ML4ML: Automated Invariance Testing for Machine Learning Models

Zukang Liao  
University of Oxford  
zukang.liao@eng.ox.ac.uk

Pengfei Zhang  
Tencent AI R&D Centre  
joepfzhang@tencent.com

Min Chen  
University of Oxford  
min.chen@oerc.ox.ac.uk

Abstract—In machine learning (ML) workflows, determining the invariance qualities of an ML model is a common testing procedure. Traditionally, invariance qualities are evaluated using simple formula-based scores, e.g., accuracy. In this paper, we show that testing the invariance qualities of ML models may result in complex visual patterns that cannot be classified using simple formulas. In order to test ML models by analyzing such visual patterns automatically using other ML models, we propose a systematic framework that is applicable to a variety of invariance qualities. We demonstrate the effectiveness and feasibility of the framework by developing ML4ML models (assessors) for determining rotation-, brightness-, and size-variances of a collection of neural networks. Our testing results show that the trained ML4ML assessors can perform such analytical tasks with sufficient accuracy.

Index Terms—Machine Learning Testing, Invariance Testing, ML4ML, Testing Framework

I. INTRODUCTION

In many applications, it is desirable for machine-learned models to be invariant in relation to the variation of some aspects of input data. Especially in the field of computer vision, much effort has been made to improve rotation invariance, brightness invariance, size invariance, and other types of invariance, e.g., [4]. Given a machine-learned model, manually determining whether the model is $T$-invariant, where $T$ is a type of invariance property such as rotation, can be a data-intensive and time-consuming process, which typically leads to inconsistent judgement. Hence, it is highly desirable to standardize and automate such processes.

Invariance testing has been an important part of robustness testing, and the level of invariance is traditionally measured using a relatively simple formula (e.g., [14], [15]). Because such a formula aggregates a huge amount of measured data quickly to a single score, it cannot fully encapsulate all variance-related patterns in the measured data. Later in Section IV we will demonstrate that some variance patterns that should have raised some concerns may sail through a formula-based assessment. Similar to many statistical measures (e.g., correlation index), visualization (e.g., a scatter plot) can help validate whether a statistical measure is indicative or does not meet the condition for its application (e.g., bivariate normal distribution for correlation estimation).

However, it is costly for human experts to inspect visual patterns in routine invariance testing in machine learning (ML) workflows, since there are different types of $T$-invariance qualities and one can collect variance measurement data in different structural locations in an ML model. It is thus highly desirable to automate such visual inspection using ML.

In this paper, we address the needs (1) for improving invariance testing by allowing more detailed inspection instead of simple formula-based scores, and (2) for automating such detailed inspection instead of relying on manual visual analysis. We propose to use machine learning for machine learning (ML4ML), i.e., by training classifiers or regressors (ML4ML assessors) to analyse the visual patterns resulting from invariance testing. Here we use the term “assessors” for ML4ML classifiers or regressors to avoid confusion with those ML models to be tested (the test candidates). The main contributions of this work are:

- We propose a technical framework for automated invariance testing, and we evaluate the framework for different types of invariance qualities.
- We use visual patterns generated at different locations using different statistical functions to show invariance qualities are more complex than formula-based scores.
- We collect a model repository that consists of over 600 models featuring different invariance properties for aiding the training of ML4ML assessors.
- We have made all the relevant models, data, and documentation available at the github [20].

II. RELATED WORK

Existing topics of machine learning (ML) testing include security, interpretability, fairness, efficiency, and so on. Zhang et al. provided a comprehensive survey on these topics [43]. In this section, we focus on invariance testing and the related robustness testing.

The invariance qualities of ML models have been studied for several decades [25]. One common approach to improving such quality is through augmentation [35]. Engstrom et al. [9] found that augmentation was crucial for improving the robustness of ML models since adversarial attacks may be carried out by rotating or translating inputs. Zhang [44] found that CNNs were not fully translation-invariant because MaxPooling layers were not optimal in terms of Nyquist sampling theory. However, in previous works, invariance qualities were mostly measured by accuracy-based formulas, e.g., [31], [41]. This work extends such evaluation by considering visual patterns generated at different positions of ML models.
Compared with gradient-based methods (e.g., [21], [36]), adversarial examples generated using affine transformation are more closely related to invariance testing. The works on neuron coverage, such as DeepXplore [30] and DeepGauge [23] are highly representative. TensorFuzz [27] introduced a property-based fuzzing procedure to enhance its convergence. DLFuzz [16] exploited a coverage-guided fuzzing approach to generate transformed adversarial examples. DeepHunter [40] introduced a frequency-aware seed selection strategy. These methods were able to find adversarial examples using affine transformations, while our framework focuses on evaluating how robust a model is under transformation(s).

Selective classification (or reject options) [1] and out-of-distribution detection [42] are also closely related to robustness testing. Werpachowski et al. [38] trained a reject function for each trained CNN, and they used either softmax response (confidence score) or uncertainty [13] to judge if the CNN is confident enough for a given data object \(x\). We adopt confidence scores as one of the testing criteria in our framework. The approach was typically for evaluating the likelihood of untrustworthy predictions (e.g., [32]), while our work focuses on evaluating invariance qualities of ML models as a whole.

Metamorphic testing [33] is also strongly related to invariance testing. Different types of mutations can be applied to permutation on input channels or the order of training/testing data [7] as well as model structures [24], [34]. Both metamorphic and invariance testing can be used to prevent unwanted deployment, although they serve different purposes for different properties.

III. DEFINITIONS AND MOTIVATION

In machine learning, the invariance quality of a trained model \(M\) is typically defined as the level of consistency when the model makes predictions. Consider a transformable attribute of input data objects, such as the rotation, brightness, or size of images. The more consistent the predictions of the model regardless of the variation of the attribute are, the better invariance quality the model has.

Let \(\mathbb{D}\) be the set of all possible data objects that could be the input of model \(M\), which is commonly referred to as the data space of \(M\). Let \(x\) and \(y\) be two arbitrary data objects in \(\mathbb{D}\). Let \(D \subset \mathbb{D}\) be a subset where all data objects are largely similar but vary in one particular aspect, denoted as variation \(T\). Consider every such subset, \(D_1, D_2, \ldots \) in \(\mathbb{D}\).

**Definition.** \(M\) is said to be \(T\)-invariant if for any pair of data objects both belonging to the same subset \(D_i (i = 1, 2, \ldots)\), i.e., \(\forall x, y \in D_i, \) we have \(M(x) = M(y)\).

When \(M\) is not \(T\)-invariant, ideally we can assess the level of inconsistency (or \(T\)-variance) with a function \(\Upsilon\):

\[
\text{Level}_T = \Upsilon(M, D_1, D_2, \ldots)
\]

Obtaining such an ideal measurement \(\text{Level}_T\) is almost impossible because (i) it is difficult to identify all subsets \(D_1, D_2, \ldots\) and obtain all data objects in such a subset; (ii) the definition of “largely similar” is imprecise and subjective; and (iii) there is no ground truth function for \(\Upsilon\).

Hence the level of \(T\)-variance of a model is commonly estimated by using a simplified process. One such example is to use a family of transformations \(T\) to simulate variation \(T\) [9]. One obtains a testing dataset \(X \subset \mathbb{D}\), and for each data object \(x_i \in X (i = 1, 2, \ldots, n)\), one applies different transformation \(t_j \in T (j = 1, 2, \ldots, n)\) to \(x_i\) to create a set of data objects:

\[
\tilde{D}_i = \{d_{i,0}, d_{i,1}, \ldots, d_{i,n}\} = \{t_0(x_i), t_1(x_i), \ldots, t_n(x_i)\}
\]

One can define a function \(f\) to measure the inconsistency between two predictions of model \(M\). With \(f\) one may approximate \(\Upsilon\) with the following simplistic function \(\Phi\):

\[
\Phi(M, X) = \frac{1}{m} \sum_{i=1}^{m} \prod_{j=1}^{n} f(M(x_i), M(t_j(x_i)))
\]

When \(f = 1\), \(x_i\) and \(t_j(x_i)\) are consistently classified, i.e., \(M(x_i) = M(t_j(x_i))\), and when \(f = 0\) the contrary. Because Gao et al. [14] also associated consistency with correctness, i.e., \(M(x_i) = M(t_j(x_i)) = \text{correct prediction}\), they referred such a \(\Phi\) to as robust accuracy. Eq. (1) is served as a simple baseline in this work.

As mentioned earlier, this is not a ground truth function, and \(\Phi\) can be defined differently, e.g., using a mean function instead of the product in Eq. (1). In this work, we made use of the simulation approach based on a family of transformations \(T\). While exploring more general ways of defining the function \(f\), e.g., different statistical functions, we utilise ML techniques to obtain the overall function \(\Phi\). Using ML techniques to aid the evaluation of the invariance quality of ML models allows us to capture and encode complex knowledge about multivariate invariance assessment.

IV. METHODOLOGY: TESTING FRAMEWORK

As shown in Fig. 1, our ML4ML framework for invariance testing includes two workflows: a development workflow and a deployment workflow. In the former workflow (lower part of Fig. 1), one collects a repository of models and uses them to train ML4ML assessors for invariance testing. For each type of invariance \(T\), this workflow will yield one assessment model or an ensemble of models, which are to be used in the latter workflow (upper part of Fig. 1).

A. Deployment Workflow

a) Transformations and Sampling Variable.: Let us first describe the deployment of an ML4ML assessor for testing a type of invariance \(T\). Given an ML model \(M\) and a testing dataset \(X\), we first define a family of transformation functions \(T = \{t_0, t_1, \ldots, t_n\}\) to simulate the variations for testing \(T\). We use an independent variable \(V\) to characterize such variations, e.g., the rotation degree of an image for testing rotation-invariance or the average image brightness for testing brightness-invariance. When \(V = 0\), the corresponding \(t(0)\) is an identity transformation. We can thus order the functions in \(T\) incrementally according to \(V\).

We recommend to define \(T\) to enable the sampling of \(V\) in a fixed interval \([-\alpha, \alpha]\) with \(2k + 1\) instances (\(k \in \mathbb{N}^+\)).
Hence $n = 2k$ and $t_k() = t_{n/2}()$ is an identity transformation. For example, to simulate rotation variations, one may create a family of transformations $T_r$ corresponding to:

$$V_r = \{ -90^\circ, -87^\circ, \ldots, -3^\circ, 0^\circ, 3^\circ, \ldots, 87^\circ, 90^\circ \}$$

To simulate brightness variations, one may create a family of transformations $T_b$ corresponding to:

$$V_b = \{-25\%, -24\%, \ldots, -1\%, 0\%, 1\%, \ldots, 24\%, 25\% \}$$

Note that $T_r$ may feature other variations in addition to rotation degrees (e.g., image cropping or background filling), and $T_b$ may feature other variations in addition to brightness changes (e.g., contrast normalization or color balancing). These additional adjustments are usually there to minimize the confounding effects that may be caused by the primary variation $V_r$ or $V_b$. If one wishes to test the invariance quality related to any of these additional variation qualities, one can specify a new type $T_{new}$, define a new set of transformations $T_{new}$, and a new sampling variable $V_{new}$.

b) Modality: As described earlier, with the defined transformations $T$, we can create a testing data set $D_t$ for each data object $x_i$ in the original testing dataset $X$. We can then measure the impact of the transformations $t_j (j = 0, 1, \ldots, n)$ upon $x_i$. Eq. [1] used a simple function $f$ to measure the difference between the model predictions based on $x_i$ and $t_j(x_i)$ respectively. Inspired by medical imaging [26], we replace the simple $f$ with different statistical functions. And we also replace the “summation on production”, i.e., $\sum_j \prod_i$, operation in Eq. [1] with a different cumulative function. We refer a combination of a statistical function and cumulative function to as a *modality*. Additionally, we would like to observe not only the output of a model $M$ but also the internal status of $M$ in response to $x_i$ and $t_j(x_i)$, which are referred to as different *modalities* at different *locations*.

We thus extend the definition of $f$. Consider the data flowing from an input $x \in X$, through a model $M$, to the final prediction of $M$. We can measure the signals at an internal position $p$ of $M$ or its output layer. We define $f$ that maps a set of position-sensitive signals to a measurement. Let $S$ be a set of signals at a position of $M$, such as the output vector/tensor at a layer in a CNN model, a feature vector handled by a decision-tree model, or a confidence vector generated by a classifier. $S$ is thus sensitive to the model $M$, a specific position $p$, and a specific input data object $x$. We denote this as $S(M, p, x)$.

Given two different input data objects $x$ and $y$, we can obtain two signal sets $S(M, p, x)$ and $S(M, p, y)$. We can measure the difference between $S(M, p, x)$ and $S(M, p, y)$ using a difference function:

$$\text{dif}(S(M, p, x), S(M, p, y))$$

Because $S$ is a set of signals, dif() is more complicated than the simple $f()$ function used in Eq. [1]. Recall the notations of an original testing dataset $X$, a set of transformations $T$ and its corresponding sampling variable $V$. We can use dif() to measure the difference between any two derived testing data objects $t_i(x)$ and $t_j(x)$ as:

$$\text{dif}(S(M, p, t_i(x)), S(M, p, t_j(x)))$$

(2)
Let us rewrite $S(M, p, t_j(x))$ as $S_i = \{s_i, t_j(x)\}$ and $S(M, p, t_j(x))$ as $S_j = \{s_j, t_j(x)\}$. The difference function can be as simple as $\text{dif}_a = \max(S_i) - \max(S_j)$ or $\text{dif}_b = \text{mean}(S_i) - \text{mean}(S_j)$, or as complex as a Minkowski distance metric or other difference functions.

For all $x_k \in X$, we can measure the cumulative effect as:

$$\delta_{i,j} = \text{cumul}_{x_k \in X}(\text{dif}(S(M, p, t_i(x_k)), S(M, p, t_j(x_k))))$$

The functions $\text{dif()}$ and $\text{cumul()}$ define the essence of a mapping from a set of signals detected at a position $p$ of a model $M$ to a measurement, and can thus be considered as modalities in analogy to medical imaging [26]. We refer a combination of $\text{dif}()$ and $\text{cumul}()$ as a modality.

In this work, we use the same cumul function for all our modalities. When there is no confusion which model $M$ and which testing position $p$ we are referring to, we denote Eq. 2 as $\text{dif}_{i,j}()$. The function $\text{cumul}$ is thus defined as:

$$\delta_{i,j} = \text{cumul}_{x_k \in X} = \sqrt{\mathbb{E}_{x_k \in X} \left\{[\text{dif}_{i,j}(x_k)]^2\right\}}$$

where $\mathbb{E}$ is the expected value. Using standard concentration inequalities, such as the Hoeffding or Bernstein bounds [37], we can show that with this cumul function, $\delta_{i,j}$ concentrates around the true expected value $\delta_{i,j}$ unless there are extreme values or numerical issues. The detailed proof can be found in Appendix D on our github repository for this work [20].

c) Variance Matrix.: When we measure $\delta_{i,j}$ for all pairs of transformations in $T = \{t_0, t_1, \ldots, t_n\}$, we can obtain a variance matrix:

$$\Delta(M, T, X, p, \text{dif}, \text{cumul}) = \begin{bmatrix}
\delta_{n,0} & \delta_{n,1} & \cdots & 0 \\
\vdots & \ddots & \vdots & \vdots \\
\delta_{1,0} & 0 & \cdots & \delta_{1,n} \\
0 & \delta_{0,1} & \cdots & \delta_{0,n}
\end{bmatrix}$$

where the elements in the matrix are ordered according to the sampling variable $V$ horizontally and vertically. In Fig. 1, we use $f_1, f_2, \ldots$ to denote different modalities ($\text{dif}$, $\text{cumul}$), and use $F$ to denote a set of modalities. When different modalities are used to probe different positions of a model, we can obtain a collection of variance matrices like multi-modality in medical imaging [19].

The variance matrices can indeed be viewed as images. We thus intentionally place the matrix element $\delta_{0,0}$ at the bottom-left of the matrix to be consistent with the directions of coordinate axes. Fig. 2 shows six examples of such images, which are referred to as variance matrices.

d) Image Analysis.: One can observe a variety of imagery patterns from the variance matrices as illustrated in Fig. 2. Each $\delta_{i,j}$ value is encoded using a colormap with dark blue for little cumulative difference and yellow for greater difference. The typical patterns that one may observe include:

- **Incremental Direction.** By definition, the matrix cells on the diagonal lines, $\delta_{i,i}$, should exhibit no difference. As both axes of the matrix correspond to the range $[-\alpha, \alpha]$ of an independent variable $V$ that characterizes a type of variance $T$, one may anticipate that the amount of inconsistency will normally increase from the matrix center $(0,0)$ towards the cells furthest away from the diagonal lines, i.e., $(-\alpha, \alpha)$ and $(\alpha, -\alpha)$. Examples (a), (b), and (e) in Fig. 2 exhibit the expected incremental direction, while (c), (d), and (f) reveal less expected patterns.

- **Incremental Rate.** The colors depict the rate of increasing difference effectively. Among the three expected patterns, (b) shows a slow increment and (e) shows a rapid increment, while (a) is somewhere in the middle.

- **Transitional Smoothness.** While the transitional patterns in (a), (b), (d), and (e) are smooth, those in (c) and (f) show some irregularities that attract viewers’ attention.
Four example features are given in Table II. From the images, we can easily observe different visual patterns. We extract 80 features from each variance matrix.

Table II

| Measurements               | Model 1 | Model 2 | Model 8 | Model 12 | Model 13 | Model 17 | Model 35 | Model 61 | Model 73 | Model 85 | Model 96 | Model 264 |
|----------------------------|---------|---------|---------|----------|----------|----------|----------|----------|----------|----------|----------|-----------|
| Max@CONF: Square Mean      | 0.015   | 0.013   | 0.014   | 0.015    | 0.012    | 0.007    | 0.010    | 0.013    | 0.016    | 0.008    | 0.011    | 0.014     |
| Max@CONV-1: Gradient score | 0.594   | 0.819   | 0.897   | 0.991    | 0.939    | 1.028    | 0.928    | 1.072    | 1.032    | 1.002    | 0.853    | 0.942     |
| Max@CONV-2: Discontinuity  | 0.936   | 0.810   | 0.878   | 1.401    | 0.929    | 1.345    | 0.944    | 1.456    | 2.416    | 2.059    | 0.758    | 1.508     |
| Mean@CONV-1: Asymmetry     | 0.047   | 0.009   | 0.005   | 0.014    | 0.007    | 0.012    | 0.008    | 0.009    | 0.041    | 0.024    | 0.048    | 0.012     |
| 'Labels' for invariance    | ✔️      | ✔️      | ✔️      | ✔️       | ✔️       | ✔️       | ✔️       | ✔️       | ✔️       | ✔️       | ✔️       | ✔️        |

- **Irregularity:** Sub-domain Partition. There can be many types of irregularities. Example (c) shows one type of irregularity, which may suggest that the model under testing has different behaviors in different sub-domains of \([-\alpha, \alpha]\) \([-\alpha, \alpha]\). In this case, positive and negative variations of \(V\) impact the model differently.

- **Irregularity:** dot-patterns. Example (d) shows relatively fast increments (from the center of the matrix) towards the centers of the four edges, i.e., \((0, \pm \alpha)\) and \((\pm \alpha, 0)\). This indicates that the inconsistency between 0 and \(\pm \alpha\) is greater than the inconsistency between \(+\alpha\) and \(-\alpha\).

- **Irregularity:** Abrupt Transition. Example (f) shows one weak and one strong abrupt pattern. The strong pattern (shown in yellow) suggests that the testing may be affected by a specific transformation with an adversary effect, or the model being tested cannot handle certain input conditions very well. The weak pattern (shown in cyan) suggests the possibility of combined irregularities of sub-domain partition and dot-patterns.

All the different visual patterns mentioned above show that a simple formula-based score does not suffice to approximate invariance qualities. In Fig. 3, we also show twelve models that have similar robust accuracy on the interval of \([-15^\circ, 15^\circ]\) \(\sim 66.5\% \pm 1.5\%\), however, their variance matrices are all different. Therefore, we propose to use the aforementioned visual patterns as a general guideline to provide invariance annotations.

In many fields, e.g., facial action units [22], natural language processing (NLP) [17], segmentation [5], etc, annotations are not always accurate or reliable. When there is no "clean ground truth" for evaluation, we adopt majority votes as the "pseudo ground truth" which was proven to be superior than individual annotators by [3] using differential comparison. As suggested by [8], when providing new research approaches/annotations, we use two inter-rater reliability (IRR) measures, i.e., Cohen’s kappa [2] and Fleiss’ kappa [10], to evaluate how much confidence we can have on our invariance labels.

We consider that the level of \(T\)-invariance of ML models should be assessed as a generic quality measurement, while the requirements for invariance quality may be application-dependent. For example, given two ML models for object detection, the annotations for \(T\)-invariance should be a score consistently regardless of their intended applications. Meanwhile, different applications, e.g., detecting vegetables in photos and detecting humans in CCTV imagery, likely require different levels of rotation- or brightness-invariance.

![Fig. 3. Five variance matrices for twelve models with almost the same robust accuracy were arbitrarily selected from the repository of 600 CNNs. All twelve models were trained on CIFAR10. The transformations simulate rotation variance, in the range of \([-15^\circ, 15^\circ]\) \(\pm 1^\circ\). The metadata of these models can be found in Table 6 in the Appendix C [20]. From the images, we can easily observe different visual patterns. We extract 80 features from each variance matrix. Four example features are given in Table II.](image-url)
B. Development Workflow

In order to avoid manually evaluating invariance qualities, we proposed ML4ML where we train an ML4ML assessor to automatically perform $T$-invariance assessment. For a given simulation of $T$, to train an ML4ML assessor, we first need to collect “training data”, which in our case are the variance matrices captured using a group of modalities $F$ at different locations. In Fig. 1 variance matrices are labelled with the prefix “TF-” to emphasize the dependency of the transformations on the testing data (i.e., the simulation) and modalities used in the workflow.

Because the ML4ML assessor for $T$-invariance is application-independent, we can use different ML models and different versions of them to generate training data, i.e., variance matrices, as long as each variance matrix is clearly identified with its modality and position. An ML4ML assessor is normally trained with data sharing the same modality and position properties, though future ML techniques may enable learning from heterogeneous training data. In most serious machine learning workflows, e.g., in academic research and industrial development, many models or many versions of a model are being trained and tested. Hence, gathering a repository of variance matrices from different workflows collectively will be attainable in the longer term. For this work, we demonstrate the feasibility of the proposed framework using a small repository of 600 CNNs.

a) Decision Tree Learning.: We use techniques in the decision tree (DT) family to train our ML4ML assessors, because (1) we have a relatively smaller model repository, for which DT techniques are more suitable; (2) invariance quality evaluation needs to be explainable and DT models are more transparent and understandable; and (3) annotations of invariance qualities may include noise due to subjectivity, bagging methods often achieve superior results \cite{11, 13}.

Given a family of transformations $T = \{t_0, t_1, \ldots, t_n\}$, a set of $m$ modalities $F = \{f_1, f_2, \ldots, f_m\}$ and a set of $l$ positions $P = \{p_1, p_2, \ldots, p_l\}$, there are $m \cdot l$ variance matrices $\Delta$, each of $(n+1)^2$ dimensions. The goal of an ML4ML assessor $Q$ is to take the $m \cdot l$ variance matrices as input and predict the level of invariance quality.

We use binary labels (0: invariant, 1: variant) to annotate models in a model repository $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots\}$. Since $Q$ is trained with such labelled data, it produces a binary class label $\in \{0, 1\}$ normally, or a real value $\in [0, 1]$ if it is an ensemble or regression model. Note that the formula in Eq. 1 by \cite{13} defines 1 as fully invariant.

b) Features in Variance Matrices.: When the DT techniques and regression method are applied to variance matrices, it is necessary to extract features from these matrices. Given an $(n+1) \times (n+1)$ variance matrix $\Delta = [\delta_{i,j}] (i, j \in [0, n])$, a feature is a variable that characterizes one aspect of the matrix. Similar to many ML solutions in medical image analysis, we adopt the methodology of “a bag of features” by defining a collection of features for characterizing different aspects of a variance matrix, including but not limited to gradient, asymmetry, continuity, etc (details in the Appendix A \cite{20}).

Each feature extraction algorithm is thus a function $h : \mathbb{R}^{(n+1) \times (n+1)} \rightarrow \mathbb{R}$. In this work, we have defined a total of 16 features (see the Appendix A \cite{20}) for each variance matrix. The $m \cdot l$ variance matrices thus yield $16m \cdot l$ features. For example, Fig. 3 shows five variance matrices for each of the 12 models in the model repository. Two modalities (labelled as Max and Mean) and three positions (CONF, CONV-1, and CONV-2) are shown in the figure, from which we can observe different visual features. Metadata about these 12 models can be found in Table 6 in Appendix C \cite{20}. Such metadata is not used in training any ML4ML assessor.

Table III shows four types of feature measurements obtained from some of these images, together with the labels of invariance quality. In each row, there are one or a few numbers (in bold) that are higher than most of the others, indicating the ability of a feature to differentiate some anomalies.

V. Experimental Results and Analysis

a) Model Repository.: To build a model repository, we trained 600 CNNs under different settings. The types of variations of the settings include: (a) CNN structures (VGG13bn and CNN5); (b) Training datasets (CIFAR10 and MNIST); (c) Normalization of pixel values \([-0.5, 0.5], [0, 1]\); (d) Learning rate \(e^{-2}, e^{-3}, e^{-4}, e^{-5}\); (e) Number of epochs \(\in [1, 2000]\); Batch size \(\in \{4, 8, 1024, 16384\}\); (f) Optimizers (Adam or SGD); (g) Anomalies \(\text{none, data leakage}\); (h) Training with adversarial attacks \(\text{none, FGSM}\); (i) Hardware (CPU, GPU); (j) Added variance \(\text{none, rotation, scaling, brightness}\); (k) Transformation ranges \(\text{none, different }[-\alpha, \alpha]\); and (l) some minor variations.

The majority of the models were trained using Tesla V100 32GB. When they were used for training or testing ML4ML assessors, the experiments were run on a single laptop (Apple M1 Chip) without GPU. In Appendix B \cite{20}, we provide further details on the training parameters, while in Appendix C \cite{20} we describe the structure of CNN5 and the metadata of the models trained with this structure.

b) Invariance labels.: To assess the quality of the data labels used for training, we measured the inter-rater reliability (IRR) of the acquired data labels. As shown in Table III the IRR scores (i.e., the Cohen’s and Fleiss’ kappa scores) are around 0.76 to 0.82, which are similar to the quality of the annotations measured in a few NLP applications \cite{3}.

c) Training and Testing ML4ML Assessors.: We trained three ML4ML assessors using the techniques in the decision tree family, including decision trees, random forests, and AdaBoost. We also trained an ML4ML assessor using linear regression. We used the scikit-learn libraries \cite{29} for the implementation. We set all the hyper-parameters of the
assessors, e.g., depth, to default values without tuning. For regressors, we set the threshold to 0.5 without further tuning.

In addition, we implemented a numerical assessor using Eq. [1] as the baseline for comparison. We used greedy search to find the threshold on the training data.

To train each ML4ML assessor, we selected a relatively balanced subset of “variant” and “invariant” models according to the type of invariance $I$ that the assessor was targeted for. This selection process resulted in four different development workflows denoted in Table IV as:

- $W_1$: Rotation with VGG13bn (models) on CIFAR10,
- $W_2$: Rotation with CNN5 (models) on MNIST,
- $W_3$: Brightness with VGG13bn (models) on CIFAR10,
- $W_4$: Scaling with VGG13bn (models) on CIFAR10.

Details on the setup of training and testing the ML4ML assessors of the four workflows $W_1 \sim W_4$ can be found in Appendix C [20] and an ablation study in Appendix E [20].

d) Variance Matrices: To create variance matrices for the models, we used four families of 31 transformations. The ranges of these transformations and the fixed intervals are:

- $W_1, W_2$: $[-15^\circ, 15^\circ] \pm 1^\circ$,
- $W_3, W_4$: $[-30\%, +30\%]$ or $[\times 70\%, \times 130\%] \pm 2\%$.

For the 600 CNNs, we select three interested positions: at confidence score level (CONF), and at the last two convolutional layers (CONV-1 and CONV-2). And we used two dif functions: $\max(S_x) - \max(S_y)$ and $\text{mean}(S_x) - \text{mean}(S_y)$. The former was applied to CONF, and both were applied to CONV-1 and CONV-2. Details on the selection of testing locations can be found in Appendix B [20].

Feature Extraction: For each variance matrix, we generated 16 feature measurements (a full list in Appendix A [20]), resulting in a $5 \times 16 = 80$ dimensional feature vector. Table 5 in Appendix A [20] shows that some feature measurements at the CONF level have a strong correlation with the robust accuracy defined in Eq. [1]. However, as shown in Fig. 3, it does not suffice to use robust accuracy only to approximate invariance qualities. Therefore we extracted multiple feature measurements at different locations to facilitate the training of ML4ML assessor(s), and thus replace the simple formula-based accuracy score defined in Eq. [1].

e) Result Analysis: In each of four development workflows, $W_1 \sim W_4$, we trained four ML4ML assessors, together with a numerical assessor as the baseline for comparison. The performances of these assessors are shown in Table IV from which we can make the following observations:

- The three DT assessors performed noticeably better than linear regression and the numerical assessor (Eq. [1]). The only exception is perhaps with workflow Rotation with CNN5 on MNIST, where the numerical assessor performed similarly to the three DT assessors.
- Random forest consistently performs better than AdaBoost. This is aligned with the observation in [3], [11], [12] where bagging methods were shown to counteract noisy annotations, while AdaBoost tends to place larger weights on noisy or mislabelled instances.
- The results of CNN5 on MNIST are not as good as other workflows, except for the numerical assessor. Further investigation will be necessary to see if the main factors reside with the CNN5 structure, the MNIST dataset, or the training of the ML4MMS assessors.

The results in Table IV allow us to draw the following conclusions: (i) Our feature selection was effective, otherwise, the DT assessors would not perform notably better than others; (ii) The notion of variance matrices may be generally applicable to different types of variance, otherwise, the results could vary more dramatically among different variance testing; (ii) The size of our model-repository is relatively small, and the statistics (e.g., entropy) is less accurate. Although random forest and boosting can help, it is desirable to continue expanding the model repository.

VI. CONCLUSIONS

In this work, we propose a novel framework for evaluating the invariance qualities of ML models. This framework can be used to automate the testing procedure, replace labour-intensive analysis, and prevent unwanted deployment of the ML models.

We use visualisation to show that simple formula-based scores, e.g., robust accuracy, does not suffice to approximate invariance qualities. The proposed framework enables us to evaluate invariance qualities from different perspectives, i.e., different modalities, testing positions and visual patterns.

We also show the inter-rater reliability scores ($\sim 0.76-0.82$) of invariance annotations, which would be considered between “Good” and “Excellent” in typical NLP applications [8].

Our experimentation on three different types of invariance qualities with 600 CNN models has confirmed the usability and feasibility of the proposed technical framework, as the three ML4ML assessors based on DT techniques can achieve around $83 \sim 95\%$ accuracy, noticeably better than linear regression and the robust accuracy.

| Classifier/Regressor | Transformation | Rotation with VGG13bn on CIFAR10 | Brightness with CNN5 on MNIST | Scaling with VGG13bn on CIFAR10 |
|----------------------|----------------|----------------------------------|-------------------------------|---------------------------------|
| Decision Tree        | W1             | 84.73\%±4.03\%                  | 82.93\%±3.73\%               | 91.33\%±9.19\%                 |
| Random Forest        | W2             | 91.31\%±2.02\%                  | 87.66\%±1.02\%               | 94.66\%±1.13\%                 |
| AdaBoost             | W3             | 89.90\%±4.62\%                  | 85.40\%±2.26\%               | 93.86\%±3.33\%                 |
| Linear Regression    | W4             | 75.46\%±3.13\%                  | 68.26\%±2.93\%               | 78.40\%±3.20\%                 |

All ML4ML results are averaged on ten repeated experiments. For each experiment, the standard 3-fold cross-evaluation is used.

TABLE IV COMPARING THE PERFORMANCE (ACCURACY) OF ML4ML ASSESSORS AND A NUMERICAL ASSSESSOR IN FOUR DEVELOPMENT WORKFLOWS.
To provide abstraction of each variance matrix, we consider 16 types of feature measurements. All ML models involve abstraction. Feature engineering is an effective way for introducing human analytical knowledge to ML models, while data labelling provides spontaneous human knowledge to models. Both can have biases and be costly. The former is usually provided by one or a few experts, while the latter is usually provided by many less-skilled data annotators. Learning only from labelled knowledge usually requires very large training datasets. In ML, feature engineering can reduce such demand, allowing models to be trained with relatively small training datasets.

a) 1. Squared Value Mean (all elements). The mean of the squared values of all elements in the variance matrix.

\[
\phi_{\text{svm}} = \frac{1}{2(n+1)} \sum_{i=0}^{n} \sum_{j=0}^{n} \delta_{i,j}^2
\]

b) 2. Mean (“meaningful” elements). The mean value of all “meaningful” elements in the variance matrix (i.e., the upper left part) after excluding the duplication and the diagonal line (self-comparison).

\[
\phi_{\text{mean}} = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=0}^{i-1} \delta_{i,j}
\]

c) 3. Standard Deviation. The standard deviation of all “meaningful” elements considered in (2) in the variance matrix (i.e., the upper left part).

\[
\phi_{\text{std}}^2 = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=0}^{i-1} (\delta_{i,j} - \phi_{\text{mean}})^2
\]

d) 4. Amount of Significant Variance. Given a threshold value \(\tau\), any value \(\delta_{i,j}\) (pairwise variance) is considered to be undesirable or significant if \(\delta_{i,j} > \tau\). In this work, we set \(\tau = 0.15\).

\[
\nu_{i,j} = \begin{cases} 1 & \text{if } \delta_{i,j} > \tau \\ 0 & \text{if } \delta_{i,j} \leq \tau \end{cases}
\]

\[
\phi_{\text{asv}} = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=0}^{i-1} \nu_{i,j}
\]

e) 5. Sensitivity. Sensitivity level to the size of testing dataset. Normally the value of \(\delta_{i,j}\), i.e.,

\[
\delta_{i,j} = \text{cumul}_{x_k \in X} \frac{E_{x_k \in X} \{ (\text{diff}_{i,j}(x_k))^2 \}}{\text{size}_{x_k \in X}}
\]

is obtained using 100% of the data objects in \(x_k \in X\). Here we denote \(\delta_{i,j}^r\) as being obtained using \(r\%\) of \(X\) (0 < \(r < 100\)). In this work, we set \(r = 90\).

\[
\phi_{\text{asy}} = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=0}^{i-1} (\delta_{i,j} - \delta_{i,j}^r)^2
\]

f) 6. Horizontal Gradient (Mean). The mean value of the gradient values of the variance matrices in the horizontal direction.

\[
e_{i,j} = \delta_{i,j+1} - \delta_{i,j} \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, i
\]

\[
\phi_{\text{hg-mean}} = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=1}^{i} e_{i,j}
\]

when computing all gradient measurements, the primary diagonal of the variance matrix is filled in using the average value of its neighbour(s).

g) 7. Horizontal Gradient (Standard deviation). The standard deviation value of the gradient values of the variance matrices in the horizontal direction.

\[
\phi_{\text{hg-std}} = \frac{2}{n(n+1)} \sum_{i=1}^{n} \sum_{j=1}^{i} (e_{i,j} - \phi_{\text{hg-mean}})^2
\]

h) 8. Horizontal Gradient (row-based standard deviation). The averaged standard deviation value of each column (valid elements) of the gradient map \([e_{i,j}]\).

\[
\mu_i = \frac{1}{n} \sum_{j=1}^{i} e_{i,j} \quad i = 1, 2, \ldots, n
\]

\[
\phi_{\text{hg-rad}} = \frac{1}{n} \sum_{i=1}^{n} \sqrt{\frac{1}{i} \sum_{j=1}^{i} (e_{i,j} - \mu_i)^2}
\]

i) 9. Vertical Gradient (Mean). The mean value of the gradient values of the variance matrices in the vertical direction.

\[
e_{i,j} = \delta_{i+1,j} - \delta_{i,j} \quad j = 0, 1, \ldots, n-1; i = j, \ldots, n\]

\[
\phi_{\text{vg-mean}} = \frac{2}{n(n+1)} \sum_{i=0}^{n-1} \sum_{j=0}^{i} e_{i,j}
\]

j) 10. Vertical Gradient (Standard deviation). The standard deviation value of the gradient values of the variance matrices in the vertical direction.

\[
\phi_{\text{vg-std}} = \frac{2}{n(n+1)} \sum_{i=0}^{n-1} \sum_{j=0}^{i} (e_{i,j} - \phi_{\text{vg-mean}})^2
\]

k) 11. Vertical Gradient (Column-based standard deviation). The averaged standard deviation value of each row (valid elements) of the gradient map \([e_{i,j}]\).

\[
\mu_j = \frac{1}{n-j} \sum_{i=j}^{n-1} e_{i,j} \quad j = 0, 1, \ldots, n-1
\]

\[
\phi_{\text{vg-rad}} = \frac{1}{n} \sum_{j=0}^{n-1} \sqrt{\frac{1}{n-j} \sum_{i=j}^{n-1} (e_{i,j} - \mu_j)^2}
\]

l) 12. Diagonal Gradient (Mean). The mean value of the gradient values of the variance matrices in the diagonal direction.

\[
e_{i,j} = \delta_{i+1,j+1} - \delta_{i,j} \quad i = 1, 2, \ldots, n-1; j = 1, 2, \ldots, n-1
\]

\[
\phi_{\text{dg-mean}} = \frac{2}{n(n+1)} \sum_{i=1}^{n-1} \sum_{j=1}^{i} e_{i,j}
\]
m) 13. Diagonal Gradient (Standard deviation): The standard deviation value of the gradient values of the variance matrices in the diagonal direction.

\[
\phi_{dg-std}^2 = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=1}^{i} (e_{i,j}^g - \phi_{dg-mean})^2
\]

n) 14. Overall Gradient.: The averaged ratio of the mean and the standard deviation value of the gradient in horizontal/vertical/diagonal direction.

\[
\phi_{G-overall} = (\phi_{hg-mean} / \phi_{hg-std} + \phi_{vg-mean} / \phi_{vg-std} + \phi_{dg-mean} / \phi_{dg-std}) / 3
\]

o) 15. Discontinuity.: The discontinuity (flatness) value of the variance matrix.

\[
\mu_r = \sum_{j=0}^{n-r} \frac{\delta_{i+j+r}}{n-r+1} \quad r = 1, 2, \ldots, n - 1
\]

\[
\phi_{dctny} = \sum_{r=1}^{n-1} \sum_{j=0}^{n-r} \frac{(\delta_{i+j+r} - \mu_r)^2}{\phi_{mean}}
\]

p) 16. Asymmetry.: The asymmetry (about the second diagonal) value of the variance matrix.

\[
\phi_{asymm} = \sum_{i=1}^{n} \sum_{j=0}^{i-1} |\delta_{i,j} - \delta_{n-j,n-i}| / \phi_{mean}
\]

**Proposition 1** When the \(\text{dif}_{i,j}()\) function can be written as \(f_i() - f_j()\), e.g., \(\max\) or \(\text{mean}\), the square value mean \(\phi_{sym}\) is proportional to the difference between the variance of \(f(y)\), where \(y \in T(X)\), and the covariance of \(f(z_i)\) and \(f(z_j)\), where the latent variables \(z_i = t_i(x)\) and \(z_j = t_j(x)\) (of the same data object \(x \in X\)) are identically and dependently distributed. \(T(X)\) consists of all transformed data objects.

**Proof:** Square Mean of the Matrix:

\[
\frac{1}{2\tau^2} \sum_i \sum_j \sum_j \delta_{i,j}^2
\]

\[
= \frac{1}{2\tau^2} \sum_i \sum_j \sum_j \delta_{i,j}^2
\]

\[
= \frac{1}{\tau} \sum_i \sum_j \text{E}_X[f_i^2(x) + f_j^2(x) - 2f_i(x)f_j(x)]
\]

\[
\approx \sum_{i, j} \text{E}_{X \in T(X)}[f_i^2(y)] - \text{E}_{X \in T(X)}[f_i(z_i)] \cdot \text{E}_{X \in T(X)}[f_j(z_j)] - \text{COV}_{X \in T(X)}[f_i(z_i), f_j(z_j)]
\]

\[
= \text{VAR}[f(y)] - \text{COV}[f_i(z_i), f_j(z_j)]
\]

where we let \(\tau = n+1\), and the expected value of \(f(y), f(z_i),\) and \(f(z_j)\) are the same. \(\text{COV}\) stands for covariance and \(\text{VAR}\) stands for variance.

Table VII shows the correlation between the accuracy evaluation \([14]\) and our measurements. At the confidence score level (CONF), \(\phi_{asym}, \phi_{dctny}\) and \(\phi_{G-overall}\) have a strong correlation to robust accuracy.

**TABLE V**

| Square Mean | Discontinuity | Gradient Overall |
|-------------|---------------|------------------|
| RobustAcc   | 1.13          | 0.03             |
| Accuracy    | -0.12         | -0.03            |

Fig. 4. Rotation variance matrices mean@CONF-1 from a VGG13bn trained on CIFAR10 (model ID: 1). (a) abnormal data causes the variance matrix to be “jagged”. (b) Abnormal data lead to extreme activation values which are far away from the distribution. (c) After cleaning, the patterns become normal.

**VIII. LABELLING CONVENTION**

For each model, we provide the following hyper-parameters:

- CNN structures (VGG13bn and CNN5).
- Weight initialisation: (default random initialisation, or pre-trained on ImageNet [6]), weights are loaded by Pytorch [23].
- Databases the CNNs are trained on: (CIFAR10, or MNIST).
- Different pre-processing: 1) Normalise pixel values to [-0.5, 0.5]. 2) Normalise pixel values to [0, 1].
- Learning rate \((e^{-1}, e^{-2}, e^{-3}, e^{-4}, e^{-5})\).
- Number of epochs (varying from 1 epoch to 2000 epochs).
- Batch size (4, 8, 1024 and 16384).
- Optimizer (e.g., Adam [18] or stochastic gradient descent(SGD)).
- Some models include anomalies (noisy data, data leakage etc).
- Whether the models are trained using gradient-based adversarial attacks (FGSM [36]).
- Models trained on CPU or GPU.
- Different types of the transformation set \(T\): rotation, scaling and brightness.
- Different range \([-\alpha, \alpha]\) of the transformation set \(T\); e.g., \(\alpha = 5^\circ, 10^\circ\), etc.

Additionally, we also provide the loss functions curve on both training and testing set. The assessment labels are provided by three professional researchers/scientists who have minimum seven-year experience in both academia and industry. Note that, the labels are not the “ground truth”, since there is no
standard definition about levels of invariance. Instead, they are a kind of professional annotation, which is commonly adopted for complex annotation tasks, e.g., linguistic labelling [39], and lying/emotion detection annotation [22].

Here we describe our general guideline for labelling. Generally, we consider models as “invariant” if the models satisfy the following conditions:

- (1) they are trained with sufficient augmentation, i.e., covering the target testing interval. In our case, at least -15 to 15 for rotation, and 0.7 to 1.3 for scaling or brightness.
- (2) they do not obviously underfit or overfit according to the loss function curve.
- (3) all the concerned variance matrices are considered normal (no large values, “cross” or “jagged”).
- (4) during training, no anomaly that would doubts about the invariance quality of the ML model being labelled. For example, when 50% of the training data objects were labelled randomly, the trained models should be labelled as “variant”.

For anomalies, we consider a CNN abnormal if we manually introduce any of the following artefacts while training:

- **Noisy data**: we add some meaningless random noise to our training set and label them randomly.
- **Data leakage**: we train our models using both training set and part of testing set.
- **Impaired label**: we manually and randomly place a wrong label to some training images.
- **Others**: for example, no shuffling, or only use a smaller portion of the training data, or remove some intervals for augmentation, e.g., no augmentation from 0° to 5°.

In addition to the anomalies above, we also found extreme values and numerical issues for VGG13bn that trained with a small batch size, e.g., 8 examples per batch. In this case, the distribution of the $\text{dif}()$ function would not be sub-gaussian, therefore this leads to abrupt changes on the variance matrices, as shown in Figure [2]. After those data objects that cause the extreme values have been removed, the variance matrices appear to be normal and smooth. Note that, for those CNNs with extreme values or numerical issues, they are able to achieve satisfactory classification rate (e.g., $\geq 85\%$ on CIFAR), and all of the abnormal data objects (which cause the issue) are correctly classified with a low confidence score (around 0.23 on average).

In order to investigate if one testing position attracts more attention than the others, we ask our three annotators the following questions:

- Q1: Whether CONV-1 and CONV-2 should be checked.
- Q2: Whether the coder(s) always check CONV-1 and CONV-2 when labeling.

Coder 1: CONV-1 is strongly related to the feature vector(s) a model can provide for transfer learning / metric learning / self-supervised learning. Coder 2: activations at CONV-1 are often informative for invariance evaluation. Coder 3: while CONV-1 is often informative for invariance evaluation, CONV-1 should still be checked to ensure the invariance qualities do not only come from the final linear layer(s). All the three coders always checked CONF. Coder 1 and Coder 2 always checked CONV-1, while Coder 3 checked CONV-1 when they had doubts in the provided hyper-parameters or loss function curve but could not make judgement depending on variance matrices generated at CONV. CONV-2 was checked by the three coders only when they could not make a decision after they checked variance matrices generated at CONV and CONV-1.

**IX. Models Used for Training**

We provide a model-repository with 600 models. The metadata (including all hyper-parameters) for the 600 models can be found in our metadata appendix. In order to train the ML4ML assessors, we select four “balanced” partitions, each of which is used for training and testing the workflow $W_1 \sim W_4$ introduced in Section [V].

When testing the ML4ML assessors, although we use the standard cross-validation protocol, we also separate a “hold-out testing set” for each partitions as shown in Table [IX]. More specifically, partition I starts from model id: “mid: 1" to “mid: 100”, and “mid: t1” to “mid: t50”. When using cross-validation, the “hold-out” testing set, i.e., “mid: t1” to “mid: t50”, is always treated as a single fold, while the rest 100
models are randomly divided into two other folds. Note that, when training the ML4ML assessors, we did not tune any hyper-parameters, e.g., the depth for decision trees. Therefore, we did not further split the training set into training and validation set.

For each partition, the statistics of the assessment labels are listed in Table X. Each partition is considered relatively balanced.

For partition (d), we train 150 CNN5 models on MNIST. The CNN5 structure is not tuned. Instead, we randomly chose the hyper-parameters, e.g., the number of hidden layers, kernel size, dropout rate, etc. The CNN5 structure we use in this work is:

- Input: (1, 28, 28)
- Conv1: kernel size (5, 5), output: 6 channels
- Dropout: 25% + ReLU + MaxPooling (2, 2)
- Conv2: kernel size (5, 5), output: 16 channels
- Dropout: 25% + ReLU + MaxPooling (2, 2)
- Conv3: kernel size (3, 3), output: 32 channels
- Dropout: 25% + ReLU + Flatten
- Linear4: input 128-d, output 64-d
- Linear5: input 64-d, output 10-d

We used Pytorch [28] to implement the structure and the way of initialisation for the weights and biases were set to default.

We also conducted experiments on the entire 450 VGG13bn in the model-repository where the data are imbalanced, i.e., there are more “variant” models than “invariant” ones. As shown in Table XI, we still observe similar results, e.g., random forest performs the best. All the results were also obtained on 10 repeated experiments. And we use the same three-fold cross-validation protocol mentioned above to carry out each experiment. The proposed framework performs ~5% better than the baseline.

a) Metadata of the models:: Table VI shows all the metadata of the 12 models shown in Fig 3, and we provide all metadata of all the 600 models in our metadata-code appendix. Note that, when conducting machine learning testing, we should NOT rely on the metadata (they are not used to train ML4ML assessors).

X. MATHEMATICAL PROPERTY

Proposition 2 When the dif_{i,j} function can be written as f_i() - f_j(), e.g., max or mean, the element δ_{i,j} of the
variance matrix is strongly related to the Pearson’s correlation coefficient \( \rho_{f(z_i), f(z_j)} \), where the latent variables, i.e. \( z_i := t_i(x) \in T(X) \) and \( z_j := t_j(x) \in T(X) \) (of the same data object \( x \in X \)) are dependent and identically distributed. 

**Proof:** 

\[
\delta_{i,j}^2 / 2 = \mathbb{E}_{T(X)} (f(z_i) - f(z_j))^2 / 2 \\
= \mathbb{E}_{T(X)} (f(y))^2 - \mathbb{E}_{T(X)} (f(z_i) * f(z_j)) \\
= \left[ \mathbb{E}_{T(X)} (f(y))^2 - \mathbb{E}_{T(X)} (f(y))^2 \right] - \\
\left[ \mathbb{E}_{T(X)} (f(z_i) * f(z_j)) - \mathbb{E}_{T(X)} (f(z_i)) \mathbb{E}_{T(X)} (f(z_j)) \right] \\
= \text{VAR}(f(y)) * (1 - \rho_{f(z_i), f(z_j)}) \\
\] 

where \( \text{VAR} \) stands for variance, \( y \in T(X) \) is a transformed data object which has the same distribution as \( z_i/z_j \), and the expected values of \( f(y) \), \( f(z_i) \) and \( f(z_j) \) are the same. Moreover, \( \sigma_{f(z_i)} * \sigma_{f(z_j)} = \text{VAR}(f(y)) \), where \( \sigma \) stands for the standard deviation.

The elements of the variance matrices \( \delta_{i,j} \) concentrate around the true expected value \( \hat{\delta}_{i,j} \) when the \( \text{dif}_{i,j}() \) function is subgaussian distributed. We denote \( f^+ \) as the upper bound of the \( \text{dif}_{i,j}() \) function, for any \( \epsilon > 0 \), by the Hoeffding inequality we have:

\[
P_r\left( \frac{1}{m} \sum_{k=1}^{m} \delta_{i,j}(x_k) - \hat{\delta}_{i,j}^2(x_k) \right) \geq \epsilon \right) \leq 2e^{-2 \epsilon^2 m^2 / f^+} \quad (3)
\]

Therefore, when \( m \gg f^+ \), \( \delta_{i,j} \) should not be significantly affected by the size of \( X \). In Fig. 5 to Fig. 7 we empirically show that our variance matrices are not heavily affected by the size of \( X \), unless the distribution of \( \text{dif}_{i,j}() \) function is not subgaussian, e.g., Fig 4 which is caused by anomalies such as numerical issues or extreme values.

**XI. Ablation study**

**Size of testing suite.** In Fig 5 to 7 we show that neither the variance matrices nor the measurements will be heavily affected by the size of the testing suite \( X \).

**Splitting testing suites.** We split the testing suite \( X \) into two parts, and use one part to generate variance matrices and measurements for the models on the training set of the balanced model-repository, and use the other part to generate variance matrices and measurements for the models on the testing set of the balanced model-repository. In Table VII we show that, under this setting, our testing framework (the ML4ML assessors) can still achieve satisfactory results (over 90% on the testing set of the balanced model-repository). Note that this setting, there is no overlapping for both the ML models, or the testing suite for the training/testing set of the model-repository.

Moreover, it is not necessary to keep the same proportion of testing suite \( X \) for either training or testing the ML4ML assessors. In Table VIII we show that when generating the measurements using random proportion of the testing suite \( X \), our testing framework (the ML4ML assessors) is still able to achieve satisfactory results (over 90% on the testing set of the balanced model-repository).
Fig. 5. Rotation variance matrices: (1) First row: \( \text{max}@\text{CONF} \); (2) Second row: \( \text{max}@\text{CONF}-1 \), i.e., the last convolutional layer (The ninth convolutional layer for VGG13bn); (3) Third row: \( \text{mean}@\text{CONF}-1 \) of a randomly chosen VGG13bn (model id: 15) trained without anomaly. The visual patterns are obtained using a smaller proportion of the testing data. As shown in the figures, the matrices (visual patterns) are not heavily affected by the size of testing data set (i.e. the testing set of the original CIFAR10/MNIST dataset). When the proportion is greater than 30\%, we see no significant difference between the visual patterns for the first row (\( \text{max}@\text{CONF} \)). And we see no significant difference between the visual patterns on the last two rows.

**TABLE XII**

Measurements on rotation variance matrices (\( \text{max}@\text{CONF} \)) of a randomly chosen VGG13bn (model id: 15) trained without anomaly, and the visual patterns are obtained using a smaller proportion of the testing data. The measurements are not heavily affected by the size of testing data set (i.e. the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.085 | 0.089 | 0.089 | 0.087 | 0.087 | 0.087 | 0.087 | 0.087 | 0.087 | 0.088 |
|            | Discontinuity| 1.523 | 1.418 | 1.326 | 1.364 | 1.372 | 1.283 | 1.274 | 1.278 | 1.275 | 1.258 |
|            | Asymmetry   | 0.074 | 0.043 | 0.034 | 0.033 | 0.032 | 0.029 | 0.031 | 0.032 | 0.029 | 0.029 |
|            | Gradient    | 0.730 | 0.766 | 0.780 | 0.806 | 0.826 | 0.824 | 0.836 | 0.834 | 0.834 | 0.835 |

**TABLE XIII**

Measurements on rotation variance matrices (\( \text{max}@\text{CONV}-1 \)) of a randomly chosen VGG13bn (model id: 15) trained without anomaly, and the visual patterns are obtained using a smaller proportion of the testing data. The measurements are not heavily affected by the size of testing data set (i.e. the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.955 | 0.931 | 0.926 | 0.926 | 0.938 | 0.934 | 0.934 | 0.934 | 0.934 | 0.934 |
|            | Discontinuity| 1.889 | 1.755 | 1.733 | 1.717 | 1.725 | 1.700 | 1.713 | 1.707 | 1.707 | 1.707 |
|            | Asymmetry   | 0.027 | 0.018 | 0.016 | 0.019 | 0.017 | 0.018 | 0.021 | 0.020 | 0.019 | 0.018 |
|            | Gradient    | 1.033 | 1.058 | 1.048 | 1.045 | 1.053 | 1.056 | 1.054 | 1.059 | 1.060 | 1.058 |

**TABLE XIV**

Measurements on rotation variance matrices (\( \text{mean}@\text{CONV}-1 \)) of a randomly chosen VGG13bn (model id: 15) trained without anomaly, and the visual patterns are obtained using a smaller proportion of the testing data. The measurements are not heavily affected by the size of testing data set (i.e. the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.521 | 0.512 | 0.508 | 0.511 | 0.517 | 0.516 | 0.517 | 0.517 | 0.516 | 0.517 |
|            | Discontinuity| 1.824 | 1.727 | 1.768 | 1.739 | 1.761 | 1.742 | 1.744 | 1.745 | 1.740 | 1.750 |
|            | Asymmetry   | 0.026 | 0.017 | 0.013 | 0.011 | 0.014 | 0.014 | 0.015 | 0.012 | 0.010 | 0.011 |
|            | Gradient    | 1.040 | 1.052 | 1.037 | 1.026 | 1.043 | 1.049 | 1.053 | 1.049 | 1.050 | 1.050 |
Fig. 6. Brightness variance matrices: (1) First row: \( \text{max}@\text{CONF} \); (2) Second row: \( \text{max}@\text{CONF-1} \), i.e., the last convolutional layer (CONV9) for VGG13bn; (3) Third row: \( \text{mean}@\text{CONF-2} \), i.e., the penultimate convolutional layer (CONV8) for VGG13bn, of a randomly chosen VGG13bn (model id: 102) trained without anomaly obtained using smaller proportions of the testing data. The matrices are not heavily affected by the size of testing data set (i.e., the testing set of the original CIFAR10/MNIST dataset).

| TABLE XV | MEASUREMENTS ON BRIGHTNESS VARIANCE MATRICES (\( \text{max}@\text{CONF} \)) OF A RANDOMLY CHOSEN VGG13BN (MODEL ID: 102) TRAINED WITHOUT ANOMALY OBTAINED USING SMALLER PROPORTIONS OF THE TESTING DATA. THE MEASUREMENTS ARE NOT HEAVILY AFFECTED BY THE SIZE OF TESTING DATA SET (I.E., THE TESTING SET OF THE ORIGINAL CIFAR10/MNIST DATASET). |
|-----------|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Proportion | Measurement       | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
| Mean       | 0.041             | 0.041| 0.040| 0.040| 0.040| 0.040| 0.040| 0.040| 0.040| 0.040| 0.040|
| Discontinuity | 2.445            | 2.509| 2.440| 2.300| 2.133| 2.157| 2.179| 2.236| 2.205| 2.246|
| Asymmetry  | 0.260             | 0.252| 0.246| 0.232| 0.218| 0.224| 0.230| 0.236| 0.230| 0.236|
| Gradient   | 1.923             | 2.035| 2.064| 2.050| 2.023| 2.057| 2.049| 2.061| 2.049| 2.051|

| TABLE XVI | MEASUREMENTS ON BRIGHTNESS VARIANCE MATRICES (\( \text{max}@\text{CONV-1} \)) OF A RANDOMLY CHOSEN VGG13BN (MODEL ID: 102) TRAINED WITHOUT ANOMALY OBTAINED USING SMALLER PROPORTIONS OF THE TESTING DATA. THE MEASUREMENTS ARE NOT HEAVILY AFFECTED BY THE SIZE OF TESTING DATA SET (I.E., THE TESTING SET OF THE ORIGINAL CIFAR10/MNIST DATASET). |
|-----------|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Proportion | Measurement       | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
| Mean       | 0.045             | 0.035| 0.031| 0.029| 0.029| 0.028| 0.027| 0.026| 0.027| 0.026|
| Discontinuity | 1.545           | 1.562| 1.547| 1.266| 1.113| 1.092| 1.048| 1.060| 1.022| 0.986|
| Asymmetry  | 0.225             | 0.183| 0.121| 0.131| 0.129| 0.127| 0.138| 0.131| 0.137| 0.137|
| Gradient   | 1.963             | 2.027| 2.006| 2.101| 2.081| 2.019| 2.035| 2.046| 2.060| 2.069|

| TABLE XVII | MEASUREMENTS ON BRIGHTNESS VARIANCE MATRICES (\( \text{mean}@\text{CONV-2} \)) OF A RANDOMLY CHOSEN VGG13BN (MODEL ID: 102) TRAINED WITHOUT ANOMALY OBTAINED USING SMALLER PROPORTIONS OF THE TESTING DATA. THE MEASUREMENTS ARE NOT HEAVILY AFFECTED BY THE SIZE OF TESTING DATA SET (I.E., THE TESTING SET OF THE ORIGINAL CIFAR10/MNIST DATASET). |
|------------|---------------------|-----|-----|-----|-----|-----|-----|-----|-----|-----|-----|
| Proportion | Measurement       | 10% | 20% | 30% | 40% | 50% | 60% | 70% | 80% | 90% | 100% |
| Mean       | 0.012             | 0.012| 0.012| 0.012| 0.012| 0.012| 0.012| 0.012| 0.012| 0.012| 0.012|
| Discontinuity | 1.892          | 1.834| 1.895| 1.960| 1.934| 1.881| 1.845| 1.817| 1.822| 1.812|
| Asymmetry  | 0.202             | 0.183| 0.194| 0.206| 0.198| 0.194| 0.191| 0.190| 0.191| 0.188|
| Gradient   | 2.113             | 2.032| 2.081| 2.090| 2.076| 2.066| 2.060| 2.079| 2.078| 2.081|
Fig. 7. Scaling (Size) variance matrices: (1) First row: \( \text{max}@\text{CONF} \); (2) Second row: \( \text{max}@\text{CONF-1} \), i.e., the last convolutional layer (CONV9) for VGG13bn; (3) Third row: \( \text{mean}@\text{CONF-1} \) of a randomly chosen VGG13bn (model id: 202) trained without anomaly obtained using smaller proportions of the testing data. The matrices are not heavily affected by the size of testing data set (i.e., the testing set of the original CIFAR10/MNIST dataset).

### TABLE XVIII

Measurements on scaling variance matrices (\( \text{max}@\text{CONF} \)) of a randomly chosen VGG13bn (model id: 202) trained without anomaly obtained using smaller proportions of the testing data. The measurements are not heavily affected by the size of testing data set (i.e., the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.137 | 0.139 | 0.141 | 0.139 | 0.138 | 0.138 | 0.139 | 0.138 | 0.138 | 0.139 |
|            | Discontinuity | 1.914 | 1.848 | 1.802 | 1.787 | 1.784 | 1.736 | 1.747 | 1.738 | 1.738 | 1.745 |
|            | Asymmetry   | 0.202 | 0.196 | 0.193 | 0.193 | 0.187 | 0.189 | 0.189 | 0.190 | 0.188 | 0.190 |
|            | Gradient    | 0.656 | 0.718 | 0.717 | 0.716 | 0.716 | 0.724 | 0.734 | 0.719 | 0.725 | 0.721 |

### TABLE XIX

Measurements on scaling variance matrices (\( \text{max}@\text{CONV-1} \)) of a randomly chosen VGG13bn (model id: 202) trained without anomaly obtained using smaller proportions of the testing data. The measurements are not heavily affected by the size of testing data set (i.e., the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.107 | 0.107 | 0.107 | 0.107 | 0.107 | 0.106 | 0.107 | 0.106 | 0.106 | 0.106 |
|            | Discontinuity | 1.941 | 1.809 | 1.751 | 1.748 | 1.746 | 1.780 | 1.757 | 1.772 | 1.785 | 1.782 |
|            | Asymmetry   | 0.211 | 0.205 | 0.206 | 0.205 | 0.203 | 0.203 | 0.202 | 0.202 | 0.201 | 0.201 |
|            | Gradient    | 0.757 | 0.768 | 0.779 | 0.763 | 0.756 | 0.747 | 0.744 | 0.751 | 0.755 | 0.754 |

### TABLE XX

Measurements on scaling variance matrices (\( \text{mean}@\text{CONV-1} \)) of a randomly chosen VGG13bn (model id: 202) trained without anomaly obtained using smaller proportions of the testing data. The measurements are not heavily affected by the size of testing data set (i.e., the testing set of the original CIFAR10/MNIST dataset).

| Proportion | Measurement | 10%   | 20%   | 30%   | 40%   | 50%   | 60%   | 70%   | 80%   | 90%   | 100%  |
|------------|-------------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
|            | Mean        | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 | 0.076 |
|            | Discontinuity | 1.974 | 2.184 | 2.027 | 2.011 | 2.009 | 2.017 | 1.992 | 1.987 | 2.020 | 2.030 |
|            | Asymmetry   | 0.215 | 0.234 | 0.216 | 0.214 | 0.214 | 0.213 | 0.214 | 0.210 | 0.214 | 0.215 |
|            | Gradient    | 0.808 | 0.835 | 0.872 | 0.871 | 0.863 | 0.865 | 0.858 | 0.859 | 0.861 | 0.856 |