{M}$, X_{test}, Y_{test}

DeepNaqqal trains DNN-based supervised validators using images that trigger misbehaviors in the classifier under test along with the corresponding validity labels provided by human assessors. It supports convolutional classifiers of varying complexity, ranging from shallow architectures to deeper and more expressive ones. By systematically comparing multiple configurations, our framework enables an empirical assessment of how well supervised validators replicate human validity judgments across tasks of increasing complexity against state-of-the-art validators. DeepNaqqal's training procedure is structured into three main phases, as shown in Algorithm 1: (1) framework initialization and architecture selection, (2) data preparation, and (3) validator training.

A. Initialization and Architecture Selection

The pseudo-code implemented in DeepNaqqal (see Algorithm 1) takes as input the following data:

- X, a set of misclassification-inducing images from TIGs;
- Y, the corresponding *ground-truth validity labels* provided by human annotators;
- Y_{exp}, the *expected class labels* for each image as specified by the TIGs, identifying the class that the classifier under test should have assigned to the input according to the TIG (this class by construction differs from the class assigned to the input by the classifier under test, as only misclassified inputs are considered).

- T, information about the TIG that generated each image.

For each dataset, the user can choose among multiple validator model architectures M, enabling an assessment of how the validator complexity influences the validation performance.

The augmentation threshold $\mathbf{tshd_{aug}}$ is also configured at initialization. This parameter defines the minimum desired size for the valid/invalid subset of the training data used to train the validator. The goal is to expand under-represented subsets of the validator's training set by synthetically producing more samples, using transformations that preserve their validity status, thus helping the validator to learn minority-class patterns. The procedure begins by fixing a user-defined random seed (line 1) to ensure reproducibility. DEEPNAQQAL loads the classifier under test *SUT* for the given classification task (line 2). This classifier is later used for data augmentation.

B. Data Preparation

To prepare human-annotated labels for validity prediction, DEEPNAQQAL encodes them into Y_{enc} as binary variables, where 0 denotes an invalid input and 1 represents a valid one (line 3). Then, DEEPNAQQAL draws a user-defined percentage of the initial dataset for training (line 4), by applying stratified random sampling. The remainder of samples (i.e., the test set) are reserved for benchmarking and comparing the trained classifier against other validators. Stratification is performed using both Y_{enc} and T as targets, ensuring that the resulting test set preserves (i) the original distribution of validity labels and (ii) the relative proportions of inputs generated by each TIG. Since every TIG in our dataset provides more than two samples, this approach guarantees that each TIG is represented in both the training and test partitions [27], preventing degenerate cases in which an entire generator would be absent from one of these partitions. Random split (rather than a fixed or sequential one) avoids biases introduced by any ordering or clustering present in the dataset (e.g., TIG-specific sample clusters), thus improving the validator's generalization ability.

To mitigate data scarcity and imbalance in the training set, DEEPNAQQAL applies targeted data augmentation guided by the user-specified augmentation threshold $\mathbf{tshd_{aug}}$ parameter. This threshold specifies the minimum number of samples required for the valid/invalid subset of the training data used to train the validator. Whenever a subset contains fewer than $\mathbf{tshd_{aug}}$ samples, DEEPNAQQAL generates synthetic additional samples starting from a seed belonging to the subset until it reaches the target threshold (lines 9—16). After this process, every subset has at least $\mathbf{tshd_{aug}}$ samples. As an example, consider a task where valid/invalid $\times$ 5 TIGs produce 10 (class, TIG) subgroups, and let $\mathbf{tshd_{aug}} = 100$. Suppose two subgroups are under-represented: $\{TIG_1, \texttt{valid}\}$ has 10 samples and $\{TIG_2, \texttt{invalid}\}$ has 20 samples. DEEPNAQQAL therefore generates $100 - 10 = 90$ augmented samples for $\{TIG_1, \texttt{valid}\}$ and $100 - 20 = 80$ samples for $\{TIG_2, \texttt{invalid}\}$, so that both subgroups contain exactly 100 samples. Augmentations are chosen to produce slight transformations that reflect the natural variability of each dataset and preserve human recognizability [28]. Since augmentation may change an input's semantic label or affect its validity, we applied only natural-image transforms (resized cropping, small rotations, slight color and sharpness jitter).

After generating a candidate augmented image, DEEPNAQQAL submits it to the classifier under test (*SUT*) (line 14). We retain the augmented sample in the training set only if the SUT misclassifies it. This design choice aligns augmented data with the original distribution of interest (misclassification-inducing inputs). Retaining only SUT-misclassified augmentations allows training the validator on inputs that are relevant for the problem. To avoid overfitting the validator, we augment candidates randomly sampled from the original seeds (line 12).

C. Validator Training

For the training of the validator (lines 17—19), DEEPNAQQAL employs Stratified $\mathcal{K}$-Fold Cross-Validation, which partitions the dataset into $\mathcal{K}$ equally sized folds while preserving the original class distribution within each fold (line 19). This procedure reduces the risk of overfitting by exposing the model to multiple train–validation splits, allowing performance to be assessed across diverse subsets of the data [29]. After cross-validation, the best-performing model configuration is selected and used in the final testing phase to perform our experimental evaluation.

IV. EMPIRICAL SETUP

A. Research Questions

$\mathbf{RQ_1}$ (Impact of Model Complexity): *How does the architectural complexity of* DEEPNAQQAL *affect its accuracy?*

The architecture used to implement DEEPNAQQAL may influence its ability to distinguish valid from invalid inputs. Simpler models may be misled by fine-grained input perturbations generated by TIGs, while more complex architectures, particularly when enhanced through transfer learning, may generalize better to challenging inputs. At the same time, higher complexity may introduce a greater risk of overfitting to TIG-specific patterns. This RQ investigates whether increasing architectural complexity leads to more accurate validation.

$\mathbf{RQ_2}$ (Comparison with State of the Art): *Which automated validation approach most accurately approximates human validity judgments?*

Automated input validation techniques aim to distinguish valid from invalid inputs, reducing the need for human inspection. However, their usefulness depends on how reliably they replicate human judgments of validity. This RQ compares supervised validation with automated validators from the literature. For this RQ, we use a validator's training set obtained from the labeled datasets available for *all* considered TIGs (i.e., in Algorithm 1, T contains all available TIGs).

$\mathbf{RQ_3}$ (Intra-TIG Generalization) *How does* DEEPNAQQAL *generalize to unseen inputs generated by the same TIG?*

In this RQ, we assess whether DEEPNAQQAL can generalize to new inputs produced by the same TIG used during training. This means that we use a validator's training set obtained from the labeled datasets available for just *one* of the considered TIGs at a time (i.e., in Algorithm 1, T contains only one

of the available TIGs). This intra-TIG evaluation isolates the validator's ability to learn TIG-specific validity patterns. This experiment allows us to perform a detailed analysis for each TIG, under the assumption that only one is available for training a validator that will be later used with the same TIG.

RQ_4 (Leave-One-Out Cross-TIG Generalization) *How does* DEEPNAQQAL *generalize to TIGs not seen during training?*

This RQ evaluates the extent to which DEEPNAQQAL can correctly assess the validity of inputs produced by TIGs that were not observed during training. This means that we use a validator's training set obtained from the labeled datasets available for TIGs different from the one used to test the validity of the generated inputs (i.e., in Algorithm 1, T contains all available TIGs except one). This scenario is important in practice, as new/updated TIGs may use input perturbations that differ substantially from those used to train DEEPNAQQAL. To approximate unseen TIGs, we adopt a leave-one-out evaluation strategy: for each TIG, we train the validator on the inputs generated by all other TIGs and then test it exclusively on the held-out TIG. This setup quantifies robustness against shift in TIG mechanisms, evaluating if DEEPNAQQAL can generalize beyond the TIGs it was trained on.

RQ_5 (Data Efficiency): *How does reducing the size of the training dataset affect the accuracy of* DEEPNAQQAL*?*

This RQ evaluates the level of sensitivity of DEEPNAQQAL to the amount of training data available. In practical scenarios, collecting validated inputs can be costly or time-consuming, thus a validator should ideally learn effectively from limited data. Therefore, we investigate whether DEEPNAQQAL can retain acceptable performance when trained on progressively smaller subsets of the data.

B. Classification Tasks and DL Systems Under Test

We considered the three popular image datasets from the benchmark of Riccio and Tonella [7]: MNIST [30], SVHN [31], and ImageNet [32]. These datasets span classification tasks of increasing complexity, ranging from recognizing 10 classes in small centered grayscale digits to identifying 1K classes in large, colored images representing diverse real-world objects. For each dataset, we selected a corresponding pre-trained DL model that is (1) well-performing in terms of classification accuracy, (2) popular, and (3) widely adopted in DL testing research.

MNIST consists of 70K grayscale images of handwritten digits. The images are small (28×28 pixels) with intensity values ranging from 0 to 255. The images are centered and presented on a uniform black background, which makes the task relatively simple. As DL classifier, we used the LeNet convolutional neural network [30], with the weights released by Pei et al. [33], a standard model adopted in several DL testing studies [8], [33], [34]. Its architecture consists of two convolutional layers, followed by a fully connected layer.

SVHN contains 600K real-world digit images (from 0 to 9) extracted from photographs of house number plates. Unlike MNIST, these images are RGB, slightly larger (32×32 pixels), and exhibit visual variability: digits may appear with irregular backgrounds, varying illumination, and adjacent numbers partially visible in the crop. We used the *All-CNN-A* architecture by Springenberg et al. [35], consisting of 7 convolutional layers and the weights provided by Dola et al. [8].

ImageNet is a large image database containing 1.43M high-resolution RGB images belonging to 1K classes. This database is one of the most influential large-scale benchmarks in computer vision [36]. Compared to MNIST and SVHN, ImageNet inputs have a higher resolution (most classifiers use 225×225 resolution) and depict diverse real-world objects such as animals, tools, and foods. For this dataset, we adopted the vgg16 deep convolutional neural network [37], which includes 13 convolutional layers followed by 3 fully connected layers, with standard pre-trained weights from the Keras library [38].

C. Experimental Dataset

We used and extended the dataset from Riccio and Tonella [7], as it is the largest available source of misclassification-inducing inputs, generated by 5 representative TIGs under comparable budgets and across tasks of increasing difficulty. Most importantly, each input was annotated by humans to determine its validity. These human labels are essential, as they enable supervised training of validators and allow us to assess if validator predictions align with human judgement. In the following, we describe our benchmark.

1) Misclassification-Inducing Images: We used images generated from the 5 TIGs included in the original benchmark [7], as they cover the major test generation paradigms: Raw Input Manipulation (RIM), Generative DL models (GDLM), and Model-based Input Representation (MIR).

DeepXplore (dx) is a RIM technique perturbing images directly in pixel space to expose misclassifications while maximizing neuron coverage, using simple transformations such as occlusion, lighting changes, and blackout regions [33].

DLFuzz (dlf) is a RIM technique that injects small pixel-level perturbations, guided by neuron coverage and output-layer confidence to increase misclassification likelihood [34].

Sinvad (sv) uses generative DL models (VAE, GAN) to explore the latent space of the input distribution, perturbing latent variables to generate misclassification-inducing inputs [39].

Feature Perturbations (fpt) modifies intermediate feature representations within a generative model, guiding perturbations toward misclassification while constraining them to observed training ranges [40].

DeepJanus (dj) is a MIR technique that abstracts inputs into a domain-specific representation (e.g., SVG for handwritten digits), mutates the model parameters to explore decision boundaries, and then re-renders concrete inputs [41].

For each classification task, we assembled test suites of equal size, consisting of the same number of misclassification-inducing inputs for each TIG. We use the same fixed ground-truth class labels as in the original study [7]: *digit 5* for MNIST and SVHN, and *pizza* for ImageNet. For MNIST and SVHN, we directly used the inputs provided in the original benchmark, i.e., 500 images per classification task (100 inputs from each of the five TIGs). For ImageNet, we extended the original

dataset, as it was composed of only 80 images, which are not sufficient to robustly train supervised validators or test their performance. Following the experimental setup used by Riccio and Tonella, we generated a test suite of 480 ImageNet inputs, consisting of 120 images from each of the 4 applicable TIGs (dx, dlf, sv, and fpt), as dj does not support ImageNet.

2) Validity Labels: We assigned a validity label to each misclassification-inducing input using the human assessments collected by Riccio and Tonella [7] from 220 independent assessors. Each image was independently evaluated by two crowdworkers and labeled as *valid/invalid* only if both agreed that it belonged to the task domain; inputs with disagreement were discarded to ensure high label reliability.

Since we extend the ImageNet portion of the benchmark, we conducted an additional round of crowdsourcing-based human assessment following the same protocol on Amazon Mechanical Turk. We applied standard quality-control measures [42], including attention-check questions and restricting participation to workers with a minimum 95% approval rate. To ensure consistency across datasets, we harmonized the ImageNet labeling scheme with MNIST and SVHN following the protocol of Maryam et al. [43]. Assessors selected the most appropriate label among the expected class, the eight most frequently mispredicted classes by the classifier under test, "Another real-world object", and "No real-world objects" (indicating invalidity). This resulted in a uniform and comparable validity labeling with 11 options across all datasets. In total, our ImageNet validation consisted of 48 surveys and involved 96 unique assessors.

D. Automated Input Validators Setup

1) DEEPNAQQAL*:* We trained multiple supervised validators, differing by the complexity of their architecture, on the dataset described in Section IV-C, which pairs images generated by TIGs with human-annotated validity labels. All validators are implemented as standard image classification DNNs in which the original output layer is replaced with a binary classification head predicting whether an input is valid or invalid. This ensures that the validators leverage the representational power of deep networks while directly optimizing for the validity-prediction task. Each pre-trained model is fine-tuned starting from the initial weights of the DNN architecture. The fine-tuning process updates only the newly introduced classifier layers, while the weights of the remainder of the network are "frozen", thus acting as fixed feature extractors. To study the impact of model complexity (RQ1), we considered for MNIST and SVHN two alternative architectures: (1) the same, simpler DNN used for the classifiers under test, or (2) more complex architectures obtained from the pretrained vgg16 network we considered in the ImageNet task. We adopted a 70/30 train–test split, ensuring that the model has sufficient data for learning, while holding out a substantial portion for evaluation. To further improve robustness and reduce overfitting, we applied stratified $\mathcal{K}$-fold cross-validation with 8 folds, preserving the original class distribution in each fold and fine-tuning the model for 16 epochs in total [29]. After training, the best-performing model (based on cross-validation performance) is applied to the test sets adopted in our empirical assessment, allowing us to measure accuracy as a primary metric of validation effectiveness.

2) Reconstruction-Based: We set up reconstruction-based validators, i.e., DAIV and SelfOracle, according to the study by Riccio and Tonella [7]. Both methods rely on a VAE trained on the same data used to train the DL system under test. After training, these validators assign a validity score based on its reconstruction-based metric: reconstruction fidelity for DAIV and reconstruction loss for SelfOracle. For DAIV, the anomalous datasets are FashionMNIST [44] for MNIST, CIFAR-10 [45] for SVHN, CelebA [46] for ImageNet. For SelfOracle, we configured its threshold to achieve a very low false-alarm rate of 10^{-4}, since most inputs in the original test sets represent nominal data by construction. Our threshold is $100\times$ smaller than the original benchmark [7], as it demonstrated better results in its replication package. All VAE architectures are from the original benchmark [7].

3) Distance-Based: We set up DeepSVDD following the configuration proposed by Ruff et al. [17], which performed particularly well in the empirical comparison by Zhang et al. [18]. For MNIST, we adopted the exact configuration used by Zhang et al. [18], as MNIST was evaluated in their study. For SVHN, we used the architecture proposed by Ruff et al. [17] for CIFAR-10, due to the similarity between the two datasets in terms of color channels and image resolution. For ImageNet (not considered by Ruff et al. [17] nor Zhang et al. [18]), we adopted a feature-extraction approach: we used a ResNet-50 model pre-trained on ImageNet to obtain high-level representations and train DeepSVDD on these features. This avoids the prohibitive cost of training DeepSVDD on feature-rich, large-scale datasets. In particular, DeepSVDD learns a center in feature space and then determines a radius that encloses a predefined portion of the training data. We use the standard setting for which up to 10% of the most distant training samples may lie outside the sphere, consistently with the official implementation [17]. Once trained, the DeepSVDD validator is used to assess the validity of images based on their distance to the hypersphere center.

4) LLMs: We selected three representative and widely adopted multi-modal LLMs, i.e., GPT-4o [47], GPT-5.1 [25], and LLaVA [48]. They represent distinct families in terms of scale, openness, architecture, and training objectives, allowing us to test whether the validation ability is consistent across different LLMs. Evaluating them enables us to assess whether input validity assessment is an emergent capability across different LLMs or if it depends on proprietary training pipelines.

We evaluated LLMs as *zero-shot* validators, i.e., models that judge whether an input image looks valid or invalid without being explicitly trained on labeled examples. The validator is provided only with the test image and a carefully designed prompt that describes the notion of validity, and it must decide whether the input represents a plausible instance of the target domain. We adopted this setting since preliminary experiments showed that few-shot prompting degrades performance. In fact,

when we supplied the models with labeled examples (few-shot), they overfitted to the examples in the prompt, repeating the same labels regardless of the actual content of the image. Moreover, zero-shot is the realistic deployment setting for LLMs: when new TIGs or new domains appear, developers would want to use an LLM without constructing a new labeled dataset for fine-tuning. For all LLMs, we set the temperature to 0.0 to maximize determinism and minimize hallucinated responses [26]. The model's output (e.g., "valid digit", "invalid image", etc.) is parsed into a binary decision, enabling direct comparison with other validators.

In our experiment, the LLM receives a test image together with a fixed prompt. In particular, each validator is configured using a dataset-independent *user prompt* and dataset-specific *system prompts*. The system prompt defines the notion of validity for the target domain (e.g., handwritten digits for MNIST), while the user prompt requests a binary validity decision. This separation ensures that the instructions defining validity adapt to each dataset, yet the interaction pattern with the model is stable and comparable across datasets. The prompts used in our experiments are available in the replication package [11].

E. Experimental Procedure

We conducted a comprehensive comparison between DEEPNAQQAL and state-of-the-art automated validators described in Section IV-D. For each configuration, we evaluated how accurately the validator reproduces the human ground-truth validity labels obtained through crowdsourcing. The **accuracy** metric is defined as the proportion of images for which the validator's output matches the human judgment. We computed the accuracy of each validator using our extended benchmark, by directly comparing the predicted validity label for each image with its corresponding ground-truth validity label.

To mitigate randomness, we trained each supervised configuration and repeated validator predictions multiple times. In particular, we performed 7 independent runs for each research question and classification task, which we found sufficient to capture variance within a reasonable computational budget. To ensure reproducibility, all runs were executed with fixed random seeds. To assess if the observed differences in accuracy between validators were statistically significant, we applied the Mann–Whitney U test [49]. To quantify the magnitude of differences, we computed the effect size using Cohen's d [50]. A threshold of $p < 0.05$, combined with a non-negligible effect size, was used to determine statistical significance.

In RQ1, we assessed the impact of model complexity in MNIST and SVHN tasks by comparing the simpler DNN architecture against the deeper vgg16-based validator. We did not perform this comparison on ImageNet since vgg16 already represents the most complex model in our setup, while simpler architectures would not be expressive enough for ImageNet-scale classification, and training additional alternative models from scratch would be prohibitively expensive. Thus, for ImageNet we did not include any complexity impact analysis.

In RQ2, we compared DEEPNAQQAL against state-of-the-art validators across the considered classification tasks.

TABLE I
RQs 1-2: ACCURACY ACHIEVED BY THE CONSIDERED AUTOMATED VALIDATORS. BEST RESULTS IN BOLD; UNDERLINED VALUES ARE NOT STATISTICALLY DIFFERENT FROM THE BEST.

Dataset	Model	DNQ	DAIV	SO	SVDD	LLaVA	GPT4	GPT5
MNIST	LeNet	**0.972**	0.677	0.834	0.826	0.667	0.777	0.782
	vgg16	**0.980**	0.677	0.834	0.826	0.667	0.777	0.782
SVHN	CNN-A	<u>0.769</u>	0.701	0.759	**0.792**	0.610	0.682	0.752
	vgg16	**0.859**	0.701	0.759	0.792	0.610	0.682	0.752
ImageNet	vgg16	**0.961**	0.947	0.939	0.684	0.867	0.896	0.922

In RQ3, we evaluated intra-TIG generalization. For every TIG, we split its dataset into 70% training and 30% testing, train the validator on the former, and evaluate it on the held-out samples from the same TIG.

In RQ4, we assessed inter-TIG generalization through leave-one-out validation. The validator is trained on all TIGs except one, which is used exclusively for testing.

In both RQ3 and RQ4, we mitigated data scarcity by adopting a $\mathbf{tshd_{aug}}$ threshold of 100.

In RQ5, we measured the effect of training data size by progressively reducing the available training data in steps of 20%, while keeping all other factors fixed.

V. RESULTS

A. RQ1 (Impact of Model Complexity)

To answer RQ1, we investigated how the complexity of the architecture of DEEPNAQQAL affects its accuracy. As shown in the second column of Table I, we compared lightweight architectures (LeNet for MNIST and CNN-A for SVHN) against a deeper model based on vgg16, using transfer learning from large-scale image datasets (i.e., ImageNet).

Across all experiments, a consistent trend emerged: validator accuracy improved as model complexity increased. On MNIST, the simpler architecture (LeNet) already achieved strong accuracy (97%). However, using the deeper model allowed a further improvement of approximately +1%. A similar, yet more pronounced, pattern was observed for SVHN. In this case, switching from CNN-A to the VGG16-based validator resulted in a statistically significant gain of +8%. Overall, these findings show that more expressive architectures provide a clear advantage in DEEPNAQQAL.

> RQ1: Higher model complexity consistently strengthened DEEPNAQQAL, achieving an improvement of +4.7% accuracy across TIGs and datasets. Our findings clearly indicate that more complex architectures using transfer learning provide a more accurate validator.

B. RQ2 (Comparison with State of the Art)

To determine which automated validator performs best, we compared their accuracy across the considered tasks. Table I (columns 3—10) reports the accuracy achieved by each validator on the same test sets. Thus, the rows report how closely each method reproduces human validity assessments.

On MNIST, DEEPNAQQAL achieves the highest agreement with human labels, with both the LeNet and vgg16 versions reaching $> 97\%$ accuracy. This is an improvement of nearly 14% over the best-performing alternative (SelfOracle). The weaker performance of the remaining validators might be attributed to overfitting to low-level features (e.g., MNIST's constraints: centered digits on a black background). Methods such as reconstruction- or distance-based validation tend to penalize even small positional deviations or noisy background pixels. In contrast, DEEPNAQQAL (like human annotators) prioritizes the semantic content of the digit and remains robust to minor visual variations and noise.

For SVHN, the simpler DEEPNAQQAL achieves an accuracy comparable to DeepSVDD (p-value > 0.05). However, with a more complex architecture, DEEPNAQQAL improves up to 10.5%. In particular, the vgg16-based validator is significantly more accurate than all other approaches, outperforming the best competitor (DeepSVDD) by up to 0.8%. The lower accuracy observed across most validators on SVHN with respect to MNIST is expected, as the former contains more background clutter and variation in illumination. As a result, shallow architectures struggle to capture the complex visual structure of SVHN images, while deeper supervised models benefit from the complexity of the feature extractor.

Even on ImageNet, which is characterized by high intra-class variability and visual complexity, DEEPNAQQAL outperforms all other approaches, consistently achieving over 96% accuracy. DeepSVDD performs poorly with ImageNet since its core assumption does not always hold on this dataset. Each class contains substantial variation (e.g., in background), which results in several distinct clusters rather than a single compact region in the feature space. Since DeepSVDD models normal data with a single hypersphere, struggling to capture such distributions; a tight hypersphere rejects many valid samples, while a loose one admits invalid samples.

> RQ2: DEEPNAQQAL achieves the highest agreement with humans across all tasks (up to 14% accuracy over the best alternative on MNIST) and maintains this advantage as complexity increases (over 96% accuracy on ImageNet).

C. RQ3 (Intra-TIG Generalization)

To evaluate how well DEEPNAQQAL generalizes to unseen test inputs from each specific TIG, we performed a 70/30 train–test split within each TIG at augmentation level 100, using the vgg16 model for all experiments, as it was the best-performing architecture in previous RQs. Each row in Table II reports the average accuracy obtained by training DEEPNAQQAL on a specific TIG (column 3) and compares it with the accuracy of alternative validators evaluated on the same test sets (columns 4—9). Missing rows correspond to dataset–TIG combinations where DEEPNAQQAL was not applicable since the TIG contained only valid examples. In such cases, a binary classifier cannot be trained in a meaningful way, as the absence of invalid samples prevents learning a decision boundary between valid and invalid inputs.

TABLE II
RQ3: INTRA-TIG ACCURACY ACHIEVED BY DEEPNAQQAL AND PERFORMANCE OF OTHER VALIDATORS ON THE SAME TEST SET. BEST RESULTS IN BOLD; UNDERLINED VALUES ARE NOT STATISTICALLY DIFFERENT FROM THE BEST.

Dataset	TIG	DNQ	DAIV	SO	SVDD	LLaVA	GPT4	GPT5
MNIST	dx	**0.980**	0.102	0.153	0.566	0.674	0.781	0.873
	fpt	**1.000**	0.944	**1.000**	0.944	0.352	0.873	0.929
	avg	**0.990**	0.523	0.577	0.755	0.513	0.827	0.901
SVHN	dj	**0.944**	0.391	0.373	0.609	0.576	0.466	0.696
	dlf	0.929	**0.955**	**0.955**	**0.955**	0.604	0.753	0.818
	dx	**0.908**	0.456	0.755	0.660	0.742	0.762	0.796
	fpt	0.801	0.808	**0.826**	**0.826**	0.739	0.801	0.795
	sv	**0.919**	0.913	0.913	0.913	0.354	0.689	0.839
	avg	**0.900**	0.704	0.764	0.793	0.603	0.694	0.789
ImageNet	dx	0.972	0.972	0.968	0.885	0.885	**1.000**	0.980
	sv	**0.863**	0.778	0.762	0.741	0.799	0.486	0.683
	avg	**0.917**	0.875	0.865	0.813	0.842	0.743	0.831

For MNIST, DEEPNAQQAL achieved a perfect score on *fpt*, which matches the performance of SelfOracle. However, reconstruction-based validators (SO and DAIV) were highly sensitive to pixel-level perturbations introduced by *dx*, obtaining the lowest accuracy (15%), while DEEPNAQQAL remained robust, achieving the highest accuracy (98%). On average, DEEPNAQQAL achieved significantly higher accuracy (99%) than other alternatives on MNIST.

On SVHN, DEEPNAQQAL consistently achieves an accuracy that is significantly higher or at least comparable to other validators across TIGs. Reconstruction- and distance-based methods occasionally perform well (i.e., highest accuracy on *dlf* and *fpt*), but their performance is highly unstable, dropping by up to 57% on *dj* with respect to DEEPNAQQAL. Overall, DEEPNAQQAL generalizes reliably across most TIGs, achieving an average accuracy of 90%, which is nearly 11% higher than the best alternative (DeepSVDD).

For ImageNet, DEEPNAQQAL performs well on *dx* pixel perturbations, where 5 validators out of 7 achieved high accuracy, i.e., over 96%. Instead, *sv* produced images from the latent space of its GAN model that were harder to validate. On these images, DEEPNAQQAL achieved the highest accuracy (86%). On average, DEEPNAQQAL achieved the highest accuracy on ImageNet (91.7%), a value that is statistically comparable to reconstruction-based approaches.

> RQ3: DEEPNAQQAL consistently generalizes well to new inputs generated by the same TIG, achieving the highest or statistically comparable accuracy among the validators. It is robust to both pixel-level perturbations and semantically rich variations, providing the most reliable intra-TIG validity judgments (up to 11% more accurate on SVHN).

D. RQ4 (Leave-One-Out Cross-TIG Generalization)

Table III reports how effectively DEEPNAQQAL generalizes to unseen TIGs. To this aim, we employ a leave-one-out strategy: each row corresponds to a configuration where one TIG (shown in the second column) is excluded from training and used exclusively as the test target during evaluation.

TABLE III
RQ4: CROSS-TIG ACCURACY ACHIEVED BY DEEPNAQQAL AND PERFORMANCE OF OTHER VALIDATORS ON HELD-OUT (HO) TIG TEST SET. BEST RESULTS IN BOLD; UNDERLINED VALUES ARE NOT STATISTICALLY DIFFERENT FROM THE BEST.

Dataset	HO	DNQ	DAIV	SO	SVDD	LLaVA	GPT4	GPT5
MNIST	dj	0.988	0.958	**1.000**	0.906	0.906	0.844	0.938
	dlf	**1.000**	0.404	**1.000**	0.758	0.960	0.465	0.263
	dx	0.843	0.108	0.161	0.559	0.677	0.774	**0.849**
	fpt	**0.989**	0.904	**0.989**	0.947	0.319	0.883	0.936
	sv	0.979	0.979	**1.000**	0.958	0.448	0.927	0.927
	avg	**0.960**	0.670	0.830	0.826	0.662	0.779	0.783
SVHN	dj	**0.737**	0.395	0.382	0.605	0.579	0.461	0.645
	dlf	0.849	**0.959**	**0.959**	**0.959**	0.630	0.740	0.822
	dx	**0.868**	0.456	0.750	0.691	0.735	0.750	0.765
	fpt	**0.817**	0.800	0.813	0.813	0.693	0.773	0.773
	sv	0.826	**0.905**	**0.905**	**0.905**	0.392	0.676	0.824
	avg	**0.819**	0.703	0.762	0.795	0.606	0.680	0.766
ImageNet	dlf	0.951	**1.000**	0.983	0.880	0.769	0.966	**1.000**
	dx	0.992	0.983	0.975	0.908	0.899	**1.000**	0.992
	fpt	**1.000**	**1.000**	0.991	**1.000**	0.991	0.949	0.983
	sv	**0.773**	**0.773**	0.761	0.761	**0.773**	0.477	0.705
	avg	0.929	**0.939**	0.928	0.887	0.858	0.848	0.920

This setup approximates real-world conditions in which a validator must recognize validity on previously unseen types of misclassification-inducing inputs.

On MNIST, DEEPNAQQAL achieves the highest average accuracy (96%), statistically comparable to SelfOracle and outperforming all remaining baselines. DEEPNAQQAL demonstrates strong cross-TIG generalization (above 84% in all configurations), while other validators exhibit weaknesses on specific TIGs. Raw input perturbations are particularly challenging: reconstruction-based validators drop below 17% accuracy on *dx*, while GPT5 performs poorly on *dlf* (26.3%).

On SVHN, DEEPNAQQAL achieves the highest accuracy on one representative TIG from each category, i.e., RIM (*dx*), GDLM (*fpt*), and MIR (*dj*). Overall, DEEPNAQQAL reaches the highest average accuracy (81.9%), statistically outperforming all validators except SVDD. Interestingly, DEEPNAQQAL shows strong complementarity with DAIV, SO, and SVDD: they outperform DEEPNAQQAL on *dlf* and *sv*, whereas DEEPNAQQAL remains the most accurate for the remaining TIGs.

On ImageNet, DEEPNAQQAL achieves an average accuracy of 92.9%. Some validators show perfect performance on isolated TIGs (SVDD on *fpt*, GPT4 on *dx*, and GPT5 on *dlf*), yet none maintains top performance across TIGs. In contrast, DEEPNAQQAL and DAIV provide consistently stronger cross-TIG generalization, i.e., they obtain statistically comparable accuracy and significantly outperform other baselines.

> RQ4: DEEPNAQQAL provides the most stable cross-TIG generalization. Unlike the baselines, whose performance varies widely across TIGs and classification tasks, DEEPNAQQAL delivers consistently high accuracy even when confronted with perturbations never seen during training.

E. RQ5 (Data Efficiency)

Table IV summarizes how the size of the training set impacts the accuracy of DEEPNAQQAL. Each row corresponds to a different model architecture. The third column reports its accuracy when trained on the full training set, while each subsequent column shows the change in accuracy when progressively decrementing the available training data by 20%, down to a minimum of 10% of the original training set.

TABLE IV
RQ5: ACCURACY OF DEEPNAQQAL ACROSS VARYING TRAINING SET SIZE. UNDERLINED VALUES ARE NOT STATISTICALLY DIFFERENT FROM THE ACCURACY ON THE WHOLE TRAINING SET REPORTED IN COLUMN 3.

Dataset	Model	Total	Δ90	Δ70	Δ50	Δ30	Δ10
MNIST	LeNet	0.972	-0.009	-0.006	-0.002	-0.005	-0.032
	vgg16	0.981	+0.001	-0.014	0.000	-0.004	-0.014
SVHN	CNN-A	0.769	+0.011	-0.005	-0.035	-0.038	-0.143
	vgg16	0.859	+0.016	+0.006	-0.003	-0.041	-0.033
ImageNet	vgg16	0.965	+0.002	0.012	-0.010	+0.003	-0.028

On MNIST, the simpler LeNet model remains statistically comparable to its full-data performance even when trained on only 10% of the data, with its largest accuracy drop being just 3% at 10% training set size. The more complex vgg16 architecture also maintains comparable performance down to 30% of the training set, but when reduced to only 10%, it exhibits a statistically significant accuracy decrease of 1.4%.

For the simpler SVHN architecture, the decrease in accuracy is never statistically significant, exceeding 10% only when the training set is reduced to 10%. For the more complex SVHN architecture, the accuracy drop becomes statistically significant only at 30% and 10% of the training set. For vgg16 the accuracy slightly improves when the training set is reduced up to 70%. This behavior is likely due to the higher variability of the SVHN dataset: with the full dataset, vgg16 may overfit some TIG-specific patterns, whereas moderate data reduction acts as a form of regularization, removing redundant or noisy samples and yielding a marginal (yet not significant) improvement. On ImageNet, accuracy oscillates as the training set is reduced and even shows a statistically significant improvement at 70%.

> RQ5: DEEPNAQQAL remains robust even when trained on reduced datasets. While its accuracy can degrade as less training data becomes available, such decrease is generally modest, and often not statistically significant until the training set becomes very small.

F. Threats to Validity

Internal Validity concerns the extent to which the observed effects are caused by experimental factors rather than uncontrolled variables. To reduce these threats, all validators were integrated into our framework, ensuring consistent experimental conditions, i.e., identical training hyperparameters, TIG-generated datasets, and test sets across runs. We also evaluated architectures with varying complexity to assess the impact of model complexity on the validator accuracy.

Construct Validity concerns whether the adopted measures capture the intended notion of input validity. We rely on the definition of validity and the human-annotated ground truth

from Riccio and Tonella [7], ensuring alignment between the construct and its measurement.

External Validity concerns the generalizability of our findings. A possible threat is the selection of datasets, models, and validators. To mitigate this, we build upon and extend the largest existing benchmark for TIG validity [7], covering datasets of increasing complexity and all major families of automated validators.

The stochastic nature of supervised training may affect *Conclusion Validity*. To mitigate this, we repeated all experiments multiple times with fixed seeds, reporting average values and applying appropriate statistical tests to assess significance.

VI. Lessons Learnt and Actionable Insight

Supervised Learning best mirrors human validity. Reconstruction- and distance-based validators fail to capture subtle, yet semantics-breaking distortions produced by TIGs, as invalid images often retain strong structural similarity to valid ones, leading to higher false negative rates compared with supervised validators. LLMs offer richer semantic reasoning, but their decisions vary across models and remain misaligned on constrained datasets (e.g., MNIST and SVHN), which are probably under-represented in their training set. By learning directly from human-annotated validity labels rather than proxy metrics, DeepNAQQAL achieves the closest alignment with human judgments, effectively capturing semantic consistency and domain boundaries. Therefore, practitioners should prioritize supervised validators trained on human-labeled validity data when validating misclassification-inducing inputs, especially in safety-critical pipelines where subtle invalidities must not be missed.

Prefer deeper, pretrained models: transfer learning delivers stronger validators. Supervised validators built on top of deep, pretrained architectures consistently outperform other validators. Richer feature extractors (e.g., vgg16) enable more accurate validity judgments, particularly on complex domains such as SVHN and ImageNet. Transfer learning should be the default choice when constructing supervised validators, as it substantially improves reliability with minimal additional engineering effort.

Supervised validators remain effective with limited labels and unseen TIGs. Despite relying on human annotations, supervised validators maintain high accuracy even when trained on a small fraction of labeled data, thus reducing annotation costs. Moreover, they generalize well to unseen TIGs, remaining reliable for inputs not represented during training. This enables teams to deploy supervised validators under limited annotation budgets and retrain them selectively only when new invalidity patterns emerge, e.g., increasing false negative/positive rates.

VII. Related Work

Prior work evaluates DL systems by generating test inputs that trigger misbehaviors. Most studies assess test adequacy using metrics, such as the number of exposed misclassifications [51]–[53], input and failure diversity [41], [54], [55], input or feature space coverage [56]–[58], activation of internal components [33], [34], [59], [60], or mutation killing [61]. Despite the breadth of this literature, these approaches assume that generated inputs remain valid and rarely assess whether test inputs actually belong to the task domain, potentially compromising the conclusions drawn from such quality assessments. Only a few studies explicitly addressed test validity.

Dola et al. [8] integrated reconstruction-based validators into TIGs and showed that many generated inputs are invalid, misleading adequacy metrics. Their study focused only on raw input perturbations and in-distribution approximations of validity, without comparison to human judgment.

Other studies relied exclusively on human assessment [13], [43]. Human assessors evaluated the effectiveness of their TIGs by inspecting generated inputs, e.g., DeepJanus inputs were deemed more likely to be invalid by human assessors when generated against high-quality models [41]. These works do not compare human judgment with automated validators.

Riccio and Tonella [7] conducted the most comprehensive study comparing automated validation and human assessment across multiple TIG families. They revealed a crucial mismatch between automated validators and human validation, but their study is limited to reconstruction-based validators.

Subsequent work considered more validators. Zhang et al. [18] compared multiple automated validators and identified DeepSVDD as the best, but evaluated a limited set of TIGs and tasks, notably excluding SelfOracle, which was the strongest validator in prior work [7]. Ghobari et al. [62] proposed HiL-TV, a supervised human-in-the-loop validator relying on predefined validation metrics as training features. They considered an industrial scenario where validity is defined relative to a reference image, i.e., if a transformed input is an acceptable modification of the original. Unlike HiL-TV, DeepNAQQAL does not rely on predefined input features, but learns them directly from data through the underlying DL architecture.

Our work provides the most comprehensive empirical comparison of automated test input validation to date. We extend the benchmark by Riccio and Tonella with additional validator families, including LLMs, and directly compare all techniques against human ground truth across multiple datasets and TIGs.

VIII. Conclusions and Future Work

We introduced DeepNAQQAL, a supervised validator for misclassification-inducing inputs, and conducted the most comprehensive empirical comparison of automated input validation techniques for DL image classifiers to date. Our results show that DeepNAQQAL achieves the highest agreement with human judgments across all datasets and TIG families, outperforming reconstruction-, distance-, and LLM-based validators by up to 14%. Moreover, it generalizes well to unseen TIGs and remain effective even when trained on substantially reduced labeled datasets. These findings establish DeepNAQQAL as a practical and reliable baseline for filtering invalid test inputs in DL testing pipelines. We plan to extend the empirical study to additional datasets, TIGs, validators, and software under test, including industrial case studies.

REFERENCES

[1] P. Wang, E. Fan, and P. Wang, "Comparative analysis of image classification algorithms based on traditional machine learning and deep learning," *Pattern Recognition Letters*, vol. 141, pp. 61–67, 2021.

[2] V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, "Testing machine learning based systems: a systematic mapping," *Empir. Softw. Eng.*, vol. 25, no. 6, pp. 5193–5254, 2020. [Online]. Available: https://doi.org/10.1007/s10664-020-09881-0

[3] A. Guerriero, R. Pietrantuono, and S. Russo, "Operation is the hardest teacher: estimating DNN accuracy looking for mispredictions," in *Proceedings of the 43rd International Conference on Software Engineering*, ser. ICSE '21. Madrid, Spain: IEEE Press, 2021, p. 348–358.

[4] T. Zohdinasab, V. Riccio, and P. Tonella, "DeepAtash: Focused Test Generation for Deep Learning Systems," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 954–966.

[5] J. M. Zhang, M. Harman, L. Ma, and Y. Liu, "Machine learning testing: Survey, landscapes and horizons," *IEEE Transactions on Software Engineering*, vol. 48, no. 1, pp. 1–36, 2022.

[6] H. B. Braiek and F. Khomh, "On testing machine learning programs," *Journal of Systems and Software*, vol. 164, 2020.

[7] V. Riccio and P. Tonella, "When and why test generators for deep learning produce invalid inputs: An empirical study," ser. ICSE '23. IEEE Press, 2023, p. 1161–1173. [Online]. Available: https://doi.org/10.1109/ICSE48619.2023.00104

[8] S. Dola, M. B. Dwyer, and M. L. Soffa, "Distribution-aware testing of neural networks using generative models," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 226–237.

[9] O. Weißl, A. Abdellatif, X. Chen, G. Merabishvili, V. Riccio, S. Kacianka, and A. Stocco, "Targeted deep learning system boundary testing," *ACM Transactions on Software Engineering and Methodology*.

[10] A. Stocco, M. Weiss, M. Calzana, and P. Tonella, "Misbehaviour prediction for autonomous driving systems," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ser. ICSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 359–371. [Online]. Available: https://doi.org/10.1145/3377811.3380353

[11] M. Maryam, M. Biagiola, P. Tonella, and V. Riccio, "Deepnaqqal replication package." [Online]. Available: https://github.com/deeptestai/DEEPNAQQAL

[12] S. Yoo, "Sbst in the age of machine learning systems - challenges ahead," in *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*, 2019, pp. 2–2.

[13] M. O. Attaoui, H. Fahmy, F. Pastore, and L. Briand, "Black-box safety analysis and retraining of dnns based on feature extraction and clustering," *arXiv preprint arXiv:2201.05077*, 2022.

[14] J. Yang, K. Zhou, Y. Li, and Z. Liu, "Generalized out-of-distribution detection: A survey," *arXiv preprint arXiv:2110.11334*, 2021.

[15] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[16] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[17] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 4393–4402. [Online]. Available: https://proceedings.mlr.press/v80/ruff18a.html

[18] J. Zhang, J. Keung, X. Ma, X. Li, Y. Xiao, Y. Li, and W. K. Chan, "Enhancing valid test input generation with distribution awareness for deep neural networks," in *IEEE 48th Annual Computers, Software, and Applications Conference (COMPSAC)*. IEEE, 2024, pp. 1095–1100.

[19] O. et al., "Gpt-4 technical report," 2024, accessed: March 11, 2025. [Online]. Available: https://arxiv.org/abs/2303.08774

[20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019. [Online]. Available: https://arxiv.org/abs/1810.04805

[21] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023. [Online]. Available: https://arxiv.org/abs/2302.13971

[22] OpenAI, "Hello gpt-4o," 2024, accessed: March 11, 2025. [Online]. Available: https://openai.com/index/hello-gpt-4o/

[23] O. et al., "Gpt-4o system card," 2024, accessed: March 11, 2025. [Online]. Available: https://arxiv.org/abs/2410.21276

[24] OpenAI, "GPT-5.1 model card," https://platform.openai.com/docs/models/gpt-5.1, 2025, accessed: 2025-11-18.

[25] OpenAI, "Gpt-5 system card," OpenAI, Tech. Rep., Aug. 2025, published August 13, 2025. [Online]. Available: https://www.openai.com

[26] S. Ouyang, J. M. Zhang, M. Harman, and M. Wang, "An empirical study of the non-determinism of chatgpt in code generation," *ACM Trans. Softw. Eng. Methodol.*, vol. 34, no. 2, pp. 42:1–42:28, 2025. [Online]. Available: https://doi.org/10.1145/3697010

[27] *scikit-learn 1.7.2 documentation*, scikit-learn Developers, 2025, accessed: 2025-01-05. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[28] S. Raschka, Y. H. Liu, and V. Mirjalili, *Machine Learning with PyTorch and Scikit-Learn: Develop machine learning and deep learning models with Python*. Packt Publishing Ltd, 2022.

[29] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'95. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1995, p. 1137–1143.

[30] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[31] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," in *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[33] K. Pei, Y. Cao, J. Yang, and S. Jana, "Deepxplore: Automated whitebox testing of deep learning systems," *Commun. ACM*, vol. 62, no. 11, p. 137?145, Oct. 2019. [Online]. Available: https://doi.org/10.1145/3361566

[34] J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, "Dlfuzz: Differential fuzzing testing of deep learning systems," in *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, pp. 739–743.

[35] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[36] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[38] Keras, "Vgg model by keras," https://keras.io/api/applications/vgg, accessed: 2022-08-29.

[39] S. Kang, R. Feldt, and S. Yoo, "Sinvad: Search-based image space navigation for dnn image classifier test input generation," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, pp. 521–528.

[40] I. Dunn, H. Pouget, D. Kroening, and T. Melham, "Exposing previously undetectable faults in deep neural networks," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2021. New York, NY, USA: Association for Computing Machinery, 2021, pp. 56–66. [Online]. Available: https://doi.org/10.1145/3460319.3464801

[41] V. Riccio and P. Tonella, "Model-based exploration of the frontier of behaviours for deep learning system testing," in *Proceedings of the ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE '20. Association for Computing Machinery, 2020, p. 13 pages.

[42] E. Peer, J. Vosgerau, and A. Acquisti, "Reputation as a sufficient condition for data quality on amazon mechanical turk," *Behavior Research Methods*, vol. 46, no. 4, pp. 1023–1031, Dec 2014. [Online]. Available: https://doi.org/10.3758/s13428-013-0434-y

[43] M. Maryam, M. Biagiola, A. Stocco, and V. Riccio, "Benchmarking generative ai models for deep learning test input generation," in *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2025, pp. 174–185.

[44] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[45] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[46] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3730–3738.

[47] OpenAI, "Openai images api guide," 2025, accessed: March 11, 2025. [Online]. Available: https://platform.openai.com/docs/guides/images?api-mode=chat

[48] M. Andersland, "Amharic llama and llava: Multimodal llms for low resource languages," 2024. [Online]. Available: https://arxiv.org/abs/2403.06354

[49] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, 1945.

[50] J. Cohen, *Statistical power analysis for the behavioral sciences*. Hillsdale, N.J: L. Erlbaum Associates, 1988.

[51] R. B. Abdessalem, S. Nejati, L. C. Briand, and T. Stifter, "Testing vision-based control systems using learnable evolutionary algorithms," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. ACM, 2018, pp. 1016–1026. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180160

[52] M. Zhang, Y. Zhang, L. Zhang, C. Liu, and S. Khurshid, "Deeproad: Gan-based metamorphic testing and input validation framework for autonomous driving systems," in *2018 33rd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2018, pp. 132–142.

[53] A. Gambi, M. Müller, and G. Fraser, "Automatically testing self-driving cars with search-based procedural content generation," in *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*. ACM, 2019, pp. 318–328. [Online]. Available: https://doi.org/10.1145/3293882.3330566

[54] Z. Aghababaeyan, M. Abdellatif, L. Briand, M. Bagherzadeh *et al.*, "Black-box testing of deep neural networks through test case diversity," *IEEE Transactions on Software Engineering*, vol. 49, no. 5, pp. 3182–3204, 2023.

[55] T. Zohdinasab, V. Riccio, and P. Tonella, "An empirical study on low- and high-level explanations of deep learning misbehaviours," in *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2023, pp. 1–11.

[56] T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, "Deephyperion: exploring the feature space of deep learning-based systems through illumination search," in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, pp. 79–90.

[57] ——, "Efficient and effective feature space exploration for testing deep learning systems," *ACM Transactions on Software Engineering and Methodology*.

[58] T. Zohdinasab, V. Riccio, and P. Tonella, "Automated feature extraction for testing deep learning systems through illumination search," *Software Testing, Verification and Reliability*, vol. 36, no. 1-2, p. e70019, 2026.

[59] L. Ma, F. Juefei-Xu, F. Zhang, J. Sun, M. Xue, B. Li, C. Chen, T. Su, L. Li, Y. Liu, J. Zhao, and Y. Wang, "Deepgauge: Multi-granularity testing criteria for deep learning systems," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, ser. ASE 2018. ACM, 2018, pp. 120–131. [Online]. Available: http://doi.acm.org/10.1145/3238147.3238202

[60] Y. Tian, K. Pei, S. Jana, and B. Ray, "Deeptest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, 2018, pp. 303–314.

[61] V. Riccio, N. Humbatova, G. Jahangirova, and P. Tonella, "Deepmetis: Augmenting a deep learning test set to increase its mutation score," in *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2021, pp. 355–367.

[62] D. Ghobari, M. H. Amini, S. Park, S. Nejati, M. Sabetzadeh *et al.*, "Test input validation for vision-based dl systems: An active learning approach," in *2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, 2025, pp. 630–640.