

Invariance Testing For Machine Learning Models



Zukang Liao
St Hugh's College
University of Oxford

A thesis submitted for the degree of
Doctor of Philosophy

Trinity 2024

Abstract

Machine learning models are expected to produce consistent results when the input data objects are slightly modified whilst the ground truth remains unchanged. Such characteristics are also known as invariance qualities. However, previous studies assessing ML model invariance have primarily relied on one or a few accuracy metrics, which provide limited information on how models behave under different transformations. This thesis addresses this gap by proposing a novel, systematic invariance testing framework that moves beyond aggregated accuracy scores and offers more detailed visual information to enable more complex and explainable analysis of model performance.

Our framework introduces a generic and extendable methodology for evaluating invariance across various attributes, including simple transformations (e.g., object rotation) as well as complex environmental variations (e.g., background changes). It is designed to be fully automated, enabling extensive and reproducible testing while ensuring consistent and explainable assessments of models' invariance qualities. A key challenge in invariance testing is ordering and structuring test data, especially when testing an entire input space is infeasible. To address this, we develop a novel approach to sampling large information spaces, leveraging context-sensitive training data to construct context-free models. This method ensures that our framework can effectively assess invariance properties without requiring exhaustive test cases.

By providing visualized patterns of model behavior, our approach enhances diagnostic capabilities, making it a valuable tool for both researchers and practitioners. Our findings demonstrate that the evaluation of ML models' invariance qualities based on the visualized patterns can be: more informative, automatic, reliable and consistent. This thesis contributes to the broader effort of improving ML model evaluation, ensuring that claims of invariance are substantiated with comprehensive and reproducible evidence.

Contents

1	Introduction	1
1.1	Research Gap and Problem Statement	1
1.2	Definable Variation	3
1.3	Motivations	4
1.4	Objectives	6
1.5	Outline of the Thesis	8
1.5.1	Published Papers and Technical Reports	9
2	Related Works	11
2.1	Overview of Related Works	11
2.2	Traditional definition	13
2.3	Variables	17
2.3.1	Basic data attributes	17
2.3.2	Variables of devices / data capturing	22
2.3.3	Data Quality: Imprecision (errors) and Noise	25
2.3.3.1	Imprecision and Errors	25
2.3.3.2	Noise	26
2.3.4	Styles attributes	31
2.3.5	Variables associated with applications	34
2.4	Input generation and mutation control	34
2.4.1	Generative adversarial networks	35
2.4.2	Variational Autoencoder	38
2.4.3	Coverage-based fuzzing testing	39
2.4.4	Adversarial attacks	41
2.4.5	Ontology and Scene Understanding	43
2.5	Invariance Testing Workflows	44
2.5.1	Formula-based invariance testing	45
2.5.2	Visualization-based professional analysis	46

2.5.2.1	Explainable AI (XAI)	47
2.5.2.2	Symbolic testing	47
2.5.2.3	Heatmap-based Analysis	48
2.5.3	Theory-based professional analysis	51
2.6	Conclusion	53
3	Generic Invariance Testing Framework for Basic Data Attributes	55
3.1	Introduction	55
3.2	Related work	56
3.3	Definitions and Motivation	57
3.4	Methodology: Testing Framework	59
3.4.1	Deployment Workflow	59
3.4.2	Development Workflow	65
3.5	Experimental Results and Analysis	67
3.5.1	Model Repository	67
3.5.2	Invariance Labelling Convention	68
3.5.3	Training and Testing Assessors.	70
3.5.3.1	Experiment Details on Variance Matrices	71
3.5.3.2	Feature Extraction	72
3.5.4	Ablation Studies	76
3.5.5	Result Analysis.	77
3.6	Case Study: Invariance Testing using Sparse Linear Layers	79
3.6.1	Sparse Linear Layers	80
3.6.2	Invariance Testing Workflow using Sparse Linear Layers	81
3.6.3	A New Test Oracle: Associativity	83
3.6.4	Actionable: Feature Selection	84
3.6.5	Consistency and Reliability	84
3.7	Conclusions	85
4	Invariance Testing for Advanced Data Attributes: Background Invariance Testing	90
4.1	Introduction	90
4.2	Related Work	92
4.3	Definition, Overview, and Motivation	93
4.4	Methodology	95
4.4.1	To Obtain Detected Keywords	95
4.4.2	To Obtain Expanded Keywords	96

4.4.3	To Synthesize/Generate Testing Images	99
4.4.3.1	Background Scenes Sampling	100
4.4.3.2	Simple Background Replacement	100
4.4.4	Point Clouds from ML Testing.	102
4.4.5	Variance Matrices from RBF-based Resampling.	103
4.4.6	Professional Annotations	104
4.4.7	ML4ML Assessor Training and Deployment.	106
4.5	Experiments	106
4.5.1	Model Repository	107
4.5.2	Worst-Case Accuracy versus Variance Matrices	108
4.5.3	Comparisons Between Testing Methods	109
4.5.3.1	Prior Works: Random/Nearest Sampling	109
4.5.3.2	Distance and Keyword-based Sampling	110
4.5.4	Training ML4ML Assessors	110
4.5.5	Ablation Studies	111
4.5.6	Results and Analysis	113
4.6	Conclusion	114
5	Ordering Invariance Testing Data: Image Similarity using An Ensemble of Context-Sensitive Models	116
5.1	Introduction	116
5.2	Related Work	119
5.3	Dataset	120
5.4	Methodology	121
5.4.1	Fine-tuning Models	123
5.4.2	Ensemble Strategies	124
5.5	Experiment	127
5.5.1	Performance of Context-Sensitive Models	128
5.5.2	Performance of Global Models	130
5.5.3	Performance of Ensemble Models	131
5.5.4	Result Analysis and Training Visualization	131
5.5.5	Ablation Studies	132
5.5.5.1	Evaluation using Ranking Block (Binary Classifier)	133
5.5.5.2	Cross Validation of CS Models	136
5.5.5.3	Ablation Study on Ensemble Models	137
5.5.5.4	Meta Context-sensitive models	139

5.6 Conclusions	141
6 Conclusions	142
Bibliography	148

Chapter 1

Introduction

Invariance qualities are often desired for ML models. ML models need to be robust and resilient to variations and changes in the input data. Invariance qualities help ensure that the model's performance remains consistent across different scenarios, such as variations in lighting conditions, viewpoints, or other transformations of the data. By being invariant to certain factors, models can make accurate predictions and decisions even when faced with changes in the input. Secondly, Invariance qualities aid in the generalization ability of ML models. Generalization refers to the model's ability to perform well on unseen data that may differ from the training data. Invariance ensures that the model can recognize and extract meaningful features regardless of certain variations or transformations in the input. This allows the model to generalize its learned patterns and knowledge to new, unseen examples. Finally, ML models are often deployed in real-world scenarios where the input data can exhibit variations and transformations. For example, in computer vision, images may have different lighting conditions, rotations, or occlusions. In natural language processing, text data can have different formats, spellings, or sentence structures. Invariance qualities allow ML models to be more applicable and useful in practical situations by accommodating these real-world variations. Therefore, it is important to test ML model's invariance qualities.

1.1 Research Gap and Problem Statement

As deep learning techniques have been prevalent and ubiquitous, thousands of training strategies are proposed and published every year while testing is almost always formula-based: to simply compute an accuracy score. However, comparing accuracy scores on one or some dataset(s) is simplistic and cannot truly represent the performance of the target ML model (to be tested), especially for invariance testing. Ac-

cording to Anscombe Quartet, an averaged score cannot truly represent how the data are truly distributed. Especially when conducting invariance testing, the averaged accuracy scores like worst-case accuracy are not informative enough to represent the performance of the target ML model. The most universally accepted testing strategies as well as state-of-the-art testing frameworks often encounter the following issues:

1. *Using an overly abstracted statistical measure, e.g., average accuracy score, to testing the invariance qualities:*

- Cannot observe the distribution of errors across different values of a specific interested variable. For example: an ML model can achieve satisfactory results when the inputs are rotated by less than 10 degrees, whereas the performance can drop drastically otherwise. This will lead to a poor statistical score, e.g., average accuracy score, but this does not necessarily indicate that the model is not useful.
- Cannot observe the results from different locations and components of interested ML models. For example, the target ML models might have an invariant feature extractor block whilst the classifier block might not be invariant. When only one average score is reported, it is difficult to examine different locations and components of the interested ML models in detail.
- Do not support explainable ML process. For example, two ML models can have different accuracy scores, whilst it is difficult to determine one is superior than the other one simply by comparing statistical measures. Additionally, in many cases, ML models can have similar accuracy scores on some datasets, which makes the comparisons between different ML models difficult.

2. *Difficulty in dealing with non-numerical variables, e.g., background:*

- For complex and non-numerical variables, variance is often defined semantically, and is often perceived subjectively, and typically context-sensitive. For example, unlike rotation or translation, variation in background cannot be defined numerically. The definition of background variance is often defined and perceived subjectively, e.g., via some simple background replacement [156,338].
- The similarity measure that underpins the variance definition is also defined semantically, perceived subjectively, and context-sensitive. For example, for simple background replacement, the distance or similarity measure between testing candidates and original input data object is often defined semantically,

e.g., L2/cosine distance of embeddings obtained from ML models which often take semantics into consideration.

- This makes simple robustness test less effective in reflecting the invariance quality in relation to non-numerical variations.

3. Imbalance between explainability and automation in testing:

- Fully explainable and interpretable testing frameworks, e.g., manually analyzing invariance qualities of ML models or XAI [58], can be time consuming and labour-intensive. Testing should be able to be conducted repeatedly during model development and evaluation to ensure consistent and reliable performance.

In this thesis, we mainly focus on invariance testing for discriminative ML models, especially for classification models. For testing generative models, we refer interested readers to the surveys on the commonly used metrics and testing methods [19, 213]. We also refer interested readers to a study [302] that focused on the importance of task-specific evaluations and highlighted the instability of the commonly used metrics, e.g., Parzen window estimates.

The necessity for invariance in machine learning models varies depending on applications. Some domains demand a high degree of invariance than others. For instance, surveillance systems require strong size invariance to ensure consistent detection regardless of object scale [290]. Conversely, some applications benefit from sensitivity to specific attributes; for example, agricultural ML models designed to assess vegetable size should not exhibit size invariance, as variations in size are critical to their function [95]. In this thesis, we focus on evaluating the invariance properties of ML models, while the investigation of application-specific invariance requirements is left for future work. We assume that at the moment ML model developers will determine what attributes should or should not be invariant for models to be deployed in a specific application context. We believe that most of such decisions may potentially be automated in the future when an invariance testing framework has collected sufficient knowledge about numerous applications.

1.2 Definable Variation

A variation for an input of ML models refers to a structured or measurable change made to the original input data object. In structured data, individual input features

(e.g., height, weight, age in a medical dataset) can be varied to study the model’s behavior under different input variants. The variation can influence how the models make decisions and this helps understand how changes in specific input features, affect the model’s predictions. Variations are often intentionally introduced to study the models’ invariance qualities. For example, a small amount of random noise can be added to the input data to examine how sensitive the model is to changes.

A variation is definable if it is: 1) *explicitly described*, 2) *quantified*, and 3) systematically *controlled*. A definable variation is a change that can be articulated in clear, precise terms and measured or manipulated systematically. This allows the variation to be consistent and repeatable in experiments, and allows us to observe and analyze ML models’ behavior in a consistent manner. Definable variations should be able to be applied in steps, e.g., incrementally increasing the rotation angles, or the magnitude of the Gaussian noise (*controlled*). This provides researchers with control over how the input is varied and helps test the model’s behavior at different levels of variation. The variation is normally applied uniformly across different inputs. For instance, for a Gaussian blur to images, the process can be described mathematically and applied the same way to any image. The process of perturbing or modifying inputs is often based on well-understood principles or algorithms. This allows the variation to be consistent and repeatable in experiments, and allows us to observe and analyze ML models’ behavior in a consistent manner.

A definable variation causes changes to the input that are measurable, controllable, and observable. It allows for systematic testing and evaluation of how changes in the input affect ML models’ output, making it a key concept for invariance testing. Definability ensures that variations can be reproduced, providing consistency and clarity in understanding the ML models’ behavior. However, when studying complex variables, e.g, background, additional technical components are necessary to make the variables definable. Transforming such complex variables to quantifiable and controllable variables, e.g., in feature or embedding space, in graph space constructed from semantic ontology and association analysis, and using controlled generation of synthetic data. Invariance testing for many different types of complex variables remains an open topic for future works.

1.3 Motivations

Motivation M_0 : a comprehensive survey In the literature, there are various studies strongly related to invariance testing but not traditionally referred to as an

invariance testing problem. For example, Paul and Burlina used a styleGan to explore “hidden factors” to test a model’s performance [236]. Similarly, SCEGRAM [226] removed one or a few item(s) from a scene to study “syntactic inconsistencies”. In the existing studies, multiple terms are used to describe the same invariance testing problem, which makes revisiting the problem difficult. Therefore, for the completeness of the research of invariance testing and the facilitation of future works, it is beneficial to summarise existing works related to invariance testing and re-frame them into invariance testing problems when necessary.

Motivation M_1 : a general framework The importance of invariance qualities and lack of informative testing approaches motivates us to delve into the problem of invariance testing. To revisit and advance the research and studies on this topic, firstly, we need to identify the existing challenges and to discover needs that are not immediately obvious for a new testing framework. Whilst simply reporting an average accuracy score has been widely accepted by the AI/ML community as the standard testing approach for any ML model, questions and concerns are being constantly raised when it comes to accountability of ML models. For example, as more and more ML models are published each year, researchers start doubting the significance of ML models with “0.5% improvement” on a dataset. Some ML models might not perform well against one certain type of permutation, leading to a lower average accuracy score. However, this does not suggest that such ML models cannot be used or deployed. Similarly, most ML models are vulnerable against adversarial attacks [99] and the success attack rates are normally very high. Whilst this poses a serious security and safety thread, this does not suggest that ML models cannot be deployed for real world applications. If we can thoroughly test the performance of them, and understand their behaviours holistically, we can still make use of such ML models. Therefore, we believe there is a strong need for a new testing framework to demonstrate detailed performance of ML models in a more informative way. Such testing framework could help practitioners, researchers, as well as developers to understand the performance of their ML models in detail, and therefore it could allow them to make overall decisions on whether and how their ML models should be deployed and used.

Motivation M_2 : a complex variable Whilst the general testing framework is implemented for most basic and simple variables, e.g., rotation and brightness, it is also important that the generic testing framework can be extended to more complex variables, e.g., background. For such non-numerical variables, variations cannot be

easily defined mathematically, and complex variables often introduce additional layers of variability or dynamics that can affect ML models’ behaviours. Therefore, it is desirable to bring more semantically-rich ML techniques into invariance testing to avoid randomised testing, e.g., a systematic way to select testing candidates from a gigantic data space.

Motivation M_3 : an organised test When variations are articulated through semantic definitions, subjectively and intuitively interpreted, and intricately influenced by contextual factors, it is desired to revisit the ordering of the testing data. Additionally, because semantically-rich ML techniques have to deal with sparse and context-sensitive training data, necessitating the development of new ML methodologies capable of effectively training models under such conditions. When data are sparse, the ability of machine learning models to generalize effectively across diverse inputs or contexts can be hindered. The advancements in effective model training under scarcity are indispensable for achieving an organised invariance testing where the ordering of testing data is crucial.

1.4 Objectives

In this section, we summarise the main objectives of the thesis, including some desired properties of the invariance testing framework and some existing challenges for advancing the studies on invariance testing.

- **O_0 Literature Review:** The literature includes numerous studies closely related to invariance testing, such as those focusing on resilience to hidden factors. Additionally, various existing works offer potential contributions to assist invariance testing. However, the formal formulation of the invariance testing problem remains incomplete, and existing literature has not been systematically summarized. Therefore, this study aims to systematically review and summarize existing works relevant to invariance testing to provide a comprehensive foundation for future research.
- **O_{1a} Informative testing framework:** Traditionally, invariance testing has been confined to reporting averaged accuracy scores, such as worst-case accuracy. However, these averaged formula-based metrics often fail to sufficiently capture the nuanced performance of ML models, akin to the limitations illustrated by Anscombe’s quartet problem. Variations in input permutations

can significantly affect model performance, underscoring the need for comprehensive testing methodologies that can holistically assess model behaviours. Therefore, this study aims to advance the field of invariance testing by developing and evaluating novel testing methodologies that move beyond simplistic averaged accuracy scores. The objective is to establish more informative testing frameworks that consider multiple factors and provide nuanced insights into ML model performance across diverse scenarios and input variations.

- **O_{1b} Automatic Testing Framework:** Manual testing of ML models is labor-intensive and prone to inconsistencies. Therefore, another important objective of this thesis is to develop an automated testing framework that can generate testing results automatically and consistently. This framework aims to simplify the testing procedure, making it easier to conduct and scale up in various applications.
- **O_{2a} Application to Complex Variables:** This thesis aims to develop a comprehensive framework for invariance testing that addresses diverse types of variables, ranging from fundamental transformations (e.g., rotation, translation) to more complex factors such as background variation. Advanced and complex variables pose challenges in defining and executing invariance testing, primarily due to the vast data space. This framework aims to be adaptable and scalable, specifically tailored to extend its application to complex variables like background invariance testing. Furthermore, the study aims to identify key challenges inherent in testing these variables and propose viable solutions to enhance the efficacy of invariance testing in complex scenarios.
- **O_{2b} Selection of Testing Data for Complex Variables:** For basic data attributes, e.g., rotation, it is straightforward to select the mutation, e.g., different rotation angles. However, for more complex variables, e.g., background, the data space for mutation is gigantic, and it is not plausible to enumerate all possibility. Ideally, the selected testing suite should be diverse and comprehensive. Meanwhile, the selected testing suite should not be too random. If a foreground object appears more often in outdoor scenes, the ideal testing background candidates should also be mostly outdoor scenes. In this thesis, we aim to introduce a systematic methodology to carefully select different contexts for invariance testing of complex variables, e.g., background.

- **O_3 Semantic Ordering and Ranking of Testing Data for Complex Variables:** While ordering testing data for simple attributes, such as rotation angles, is straightforward, the challenge intensifies with complex variables like background variations. Existing literature often emphasizes visual similarity in image comparison, neglecting semantic meanings. Attempts to bridge this gap have relied heavily on supplementary metrics like WordNet [293]. This study aims to advance the field by developing semantic ordering, sorting, and ranking methods based on reference images. By prioritizing semantic coherence alongside visual similarity, these methods will enable more organized and insightful evaluation of ML models’ behaviours under complex variables. This approach seeks to advance the systematic organization of testing data, ensuring more nuanced assessments and reliable model performance evaluations.

1.5 Outline of the Thesis

In the thesis, in addition to the literature reviews (survey) on invariance testing that summarise and re-frame existing studies into invariance testing problems, three main technical chapters are included to address the aforementioned challenges.

- **Technical Chapter 1:** In the first technical chapter (i.e., Chapter 3 in this thesis), we introduce our definition of invariance testing and a generic automatic testing framework (via visual patterns) to conduct invariance testing for basic data attributes, e.g., rotation, size or brightness. The main aspects of this work were published in AITest 2022 [165], CIKM 2022 [171] and ICML 2022 [170].
- **Technical Chapter 2:** In the second technical chapter (i.e., Chapter 4 in this thesis), we extend the framework to a more complicated variable, i.e., background, which requires a systematic way to generate testing data. We address this issue by using ontology and association analysis. As mentioned earlier, it is also essential to sort the generated testing data to facilitate further professional analysis on the invariance qualities. For basic data attributes, e.g., rotation, given a data object x , a sequence of “largely similar” data can be easily generated and sorted by how many degree x is rotated. The main aspects of this work were published in ICCV 2025 [169].
- **Technical Chapter 3:** In the third technical chapter (i.e., Chapter 5 in this thesis), unlike these basic data attributes, for background invariance testing,

it is not trivial to sort, rank and order the testing data. Therefore, in the third technical chapter, we would introduce a novel semantic-aware and self-explainable scene similarity method. For example, a river is semantically closer to a mountain (as they are both nature) than a building. In existing studies of data similarity, for a given reference image, only visually close images are recalled while semantically related but visually different images are distanced as far as totally irrelevant images. However, labelling image similarity is not always possible. Therefore, we introduced a novel ensemble approach to combine several proxy models. The main aspects of this work were published in KDD 2024 [168].

In the three technical chapters, we propose an automatic invariance testing framework for ML models for different types of data attributes (from basic to complex), cover the challenges we listed above, and also discuss future research topics on invariance testing for other types of variables.

1.5.1 Published Papers and Technical Reports

Some chapters of this thesis are published as research papers at AITEST, CIKM, ICML, KDD and ICCV. The published papers and technical reports related to this thesis include:

- **Zukang Liao**, Pengfei Zhang and Min Chen. “ML4ML: Invariance Testing for Machine Learning Models”, 2022 IEEE International Conference On Artificial Intelligence Testing (AITest).
- **Zukang Liao**, and Michael Cheung. “Automated Invariance Testing for Machine Learning Models Using Sparse Linear Layers”, International Conference on Machine Learning (ICML) 2022.
- **Zukang Liao**, and Michale Cheung. “Invariance Testing and Feature Selection Using Sparse Linear Layers”, Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM) 2022.
- **Zukang Liao** and Min Chen. “Background Invariance Testing According to Semantic Proximity”, International Conference on Computer Vision (ICCV) 2025.
- **Zukang Liao**, Yuanzhe Jin*, Zelin Ye*, Min Chen, “Survey on Invariance Testing”, Arxiv, 2024.

- **Zukang Liao**, Min Chen, “Image Similarity using an Ensemble of Context Sensitive Models” ACM SIGKDD international conference on Knowledge discovery and data mining 2024.

To conclude, the main objective of the thesis is to revisit invariance testing problem. As invariance qualities are desired in many fields and many applications, we find it necessary to advance invariance testing; The testing procedure should go beyond simply reporting one single averaged accuracy score. Therefore, we propose an automatic invariance testing framework that takes multiple factors into consideration. Meanwhile, we also studied several other techniques that can be used to assist invariance testing especially when some complex invariance qualities cannot simply fit into the generic invariance testing framework.

Chapter 2

Related Works

2.1 Overview of Related Works

Recently, deep learning and machine learning (ML) models have been advanced and have achieved better performance than human beings in many fields, e.g., computer vision [314] and natural language processing (NLP) tasks [303]. ML models are continuously improved and deployed for various applications in real life via different strategies, e.g., active learning [252], redesigned architectures [182], tricks on loss function [325] and etc. However, the testing part for ML models had been largely overlooked by the community; in most cases, the testing part only included statistical score(s) that were reported on a testing set, e.g., accuracy or f1 scores. In the most recent years, ML testing has attracted more researchers' attention as more and more ML models are being deployed for tasks that require a higher level of trust and security, e.g., self-driving cars [355] and surveillance [290]. In this survey, we focus on the testing procedure of an important quality of ML models: invariance.

Invariance qualities are desired in many fields, e.g., object detection [180], image matching [187] and etc. Especially, invariance to affine transformations had attracted the most attention in computer vision tasks before ML models became prevalent, e.g., the SIFT [221] algorithm. Invariance qualities to affine transformations are one of the most common properties to show ML models' robustness, efficacy and trustworthiness. In this survey, we start with the invariance qualities to affine transformations, and then extend the concept of invariance qualities to multiple other different types of variation, i.e., variables, e.g., background invariance. Unlike affine transformations, some invariance qualities cannot be easily defined mathematically. Alternatively, some were not known as a type of variation. In this survey, we group common types of variations, summarize how they are defined in the literature, and discuss how they should be framed into an invariance testing problem. We also acknowledge

that invariance qualities are not always wanted, e.g., segmentation models should not have size or scaling invariance qualities [205]. However, when any invariance qualities are required, e.g., size invariance for pedestrian detection models [23], testing how invariant the models are becomes indispensable. Furthermore, for any model handling any tasks, it is almost always the case that the model should at least be invariant against one or more types of variation. Therefore, testing for invariance qualities is important and necessary.

As the techniques of training ML models have been constantly advanced every year, more and more trained ML models are claimed to be robust and invariant. Therefore, it is meaningful and necessary to continue and advance the study of invariance testing. So far, in the literature, when a model is claimed to be invariant, in most cases, only a formula-based score, e.g., averaged worst-case accuracy, supported the claim. However, using statistical scores only cannot fully reflect models' property or performance. This is similar to the Anscombe's quartet [283] shown in Figure 2.1 where four sets of points have exactly the same statistics, e.g., mean, variance, correlation etc, while they are distributed differently. Similarly, testing for any ML models, especially invariance testing, should go beyond simply reporting statistical scores.

In the literature, several academic works that are strongly related to invariance testing are not traditionally referred to as invariance testing problems. For example, Paul and et al. used a styleGan to explore "hidden factors" to test a model's performance [236]. Similarly, SCEGRAM [226] removed one or a few item(s) from a scene to study "syntactic inconsistencies". In the existing studies, multiple terms and terminologies were used to describe invariance testing problems. Therefore, for the completeness of the research of invariance testing and the facilitation of future works, it is beneficial to summarise existing works related to invariance testing and re-frame them into invariance testing problems when necessary. In this survey, we also summarise techniques that can be potentially used to define or assist invariance testing in a systematical way. For example, it is implausible to enumerate all possible combinations of pixels to carry out background invariance testing. However, the testing procedure can be neatly re-organised with the assistance of semantic scene understanding techniques [169].

In this literature review chapter, we 1) introduce and extend the concept of invariance testing, 2) summarise existing variables and testing workflow, and 3) discuss potential research directions. The contribution of this chapter can be summarized as follows:



Figure 2.1: Anscombe’s quartet: four sets of points which have the same statistical measures, e.g., mean, standard deviation, correlation, etc. However, they are differently distributed.

- Introduce and extend the concept of invariance testing for ML models for the first time in the literature.
- Reviewing and classifying existing variables that are either not commonly known as an invariance problem or cannot be easily defined in a mathematical way.
- Providing a review of benchmark datasets for the variables.
- Reviewing and grouping existing testing frameworks. Providing a taxonomy of ML testing workflows which includes but not limited to reporting statistical scores and testing assisted by visualization.
- Reviewing input generation and mutation control techniques that can potentially assist invariance testing.
- Discuss a set of possible future research directions on invariance testing.

The rest of the survey chapter is organised as follows. In section 2, traditional definition and introduction to invariance testing are discussed. In section 3, we extend the definition of invariance testing and introduce different variations. In section 4, we introduce and group different testing workflows. In section 5, input generation and selection methods are introduced as effective tools to define and carry out invariance testing in a more systematical way. Section 6 summarises potential research directions and concludes the survey chapter.

2.2 Traditional definition

In the literature, the traditional setting of testing a discriminative machine learning model is to run the model on a set of pairs of (x, y) , i.e., testing set, where x represents a data object, e.g., an image, and y represents its annotation(s). For a model $\mathbf{M}$ (to be tested), the metric-based testing score is normally defined as:

$$\frac{1}{m} \sum L(\mathbf{M}(x), y) \quad (2.1)$$

where L is a fitness function, e.g., accuracy, cross entropy or other loss functions [325] and m is the size of the testing set. In terms of a specific type of variation $\mathcal{T}$, e.g., rotation invariance, let $V = \{\tau_1, \tau_2, \dots, \tau_n\}$ be a sequence of transformation operations, e.g., τ_i could be to rotate image by i degree(s). Let $\mathbb{D}$ be the set of all possible data objects that could be the input of the model, which is commonly referred to as the data space of $\mathbf{M}$. The model $\mathbf{M}$ is said to be $\mathcal{T}$ -invariant if $\forall x \in \mathbb{D}$:

$$\mathbf{M}(\tau_i(x)) = \mathbf{M}(\tau_j(x)), \forall \tau_i, \tau_j \in V \quad (2.2)$$

When $\mathbf{M}$ is not $\mathcal{T}$ -invariant, ideally we can assess the level of inconsistency (or $\mathcal{T}$ -variance) with a function Υ :

$$\text{Level}_{\mathcal{T}} = \Upsilon(\mathbf{M}, \mathbb{D}, \mathcal{T}) \quad (2.3)$$

However, it is impractical to obtain the above idealised invariance level because: 1) the data space $\mathbb{D}$ is gigantic, making it impractical to thoroughly traverse, 2) the transformation operations V of the variation $\mathcal{T}$ might not always be continuously and qualitatively defined, and 3) there is no ground truth function for Υ .

Due to the aforementioned limitations, in the literature, when reporting invariance qualities of a model, the following assumptions and protocols are made and used:

- To avoid traversing the huge data space, a subset $X = \{x_1, \dots, x_m\} \subset \mathbb{D}$ is normally used, e.g., a testing set.
- In the literature, when considering invariance qualities, the scope is normally limited to affine transformations which can be quantitatively defined, e.g., rotation, translation. In this work, we extend the definition of invariance testing and thus can include variations that cannot be easily defined.
- Researchers normally adopt the traditional setting of testing a discriminative model and obtain a testing score based on Eq (2.1).

To measure the inconsistency between two predictions of model $\mathbf{M}$, one can define a fitness function f similar to the loss function in Eq(2.1). With f , one may approximate Υ with the following simplistic function Φ :

$$\Phi(\mathbf{M}, X) = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^n f(\mathbf{M}(\tau_j(x_i)), M(x_i))$$

When $f = 1$, x_i and $t_j(x_i)$ are consistently predicted (or classified), i.e., $\mathbf{M}(x_i) = \mathbf{M}(t_j(x_i))$, and when $f = 0$ the contrary. In the literature [87, 339], to align with

the traditional setting of testing discriminative models, i.e., Eq(2.1), f is normally associated with correctness. To be more specific, $f = 1$ when $\tau_j(x_i)$ is correctly classified / predicted, i.e., $\mathbf{M}(\tau_j(x_i)) = y_i$, and $f = 0$ the contrary. Such evaluation function Φ is normally referred to as *robust accuracy* or *worst-case accuracy*, where $Y = \{y_1, y_2, \dots, y_m\}$ is the ground truth annotations.

$$\text{Worst-case (Robust) Accuracy: } \Phi(\mathbf{M}, X, Y) = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^n f(\mathbf{M}(\tau_j(x_i)), y_i) \quad (2.4)$$

While in general we can define invariance qualities as Eq(2.2), invariance problem can be strongly related to the study of causality [24,241] which considers data obtained under different environments, e.g., perturbations, experimental settings or different types of variations in our case: $(X^e, Y^e), e \in \varepsilon$. Note that when $e \in V \subset \varepsilon$ it becomes the invariance problem specified above. In some special cases, e.g., when considering translation invariance qualities for object tracking models, the annotations Y^e will change according to the variation. In this case, *equivariance* can be considered $\forall x \in \mathbb{D}$:

$$\mathbf{M}(\tau(x)) = \tau(\mathbf{M}(x)), \forall \tau \in V \quad (2.5)$$

Similarly to invariance testing, in the literature, the level of equivariance is normally evaluated by *worst-case accuracy* and *robust accuracy*. When the correct or expected prediction(s) of $\mathbf{M}$ for a transformed input data object $\tau(x_i)$ is $\tau(y_i)$, the *worst-case accuracy* and *robust accuracy*, i.e., Eq(2.4), in terms of equivariance becomes:

$$\Phi(\mathbf{M}, X, Y) = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^n f(\mathbf{M}(\tau_j(x_i)), \tau(y_i))$$

Although it might not always be feasible or reasonable to obtain $\tau(\mathbf{M}(x))$, e.g., outputs for classification models cannot be transformed, equivariance is beneficial for many applications, e.g., pedestrian tracking [23] or surveillance system [290] where annotations are expected to be transformed in the same way as input data objects. However, both invariance, i.e., Eq(2.2) and equivariance, i.e., Eq(2.5) are strict requirements for any models to satisfy. Therefore it is inevitable to measure the level of inconsistency, i.e., Eq(2.3). In this survey, we list existing attempts to evaluate invariance qualities for machine learning models, including but not limited to formula-based approaches, fuzzing testing frameworks, visualization-based methods and theory-based analysis.

Group theory is also strongly related to invariance testing, especially when defining the sequence of transformation operations V [39]. This is because when evaluating invariance qualities, in the literature, normally only affine transformations were taken

into consideration. A *group* G is a set with an operation (denoted as “ $\cdot$ ” : $G \times G \mapsto G$) that combines any number of elements in the set and output another element within the set; $\forall g, h \in G$ we have $g \cdot h \in G$. It is associative, i.e., $(f \cdot g) \cdot h = f \cdot (g \cdot h)$. An identity element e is defined as $g \cdot e = e \cdot g = g$. And for an element g , its reverse is defined as $g^{-1} \cdot g = g \cdot g^{-1} = e$. A group action is defined as $\phi(g, x) : G \times X \mapsto X$, for a data object $x \in X$ and an element of the group $g \in G$. According to its associative property, we have $\phi(g, \phi(h, x)) = \phi(g \cdot h, x)$, where $\forall g, h \in G$ and $x \in X$, or more commonly written in the following format:

$$g \cdot x := \phi(g, x)$$

$$g \cdot (h \cdot x) = (g \cdot h) \cdot x$$

Given the group action ϕ , the orbit of $x \in X$ is defined as the set of all possible transformed data object: $G(x) = \{\phi(g, x), \forall g \in G\}$. When considering affine transformations, the orbit and group actions can be clearly and mathematically defined under group theory. And the aforementioned set of transformations V is aligned with the group action ϕ and the group G .

In invariance testing, the input X could be any form for any ML model architectures, e.g., convolutional neural networks (CNNs) [182], transformers [114] or graph neural networks (GNNs) [336]. We use the general term X to represent the inputs while it could be a tuple of inputs for GNNs or any other architectures of models. For example, for a GNN, $X = \{A, \mathcal{X}\}$ where $\mathcal{X}$ is the actual data objects and $A \in \mathbb{R}^{|\nu| \times |\nu|}$ is the adjacency matrix where $|\nu|$ is the number of nodes in the GNN, and $a_{u,v} = 1$ when there is an edge between node u and node v , $a_{u,v} = 0$ otherwise. In this case, the transformed data objects $\tau(x)$ becomes:

$$\tau(X) = \{PAP^\top, P\mathcal{X}\}$$

where P is the permutation matrix corresponding to the transformation operation τ . Similarly, $\mathbf{M}(x)$ can be output from $\mathbf{M}$ given input data object x at any intermediate locations or any other forms for multimodal ML models [18] as long as the transformations can be hypothetically defined.

Invariance testing can go beyond simple affine transformations. In this survey, in addition to affine transformations, we group and categorise existing variations that were not normally referred as invariance problems. As shown in Figure 2.2, in this survey, we first discuss invariance qualities (variables) in the literature, and how the transformation/mutation is controlled, e.g., $X = \{x_1, \dots, x_m\}$ can be generated by

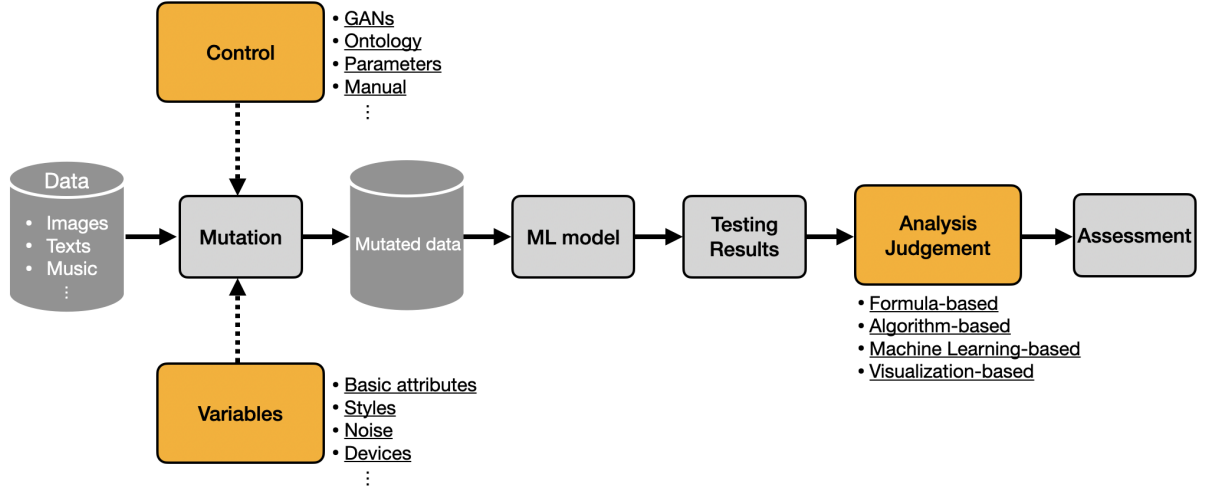


Figure 2.2: Overview of invariance testing: data are mutated based on: 1) which invariance quality is desired (variables) and 2) how the mutation/transformation is controlled. After the mutated data are sent to the ML model (to be tested), different analytical methods can be used, e.g., formula-based (accuracy) or visualization-based (e.g., XAI).

applying different parameters to the original data object or can be generated by using GANs. Finally, after the mutated data are sent to the ML model (to be tested) and testing results have been obtained, we discuss several methods to conduct the analysis (invariance test).

2.3 Variables

In this section, we introduce invariance qualities from different perspectives and revisit existing properties of ML models and re-frame them into invariance testing problems. We start with the most common variations, e.g., affine transformations, and then categorise existing variations into different groups. Finally, we also list uncommon variations that normally only fit into some specific applications. We bring in as many variables as possible, meanwhile, we also acknowledge that there are variables that do not fit into any of the categories. As shown in Figure 2.3, in this section, we group variables into four categories, i.e., basic attributes, style attributes, variables that can be introduced during data capturing, and noise/imperfection.

2.3.1 Basic data attributes

For image data, geometric data attributes are the most common variables for which most ML models are trained to be robust. In this section, we begin the discussion

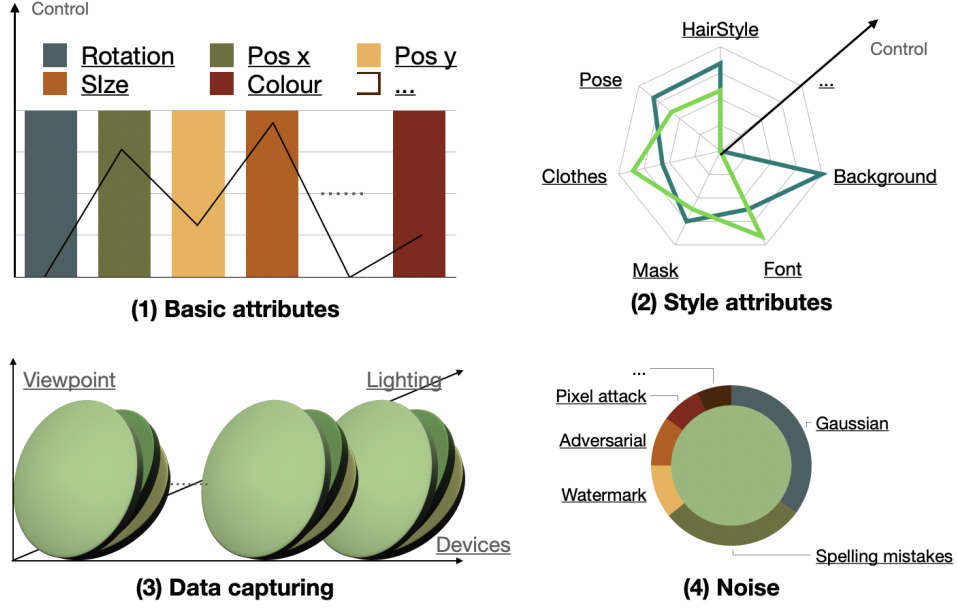


Figure 2.3: (a) Basic variables, e.g., rotation angles, that can normally be used for data augmentation. These basic variables can normally be controlled by parameters, e.g., rotate 10° . (b) Style variables, e.g., clothes, are normally controlled by generative techniques, e.g., GANs. (c) Variables that are introduced when capturing the data, e.g., different devices, lighting conditions and viewpoints. (d) Noisy data: e.g., adversarial attacks or spelling mistakes in NLP.

on **invariance testing** for these basic geometric variables. And we refer interested readers to a survey [284] which summarizes using these data attributes as data augmentation in the **training** phase.

- **Flipping:** i.e. to flip a given image horizontally (x_h), vertically (x_v) or mirrored flipping (x_m). When testing data object(s) are semantically symmetrical (which is always almost the case), the flipped counterpart represents a realistic variation of the original one. This way, given a data object x , one can generate $\{x_h, x_v, x_m\}$ to conduct invariance testing when flipping-invariance qualities are required.
- **Rotation:** usually refers to image rotation, object rotation, or 3d rotation. The most common approach to generate testing data for rotation-invariance testing is to rotate the original images by a predetermined angle. A testing suite can be defined as $\{x_{-a^\circ}, \dots, x_{0^\circ}, \dots, x_{a^\circ}\}$ for a given testing interval $[-a^\circ, a^\circ]$ and the sampling rate, e.g., every 1° . In the literature, many works focused on improving rotation invariance qualities in the training phase whilst they only reported one worst-case accuracy score (eq. 2.4) in the testing phase. Some

studies attempted to theoretically explain how the invariance qualities were acquired. For example, Benjamin Chidester et al. [47] introduced that 2D-DFT can be a mathematical foundation, [44] introduced a rotation-invariant regularizer which can theoretically align rotated and original features. Another more complex way to conduct rotation-invariance testing is to rotate the target objects only while keeping the background intact, e.g., 3D rotation [209]. In 3D rotation, the common way is to apply a virtual depth, so that a 2D image can therefore be rotated in any of the 3D direction. Commonly used methods include 3DPI [282] and MiDas (depth estimation for 3DPI) [251]. For example, a virtual depth can be used for medical image processing, which does not change the semantics [154] but provide more information for the target objects.

- **Size:** includes zooming in/out two dimensional images, object scaling (including shrinking/expanding of the object or various distances to the camera). For example, [289] evaluated size invariance qualities for different ML models that were trained under different training strategies, e.g., whether super-resolution images are provided. Although some size invariance test can be extreme, e.g., extremely small or large, the morphed images should contain the same semantic meaning; from the transformed/morphed data object, humans can still recognise the same content. The size invariance is considerably important in object detection and pedestrian detection. In the literature, various studies were conducted to improve size invariance for these tasks, e.g., image pyramid [1, 192, 329].
- **Translation:** Translation invariance property is desirable in tasks such as object recognition, where the position of objects may vary in different images. In the context of signal processing or image analysis, an image can be represented as a function of two variables: $f(x,y)$, where x and y denote the horizontal and vertical positions, respectively. If an ML model is translation-invariant with respect to both x and y , it means that the output remains unchanged under translations along both axes. In the literature, invariance qualities are tested empirically and theoretically. In Section 5.3 we summarise how Nyquist theorem can be used to support translation invariance testing [365].
- **Cropping:** Cropping images can also preserve information or labelled objects depending on the size of cropping [284]. The two most common cropping approach is 1) crop four corners and centre of the image, 2) random cropping. Both of them are assumed to be able to keep the original semantics. After

applying the cropping operation C to the original image, we obtain a cropped image, denoted as $C(x,y)$, which contains only the selected sub-region or crop. Similarly to translation invariance, the variables x and y in $C(x,y)$ represent the coordinates within the cropped image.

- Color space transformation: as the image samples are coded using RGBA color tensor, color space transformation can simply use one color channel to generate new images. Based on that, simple matrix can easily manipulate the brightness, contrasts and other characteristics of images. Without changing the semantics, it can filter out some of the noise and lower the computational cost and improve the model performance [37, 136]. However, for color-sensitive object such as blood, changing the dark red color to other color may result in the change of semantics and the model should not be color-invariant [350].

In terms of audio data, various tailored techniques can be used for invariance testing in different areas, such as speech recognition [147], animal sound classification [217], and other general audio processing techniques [69, 299]. These different applications and areas normally require different types of invariance qualities, we summaries the most common invariance qualities for ML models trained or partially/jointly trained on audio data.

- Time stretching (speed change) is the process of varying the length of an audio signal without changing its pitch [328]. Combined with random filtering, this technique can considerably improve the quality of the testing data [269]. Note that in audio processing, time stretching normally also cause changes in relatively pitch and the music content itself, e.g., the harmonics. When conducting time stretching invariance testing, one should make sure the transformation (the generation of testing data) should not involve other variables. Time stretching is important for speech recognition, e.g., dysarthric speech recognition [310] and automatic speech recognition (ASR) [222]. And another recent technique (speed perturbation [148]) provides more flexibility based on the time-stretching invariance quality within a certain threshold, e.g., $[x0.5, \dots, x0.9, x1.0, x1.1, \dots, x1.5]$ with respect to the original audio speed.
- Pitch shifting: is an audio data augmentation technique that modifies the pitch of an audio signal while preserving its duration. This technique is widely used in audio signal processing to generate new versions of audio signals with varying pitches. Pitch shifting has been researched and applied in many papers [35,

199, 301]. In [199], it includes invariance testing that shows the ML models with pitch shifting quality can achieve better result and provide generalization of audios for $\{-1, 0, +1\}$ semitones, given that the relatively pitch through sequence and the timbre of each instrument is kept unchanged for all the tests in the instrumental recognition tests.

- Volume control: changing the volume of an audio signal to simulate different recording conditions. The magnitude can be manually manipulated and the generated testing data can be used for invariance testing. Most ML models (dealing with audios) are expected to be volume-invariant. Note that when increasing the volume, distortion can happen. Similarly, when decreasing the volume, some semantics can become indistinguishable. Therefore, a reasonable interval should be selected based on the application, e.g., -10db to +10db.
- Testing based on image representation: simply convert the audio signal to the spectrogram and then apply the image augmentation. Commonly used image representation includes the chromagram, mel-frequency spectrogram, and the corresponding self similarity matrix (SSM), recurrence matrix and affinity matrix. ML models trained for various audio tasks can be tested this way, including classification [48, 53, 300], segmentation [80, 198, 200, 264] and recognition [9]. People also tend to manipulate different feature representations while maintaining the same task-related information usually by fusing different self similarity matrix [264, 305]. Although different feature image is combined and weighted averaged, the semantics and segmentation information are preserved.
- Filtering is the process of applying various filters [215] to an audio signal in order to simulate different recording conditions, such as equalisation, filter with moving window or band-pass filtering [217]. Filtering can also be used to process the image-like feature mapping [294] or image data generated from the temporal audio features, such as smoothing the edge, or strength the diagonal to erase some noise information [198]. For audio segmentation, filtering can usually eliminate those segments that are too short, or some random short clip that should not be present. And for clustering-based segmentation, models are supposed to focus on big clusters that covers considerable length which shall not be filtered out. Therefore, ML models are supposed to produce consistent results no matter the original audio data are filtered or unfiltered, i.e., invariant against different types of filters.

- Noise addition can be used to generate additional testing data for ML models which are supposed to be invariant against different levels and types of noise [69, 130]. The additive noise for the testing data can be manually collected white noise or automatically synthesized white noise, and we refer interested readers to a survey on noisy speech recognition [273] where different types of noise were discussed in detail.

2.3.2 Variables of devices / data capturing

Images / videos can be captured by different cameras, and audios can be recorded by different microphones. In addition to the variation caused by different devices, variables of brightness, lighting conditions, viewpoints, shooting angles, occlusion and blurriness can also be present while capturing the data. In the testing regime of these types of variations, one cannot easily define a transformation operation τ in a mathematical or quantitative way. Therefore, instead of having a sequence of operations $\{\tau_1, \tau_2, \dots, \tau_k\}$ for an given data object x , one can test the invariance qualities by using a set of data objects $D = \{x^1, x^2, \dots, x^k\}$ which are “largely similar” but with a certain type of interested variation. In the following parts of this section, we describe what variations have been reported in the literature, and how the “largely similar” data are collected and presented. The variations described in the section normally co-exist with each other, e.g., images captured by different cameras might well have different resolutions, lighting conditions or shooting angles. In Table 2.1, we summarise some typical datasets where these types of variation are present.

(3.2.i) Device and resolution: Robustness against variations caused by different devices are desired in most cases because the semantic meanings are almost always preserved when the same object is captured by different devices. The variation can be carefully introduced by placing different devices at the same position and capturing data at the same time. For instance, in the DPED dataset [124], scenes/objects were captured simultaneously by four cameras, i.e., iPhone 3GS, BlackBerry Passport, Sony Xperia Z, and Canon 70D DSLR. Except for some inevitable variations caused by slightly different shooting places (hence slightly different viewpoints), the variations in the database are mainly caused by devices. Similar datasets can be found in Table 2.1. These datasets are normally collected for super-resolution tasks [12], while in practice, they can also be used to conduct invariance testing.

Although some applications might heavily rely on high resolution, e.g., medical analysis [270] or background matting [176], resolution invariance is crucial for most applications because the semantic meanings can normally be preserved (if the low

resolution does not lead to distortion), e.g., resolution adaptation re-id networks [42] or resolution-invariant feature extractor [195]. The variation of resolution can be introduced by different devices or manually down-sampling the data. The latter is similar to adding Gaussian noise to the original data which will be described in detail later in Section 3.4.

(3.2.ii) Viewpoint and lighting condition: Viewpoint is one of the most common variations in many application, e.g., pedestrian detection [23], person Re-ID [348], surveillance [290] etc. In these applications, the same data objects (e.g., pedestrians) are normally captured by several different devices located (fixed) in different places. Much effort had been made to improve models’ invariance quality, e.g., domain adaptation approaches [179, 250] where “a different camera domain” often indicates variations caused by camera, resolution and most importantly viewpoint. The invariance quality is also desired in many other applications, from the generic tasks like object classification [149, 335], to more specific tasks like 3D Human Pose Estimation [266], robot visual pace recognition / classification [90, 91, 185, 356] and robot manipulation tasks [261]. In addition to placing devices in several random locations, variation in viewpoint can also be carefully introduced. For example, in the ITOP dataset [115], cameras were set to the “front” and “side”, and in the YaleFaceB dataset [94], cameras were set to frontal, 12° and 24° .

Similarly, variation in lighting condition can often occur while capturing data. This can be carefully introduced and controlled by setting up lighting experiments. For instance, in the MUIT dataset [211] and YaleFaceB dataset [94], light bulbs were fixed in multiple locations (in a lighting dome), which provided 25 and 64 unique yet replicable lighting conditions respectively. The variation of lighting condition also naturally exists in any dataset that were captured in the wild, e.g., the ImageNet dataset [260] or the PASCAL VOC dataset [75]. One typical and special case where lighting variation and device variation coexist is: images captured by normal standard camera and infrared camera, e.g., [364].

(3.2.iii) Occlusion and blurriness: Blurriness can be caused by object motion, camera lens focus, shaky hands or camera blemish etc. Recent work collected a large dataset (SmartBlur [368]) consisting of 10k images with annotation of desirability of blurriness. Backgrounds can be blurred on purpose so that foreground objects can be more eye-catching. In practice, in most cases, blurriness (even for those unwanted ones) does not change the semantic meaning of the original data. Therefore, ML models should be invariant against blurriness and capture the original semantic information, e.g., [143]. For this reason, in a special case, ML models should also be able to conduct

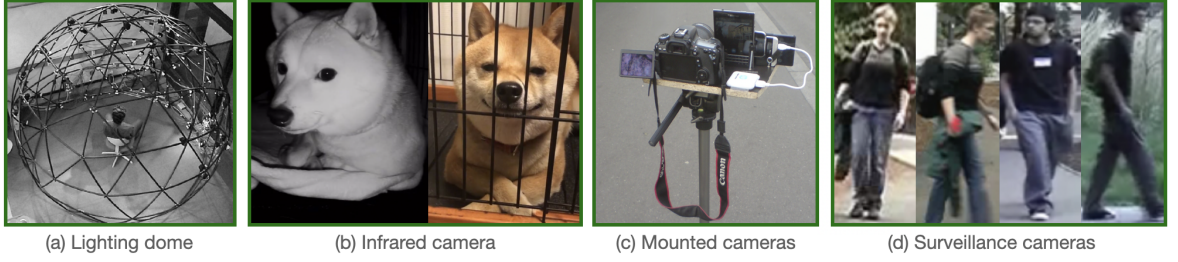


Figure 2.4: For variables of devices and data capturing, testing data are normally manually collected. (a) A lighting dome that precisely introduced replicable lighting variation. (b) Infrared camera: a combination of viewpoint, device, and most importantly lighting variation. (c) Cameras were mounted tightly together (almost no viewpoint variation) and images were captured simultaneously [124]. (d) Cameras were located in different places for person ReID [105]. Figures are re-created, re-sampled or re-collected based on the ideas of the papers.

Table 2.1: Existing datasets that contain “largely similar” data under variations of devices / data capturing

Dataset	Size	Devices	Resolution	Viewpoint	Lighting	Occlusion	Blurriness	Source
Market1501 [369]	1501	2 - 6	2	2 - 6	Yes	Yes	Yes	ReID
CUHK03 [162]	1,467	6 (2 each)	Yes	6 (2 each)	Yes	Yes	Yes	ReID
DPED [124]	22k	4	4	Almost No [†]	No	No	No	Super-resolution
CameraFusion [326]	146	2	2	Slightly	No	No	No	Super-resolution
ITOP [115]	110k	multiple	multiple	front/side	No	No	No	3D Pose
ModelNet [335]	660	1	1	multiple	No	No	No	3D classification
Stanford 3D [149]	207	1	1	multiple	No	No	No	3D classification
MUIT [211]	1016	1	1	No	25	No	No	Illumination prediction
YaleFaceB [94]	38	1	1	9	64	No	No	Face recognition
VAIS [364]	1088	2	2	Almost No [†]	2	No	No	Infrared Camera
SmartBlue [368]	10k	multiple	multiple	multiple	Yes	No	4	Image Blur
OccludedPASCAL3D+ [316]	10812	multiple	multiple	multiple	Yes	9	No	Invariance Test

[†] Devices are tightly mounted together as shown in Figure 2.4 (c).

the task of de-blur, e.g., [223, 361]. Similar to the variation in resolution, blurriness can be manually introduced by adding Gaussian noise to the data (see Section 3.4).

Occlusion variation can be manually introduced by overlaying a texture patch [256]. It can also happen naturally, e.g., pedestrians can be occluded by trees, cars, lamp post or other people [127]. Unlike the aforementioned variations, occlusion might change the semantic meaning of the original data. Therefore, when introducing variation in occlusion, one should deliberately exclude or annotate those that are heavily affected. For instance, in the OccludedPASCAL3D+ dataset [316], nine occlusion levels were included, namely all combinations of three levels of foreground occlusion and three levels background occlusion.

The evaluation of to what extent the variations (**B**lur and **O**cclusion) affect the semantic meaning of the original data remains as an open research direction. In the

literature, the following approaches were used:

- **B/O**: By number of affected pixel (foreground, background) [256, 316, 368]
- **B**: By spectrum information, e.g., local power spectrum slope [181], Laplacian [228].
- **O**: By semantic parts of data objects, e.g., [172, 362] occludes part(s) of a face each time, e.g. mouth, eyes etc.

2.3.3 Data Quality: Imprecision (errors) and Noise

In addition to the variables introduced during data capturing, imprecision and noise can also appear in the collected data. Imprecision usually refers to lack of exactness or accuracy in written, image, audio, or video information. Humans are prone to making mistakes when performing repetitive or intricate tasks. Even individuals with excellent skills can occasionally make errors due to lapses in attention or motor control. For example, in text analysis, misspelling errors occur when words are spelled incorrectly, often as a result of typos or a lack of knowledge about proper spelling. Punctuation errors can also occur when comma, period, and other punctuation rules are not followed correctly. Context, grammar, and translation errors are also possible, especially when writing in a second language or dealing with complex sentence structures. Note that when words or phrases are placed in the incorrect part of a sentence, it can cause confusion or ambiguity. However, most ML models are desired to be tolerant of common minor and trivial human errors, i.e., imprecision-invariant.

Noise is another variable that can affect the quality of the collected data. Unlike unintended imprecision, noises can be deliberately added into the originally collected data as extra information for some purpose like making a watermark or adversarial attack. In the following section, we will discuss the common errors and noise related to invariance testing.

2.3.3.1 Imprecision and Errors

- Imprecision (Errors) in writing, spelling, pronunciation

Invariance testing for writing, spelling, and pronunciation aims to assess the robustness of a system or model to variations in how words are written, spelled, or pronounced. To study the invariance qualities for algorithms/models, in the literature, Alhaisoni et al. [10] analyzed spelling errors found in writing samples of Saudi beginner

Table 2.2: Representative Datasets containing data with imprecision, errors and noises

Dataset	Variable (Description)	Number of Examples	Collection or Generation
BEA-60K [132]	Spelling Mistakes	70K spelling errors from 63,044 sentences	Collected and Generated
FCE [140]	Grammar Mistakes	450k tokens	Generated
Github Typo Corpus [113]	Spelling and Grammar Mistakes	350k edits and 65M characters	Collected by “fix typo” commits
Accent Dataset [281]	Accent in English	200 Hours, 10 accents	Collected
Off-Pitch Singing [315]	Amateur singing performance	4702 separate vocal tracks	Collected
Corrupted LRS2 [120]	Babble Noise	114k utterance	Generated

learners of English and classified them into four categories according to Cook’s Classification: omission, substitution, insertion, and transposition. The study found that errors of omission constituted the highest proportion of errors and that the majority of learners’ spelling errors were related to a wrong use of vowels and pronunciation. And Toutanova et al. [304] presented a method for incorporating word pronunciation information in a noisy channel model for spelling correction. The proposed method built an explicit error model for word pronunciations. By modelling and testing pronunciation similarities between words, the authors established a substantial testing mechanism for the performance of spelling corrections (or spelling-invariance quality). In this case, errors can be detected [10] by looking at the invariance properties based on the use of vowels, typical pronunciation etc.

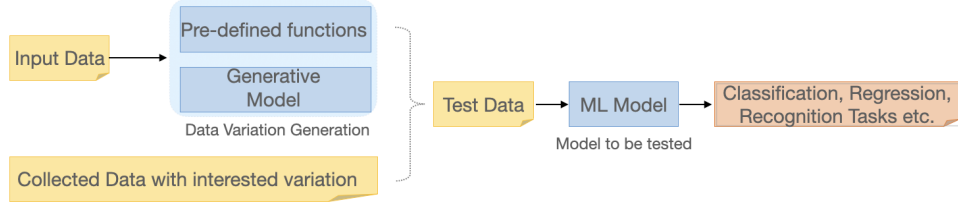
- Cross modal: Video audio mismatch and asynchronism

When models are handling data in different forms, multi-modal errors can happen [321]. This type of error can occur due to a number of factors, such as encoding issues, synchronisation issues during the recording process, or a mismatch between different formats of the data. Multi-modal models are expected to be robust against these types of variable. For example, models’ performance should not be significantly affected when the audio lags behind or ahead of the video within a reasonable range. In the literature, these types of errors are normally detected/corrected via pre-processing or a separated module, e.g., alignnet is trained to align audio and video clips [318]. The designated modules are normally tested on some manually collected dataset(s), e.g., Dance50 [318].

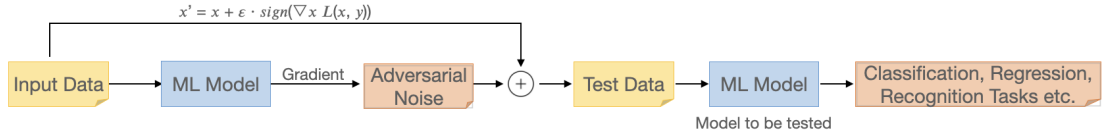
2.3.3.2 Noise

As shown in Figure 2.5, invariance testing pipelines for noise can be different from other types of invariance qualities, depending on how testing data are generated. For example, the generation of noise might require access to the original model (to be tested). The generated noises can be added into the original data.

Invariance Testing for most Variable (not related to data quality/noise)



Invariance Testing for Adversarial examples



Invariance Testing for Watermark attacks

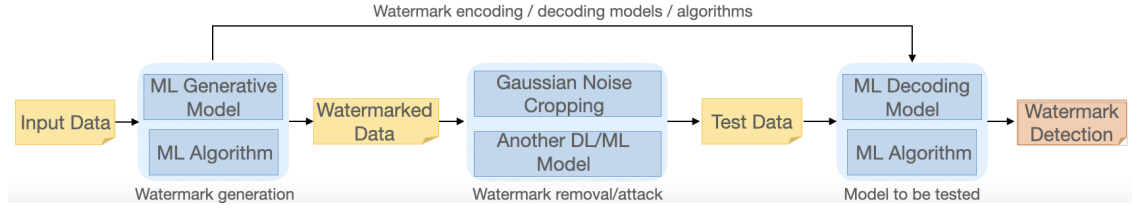


Figure 2.5: Testing pipeline for noise related variables is different from most other types of variables. Adversarial examples involves obtaining gradients from the original ML model (to be tested) to generate additive adversarial noise. Instead of adding noises, watermark attacks are removing encoded watermarks from watermarked data, which requires a different testing pipeline for ML models/algorithms.

- Gaussian noise

In nature, Gaussian noise is a kind of random and regular data noise, which can be used to simulate the uncertainty and noise of data in practical applications [313,376]. Gaussian noise invariance testing evaluates the sensitivity of the model to random disturbances in the data [268]. Unlike the aforementioned traditional/common variables in Section 3.1, e.g., rotation, the generated testing data are controlled by the three parameters: [30]

$$\hat{\mathcal{X}} = \mathcal{X} + \lambda * \mathcal{N}(\sigma, \mu)$$

where λ controls the amplitude, σ and μ control the distribution of the additive Gaussian noise.

- Adversarial examples / attacks

An adversarial attack (example) refers to a deliberately synthesized/generated input that can mislead a trained model. Adversarial training injects these adversarial

examples during the training process to boost the robustness of the trained ML models. Similarly to the Gaussian noise, the generated adversarial noise can be treated as a variable as a whole, and invariance testing can be conducted by the invariance testing methods/frameworks introduced in Section 4, e.g., by observing the change in models’ performance whilst varying the magnitude of the additive adversarial noise. In this survey chapter, we briefly introduce adversarial examples and adversarial training/testing. And we refer interested readers to a more comprehensive survey on adversarial attacks and adversarial training [7].

The adversarial attack has different implementation principles [38, 72, 232], most of which are to add noise that is not easily observed by people to the original data to affect the prediction results of the machine learning model. Adversarial attacks can be divided into gradient-based attacks and non-gradient attacks from the attack method [8]. From the perspective of invariance testing, both attack methods change the original data to a certain extent by adding noise or interference, which makes the prediction of the machine learning model deviate.

In 2014, the Fast Gradient Sign Method (FGSM) was first proposed by Goodfellow et al., which explained the basic principles of adversarial examples and proved that the state of high-dimensional networks is linear. Research shows that the high-dimensional nonlinearity of networks is responsible for adversarial attacks in networks [100]. After their work, other researchers successively proposed many derivative algorithms based on FGSM, such as RFGSM [307], NI-FGSM [175], M-DI2-FGSM [341], etc., making FGSM It can carry out more efficient black-box attacks and improve the attack success rate of its white-box attacks. In addition, Projected Gradient Descent (PGD) [194] is an iterative optimization algorithm that generates adversarial examples by continuously optimizing the loss function. Carlini and Wagner (C&W) Attack [33] is an effective adversarial example generation method, which generates adversarial examples by solving an optimization problem. DeepFool [207] is a method of using linear programming to construct adversarial examples, which can quickly generate adversarial examples. For different adversarial attacks, Yao et al. compared the performance of various adversarial attacks on different machine learning models [163].

Adversarial training for adversarial attacks can improve the pre-resistance of machine learning models against adversarial attacks to a certain extent without additional cost [276]. The paper [17] reviews recent progress in adversarial robust adversarial training using novel taxonomies. One thing to note is that adversarial training does not always have a positive effect on the improvement of model invariance testing.

In adversarial attacks, perturbation-based adversarial examples are not robust enough to achieve general robustness in some cases. A more precise definition is needed for general robustness [129]. In the paper [137], some machine learning models are improved in rotation invariance by gradually increasing the training enhancement of rotation, but while doing so, their adversarial robustness does not improve, or worse, it can even drop significantly on datasets like MNIST. In his article, Bühlmann analyzes the relationship between Invariance, Causality, and Robustness, using statistical methods for more robust and better "causality-oriented" explanations [28].

The relationship between robustness against adversarial attacks and other types of invariance qualities also drew some researchers attention. [107] investigated the trade-off between adversarial robustness and accuracy in neural networks by adding additive noises to either the input layer or all hidden layers. They studied the recovery of adversarial examples and misclassification of clean examples as the amount of noise varied (i.e., invariance testing), to understand the model's robustness [107]. Similarly, research [306] demonstrates fundamental trade-offs between robustness levels of two different types of adversarial examples. They find that defending against sensitivity-based attacks actively compromised model accuracy against invariance-based attacks. [351] found that affine transformation invariance qualities are closely related to adversarial robustness [20, 89]. However, being invariant to affine transformations does not guarantee robustness against all types of adversarial attacks [351], as there are other types of adversarial attacks that do not correspond to affine transformations.

- Watermark attacks (noise / cropping)

Watermarks are divided into traditional physical watermarks and digital watermarks [3]. A digital watermark is a marker covertly embedded into a noise-resistant signal such as audio, video, or image data [244]. It is often used to identify copyright ownership of such data. R.G. van Schyndel proposed digital watermarking in December 1994 [311]. Like traditional physical watermarking, digital watermarking is usually only perceptible under certain conditions, and since the digital copy of data is identical to the original data, digital watermarking is a passive protection tool [242]. In the Internet era, people use software watermarks to embed some secret information in the software as a sign of the copyright owner of the software to protect the software [375]. For an ML model or algorithm that are designed to encode/decode the watermarks for copyright protection, its performance should not be affected by any sort of attacks, e.g., noise or cropping. For example, when an image

with watermark is cropped or added with some noise, the ML algorithm/model should still be able to identify it is a watermarked image. For example, [49] introduced a transform-invariant public fragile watermarking scheme for 3D objects authentication. The proposed scheme embeds the watermark into 3D objects, making the 3D objects invariant to translation, rotation, and uniform scaling operations.

For generative ML models, generated outputs (images/videos/3D objects) can also be watermarked. Given a generative ML model to be protected, an invisible watermark was added into its outputs. When another ML model tried learning from its input-output pairs, the watermarks will be learned instead of the real semantic contents. For these generative ML models, invariance testing is important to guarantee the generated outputs' watermarks will still be effective under different types of attacks, e.g., affine transformation/noises/cropping. For example, [333] used a combined loss function so that the trained ML models are able to generate watermarked images. The authors conducted cropping (pre-defined cropping strategies, see Section 3.1) and noise (controlled by a standard deviation as shown in Section 3.3.2) invariance testing to ensure the robustness of the proposed methodology. They used the Peak signal-to-noise ratio (PSNR) between the original watermark and the watermark reconstructed from a cropped marked image to evaluate how strong the cropping attacks are. Similarly, [359] introduced a watermark protection framework for ML models. The authors conducted occlusion, noise (based on discrete cosine transform (DCT) and discrete wavelet transform (DWT)), adversarial noise (by surrogate models) invariance testing to ensure the trained watermarked generative ML models have the corresponding invariance qualities.

In addition to traditional watermark attacks, e.g., cropping, noise, other more advanced techniques can be used to test the invariance qualities of ML models that involve encoding and decoding watermarks. For example, Wdnet [183] introduced a watermark-decomposition network for visible watermark removal, as well as its anti-attack techniques to boost the invariance qualities in watermark attacks. They evaluated the proposed method on the Large-scale Visible Watermark Dataset (LVW) [43] which contained 60K watermarked images with 80 different types added watermarks. The authors also introduced a newly collected dataset CLWD (Colored Large-scale Watermark Dataset) which includes 60K watermarked images with 160 colored watermarks for training and 10K watermarked images with 40 colored watermarks for testing. Similarly, REFIT [41] introduced a unified watermark removal framework for deep learning systems using only a limited amount of data. The authors used



Figure 2.6: (a) Subjects (hairstylists) were captured by camera with various hairstyles [146]. (b) [29] overlaid face masks using facial landmarks. (c) - (e): Variation in clothes (ReID [320]), time (different years [118]) and fonts (SVHN dataset [219]) respectively. (f) Variation of gender can be obtained using generative algorithms, e.g., GANs [236]. (g) Replacing background for the same object [169, 338].

elastic weight consolidation (EWC) algorithm adaptation and unlabeled data augmentation to improve effectiveness. Evaluation of REFIT against diverse watermark embedding schemes. They used their watermark removal technique to test the watermark attack invariance qualities of existing algorithms, including pattern-based techniques [40, 108, 358], embedding samples drawn from other data sources as the watermarks [2], exponential weighting [216], and adversarial frontier stitching [158]. And they found that all of the ML algorithms were vulnerable against watermark attacks.

2.3.4 Styles attributes

In the literature, some attempts of invariance testing were carried out on variation of styles. For example, [236] used a StyleGan [134] to explore hidden factors (variation) that the model is sensitive to. Similarly, many self-explainable models demonstrate its interpretability through the variation of styles [58]. As mentioned in the beginning of this section, the level of desirability of one invariance quality heavily varies, especially for different applications. This is especially true for the variables of styles. Some variables of styles can also be interpreted as properties / features / attributes. Therefore, in some cases, instead of invariance, sensitivity to the variables can be more desired. For examples, hundreds of models were trained to conduct age, and gender classification [172, 308]. In these cases, age / gender are attributes that need to be recognised

Table 2.3: Existing datasets that contain “largely similar” data under variations of styles

Dataset	Source	Size	Variation
MaskNet [29]	Face mask detection	137,016	Face mask
K-HairStyle [146]	Hair style classification	500,000	Hair style & viewpoint
NKUP [320]	ReID	9738	Clothes & viewpoint
RFW [323]	Fairness	3000 individuals	Ethnicity
SVSF [118]	Out-of-Distribution	10k	Year, country, viewpoint
ImageNet-9 [338]	Invariance testing	50k	Background

and detected. However, when ML models are not specifically trained for detecting / distinguishing these attributes, the invariance qualities are indeed important. For example, gender and race invariance qualities are critical for face recognition models [151]. In this section, we summarise existing variation of styles in the literature, and we also include other research directions where there are strong connections to invariance testing. For other research directions, we defer interested readers to holistic surveys, e.g., fairness [201], domain adaptation [322], out-of-distribution [346], representation bias [278] etc.

For variables of styles, some attempts of invariance testing (including those research directions that were not traditionally referred to as invariance testing, e.g., the study of fairness) can be different from the Eq.2.2 we specify in Section 2. Instead of having a sequence of transformation operations $\{\tau_1(x), \tau_2(x), \dots\}$ for one given input data object x , several sub datasets $\{D_1, D_2, \dots\}$ can be formed based on the variables or attributes being present or absent, where each subset contains distinct and non-overlapping data objects, i.e., $D_i \cap D_j = \emptyset$ when $i \neq j$. For example, the RFW dataset [323] consists of four different subsets, namely Caucasian, Asian, Indian and African. And then models’ performance can be evaluated based on these subsets. Under such setting, the variation of data in each subset is often too complex and consists of multiple variables. Therefore, these types of datasets are more suitable for other related research directions, e.g., fairness studies. Ideally, in invariance testing, for each variable, we should have datasets that only contain the variable, i.e., Eq.2.2. Alternatively, the dataset should contain only a few strongly related variables e.g., device and resolution. Therefore, in this survey, we coherently follow Eq.2.2 and its lenient form specified in Section 3.2., i.e., given one data object x , to find a set of data objects $\{x^1, x^2, \dots\}$ that are “largely similar” but vary only in one or a few related aspects. In this section, we summarise the attempts that such data (which can be potentially used for invariance testing) are collected or synthesized.

(3.4.1) Face/Clothes/Attributes: The variation in “human” can be divided into two groups: intrinsic and extrinsic. Intrinsic variation such as age, gender and race can be found in most face datasets, e.g., VggFace dataset [31]. However, for one input data x , it is not straightforward to manually introduce those intrinsic facial variables and obtain a series of “largely similar” data $\{x^1, x^2, \dots\}$. For this reason, instead of manually collecting datasets to study the invariance qualities to these variables, generative methods can be used to synthesise the required data, e.g., a FaceAge-GAN [184] can be used to generate images for the same subject at different age groups (as shown in Figure 2.6). In terms of voices, we defer interested readers to a more specific survey of accented voice [119]. By contract, extrinsic variations can be manually introduced by deliberately collecting data, e.g., mask, beard, glasses, hairstyle, expression, etc [155]. For example, K-HairStyle [146] consists of 500,000 images taken from 300 hairstylists (in different hairstyles). In addition to data collection, researchers also synthesize data to study extrinsic facial variables. For example, MaskNet [29] overlaid masks based on detected facial landmarks [334]. Variation in clothing is similar to extrinsic facial variables. The invariance qualities of clothing has recently attracted researchers’ attention for tasks like ReID, surveillance [372] and gait recognition [121]. For example, researchers have collected 9738 images for pedestrians in different clothes to improve the invariance qualities for ML models (NKUP dataset [320]).

(3.4.2) Background/Domain: In the literature, especially in the studies of domain adaptation / generalization and out-of-distribution, domain shifts are sometimes referred to as “style” shifts / bias, e.g., [214]. To be more specific, in the waterbirds vs landbirds dataset [262], waterbirds are normally captured on water while landbirds are normally captured on land. As a result, the models can be trained to distinguish the backgrounds instead of the interested objects. Another example is when models are trained on objects with clean background, e.g., the MNIST dataset, their performance on objects with noisy background, e.g., the SVSF dataset can heavily vary. The aforementioned problems can be referred to as domain adaptation, out-of-distribution or spurious correlation in the literature. Meanwhile, they are also typical invariance testing problems where the variable can be different backgrounds or domains. Given one input data object, in order to introduce the variation in domain/background, several generative methods can be used, e.g., [169] used association analysis to build an ontology to find a set of related and suitable backgrounds, [4, 291] deliberately removed some related items from its background (See Section 5 for more details of generative methods for invariance testing).

Table 2.4: Datasets that include different weather conditions

	Source	Size	Variables
Robotcar [193]	Autonomous Driving	Over 25M	Weather, lighting, devices & modalities (LiDAR)
Synthia [255]	Autonomous Driving	213,400	View point, weather & brightness
RADIATE [279]	Autonomous Driving	200k	Weather & modalities (LiDAR)
DAWN [142]	Surveillance	1K	Weather, illumination & occlusion

2.3.5 Variables associated with applications

In addition to the aforementioned variables, there are some other variables that are commonly linked to some specific applications. For example, for autonomous driving, weather invariance is crucial to any ML models [91]. Some tasks require sensitivity to the variables instead of invariance qualities. However, when ML models are not trained specifically to detect these attributes, the invariance qualities are normally desired.

In both surveillance and autonomous driving, weather invariance is indispensable for ML models. In the literature, many datasets have been collected to study the robustness of ML models under different weather conditions, and we list some commonly used datasets in Table 2.4.

There are other types of variables that are normally associated with applications. For example, in agriculture, humidity invariance and shape invariance are often desired. We defer interested readers to their surveys that are linked to the industries, e.g., agriculture [95, 138], energy system [81], finance [6], drug design [324] etc.

In Figure 2.7, we summary how data with different types of variation are generated, synthesized, and collected. Manually collecting testing data with variation is the most reliable approach for invariance testing as the collected data truly reflect the variation in real life. However, it is labour-intensive and time-consuming to manually collect data. Some types of variation can be generated using a pre-defined deterministic function, e.g., affine transformation, and Fourier transformation. And the other types of variation will require a generative model/algorithm, e.g., GANs. In the following section, we will introduce the evolution of such generative models/algorithms in the literature, and how they can be used for invariance testing to generate testing data with controlled variations.

2.4 Input generation and mutation control

In most cases, input generation and mutation control can be done via parameters. For example, rotate an input data object by 1° , 2° , 3° , ..., k° , i.e., the muta-

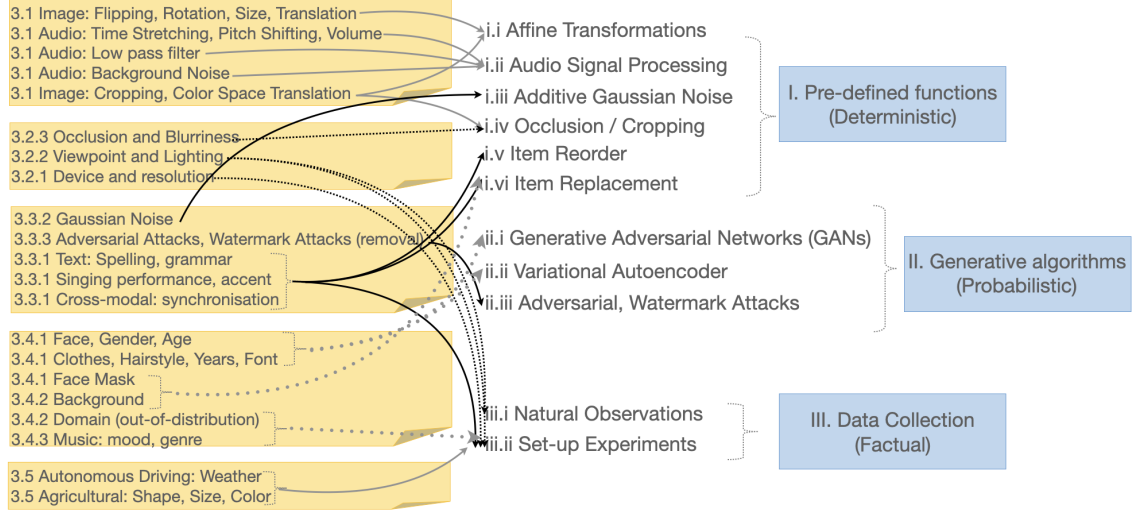


Figure 2.7: Data with variations can be obtained via a pre-defined deterministic function, e.g., affine transformations, a probabilistic generative algorithm, e.g., GANs, or deliberately collected. We summarise how testing data for different types of variables are generated or collected.

tion/transformation is controlled by the rotation angle (parameter). Similarly, when injecting Gaussian noise, we can use λ to control how strong the noise is, and use σ to define the distribution of the noise. When the variables are complicated and the mutation cannot simply be controlled by using parameters, data collection is another common way to “generate” the data and “control” the mutation, e.g., devices as mentioned in Section 3. In addition to these two common input generation and mutation control methods, there are multiple other techniques that can be used to generate input and control mutation. As shown in Figure 2.8, in this section, we summarize existing studies on testing data generation and synthesizing, especially those that were specifically designed to assist testing.

2.4.1 Generative adversarial networks

Generative Adversarial Networks (GANs) can be useful in invariance testing by generating synthetic data samples that represent variations in the input space. This is especially useful when some variables are impossible to control by simply applying parameters or collecting data. As mentioned in Section 3, GANs can smoothly morph a given data object with respect to one or a few strongly related attributes, e.g., [184] can change the same person from a male version to a female version gradually and smoothly. This way, a sequence of largely similar testing data objects $X = \{x_1, x_2, \dots, x_m\}$ are obtained and can be used for invariance testing analysis

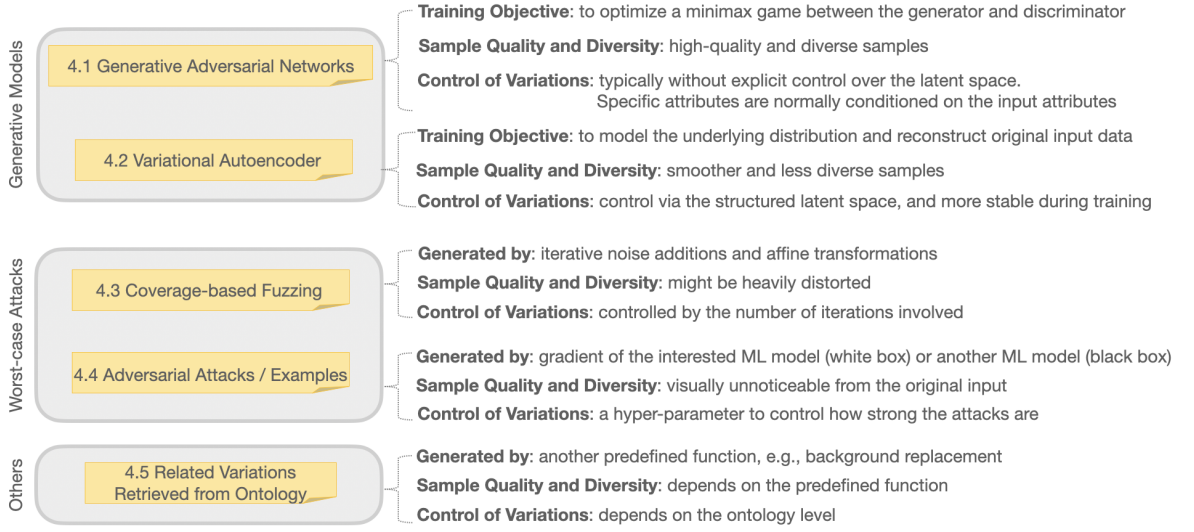


Figure 2.8: We group recent research and studies on generation/synthesizing of invariance testing images into three main groups, i.e., generative models, adversarial attacks, and related information retrieved from ontology. In this section, we introduce these invariance testing data generation methods, especially those specifically designed to assist testing.

in the next steps. In this section, we briefly introduce the basics of GANs and refer interested readers to more detailed survey papers [98, 231].

Generative Adversarial Networks consist of two important components: a generator and a discriminator. GANs are designed to generate new data samples that resemble a given training dataset by simultaneously training both generator and discriminator using the following loss function (L_D is the loss function of the discriminator while L_G is that of the generator):

$$L_D = -[\log(D(x)) + \log(1 - D(G(z)))]$$

$$L_G = -\log(D(G(z)))$$

where z is random noise, $G(z)$ is the generated image based on the random noise, x is an input data object for the discriminator, $D(x)$ is the output of the binary classification (x is labelled as “class 0” if it is generated by the generator, “class 1” otherwise). By training with these two loss functions, the generators are expected to generate realistic images that the discriminator cannot tell the difference between the generated image and images in the training data set. The training of GANs is normally not a stable process. In order to stabilize the training, several methods were introduced in the literature, e.g., WGANs [110] used Wasserstein distance instead of KL distance or JS distance. In this survey, we focus on the control of the generation

and how GANs can be used for invariance testing, and we refer interested readers to a detailed survey about stabilization of GANs [331].

The generation process can be controlled and conditioned on attributes or images. Instead of generating images from random noise, conditional GANs can generate images based on an input data object and random noise [128]:

$$\begin{aligned} L_D &= -[\log(D(x, y)) + \log(1 - D(x, G(x, z)))] \\ L_G &= -\log(D(x, G(x, z))) \end{aligned}$$

where x is a given input data object, z is random noise. The generator (G) takes the random noise z and input data object x and generates an image $G(x, z)$. Similarly, the discriminator (D) takes the image x and a real image y from the training dataset. The discriminator does not necessarily to be conditioned on the given input data object x , but [235] found that conditioning on the original data object x was beneficial to mix the objectives with a more traditional loss, such as L2 distance. This way, given an input data object, a sequence “largely similar” data can be generated.

In order to limit the variation of the generated images and control the variation of each generated image, StyleGan [134] can be used to morph a given input data object with respect to one or a few strongly related variables. However, it's difficult to relate one latent code (latent factor) and one variable. Therefore, in the literature, GANs are normally used to generate images based on different latent codes which correspond to one or some variables. For example, [236] controlled image generation / mutation via maximizing mutual information between latent factors and generated images. In order to ensure the latent factor (C_i) aligns with a potential variable (A_i) in images, they trained an auxiliary network that predicted the latent factor (C_i) from the generated image. After a sequence of “largely similar” data are generated, they calculate the overall accuracy to test the invariance qualities of the target ML model. To control and edit the mutation, EditGANs [178] modelled a joint distribution between the original input data object and its segmentation map. And then the segmentation map can be edited and a new image can be re-generated based on the edited segmentation map. This way, the newly generated images are almost identical to the original one except for the edited part in the segmentation map, e.g., cars with bigger wheels or human faces with different shapes of eyes / nose / mouth. Furthermore, to improve the controllability of mutation, DragGANs [230] allowed users to drag a few handling key points to a new position and morph the original data object accordingly via motion supervision (a special loss function) and point tracking.

2.4.2 Variational Autoencoder

Variational Autoencoders (VAEs) are generative models which are similar to GANs. In a VAE, there are also two important elements, namely an encoder and a decoder. This section, we briefly introduce VAEs and we refer interested readers to a survey paper on deep generative models [227].

The encoder converts input data objects, e.g., images, into a latent space representation called the “latent variables” or “latent code.” The encoder network then maps the input data objects to the mean and variance of a distribution in the latent space. This distribution is often assumed to be Gaussian. From this latent distribution, a sample can be drawn to produce the latent code. The decoder network takes a latent code and generate a data object. If the latent code is obtained by feeding the original input data objects into the encoder, then the decoder is supposed to reconstruct the original data objects.

The VAE is trained by optimizing a loss function that consists of two components: a reconstruction loss, which measures the performance of the decoder (the difference between the reconstructed input data and the original ones), and a regularization term, which encourages the latent distribution to resemble a prior distribution, typically a unit Gaussian. This probabilistic representation allows for sampling from the latent space, enabling the generation of new data samples. It also provides a natural way to explore the latent space and interpolate between different data points, offering more expressive and creative capabilities compared to the fixed encoding in autoencoders (AEs). An example of the loss functions for VAEs is:

$$\begin{aligned}\text{reconstruction_loss} &= \text{MSE}(x, x') \\ \text{regularization_term} &= \sum (1 + \log(\sigma^2) - \mu^2 - \sigma^2) \\ \text{Loss_VAE} &= \text{reconstruction_loss} + \lambda * \text{regularization_term}\end{aligned}$$

where the reconstruction loss can be any discrepancy functions instead of mean square error (MSE) between the original data x and the reconstructed data x' , the regularization term can also be flexible if the desired mean μ and variance σ are not 0 and 1 respectively, and λ is a coefficient controlling the ratio of the two terms.

To disentangle the latent representation and control the generation of images, [370] optimized the VAEs so that the latent representation followed Gaussian mixture distribution. This way, the disentangled VAEs can generate images conditioned on some labels (or variables). Due to the assumption of Gaussian distribution of the latent representation, one can control the mutation by sampling the latent representation.

Therefore, once VAEs are able to learn disentangled and meaningful latent representation, they can be used to generate testing images for invariance tests. We refer interested readers to a survey on disentanglement [32].

2.4.3 Coverage-based fuzzing testing

Coverage-based fuzzing (CBF) is a testing paradigm that has been widely studied in the field of security vulnerability of software engineering [161]. It works by generating a large number of inputs or test cases, and feeding them into the program to see if any unexpected behaviors or errors occur. In the definition, a CBF test starts with a testing suite (seeds). And then multiple input data are generated by mutating the seeds (fuzzing). A criterion is pre-defined to evaluate the quality of the mutated testing inputs. For example, the number of lines (codes) that are executed using the mutated testing inputs. By having as much of the program’s code is executed as possible during testing, the tester can increase the chances of finding bugs or vulnerabilities that may not be apparent from other types of testing.

This testing paradigm can be transferred to machine learning testing. In machine learning, coverage-based fuzzing can be used to test the robustness and resilience of ML models. By generating a large number of inputs that are designed to test the model’s ability to handle unexpected or adversarial inputs, testers can identify weaknesses in the model’s behavior and improve its overall performance. For example, a CBF test could be used to generate a large number of images with subtle variations that are still recognizable as a specific object. By feeding these images into an ML model designed to recognize that object, testers can identify cases where the model fails to correctly identify the object and improve its performance.

To be more specific, a CBF test for ML models starts with a testing set. And then data objects in the testing set can be morphed by either affine transformations or injecting noise. A pre-defined criterion, e.g., neuron coverage, is used to evaluate the quality of the mutated testing inputs which are supposed to expose vulnerability of the ML model(s). For example, DeepXplore [238] defined neuron coverage as the percentage of neurons that are triggered by a testing suite. They trained multiple networks and used them to find testing data objects that did not have consistent predictions. Meanwhile output values from randomly chosen neurons were added into an objective function to find data objects that triggered more neurons and thus the mutated testing inputs will have higher neuron coverage. This way, DeepXplore was able to generate test inputs that had 34.4%/33.2% higher neuron coverage rate

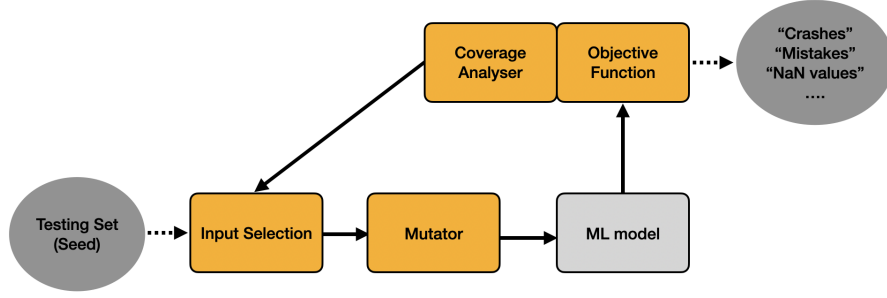


Figure 2.9: General workflow of coverage-based fuzzing (CBF) tests for ML models. The CBF tests start with a testing set (seed). The data are then mutated by affine transformation or injecting noise. The mutated data are then sent to the ML model and the outputs are analysed by the coverage analyser and objective function. The feedback is then sent back to the input selection.

than randomly chosen natural/adversarial examples. Similarly, DeepGauge [189] introduced five variants to generate “corner cases”. SV-Coverage [295] was inspired by modified condition/decision coverage (MC/DC) and they defined Sign/Value-Change to represent True or False branch predicates. DeepCT [188] introduced a novel combinatorial coverage criteria to find testing data objects that can potentially reveal vulnerability of ML models.

However, [116] [164] suggested that neuron coverage might be ineffective for fault detection with natural inputs, and no significant change in the coverage was observed when error rates increased from 0.01% to 10%. Therefore, some other metrics were proposed to replace neuron coverage such as surprise adequacy [144] and metrics computed by approximate nearest neighbour [224]. We defer interested reader to a survey on neuron coverage as long as safety and trustworthiness of deep neural networks [123].

After the coverage criteria are fixed, TensorFuzz [224] introduced a property-based testing procedure, where the objective function could be flexible, e.g., “crashes” or “numerical issues” as shown in Figure 2.9. DLFuzz [111] used a similar CBF testing framework to explore different approaches to select interested neurons. They investigated four strategies:

- To prioritise neurons which are frequently activated during training / testing with the seeds
- To prioritise neurons which are rarely activated during training / testing with the seeds

- To prioritise neurons which have large values as they might affect the final outputs more significantly
- To prioritise neurons whose average values are close to the threshold of “being activated (fired)” as they are easier to be activated by a random input data object

Their results showed that mutated data objects using the first strategy are more likely to mislead ML models. DeepHunter [342] introduced a frequency-aware seed selection strategy based on the number of times a seed has been fuzzed.

All the aforementioned methods were able to find abnormal data objects for ML models and numerically evaluate invariance qualities. Although numerically, most CBF tests still heavily rely on formula-based methods to evaluate the performance of ML models, the mutated data objects are generated under the guidance of internal status of ML models, e.g., neuron coverage. The selection of interested neurons, prioritisation of testing seeds [27, 363], the approaches to mutate the seeds, and the criteria (and objective functions) to guide the mutation remain open research topics.

2.4.4 Adversarial attacks

An adversarial attack refers to an attack against an ML model, that is, to create some inputs that are malicious and deceptive to the model, so as to test the robustness of the model. In most cases, adversarial attacks are generated by adding noise that is not easily observed by humans to the original data objects to affect the prediction / decisions of the model. The generation of adversarial attacks can be divided into gradient-based and non-gradient depending on how the adversarial noises are created [8]. From the perspective of invariance testing, both attack methods change the original data to a certain extent by adding noise or interference, which makes predictions of ML models deviate. The generation of adversarial attacks can be controlled by: 1) parameters (ϵ or number of modified pixels), 2) gradient of the ML model (gradient masking, white box or black box), and 3) methodology of generating the adversarial noise (e.g., iterative or single pixel attack). In this section, we briefly introduce the basics of adversarial attacks and refer interested readers to a detailed survey [7].

In 2014, the Fast Gradient Sign Method (FGSM) was first introduced in [297]. They found that a “robust” ML models (robust against random affine transformation

and Gaussian noise) can be easily deceived by back-propagating the gradient to the original images:

$$x^{adv} = x + \varepsilon \cdot \text{sign}(\nabla_x L(x, y_{true}))$$

where x is a data object, y_{true} is the label, L is a loss function, and ε controls the magnitude of the adversarial noise. If the loss function L is obtained from the target ML model (to be tested), the attacks are known as “white-box” attacks. Otherwise if the loss function is obtained from other ML models, the attacks are known as “black-box” attacks. In the literature, researchers successively proposed many derivative algorithms based on FGSM, such as RFGSM [307], NI-FGSM [175], M-DI2-FGSM [341], etc.. These methods can achieve a better success attack rate (SAR), which is defined as:

$$SAR = \frac{\text{Number of deceptive examples}}{\text{Total number of examples}}$$

In addition to FGSM, Projected Gradient Descent (PGD) [194] is an iterative optimization algorithm that generates adversarial examples by continuously optimizing the loss function. Carlini and Wagner (C&W) Attack [33] is an effective adversarial example generation method, which generates adversarial examples by solving an optimization problem. DeepFool [207] is a method of using linear programming to construct adversarial examples, which can quickly generate adversarial examples. For different adversarial attacks, Yao et al. compared the performance of various adversarial attacks on different ML models. They discover that using Rebel’s detection method [249] has a better effect on FGSM than C&W and PGD on the Cifar-10 dataset, but on the MNIST dataset, sometimes ODD [258] may have a stronger attack effect [163].

Adversarial training for adversarial attacks can improve the pre-resistance of ML models against adversarial attacks to a certain extent without additional cost [276]. However, adversarial training might lead to gradient masking [166]. Instead of an ML model being more robust against white box adversarial attacks, the ML model’s gradient could become “masked” and therefore become “weaker”. This way, the SAR of the generated adversarial examples can be heavily affected. Another thing worth mentioning is that adversarial training does not necessarily have a positive effect on the invariance qualities of the ML model. In the paper [137], some ML models became more rotation-invariant by gradually enhancing rotation augmentation during training, however, while doing so, their performance on adversarial examples could become worse or even drop significantly on datasets like MNIST. In the article [28], Bühlmann analyzed the relationship between invariance, causality, and robustness



(a) Toilet with a ketchup and a banana on the sink



(b) Kitchen with a toothbrush and toothpaste on the worktop

Figure 2.10: Picture recreated based on [226]. After they applied a scene understanding technique, they removed or added items from / into the scene and tested whether a scene classification ML model can still make the correct predictions. For example, (a) showed a toilet with a ketchup and banana and (b) showed a kitchen with a toothbrush and toothpaste on the worktop. If the models can still make correct predictions, then it is more trustworthy that the model was not affected by spurious correlations or did not overfit some specific items.

(against adversarial attacks), using calculation and comparison of the results of machine learning models through statistical formulas and derivations for more robust and better "causality-oriented" explanations.

2.4.5 Ontology and Scene Understanding

As mentioned earlier in this section, one can change the segmentation map of a given input image, and use a StyleGan to reconstruct a new image conditioned on the morphed segmentation map. Scene understanding techniques, e.g., segmentation maps, are indeed very important for invariance testing. In this section, we briefly introduce how scene understanding can be used for invariance testing and we refer interested readers to more detailed survey papers on scene understanding [104, 218].

After a segmentation map has been obtained, one can test the model's sensitivity to any of the detected items and any attributes of each of the items. Based on the outputs of the scene understanding technique, one can morph the original image by using a generative model or simply removing / overlaying part of the original image by a new object / image. For example, the background challenge dataset [339] separated the foreground object from its background and replace it with other images' background(s). Similarly, [60] swapped two images backgrounds, and CutMix [354] overlaid part of the foreground object with part of another foreground object. For instance, overlaid a dog's face onto a cat's body. [226] tested semantic consistency in

a scene. For example, they removed or added a toilet roll / kitchen tower / ketchup from an input data object and tested whether the target ML model can still make the correct predictions / decision as shown in Figure 2.10.

However in the literature, when conducting invariance testing using scene understanding techniques, the testing data objects were normally generated ad hoc. In the aforementioned studies, items in the testing input images were added / removed either randomly or via human knowledge, e.g., Figure 2.10. Recently, [169] used ontology, e.g., association analysis, to study the relationship between different detected items in the input data objects. Considering a set of all possible keywords K_{all} that can be extracted from all images in the repository $\mathbb{B}$, the level of association between two keywords k_a and k_b can be described by two concepts. Consider three itemsets: $s_a = \{k_a\}$, $s_b = \{k_b\}$, and $s_{ab} = \{k_a, k_b\}$

$$\begin{aligned}\text{support}(s_{ab}) &= \frac{\text{number of images where } s_{ab} \text{ is present}}{\text{total number of images}} \\ \text{confidence}(\exists s_a \rightarrow \exists s_b) &= \frac{\text{support}(s_a \cup s_b)}{\text{support}(s_a)}\end{aligned}$$

where $\exists s_a \rightarrow \exists s_b$ indicates an association rule from one itemset (s_a) to another (s_b). *support* indicates the frequency of the co-occurrence of k_a and k_b and *confidence* indicates the confidence level about the inference that if the object of keyword k_a appears in a scene, the object of keyword k_b could also appear in such a scene. And [169] used the association relationships (ontology) to control the mutation, generate new testing images, rank the distance between each mutated images, and conducted background invariance testing.

2.5 Invariance Testing Workflows

As aforementioned and defined in Section 2, invariance testing focuses on the performance of interested model(s) given a sequence of “largely similar” input data objects, which vary with respect to only one or a few related variables defined in Section 3. In order to demonstrate and display the performance, although the most common invariance testing workflow is formula-based, e.g., to compute worst-case accuracy, various other workflows can be used. In this section, we summarise existing invariance testing workflows in the literature, and group them into three main categories as shown in Figure 2.11.

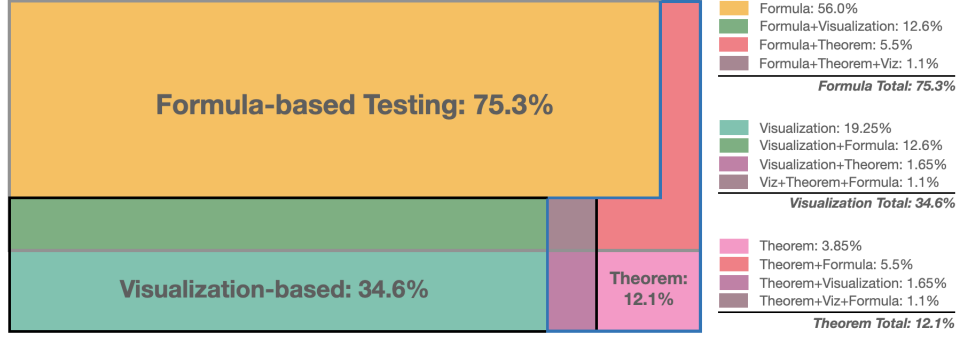


Figure 2.11: We group existing 182 studies (summarized in this survey chapter) involving invariance testing into three main categories. Some studies belong to two or all of the categories, e.g., Precision-recall curve analysis [78] can involve both visualization-based analysis and formula-based evaluation.

2.5.1 Formula-based invariance testing

In the literature, various formulas were introduced to numerically evaluate performance of ML models. In addition to worst-case accuracy [339] or robust accuracy [87], f1 score, precision and recall [102] can also be used to represent invariance testing results if the models are trained for data retrieval, e.g., for copy right protection.

$$recall = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$precision = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$f_{\beta} = (1 + \beta^2) * \frac{precision * recall}{\beta^2 * precision + recall}$$

For a sequence of “largely similar” data objects $D = \{x_1, x_2, \dots\}$, the consistency of the performance of a target model $\mathbf{M}$ can be evaluated using precision, recall or f1 score instead of accuracy. For example, for each $x_i \in D$, we can compute a recall, precision or f1 score based on the performance of $\mathbf{M}(x_i)$, e.g., $recall(\mathbf{M}(x_i))$. And then we can compute the difference between the recall, precision or f1 scores, e.g., $recall(\mathbf{M}(x_i)) - recall(\mathbf{M}(x_j))$, to evaluate the invariance quality of $\mathbf{M}$.

Similarly, various other metrics can be used for different target models that are trained for different tasks, e.g., average precision [352], mean average precision (MAP) or Area Under Receiver operating characteristic Curve (AUROC) [22] for object detection models, different rewards for reinforcement learning models [286], Kolmogorov-Smirnov statistic for ads – the optimal number of targeted customers and credit scoring scenarios [197], and Mean Intersection of Union (mIoU) for segmentation

Table 2.5: Different metrics to evaluate ML models

	Formula or description
Intraclass correlation (ICC) [77]	$\frac{1}{Ns^2} \sum_{n=1}^N (x_{n,1} - \bar{x})(x_{n,2} - \bar{x}), s^2 = \frac{1}{2N} \sum_{n=1}^N ((x_{n,1} - \bar{x})^2 + (x_{n,2} - \bar{x})^2)$
Correlation Coefficient [292]	$\rho = \text{cov}(X, Y) / \sigma_x * \sigma_y$
Concordance [106]	$\frac{2\rho\sigma_x\sigma_y}{\sigma_x^2 + \sigma_y^2 + (\mu_x - \mu_y)^2}$, where ρ is the correlation coefficient
Cohen’s kappa [70]	$\frac{p_o - p_e}{1 - p_e}$, where p_o and p_e are agreement and disagreement respectively
Gini coefficient [65]	$\frac{A}{A+B}$, the Lorenz curve is obtained using model’s probability predictions
Somer’s D [220]	$\frac{N_c - N_d}{n(n-1)/2}$, N_c/N_d are number of concordant/discordant pairs respectively

models [204]. We list several other metrics that can be used to evaluate ML models in Table 2.5.

In addition to traditional statistical metrics, some special metrics that are specially designed for ML models can also be used to evaluate invariance qualities. For example, Singular Vector Canonical Correlation Analysis (SVCCA) [248] was introduced to compare outputs from different ML models and it can be used to evaluate the relationship between outputs of the same ML model, e.g., [353] studied inter-head diversity in multi-head attention using SVCCA. Similarly, other metrics can potentially be used to analyse invariance qualities of ML models, e.g., difference between Uncertainty [92] with respect to “largely similar” input data.

In the literature, there are also metrics that were designed for invariance qualities. For example, Goodfellow, Andrew Y. Ng, etc proposed a metric evaluating invariance qualities of an ML model based on the invariance scores of top-scoring proportion of hidden units (*cumulation*) in the deepest layer (*location*) [96]. For each hidden unit i , the invariance score was defined as:

$$S(i) = \frac{\frac{1}{|Z|} \sum_{z \in Z} \frac{1}{|T(z)|} \sum_{x \in T(z)} f_i(x)}{\mathbb{E}[f_i(x)]}$$

where $f_i(x)$ is the output of the neuron given input x , $T(x)$ is a set of stimuli that are “largely similar” to an input data object x , Z is a set of inputs that activate the neuron near maximally, and $\mathbb{E}[f_i(x)]$ is the expected value (which was referred to as firing rate) of the outputs of the neuron when x is drawn from a testing set.

2.5.2 Visualization-based professional analysis

As mentioned in the beginning of this chapter, formula-based approaches (statistics) cannot fully demonstrate the real distribution (as shown in Figure 2.1). To address the limitation, visualization-based methods can be used to further assist the analysis of the invariance qualities of ML models.

2.5.2.1 Explainable AI (XAI)

is a well studied visualization-based analytic approach. XAI refers to the ability to provide understandable and interpretable explanations for the decisions, predictions and behaviors of ML models. It aims to address the "black box" nature of complex ML models (e.g., CNNs, LSTMs) by providing insights into how the model(s) make their predictions or decisions [66]. The understandable and interpretable explanations are normally delivered via visualization. For example, LiME [253] trained an interpretable "surrogate" model (sparse linear explainer) that approximates the behavior of the complex black-box ML model (to be tested). This surrogate model is trained specifically for one target input data object and should be simple enough to be easily interpretable while capturing the important patterns and relationships present in the original model, e.g., a linear layer or a decision tree as they are self-explainable. The surrogate model will provide visualization analysis for the prediction of the black-box model given the particular input data object.

2.5.2.2 Symbolic testing

also known as symbolic execution, is a technique used in software testing and program analysis. It involves executing a program with symbolic values instead of concrete inputs to explore different execution paths and analyze program behavior. In ML testing, symbolic testing [101] converts an ML model into a *program* with if/else branches, i.e., each neuron output after ReLU activation layer, given an input data object. And then the final prediction(s) / decision(s) of the ML model is written into a weighted function of a series of coefficients for all the input pixels. For example, given a DNN with 2 hidden layers and 2 hidden units at each layer respectively, its predictions (y_0 and y_1) can be written as:

$$\begin{aligned}a_0 &= \sigma(w_{0,0}^0 * x_0 + w_{0,1}^0 * x_1) \\a_1 &= \sigma(w_{1,0}^0 * x_0 + w_{1,1}^0 * x_1) \\y_0 &= \sigma(w_{0,0}^1 * a_0 + w_{0,1}^1 * a_1) \\y_1 &= \sigma(w_{1,0}^1 * a_0 + w_{1,1}^1 * a_1)\end{aligned}$$

where x_0 and x_1 are the input pixels, w^0 and w^1 are the weights of the first / second layer respectively, and σ is an activation function. This way, a coefficient for each input pixel x_i is computed with respect to an output value y_j . These coefficients are how "important" the pixels are for the final prediction / decision (y_j) of the ML model. Similarly, they can also compute pixels that are prone to adversarial

attacks. The coefficients (importance of each pixel) can then be illustrated using a visual pattern. Similarly, concolic testing combines symbolic testing and coverage-based fuzzing. Concolic testing [296] starts with executing a *program* with a concrete input data object. At the end of the concrete run, another execution path (if/else predicates) is selected by human heuristics, e.g., the selection can be guided by the forward gradient or cost gradient of the ML model. The newly selected execution path is then encoded symbolically to yield a new concrete input data object for the next iteration. The concrete run and the symbolic analysis alternate until a pre-defined structural coverage rate is reached.

The main challenge of symbolic testing is to scale to large ML models as the computational costs will increase exponentially. Therefore, for larger ML models, e.g., deep CNNs, one normally needed to balance between precision and computational cost. AI² [93] introduced an abstract transformer to utilize abstract interpretation to check that all perturbed inputs are classified the same way. The proposed testing framework used a 2-bounded set of zonotopes, and propagated the abstract elements (capturing all perturbed inputs) through all the layers. After each layer, a new shape of the abstract interpretation was obtained, and at the output layer, one can check whether all perturbed inputs are classified correctly. The bounded powerset domain allowed interested practitioners to balance between scalability and precision, and generate a testing report at a required precision level.

2.5.2.3 Heatmap-based Analysis

As shown in Figure 2.12, given a set of testing data X , heatmap-based analysis, e.g., [170, 171], normally processed the testing results (final outputs and intermediate results) via some visual patterns, e.g., variance matrices. One variance matrix measured the difference between a model($\mathbf{M}$)’ output at a location p for all pairs of transformations in $T = \{t_0, t_1, \dots, t_n\}$, and the variance matrix can be expressed as:

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

where the difference δ is defined over a cumulative function and a modality function, e.g., the difference between mean or max. They found that ML models with the same accuracy and worst-case accuracy score can have completely different variance matrices. According to the resultant variance matrices, professional annotators labelled the

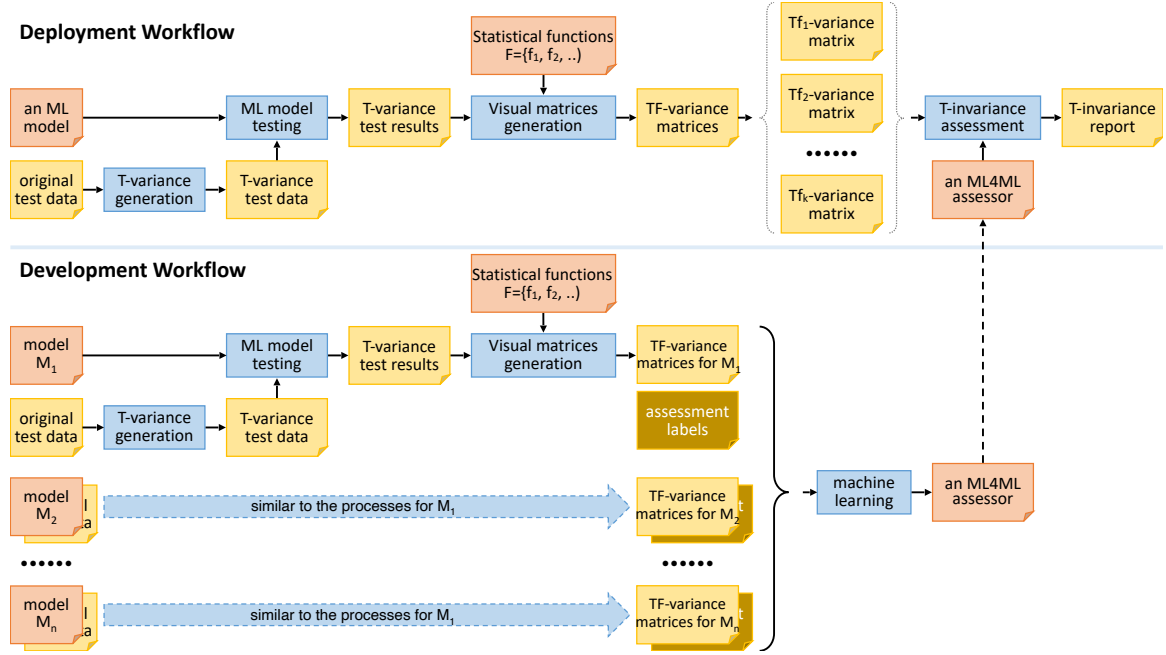


Figure 2.12: General framework for heatmap-based testing approaches. Testing data are generated and testing results are normally visualized for further professional analysis. To reduce costs in labour and time, automation can be considered by training assessors based on the visualized testing results.

target models as passed, borderline or failed. Such professional visual analysis often leads to a high inter rater reliability (IRR), e.g., around 0.8 for basic data attributes like rotation, which suggests the reliability, consistency and feasibility of such testing framework. However, manually assess the invariance qualities of ML models is labour-intensive and time consuming. Therefore, after obtaining variance matrices, [169,173] also train another ML model (assessor) using the features from the variance matrices to automate the entire testing procedure. The accuracy of the automation assessor can be around 80%~90%, and the IRR between predicted invariance qualities and professionally labelled invariance qualities is also high, e.g., around 0.8 for basic data attribute like size/brightness. This indicates that the testing framework is feasible, scalable, and can be fully automated.

Similarly, [73] used a landscape of spatial robustness to assess the models' invariance qualities, and they found that ML models are still vulnerable against some combination of affine transformations. [196] visualized the kernel of trained CNNs, and they found that CNNs can learn different filters to appear invariant against a certain type of variable. [78] conducted Precision-Recall curve analysis and demonstrated that AUPR and AUPRG often disagree in ranking the models, and they

recommended F-Gain score instead of the F-score. [16] conducted human analysis based on different visualized metrics, including feature maps, and decide whether a ML model is invariant as a whole. They mutated the original input data objects by a small amount, e.g., horizontal shift, and found that the models’ performance can be heavily affected.

These visualization-based testing paradigms normally do not rely on formulas to evaluate the performance of the models, and it can be transferred to invariance testing in the following ways:

- **Feature-wise analysis:** Visualization can provide insights into which features (or inputs) have the most significant impact on the model’s predictions or decisions. By analyzing feature importance using different variations of the input data, one can assess if certain features consistently drive the model’s decisions or if their importance changes with different conditions. This way, one can access the invariance qualities of the model. For example, to conduct invariance testing for an ML model, [170, 171] firstly extracted features using the ML model for some given input data objects. And then these features were used to train a sequence of sparse linear layers which were used to form a visual pattern to investigate the invariance quality of the features, as well as the overall invariance quality of the model.
- **Counterfactual explanations:** Visualization techniques can generate counterfactual explanations that show how the model’s predictions change when specific inputs objects are modified (with variation) [109]. This can be used to test the model’s sensitivity to variations and determine if it exhibits consistent behavior across different scenarios, e.g., [344].
- **Human-in-the-loop testing:** Visualization allows for human interpretation and validation of the model’s behavior. Invariance testing can involve incorporating human judgment and domain expertise by leveraging visual explanations to evaluate the model’s responses to different variations and assess if they align with human expectations. For example, [291] studied spurious correlation by human annotations. To be more specific, in the training set, in addition to the task that the model was trained for (e.g., ground-truth labels), data objects are also labelled as “rotated”, “occluded”, or “no transformation”. They use a visual pattern to test if the trained models have a better invariance quality.

By using visualization techniques, invariance testing can benefit from interpretability and transparency, enabling a deeper understanding of how a model behaves under different environments. Visualization also provides insights into the decision-making process, identifies limitations or biases, and assists in improving the model’s robustness and generalization capabilities across various variations or transformations of the input data.

2.5.3 Theory-based professional analysis

In addition to the aforementioned empirical testing frameworks, invariance testing can also be proved or debunked from theoretically. For example, when considering ML models (e.g., Auto Encoder) as a process of signal encoding and decoding, Nyquist-Shannon sampling theorem can be used for translation invariance testing/analysis. In signal processing, Nyquist theorem establishes the minimum sampling rate required to accurately reconstruct the original signal from its sampled values. The Nyquist theorem states that: to flawlessly reconstruct the original signal from its sampled counterpart with no aliasing (distortion), the signal sampling rate must be at least twice the highest frequency component present in the original signal. Mathematically, the theorem can be expressed as:

$$f_s > 2 * f_{max}$$

where f_s is the minimum required sampling rate (in samples per second or Hz), and f_{max} is the highest frequency component in the original signal (in Hz). The theorem indicates that when sampling a signal for digital processing, encoding or transmission, the sampling frequency is required to be greater than twice the maximum frequency present in the original signal of interest.

According to Fourier transformation [21], sampling in one domain (time or frequency domain) guarantees periodicity in the other domain. When transforming (Fourier transformation) a sampled signal into frequency domain, aliasing (or distortion) can happen due to its periodicity and low sampling rate. Nyquist theorem is the theoretical basis for designing anti-aliasing filters and kernels. [365] analyzed translation invariance qualities of ML models based on Nyquist theorem and they found that ML models are not theoretically translation invariant, especially when stride or pooling layers are used.

Group theory (as defined in Section 2) is relevant to the study of symmetries, and transformations, which can also be used to theoretically analyze invariance qualities for ML models. [39] showed that training ML models with data augmentation

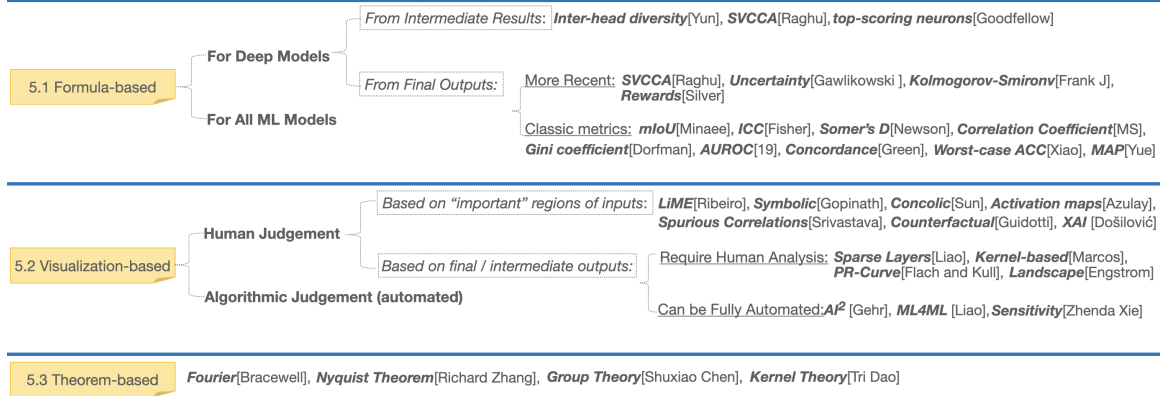


Figure 2.13: Summary of testing methods mentioned in this work into three big categories, i.e., formula-based, visualization-based and theory-based. Note that most of testing methods belong to more than one of the three big categories, and in this work, we group them based on their most distinctive contribution to the ML testing community.

was equivalent to minimizing an averaging augmented loss over the orbits of a certain group. The authors theoretically proved that data augmentation led to variance reduction, suggesting a reliable invariance quality against affine transformations. Similarly, [57] utilized kernel theory and treated augmentation as a Markov process. The authors analyzed the impact of augmentation on kernel classifiers, and showed that data augmentation can be approximately represented by 1) first-order feature averaging, and 2) second-order variance regularization terms.

Formal methods can be used to verify, test, and analyze deep learning models, ensuring the “correctness, robustness and fairness” of ML models. Mathematically rigorous techniques are normally utilized, e.g., logical reasoning, formal verification tools, and mathematical proofs. For example, formal methods were used to verify decision trees (subroutines) [139], or verify pruning algorithms for neural networks by Taylor expansion [206]. Similarly, [125] mathematically proved that models relying on non-robust features were more vulnerable to adversarial attacks, whereas emphasizing robust features enhanced model robustness. We refer interested readers to the survey on formal verification for adversarial robustness [202] and the survey on formal methods applied to machine learning [309]. The AI² [93] testing framework examined the shape of all perturbed input data after each layer of ML models, which was an attempt of using both formal methods and visualization for invariance testing.

2.6 Conclusion

As ML/DL models are becoming ubiquitous and widely deployed for many real life applications, researchers are paying more attention to testing and evaluation to ensure the robustness, safety, fairness, and invariance qualities of ML/DL models. Prior works on robustness, safety and fairness have all formed a solid and well established research direction. And existing attempts to boost robustness, safety, and fairness have been summarized in their corresponding survey papers (e.g., robustness survey [7], fairness survey [201], safety survey [274]). Unlike these properties of ML models, studies of invariance testing lack a unified formulation and established framework, resulting in scattered research efforts and terminology discrepancies.

In this survey chapter, we summarize various existing studies involving invariance testing, including but not limited to formula-based testing metrics, visualization-based (including human-centred) testing frameworks, and theorem-based testing approaches as shown in Figure 2.13. We provided a comprehensive overview of the diverse landscape of invariance testing across various disciplines, including software testing, data mining, visualization, and AI/ML.

Invariance quality of ML models is an important property of ML models. To ensure ML models are practical, reliable and consistent, invariance qualities are desired in most cases. In the literature, many studies focused on improving ML models' invariance qualities during training and fine-tuning. However, for testing, various different terms can be used to describe the same or similar concepts. The fragmented nature of the existing literature poses significant challenges for researchers and practitioners seeking to navigate this field effectively.

Our survey chapter serves as a vital contribution to addressing this challenge by synthesizing and organizing the diverse array of research findings and methodologies related to invariance testing. By providing a structured overview of the existing approaches, terminology, and research trends, we aim to facilitate a deeper understanding of this fundamental problem and foster interdisciplinary collaboration and knowledge exchange.

Moving forward, there is a pressing need for the development of standardized frameworks, methodologies, and terminology for invariance testing, as well as interdisciplinary efforts to bridge the gaps between different research communities. By promoting coherence and consistency in research practices, we can enhance the effectiveness and impact of invariance testing across various domains and contribute to the advancement of science and technology.

In conclusion, we hope that this survey chapter will serve as a valuable resource for researchers, practitioners, and educators interested in exploring the rich and multifaceted landscape of invariance testing, and inspire further research efforts aimed at addressing its challenges and unlocking its full potential.

Chapter 3

Generic Invariance Testing Framework for Basic Data Attributes

3.1 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., [54]. Given a machine-learned model, manually determining whether the model is $\mathcal{T}$ -invariant, where $\mathcal{T}$ 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., [88,97]). 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 3.4, 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 $\mathcal{T}$ -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 chapter, 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 (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 assessors.
- We have made all the relevant models, data, and documentation available at the github [165].

3.2 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 [360]. 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 [210]. One common approach to improving such quality is through augmentation [285]. Engstrom et al. [74] 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 [366] 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., [245, 345]. This work extends such evaluation by considering visual patterns generated at different positions of ML models.

Compared with gradient-based methods (e.g., [167, 298]), adversarial examples generated using affine transformation are more closely related to invariance testing. The works on neuron coverage, such as DeepXplore [239] and DeepGauge [190] are highly representative. TensorFuzz [225] introduced a property-based fuzzing procedure to enhance its convergence. DLFuzz [112] exploited a coverage-guided fuzzing approach to generate transformed adversarial examples. DeepHunter [343] 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) [25] and out-of-distribution detection [347] are also closely related to robustness testing. Werpachowski et al. [330] trained a reject function for each trained CNN, and they used either softmax response (confidence score) or uncertainty [86] 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., [267]), while our work focuses on evaluating invariance qualities of ML models as a whole.

Metamorphic testing [275] 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 [68] as well as model structures [191, 280]. Both metamorphic and invariance testing can be used to prevent unwanted deployment, although they serve different purposes for different properties.

3.3 Definitions and Motivation

In machine learning, the invariance quality of a trained model $\mathbf{M}$ 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 $\mathbf{M}$, which is commonly referred to as the data space of $\mathbf{M}$. 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

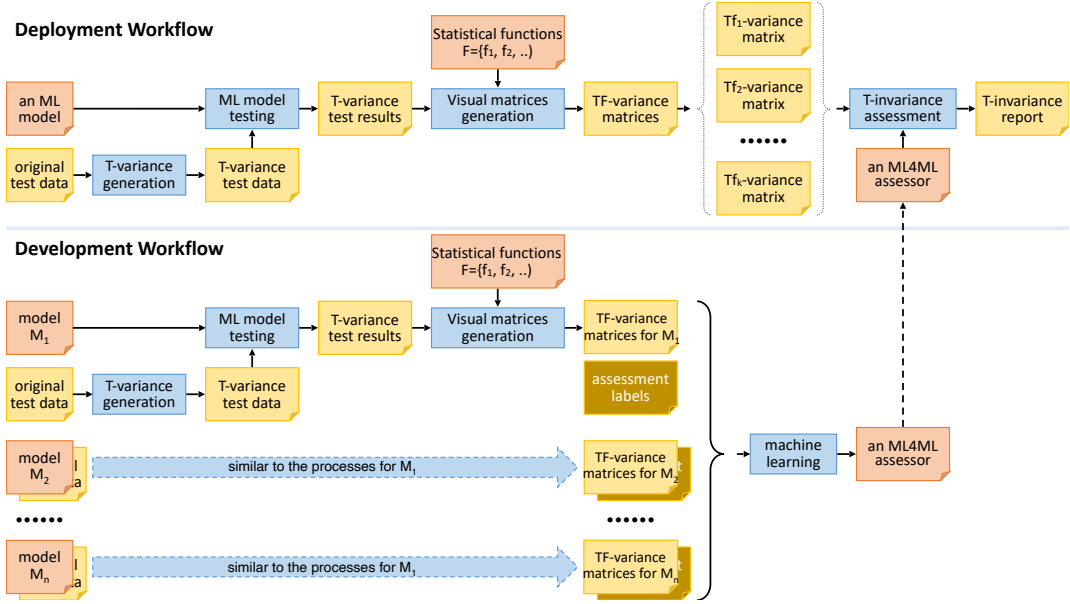


Figure 3.1: **Deployment workflow:** for each type of invariance $\mathcal{T}$, the original test data is first extended to invariance test data, which is then used to test the ML model concerned. The invariance test data is processed using different statistical functions (which are also referred to as *modalities*), yielding different visual patterns (which are also referred to as *variance matrices*). An assessor (a model / an ensemble of models) analyzes the variance matrices and reports the invariance quality of the ML model concerned. **Development workflow:** A repository of ML models is collected. The same processes for creating variance matrices are applied to these models. The labelled variance matrices are used to train an assessor that is to be used in the deployment workflow.

but vary in one particular aspect, denoted as variation $\mathcal{T}$. Consider every such subset, $D_1, D_2, \dots$ in $\mathbb{D}$.

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

When $\mathbf{M}$ is not $\mathcal{T}$ -invariant, ideally we can assess the level of inconsistency (or $\mathcal{T}$ -variance) with a function Υ :

$$\text{Level}_{\mathcal{T}} = \Upsilon(\mathbf{M}, D_1, D_2, \dots)$$

Obtaining such an ideal measurement $\text{Level}_{\mathcal{T}}$ is almost impossible because (i) it is difficult to identify all subsets $D_1, D_2, \dots$ 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 Υ .

Hence the level of $\mathcal{T}$ -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 $\mathcal{T}$ [74]. One obtains a testing dataset $X \subset \mathbb{D}$, and for each data object $x_i \in X$ ($i = 1, 2, \dots, m$), one applies different transformation $t_j \in T$ ($j = 1, 2, \dots, n$) to x_i to create a set of data objects:

$$\tilde{D}_i = \{d_{i,0}, d_{i,1}, \dots, d_{i,n}\} = \{t_0(x_i), t_1(x_i), \dots, t_n(x_i)\}$$

One can define a function f to measure the inconsistency between two predictions of model $\mathbf{M}$. With f one may approximate Υ with the following simplistic function Φ :

$$\Phi(\mathbf{M}, X) = \frac{1}{m} \sum_{i=1}^m \prod_{j=1}^n f(\mathbf{M}(x_i), \mathbf{M}(t_j(x_i))) \quad (3.1)$$

When $f = 1$, x_i and $t_j(x_i)$ are consistently classified, i.e., $\mathbf{M}(x_i) = \mathbf{M}(t_j(x_i))$, and when $f = 0$ the contrary. Because Gao et al. ([88]) also associated consistency with correctness, i.e., $\mathbf{M}(x_i) = \mathbf{M}(t_j(x_i)) = \text{correct prediction}$, they referred such a Φ to as *robust accuracy*. Eq. 3.1 is served as a simple baseline in this work.

As mentioned earlier, this is not a ground truth function, and Φ can be defined differently, e.g., using a mean function instead of the product in Eq. 3.1. 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 Φ . 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.

3.4 Methodology: Testing Framework

As shown in Figure 3.1, our framework for invariance testing includes two workflows: a *development workflow* and a *deployment workflow*. In the former workflow (lower part of Figure 3.1), one collects a repository of models and uses them to train assessors for invariance testing. For each type of invariance $\mathcal{T}$, this workflow will yield one assessment model or an ensemble of models, which are to be used in the latter workflow (upper part of Figure 3.1).

3.4.1 Deployment Workflow

Transformations and Sampling Variable. Let us first describe the deployment of an assessor for testing a type of invariance $\mathcal{T}$. Given an ML model $\mathbf{M}$ and a testing dataset X , we first define a *family of transformation functions* $T = \{t_0, t_1, \dots, t_n\}$ to

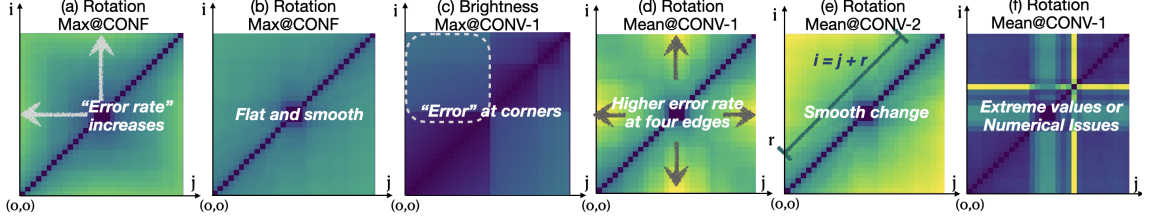


Figure 3.2: Six examples of variance matrices for different CNNs. All matrices are of size 31, and were created with a family of 31 transformations. Their testing purposes, transformations, positions, and modalities are detailed in Table 3.1.

simulate the variations for testing $\mathcal{T}$. 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 $t()$ 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:

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

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

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

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 $\mathcal{T}_{new}$, define a new set of transformations T_{new} , and a new sampling variable V_{new} .

Modality. As described earlier, with the defined transformations T , we can create a testing data set $\tilde{D}_i$ for each data object x_i in the original testing dataset X . We can then measure the impact of the transformations t_j ($j = 0, 1, \dots, n$) upon x_i . Eq. 3.1 used a simple function f to measure the difference between the model predictions

Table 3.1: Technical specifications for obtaining the six examples of variance matrices shown in Figure 3.2.

Matrix	Variance	Testing Interval	Testing Position	Function dif	Model ID
(a)	rotation	$[-15^\circ, 15^\circ], \pm 1^\circ$	confidence score (CONF)	$\max(S_x) - \max(S_y)$	$\mathbf{M}_{93}$
(b)	rotation	$[-15^\circ, 15^\circ], \pm 1^\circ$	confidence score (CONF)	$\max(S_x) - \max(S_y)$	$\mathbf{M}_2$
(c)	brightness	$[-30\%, +30\%], \pm 2\%$	last convolutional layer (CONV-1)	$\max(S_x) - \max(S_y)$	$\mathbf{M}_{101}$
(d)	rotation	$[-15^\circ, 15^\circ], \pm 1^\circ$	last convolutional layer (CONV-1)	$\text{mean}(S_x) - \text{mean}(S_y)$	$\mathbf{M}_{93}$
(e)	rotation	$[-15^\circ, 15^\circ], \pm 1^\circ$	penultimate convolutional layer (CONV-2)	$\text{mean}(S_x) - \text{mean}(S_y)$	$\mathbf{M}_2$
(f)	rotation	$[-15^\circ, 15^\circ], \pm 1^\circ$	last convolutional layer (CONV-1)	$\text{mean}(S_x) - \text{mean}(S_y)$	$\mathbf{M}_1$

based on x_i and $t_j(x_i)$ respectively. Inspired by medical imaging [212], we replace the simple f with different statistical functions. And we also replace the “summation on production”, i.e., $\sum_i \prod_j$, operation in Eq. 3.1 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 $\mathbf{M}$ but also the internal status of $\mathbf{M}$ 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 $\mathbf{M}$, to the final prediction of $\mathbf{M}$. We can measure the signals at an internal position p of $\mathbf{M}$ 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 $\mathbf{M}$, 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 $\mathbf{M}$, a specific position p , and a specific input data object x . We denote this as $S(\mathbf{M}, p, x)$.

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

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

Because S is a set of signals, $\text{dif}()$ is more complicated than the simple $f()$ function used in Eq. 3.1. Recall the notations of an original testing dataset X , a set of transformations T and its corresponding sampling variable V . We can use $\text{dif}()$ to measure the difference between any two derived testing data objects $t_i(x)$ and $t_j(x)$ as:

$$\text{dif}(S(\mathbf{M}, p, t_i(x)), S(\mathbf{M}, p, t_j(x))) \quad (3.2)$$

Let us rewrite $S(\mathbf{M}, p, t_i(x))$ as $S_i = \{s_{i,1}, s_{i,2}, \dots\}$ and $S(\mathbf{M}, p, t_j(x))$ as $S_j = \{s_{j,1}, s_{j,2}, \dots\}$. The dif 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.

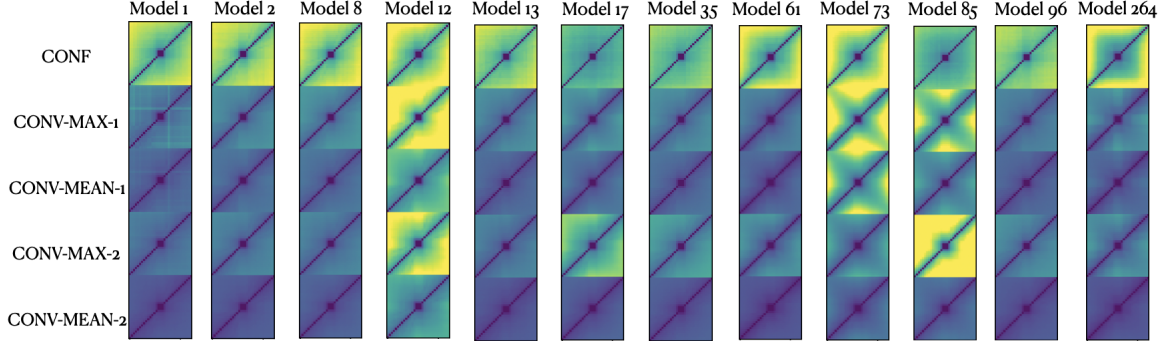


Figure 3.3: Five variance matrices for twelve models with almost the same robust accuracy were arbitrarily selected from the repository of 600 CNNs. All twelve models were trained on CIFAR10. The transformations simulate rotation variance, in the range of $[-15^\circ, 15^\circ] \pm 1^\circ$. The metadata of these models can be found in Table 3.3. 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 3.2.

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

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

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 $\mathbf{M}$ to a measurement, and can thus be considered as *modalities* in analogy to medical imaging [212]. 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 $\mathbf{M}$ and which testing position p we are referring to, we denote Eq. 3.2 as $\text{dif}_{i,j}()$. The function 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 [312], we can show that with this cumul function, $\delta_{i,j}$ concentrates around the true expected value $\hat{\delta}_{i,j}$ unless there are extreme values or numerical issues.

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

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

Table 3.2: For each of the twelve models in Figure 3.3, we show the measurements of four example features, together with the class labels used for training assessors for testing rotation invariance. The bold numbers indicate some anomalous values.

Model \ Measurements	1	2	8	12	13	17	35	61	73	85	96	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	0.879	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	1.530
Mean@CONV-1: Asymmetry	0.047	0.009	0.008	0.014	0.007	0.012	0.008	0.009	0.041	0.021	0.008	0.012
...	...	...	...	...	...	...	...	...	...	...	...	...
“Labels” for invariance	X	✓	✓	X	✓	X	✓	X	X	X	✓	X

Table 3.3: Metadata, accuracy and robust accuracy of the twelve CNNs (VGG13bn) shown in Figure 3.3.

Params Model	Accuracy	Robust Accuracy	Learning rate	Epoch	Batch size	Pre-trained	Augmentation	FGSM trained	Anomaly
1	89.4%	66.3%	e^{-5}	30	8	✓	$[-15^\circ, 15^\circ]$	X	Extreme values
2	88.0%	68.0%	e^{-5}	30	8	✓	$[-45^\circ, 45^\circ]$	X	X
8	89.0%	67.6%	e^{-5}	15	8	✓	$[-20^\circ, 20^\circ]$	X	X
12	87.8%	65.4%	e^{-4}	3	8	✓	$[-15^\circ, 15^\circ]$	X	X
13	89.4%	67.4%	e^{-5}	30	8	✓	$[-15^\circ, 15^\circ]$	X	X
17	89.1%	67.7%	e^{-5}	2000	16384	✓	$[-10^\circ, 10^\circ]$	X	X
35	88.0%	67.2%	e^{-4}	50	1024	✓	$[-15^\circ, 15^\circ]$	X	Smaller training set
61	88.1%	64.3%	e^{-4}	50	1024	✓	$[-5^\circ, 5^\circ]$	✓	Data leakage
73	91.5%	67.5%	e^{-4}	50	1024	✓	$[-5^\circ, 5^\circ]$	X	Impaired labels
85	89.8%	65.3%	e^{-4}	500	1024	✓	$[-5^\circ, 5^\circ]$	X	Noisy data
96	85.8%	65.5%	e^{-4}	50	1024	✓	$[-90^\circ, 90^\circ]$	X	X
264	93.5%	66.3%	e^{-4}	500	1024	✓	$[-10\%, +10\%]$ Size	✓	Data leakage

Robust accuracy was evaluated on $[-15^\circ, 15^\circ]$. Accuracy was evaluated on testing set of CIFAR10.

where the elements in the matrix are ordered according to the sampling variable V horizontally and vertically. In Figure 3.1, we use $f_1, f_2, \dots$ to denote different modalities (dif, 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 [153].

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. Figure 3.2 shows six examples of such images, which are referred to as *variance matrices*.

Image Analysis. One can observe a variety of imagery patterns from the variance matrices as illustrated in Figure 3.2. 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 $\mathcal{T}$, 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 Figure 3.2 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.
- *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 $[-\alpha, \alpha] \times [-\alpha, \alpha]$. 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 Figure 3.3 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.

Invariance Data Labels. The judgments about different patterns can vary from person to person, especially when such images are unfamiliar to many potential human annotators. Therefore we hire three professionals in ML/DL, and we provide them with all the metadata and the visual patterns shown in Figure 3.3.

In many fields, e.g., facial action units [174], natural language processing (NLP) [126], segmentation [59], 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 [52] using differential comparison. As suggested by [71], when providing new research approaches/annotations, we use two inter-rater reliability (IRR) measures, i.e., Cohen’s kappa [51] and Fleiss’ kappa [79], to evaluate how much confidence we can have on our invariance labels.

We consider that the level of $\mathcal{T}$ -invariance of ML models should be assessed as a generic quality measurement, while the requirements for invariance quality may be applicant-dependent. For example, given two ML models for object detection, the annotations for $\mathcal{T}$ -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.

3.4.2 Development Workflow

In order to avoid manually evaluating invariance qualities, we train an assessor to automatically perform $\mathcal{T}$ -invariance assessment. For a given simulation of $\mathcal{T}$, to train an 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 Figure 3.1, 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 assessor for $\mathcal{T}$ -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 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.

Decision Tree Learning. We use techniques in the decision tree (DT) family to train our 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 [83, 84].

Given a family of transformations $T = \{t_0, t_1, \dots, t_n\}$, a set of m modalities $F = \{f_1, f_2, \dots, f_m\}$ and a set of l positions $P = \{p_1, p_2, \dots, p_l\}$, there are $m \cdot l$ variance matrices Δ , each of $(n+1)^2$ dimensions. The goal of an assessor $\mathbf{Q}$ 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 $\mathcal{M} = \{\mathbf{M}_1, \mathbf{M}_2, \dots\}$. Since $\mathbf{Q}$ 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. 3.1 by [88] defines 1 as fully invariant.

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.

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 Section 3.5.3.2) for each variance matrix. The $m \cdot l$ variance matrices thus yield $16m \cdot l$ features. For example, Figure 3.3 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 3.3. Such metadata is not used in training any assessor.

Table 3.2 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.

3.5 Experimental Results and Analysis

In this section, we show our experimental results using the proposed automatic testing framework, as well as discussing and analysing the experimental results.

3.5.1 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 (4, 8, 1024, and 16384); (f) Optimizers (Adam or SGD); (g) Anomalies (*none*, data leakage); (h) Training with adversarial attacks (*none*, FGSM); (i) Hardware (CPU, GPU); (j) Added variance (*none*, rotation, scaling, brightness); (k) Transformation ranges (*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 assessors, the experiments were run on a single laptop (Apple M1 Chip) without GPU. The metadata (including all hyper-parameters) for the 600 models can be found in the metadata appendix.

In addition to pretrained models, we train 150 CNN5 models on MNIST from scratch. 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
- Dropout: 25% + ReLU
- Linear5: input 64-d, output 10-d

We used Pytorch [234] to implement the structure and the way of initialisation for the weights and biases were set to default.

3.5.2 Invariance Labelling Convention

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 $\times 0.7$ to $\times 1.3$ 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 cast 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”.

Note that 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 [337], and lying/emotion detection annotation [174].

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.

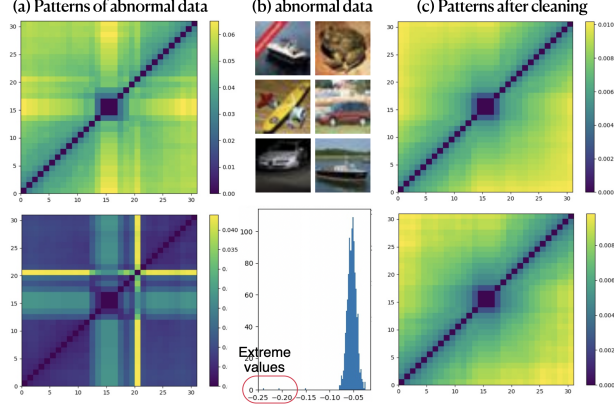


Figure 3.4: Rotation variance matrices $mean@CONF-1$ from a VGG13bn trained on CIFAR10 (model ID: 1). (a) abnormal data causes the variance matrix to be “jagged”. (b) Abnormal data lead to extreme activation values which are far away from the distribution. (c) After cleaning, the patterns become normal.

Table 3.4: IRR: Cohen’s and Fleiss’ kappa scores

Cohen’s	Coder 1	Coder 2	Coder 3	Pseudo GT	Fleiss’
Coder 1	1	0.818	0.801	0.927	0.793
Coder 2	0.818	1	0.763	0.890	
Coder 3	0.801	0.763	1	0.871	

- **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 $dif()$ function would not be sub-gaussian, therefore this leads to abrupt changes on the variance matrices, as shown in Figure 3.4. 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).

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 3.4, the IRR

Table 3.5: Comparing the performance (accuracy) of assessors and a numerical assessor in four development workflows.

Transformation Classifier/Regressor	Rotation with		Brightness with	Scaling with
	VGG13bn on CIFAR10	CNN5 on MNIST	VGG13bn on CIFAR10	VGG13bn on CIFAR10
Decision Tree	84.73%±4.03%	82.93%±3.73%	84.01%±3.34%	91.33%±5.19%
Random Forest	91.31%±2.02%	87.66%±1.02%	91.40%±2.12%	94.66%±1.13%
AdaBoost	89.90%±4.62%	85.46%±2.26%	89.93%±4.20%	93.86%±3.33%
Linear Regression	75.46%±3.13%	68.26%±2.93%	72.70%±6.77%	78.80%±3.20%
Logistic Regression	78.42%±2.33%	64.14%±1.20%	80.68%±0.79%	80.01%±1.03%
baseline [88]	76.67%	83.33%	79.33%	78.67%

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

scores (i.e., the Cohen's and Feliss' kappa scores) are around 0.76 to 0.82, which are similar to the quality of the annotations measured in a few NLP applications [71].

3.5.3 Training and Testing Assessors.

We trained multiple assessors using the techniques in the decision tree family, including decision trees, random forests, AdaBoost, linear and logistic regression. We chose these relatively simple techniques due to the limited amount of ML models we collected in our model repository. We used the scikit-learn libraries [237] 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. 3.1 as the baseline for comparison. We used greedy search to find the threshold on the training data.

To train each assessor, we selected a relatively balanced subset of “variant” and “invariant” models according to the type of invariance $\mathcal{T}$ that the assessor was targeted for. This selection process resulted in four different development workflows denoted in Table 3.5 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.

When testing the assessors, although we use the standard cross-validation protocol, we also separate a “hold-out testing set” for each partitions as shown in Table 3.6. More specifically, partition 1 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

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.

Table 3.6: Four partitions on the model-repository

	Partition (a)	Partition (b)	Partition (c)	Partition (d)
Regular	mid 1 - 100	mid 101 - 200	mid 201 - 300	mid 301 - 400
"Hold out"	mid t1 - t50	mid t101 - t150	mid t201 - t250	mid t301 - t350

For each partition, the statistics of the assessment labels are listed in Table 3.7. Each partition is considered relatively balanced.

Table 3.7: Statistics of the assessment labels on each partition

	Partition (a)	Partition (b)	Partition (c)	Partition (d)
Invariant	66	78	56	86
Variant	84	72	94	64

Table 3.8: Accuracy of assessors on the imbalanced model repository for three different types of invariance qualities.

	Rotation	Brightness	Size
Decision Tree	93.5±2.8%	94.4±2.0%	98.2±0.4%
Random Forest	97.8±0.9%	97.3±0.7%	99.0±0.3%
AdaBoost	96.5±1.0%	96.9±0.9%	98.9±0.7%
Linear Regression	94.8±0.8%	90.8±1.0%	96.2±0.7%
baseline	92.2%	93.1%	92.8%

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 3.8, we still observe similar results, e.g., random forest performs the best. All the results were also obtained on 10 repeated experiments. We use the same three-fold cross-validation protocol mentioned above to carry out each experiment. The proposed framework performs $\sim 5\%$ better than the baseline. Note that the assessors’ performances were evaluated using simple formula-based accuracy score, which we believed was not ideal. However, in this thesis, we focus on improving invariance testing, and we leave more advanced testing techniques for standard models’ performances to future works.

3.5.3.1 Experiment Details on 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:

$W_1, W_2: [-15^\circ, 15^\circ] \pm 1^\circ,$

$W_3, W_4: [-30\%, +30\%] \text{ or } [\times 70\%, \times 130\%] \pm 2\%.$

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.

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 semantic features which should be considered when conducting invariance testing. Coder 3: while *CONF* 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 *CONF*. *CONV-2* was checked by the three coders only when they could not make a decision after they checked variance matrices generated at *CONF* and *CONV-1*.

3.5.3.2 Feature Extraction

For each variance matrix, we generated 16 feature measurements, resulting in a $5 \times 16 = 80$ dimensional *feature vector*.

Table 3.9 shows that some feature measurements at the CONF level have a strong correlation with the robust accuracy defined in Eq. 3.1. However, as shown in Figure 3.3, 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 assessor(s), and thus replace the simple formula-based accuracy score defined in Eq. 3.1.

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.

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

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

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).

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

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

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

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

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

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

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

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

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 $r = 90$.

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

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

$$e_{i,j}^{hg} = \delta_{i,j-1} - \delta_{i,j} \quad i = 1, 2, \dots, n; j = 1, 2, \dots, i$$

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

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

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

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

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

$$\mu_i = \frac{1}{i} \sum_{j=1}^i e_{i,j}^{hg} \quad i = 1, 2, \dots, n$$

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

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

$$e_{i,j}^{vg} = \delta_{i+1,j} - \delta_{i,j} \quad j = 0, 1, \dots, n-1; i = j, \dots, n-1$$

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

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

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

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

$$\mu_j = \frac{1}{n-j} \sum_{i=j}^{n-1} e_{i,j}^{vg} \quad j = 0, 1, \dots, n-1$$

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

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

$$e_{i,j}^{dg} = \delta_{i+1,j-1} - \delta_{i,j} \quad i = 1, 2, \dots, n-1; j = 1, 2, \dots, i$$

$$\phi_{\text{dg-mean}} = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=1}^i e_{i,j}^{dg}$$

13. Diagonal Gradient (Standard deviation). The standard deviation value of the gradient values of the variance matrices in the diagonal direction.

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

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

$$\phi_{\text{G-overall}} = \left(\frac{\phi_{\text{hg-mean}}}{\phi_{\text{hg-std}}} + \frac{\phi_{\text{vg-mean}}}{\phi_{\text{vg-std}}} + \frac{\phi_{\text{dg-mean}}}{\phi_{\text{dg-std}}} \right) / 3$$

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

$$\mu_r = \sum_{j=0}^{n-r} \frac{\delta_{j+r,j}}{n-r+1} \quad r = 1, 2, \dots, n-1$$

$$\phi_{\text{dctny}} = \sum_{r=1}^{n-1} \sum_{j=0}^{n-r} (\delta_{j+r,j} - \mu_r)^2 / \phi_{\text{mean}}$$

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

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

Proposition 1 When the $\text{dif}_{i,j}()$ function can be written as $f_i() - f_j()$, e.g., *max* or *mean*, the square value mean ϕ_{svm} 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.

$$\begin{aligned}
& \textit{Proof: Square Mean of the Matrix} : \frac{1}{2\tau^2} \sum_i \sum_j \delta_{i,j}^2 \\
&= \frac{1}{2\tau^2} \sum_i \sum_j \sum_{x \in X} \mathbb{E} [f_i^2(x) + f_j^2(x) - 2f_i(x)f_j(x)] \\
&= \underbrace{\frac{1}{\tau} \sum_i \sum_{x \in X} \mathbb{E} [f_i^2(x)]}_{\cong \mathbb{E}_{y \in T(X)} [f^2(y)]} - \underbrace{\frac{1}{\tau^2} \sum_i \sum_j \sum_{x \in X} \mathbb{E} [f_i(x)f_j(x)]}_{\cong \mathbb{E}_{z_i, z_j \in T(X)} [f(z_i)f(z_j)]} \\
&\cong \mathbb{E}_{T(X)} [f^2(y)] - \mathbb{E}_{T(X)} [f(z_i)] * \mathbb{E}_{T(X)} [f(z_j)] \\
&\quad - \text{COV}_{T(X)} [f(z_i), f(z_j)] \\
&= \mathbb{E}_{T(X)} [f^2(y)] - \mathbb{E}_{T(X)} [f(y)]^2 - \text{COV}_{T(X)} [f(z_i), f(z_j)] \\
&= \text{VAR}_{T(X)} [f(y)] - \text{COV}_{T(X)} [f(z_i), f(z_j)]
\end{aligned}$$

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

Table 3.9 shows the correlation between the accuracy evaluation [88] and our measurements. At the confidence score level (CONF), ϕ_{svm} , ϕ_{dctny} and $\phi_{G\text{-overall}}$ have a strong correlation to robust accuracy.

Table 3.9: Correlation between measurements and Accuracy

	Square Mean		Discontinuity		Gradient Overall	
	CONF	CONV-1	CONF	CONV-1	CONF	CONV-1
RobustAcc	-0.63	-0.10	-0.53	-0.34	-0.47	-0.11
Accuracy	-0.12	-0.03	-0.06	-0.11	-0.15	-0.21

3.5.4 Ablation Studies

Size of testing suite. In the Figures 3.5 3.7 3.6, 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 3.10, we show that, under this setting, our testing framework (the 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

Table 3.10: Accuracy of assessors $\mathbf{Q}(\mathbf{M}, X)$ on different model-repository partitions. Training set of the model-repository: measurements generated using half of the testing suite X . Testing set of the model-repository: measurements generated using the other half of the testing suite X .

Transformation Classifier/Regressor	Rotation		Brightness	Scaling
	<i>VGG13bn on CIFAR10</i>	<i>CNN5 on MNIST</i>	<i>VGG13bn on CIFAR10</i>	<i>VGG13bn on CIFAR10</i>
Decision Tree	82.60%±8.06%	81.93%±3.93%	85.86%±4.53%	89.66%±3.00%
Random Forest	91.47%±2.13%	87.13%±0.46%	90.66%±1.33%	94.20%±1.13%
AdaBoost	91.67%±3.66%	86.20%±2.46%	88.73%±2.73%	92.20%±2.86%
Linear Regression	74.13%±8.53%	67.00%±4.33%	73.46%±4.53%	75.73%±6.93%
*Baseline (<i>Gao et al 2020</i>)	76.67%	83.33%	79.33%	78.67%

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

Table 3.11: Accuracy of assessors on different model-repository partitions. Variance matrices are generated using random proportion of the testing suite X for both training and testing set of the model-repository.

Transformation Classifier/Regressor	Rotation		Brightness	Scaling
	<i>VGG13bn on CIFAR10</i>	<i>CNN5 on MNIST</i>	<i>VGG13bn on CIFAR10</i>	<i>VGG13bn on CIFAR10</i>
Decision Tree	82.60%±3.93%	82.67%±2.00%	84.40%±2.40%	88.13%±2.80%
Random Forest	90.73%±1.19%	87.33%±1.66%	90.46%±1.80%	94.13%±1.20%
AdaBoost	86.86%±2.86%	86.33%±2.13%	90.66%±3.26%	93.39%±1.93%
Linear Regression	66.60%±7.13%	74.60%±5.93%	67.59%±2.93%	68.66%±4.66%
*Baseline (<i>Gao et al 2020</i>)	76.67%	83.33%	79.33%	78.67%

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

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 assessors. In Table 3.11, we show that when generating the measurements using random proportion of the testing suite X , our testing framework (the assessors) is still able to achieve satisfactory results (over 90% on the testing set of the balanced model-repository). Finally, in Table 3.21 we show that the trained assessors do not overfit despite of the small size of the model repository. When we only use 10% to 90% of the models to train the assessor, we can use the rest as the validation set. And we show that the performances of the trained assessors are similar on the validation set and the testing set.

3.5.5 Result Analysis.

In each of four development workflows, $W_1 \sim W_4$, we trained four assessors, together with a numerical assessor as the baseline for comparison. The performances of these assessors are shown in Table 3.5, from which we can make the following observations:

- The three DT assessors performed noticeably better than linear regression and the numerical assessor (Eq. 3.1). The only exception is perhaps with workflow

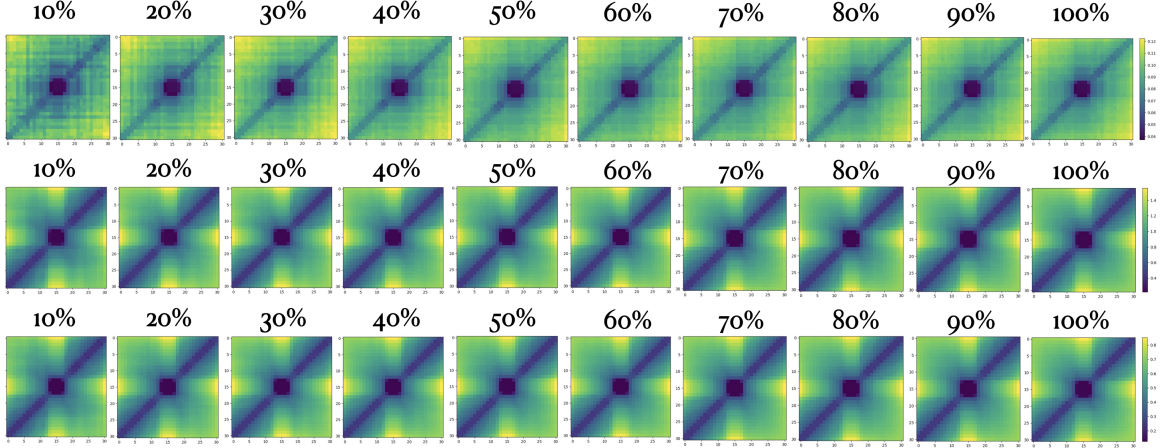


Figure 3.5: Rotation variance matrices: (1) First row: $\max@CONF$; (2) Second row: $\max@CONF-1$, i.e., the last convolutional layer (The ninth convolutional layer for VGG13bn); (3) Third row: $\text{mean}@CONF-1$ of a randomly chosen VGG13bn (model id: 15) trained without anomaly. The visual patterns 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 ($\max@CONF$). And we see no significant difference between the visual patterns on the last two rows.

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 [52, 83, 84] 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 assessors.

The results in Table 3.5 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

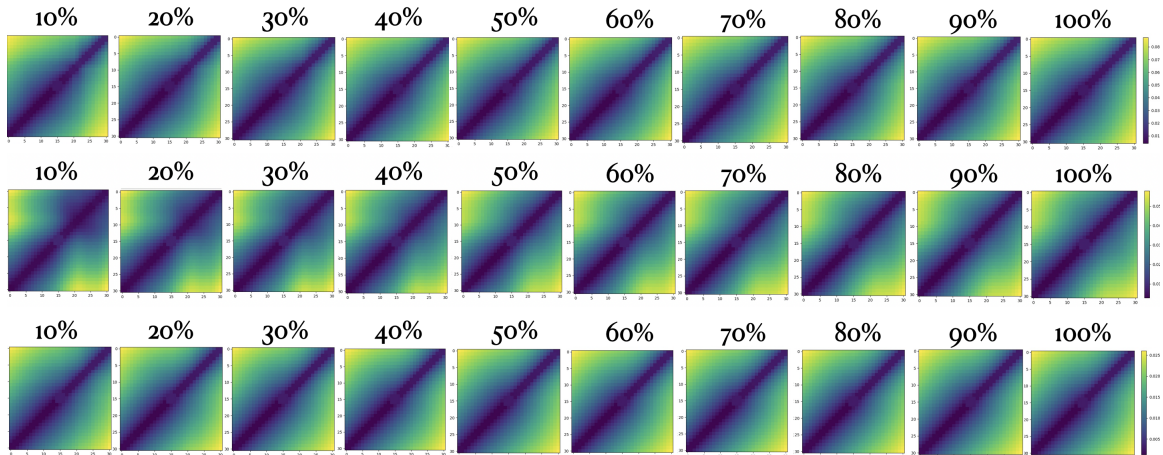


Figure 3.6: Brightness variance matrices: (1) First row: $\max@CONF$; (2) Second row: $\max@CONF-1$, i.e., the last convolutional layer (CONV9) for VGG13bn; (3) Third row: $\text{mean}@CONF-2$, i.e., the penultimate convolutional layer (CONV8) for VGG13bn, of a randomly chosen VGG13bn (model id: 102) 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).

the statistics (e.g., entropy) is less accurate. Although random forest and boosting can help, it is desirable to continue expanding the model repository.

3.6 Case Study: Invariance Testing using Sparse Linear Layers

In this Section, we discuss an alternative way to obtain variance matrices we described in Section 3.4. Instead of accumulating signals from the intermediate layers and final decisions from the ML models, we show the possibility to obtain such invariance matrices using sparse linear layers [377]. To utilize sparse linear layers for invariance testing, we treat each ML model as a feature extractor. From transfer learning [229], metric learning [152], to self-supervised learning [133], machine learning models are often used as feature extractors. Furthermore, for any machine learning model, it is always desired to investigate how robust/invariant the final feature vectors are [360]. Therefore, in this section, we focus on the final feature vectors, and provides a novel testing approach to evaluate invariance qualities with trained sparse linear layers instead of statistical scores.

When testing feature extractors, normally one would train a linear classifier (or a feature selection method), e.g., SVM [36] or a multi-layer perceptron [259]. However,

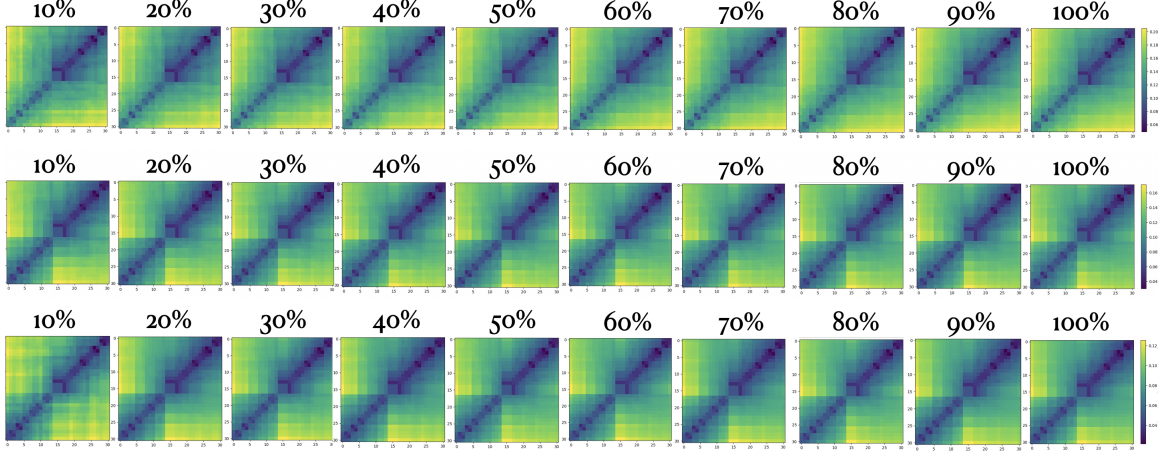


Figure 3.7: Scaling (Size) variance matrices: (1) First row: $\max@CONF$; (2) Second row: $\max@CONF-1$, i.e., the last convolutional layer (CONV9) for VGG13bn; (3) Third row: $\text{mean}@CONF-1$ of a randomly chosen VGG13bn (model id: 202) trained without anomaly obtained using smaller proportions 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).

when conducting invariance testing, these traditional methods will have two major issues: (1) training, i.e., gradient descent, for linear layers is not stable [135], resulting in different sets of weights for different runs. (2) the number of selected features is often large, e.g., $\geq 60\%$ in our case. These factors make the comparison between two sets of selected features difficult and unreliable. Therefore, we use sparse linear layers to address the two issues, and we showcase the resultant visual patterns can lead to reliable and consistent decisions, e.g., a high inter-rater reliability (IRR) score.

3.6.1 Sparse Linear Layers

In order to test whether a feature extractor $F : x \mapsto F(x) \in \mathbb{R}^d$ is invariant, where x is a data object and d is the dimension of the extracted feature vectors $F(x)$, we propose that we firstly train a series of stable sparse linear classifiers [62, 377]. We use the variable Z to represent the feature vector, and an instance is denoted as $z = F(x)$. Similarly, we have Y for the ground truth and $\tilde{Y}$ for the output of the sparse linear layers. This way, for sparse linear layers, i.e., weights ω and bias β , we have: $\tilde{Y} = Z^T \omega + \beta$. Similarly, the loss function of the sparse linear layers can be defined as $\mathcal{L}(X, Y) = \frac{1}{N} \sum (y - \tilde{y})^2$, where N is the number of input data objects. Then training a sparse linear layer is to obtain a set of weights, which are denoted as

$\theta = \{\omega, \beta\}$:

$$\min_{\theta} \mathcal{L}(X, Y) + \lambda R_{\alpha}(\theta) \quad (3.3)$$

where $R_{\alpha} = (1 - \alpha) \|\theta\|_2^2 / 2 + \alpha \|\theta\|_1$, which is referred to as elastic net regularisation [377] for the parameter α and λ . α is a hyper-parameter controlling the regularisation terms varying between LASSO ($\alpha = 1$) and ridge ($\alpha = 0$) regression. Therefore, we can obtain a series of such weights θ when using different $\lambda_1, \lambda_2, \dots, \lambda_k$:

$$\theta_i = \min_{\theta} \mathcal{L}(X, Y) + \lambda_i R_{\alpha}(\theta) \quad (3.4)$$

the greater the value of the parameter λ_i , the sparser the linear classifier is (more zero weights) given $\alpha \neq 0$. This way, we will know which features contribute to the final decisions the most, and we can largely reduce the number of features that we need to visualise / consider when testing a feature extractor.

In order to test whether a feature extractor is robust against a certain type of transformation $\mathcal{T}$, we can define a sequence of transformation operations, e.g., to rotate $-15^\circ, -10^\circ, 10^\circ$, and 15° . We denote such transformation functions as $t()$. We can then apply the same transformation function $t()$ to all data objects in the training set X , and use the transformed data objects $t(X)$ to train a sequence of sparse linear classifiers. Similarly, we will have a regularisation path:

$$\theta_{i,t} = \min_{\theta} \mathcal{L}(t(X), Y) + \lambda_i R_{\alpha}(\theta) \quad (3.5)$$

We can then visualise and compare the difference between different regularisation paths $\{\theta_{i,t} | i = 1, 2, \dots, k\}$ obtained using different transformation function $t()$. We suggest arrange the transformation operations in descending or ascending order for better visualisation purpose, e.g., for rotation:

$$t_r = \{-15^\circ, -14^\circ, \dots, -1^\circ, 0^\circ, 1^\circ, \dots, 14^\circ, 15^\circ\}$$

When visualising sparse linear layers, we fix the number of λ , and we denote the entire series of the k sparse linear layers as θ_t by omitting the i subscription (which indicates the i^{th} lambda values). For example, in Figure 3.8 we show sparse linear layers $\theta_{t_{15^\circ}}$ in the last row, and the i^{th} line represents the i^{th} lambda value.

3.6.2 Invariance Testing Workflow using Sparse Linear Layers

The workflow for the proposed framework is shown in Figure 3.9. As described above, we use sparse linear layers to conduct the feature selection process. For each feature

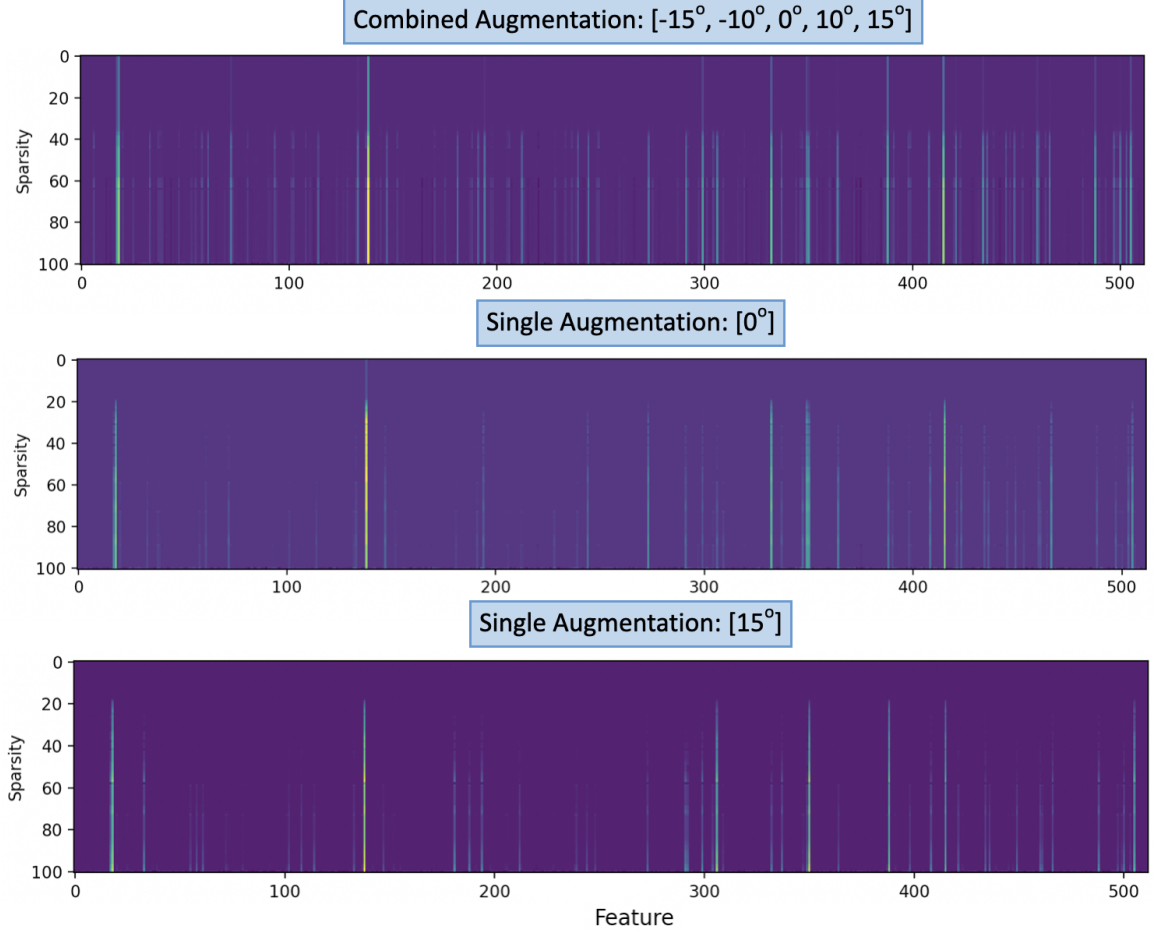


Figure 3.8: Upper row: sparse linear layers trained using combined augmentation. Middle / lower row: sparse linear layers trained using original training set X , and transformed training set $t_{15^\circ}(X)$ respectively.

extractor (or model) $\mathbf{M}$, the outputs of the feature selection process are multiple sets of sparse linear layers θ_{t_j} for j in the targeted/interested interval, e.g., $[-15^\circ, 15^\circ]$ for $\mathcal{T}$ being rotation.

We can then measure the difference between all the θ_{t_j} . However, if we consider τ transformation operations in the targeted/interested interval, e.g., τ angles for rotation, we will need to make $C(\tau, 2) = \binom{\tau}{2}$ comparisons for the τ sets of sparse linear layers. Therefore, we can use an invariance matrix to help scale up the testing procedure. We define an *invariance matrix* as:

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

Evaluation Workflow

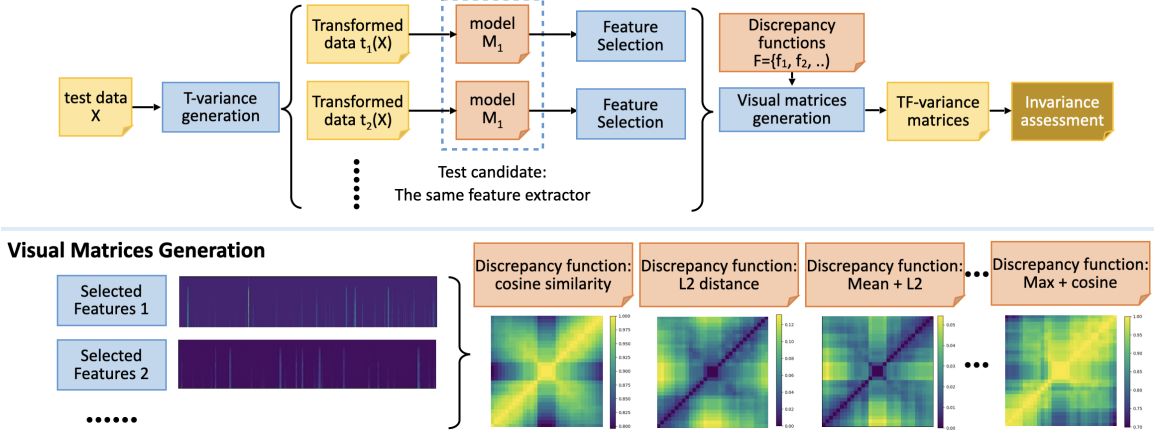


Figure 3.9: Testing procedure using sparse linear layers: We first define τ transformation operations $t()$, e.g., to rotate $\{-15^\circ, -14^\circ, \dots, 15^\circ\}$. And then we apply each $t()$ to all the data objects in the original testing suite X to form a transformed testing suite $t(X)$. For each feature extractor or model M , we train τ sets of sparse linear layers $\{\theta_j | j = 1, 2, \dots, \tau\}$ using the τ transformed testing suite $t(X)$. And then we use different discrepancy functions to evaluate the differences between the τ sets of sparse linear layers. We then use the visual patterns to analyse the invariance quality of M .

where M is the testing candidate (a feature extractor), X is a test suite that is used to train multiple sets of sparse linear layers, and we define $\delta_{i,j} = \text{dif}(\theta_{t_i}, \theta_{t_j})$ as the distance between θ_{t_i} and θ_{t_j} .

Similar to the testing framework described in section 3.4, in order to compare the difference(s) between θ_{t_i} and θ_{t_j} , we can use different $\text{dif}()$ functions, e.g., L2 distance/cosine similarity etc, to generate different variance matrices. This way, for each feature extractor, we can generate multiple variance matrices, which is similar to multi-modalities in the medical imaging testing process [153]. Examples can be found in Figure 3.9.

3.6.3 A New Test Oracle: Associativity

With the assistance of sparse linear layers for invariance testing, we introduce a new testing oracle on associativity: whether sparse layers obtained using combined augmentation (i.e., $t_{1,2,\dots,\tau}$) are the combination of the sparse layers obtained using individually augmented data, i.e., t_i , for $i=\{1, 2, \dots, \tau\}$. To be more specific:

$$\theta_{\bigcup_{k=1}^{\tau} t_k} \stackrel{?}{=} \bigcup_{k=1}^{\tau} \theta_{t_k} \quad (3.6)$$

where τ is the number of transformation operations, and we define the union operation on a set of sparse linear layers θ_{t_k} for $k = 1, \dots, \tau$ as “to choose the greatest value for each weight from the τ sets of sparse linear layers”. We can use either L2 distance or cosine similarity to evaluate associativity.

Finally, we can assess if the model is invariant or not by checking the variance matrices, the trained sparse linear layers, and the associativity score. We evaluated the invariance qualities on a small model repository consisting of 600 CNNs as specified in section 3.5.

3.6.4 Actionable: Feature Selection

As a testing tool, another advantage that sparse linear layers have is that they can provide potentially actionable items for feature selection. For example, if θ_{t_i} indicates $\{f_{i_1}, f_{i_2}, \dots, f_{i_n}\}$ are important (and hence selected), and θ_{t_j} indicates $\{f_{j_1}, f_{j_2}, \dots, f_{j_m}\}$ are important (and hence selected), then the union of these selected features can be a superior solution than any of the separate ones. Additionally, the union of these features can be superior to the weights trained by using combined augmentation (e.g., $\theta_{t_i} \cup \theta_{t_j}$).

In Figure 3.10, we show that the union of the weights consistently performs better than weights that are trained using combined augmentation. We also apply a mask to ensure the number of selected merged weights is always less or equal to the number of weights obtained from combined augmentation, i.e., masked line in Figure 3.10. To be more specific, we choose the top k non-zero weights from merged weights, where k is the number of non-zero combined weights. Again, accuracy or any formula-based scores are only one simple way to evaluate any models. However, we focus on invariance testing in this thesis and leave more sophisticated evaluation approaches in future work.

3.6.5 Consistency and Reliability

We use the proposed invariance testing process to evaluate a model repository of 600 CNNs. In order to confirm whether the proposed testing process can assist researchers to judge the invariance qualities of models more reliably and consistently, we ask two ML professionals to assess if those models are invariant or not by 1) checking a few random examples, or by 2) checking the multiple criteria we propose, i.e., the variance matrix, sparse linear layers, and the associativity score. Note that the metadata, e.g., hyper-parameters used for training the models are not provided. In Table 3.22 we

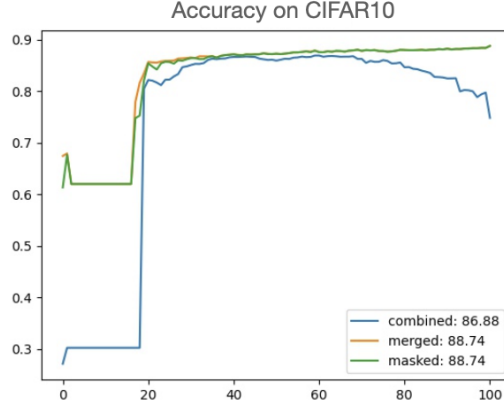


Figure 3.10: Merged weights are obtained by combining $\{\theta_{-15^\circ}, \theta_{-10^\circ}, \theta_{0^\circ}, \theta_{10^\circ}, \theta_{15^\circ}\}$. Combined weights are obtained by training sparse linear layers using augmentation of $\{-15^\circ, -10^\circ, 0^\circ, 10^\circ, 15^\circ\}$. Masked weights are obtained by constraining the number of non-zero merged weights to be less or equal to the number of non-zero combined weights. Both the merged and masked weights outperform the combined weights. x-axis: sparsity, y-axis: accuracy.

show that the proposed method is more consistent and reliable than checking only a few random examples. Additionally, the judgements using the proposed testing approach have a similar inter-rater reliability (IRR) score to many NLP tasks [71]. In addition, the reliability scores are improved when associativity scores are taken into consideration when making decisions on the invariance qualities of the ML models.

3.7 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 (~ 0.76 - 0.82) of invariance annotations, which would be considered between “Good” and “Excellent” in typical NLP applications [71].

To further consolidate our methodology, we conducted a case study on testing ML models’ invariance qualities using sparse linear layers with the introduced testing framework. The experimental results showed that our testing approach can lead to

reliable and consistent decisions and judgements on the invariance qualities of ML models.

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 assessors based on DT techniques can achieve around $83 \sim 95\%$ accuracy, noticeably better than linear regression and the robust accuracy.

Whilst the proposed invariance testing framework works for basic data attributes (e.g., brightness), theoretically, it could also work with more complex variables (e.g., background). To confirm this hypothesis, we need to investigate how it can be used for complex variables. In the next chapter, we extend the invariance testing framework introduced in this chapter to conduct background invariance testing.

Table 3.12: Measurements on rotation variance matrices ($max@CONF$) 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 3.13: Measurements on rotation variance matrices ($max@CONV-1$) 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 3.14: Measurements on rotation variance matrices ($mean@CONV-1$) 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

Table 3.15: Measurements on brightness variance matrices ($max@CONF$) 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
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 3.16: Measurements on brightness variance matrices ($max@CONV-1$) 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	0.118	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 3.17: Measurements on brightness variance matrices ($mean@CONV-2$) 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
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

Table 3.18: Measurements on scaling variance matrices ($max@CONF$) 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 3.19: Measurements on scaling variance matrices ($max@CONV-1$) 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 3.20: Measurements on scaling variance matrices ($mean@CONV-1$) 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

Table 3.21: Accuracy of assessors on validation set and testing set with varying proportion of models used for training

Proportion Accuracy	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
Validation Set	81.64%	86.93%	87.09%	87.21%	87.68%	87.24%	87.94%	88.01%	87.93%	88.49%
Testing Set	77.54%	85.46%	87.00%	86.82%	87.02%	87.18%	87.29%	87.24%	87.23%	87.96%

Table 3.22: Inter-rater reliability: Cohen’s kappa scores

	Random examples	Sparse Linear Layers	Sparse Linear Layers and Associativity Score
Cohen’s	0.195	0.683	0.708

Chapter 4

Invariance Testing for Advanced Data Attributes: Background Invariance Testing

4.1 Introduction

There are a variety of invariance qualities associated with machine-learned models. Testing these invariance qualities enables us to evaluate the robustness of a model in its real-world application where the model may encounter variations that do not feature sufficiently in the training and testing data. The testing also allows us to observe the possible biases or spurious correlations that may have been learned by a model [327] and to anticipate if the model can be deployed in other application domains [319]. This work is concerned with background invariance testing – a relatively challenging type of testing.

As mentioned in the previous chapter, visualization can be more suitable and informative for invariance testing. To leverage the visualized testing approach rather than simply reporting an accuracy score, selecting the testing results is important for observing the level of robustness in relation to the likelihood of different variations (e.g., a slightly rotated car vs. an upside-down car in an image). However, in background invariance testing, the term “background” is a multivariate variable often described qualitatively (e.g., an outdoor scene, a desert environment, etc.), which makes it challenging to sample such a complex variable space systematically as opposed to relying on random selection. Visualization-based testing frameworks often rely on human judgment of visualized patterns. When working with a vast data space, selecting only a small subset for evaluation without a systematic sampling strategy can result in inconsistent test-example selections across different runs. This inconsis-

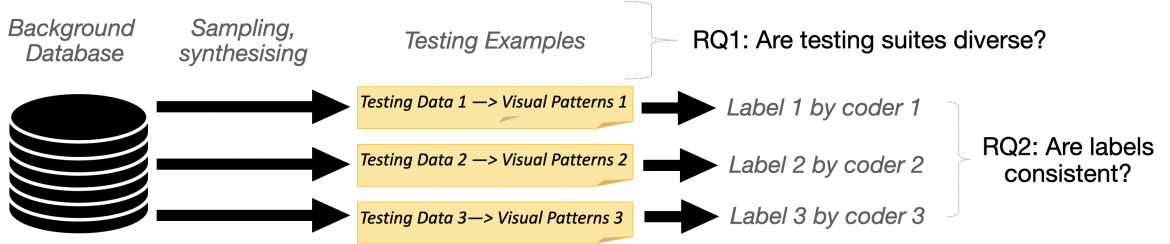


Figure 4.1: Research questions: to utilize informative visualization-based techniques for background invariance testing, this paper focused on: 1) are the selected testing examples diverse, and 2) are the resultant human decisions (based on the visual patterns) consistent.

tency leads to shifting visual patterns and unstable human judgments, as illustrated in Figure 4.2. Conversely, a sampling strategy that favours a specific type of test case may yield more consistent human judgments but at the cost of reduced diversity in the test suite. To effectively leverage visualization-based testing (more informative) for background invariance, this chapter aims to balance the trade-off between test data diversity and judgment consistency (as shown in 4.1).

In the previous chapter, we introduced a generic invariance testing framework for basic data attributes, e.g., rotation. In this chapter, to utilize the testing framework for more complex variables, i.e. background, we present a technical solution to address the need to select appropriate background scenes using semantic information. Our technical solution is built on the existing techniques of scene understanding in computer vision and those of ontological networks that are used in many text analysis applications.

Our contributions are as follows:

- a. We qualitatively confirm that the visualization-based testing method is more informative by showing that ML models with the same averaged worst-case accuracy score can behave differently (i.e., different visual patterns).
- b. We qualitatively confirm that the visualization-based testing method suffers from trade-offs between diversity of testing suites and consistency between human decisions.
- c. To overcome the trade-off, we introduce an algorithm to search for n desired background scenes based on the semantics encoded in each original image using association ontology.

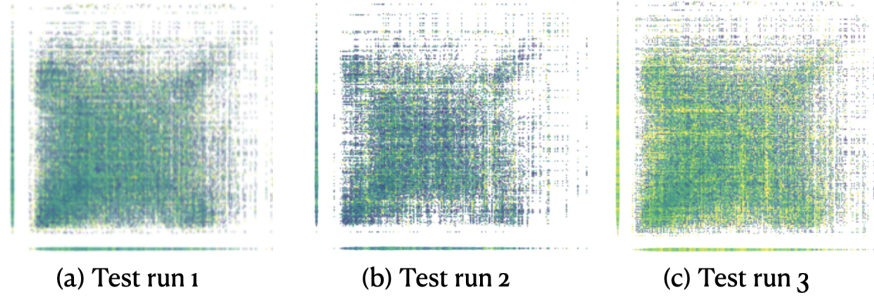


Figure 4.2: Random sampling leads to inconsistent visual representations across different testing runs, making visualization-based testing frameworks or human-centric testing methods inconsistent. We show the proposed testing framework leads to more consistent testing patterns in Figure 4.14.

- d. We quantitatively prove that our testing approach is the most balanced between diversity (*recall*) and consistency (*precision*) with the highest f1 score, compared with other testing methods.
- e. We show that the proposed background invariance testing based on visual representations can be fully automated.

4.2 Related Work

The invariance qualities of ML models have been studied for a few decades. In recent years, invariance testing has become a common procedure in invariant learning [13], [263], [55]. Among different invariance qualities, background invariance is attracting more attention. In the literature, several types of variations were introduced in background invariance testing, e.g., by replacing the original background with random noise, color patterns, and randomly selected background images. In this section, firstly, we introduce existing attempts, followed by an introduction to other techniques that are used in this work to help conduct background invariance testing.

[257] tested object detection models by transforming the original background to random noise or black pixels. They reported that all tested models failed to perform correctly at least in one of their testing cases. Similarly, [371], [45], [46] replaced parts of the images with black or grey pixels for foreground invariance testing. [61] noticed that the association between a foreground object and its background scene affected object recognition and described such association as “consistency”. [157] tested different models with consistent and inconsistent backgrounds, while using the term

“semantically-related” to describe consistent association. In particular, they used color texture to replace the original background of the target image and controlled the inconsistency using a parameterized texture model [243]. Several researchers experimented with swapping background scenes in studying background invariance, e.g., [61]. [340] provided the Background Challenge database by overlaying a foreground object to all extracted backgrounds from other images. To prepare models (to be tested), they also provided a smaller version of ImageNet with nine classes (IN9). In this work, we train a small repository of models on IN9, i.e., the models being tested in this work were trained for image classification.

To test deep models, an ideal testing suite should trigger as many neurons as possible. [239] defined the percentage of neurons triggered by a testing suite as the *neuron coverage rate* of the testing suite. Recent studies utilize coverage-based fuzzing techniques to find a testing suite that triggers more potential “bugs” for an ML model, e.g., TensorFuzz [225] and DeepHunter [343]. In this work, we adopt the most commonly used neuron coverage rate [239] to evaluate the synthesized testing images; Higher neuron coverage rates indicate more diverse testing images [145]. For these reasons, in this work, we use neuron coverage rate to indicate how diverse and comprehensive a testing suite is.

More recent works on invariance testing found that averaged accuracy scores are not informative enough to judge the performance of ML models [58, 66]. Instead, visualized representations are becoming more popular for conducting more informative tests and analyses. For example, [74] used a 3d heat-map to analyse the translation/rotation invariance qualities and [11] also visualized the testing results and showed that ML models with the same averaged worst-case accuracy score can behave differently. Furthermore, [171] showed that analyzing the invariance qualities based on visualized patterns can achieve a higher inter-rater reliability (IRR) score than many NLP tasks, confirming the plausibility of conducting invariance testing based on visual patterns. In this work, we show that our visualization-based background invariance testing framework can lead to a satisfactory IRR score with the assistance of our novel technical components, e.g., ontology built on association analysis.

4.3 Definition, Overview, and Motivation

Let $\mathbf{x}_i$ be the i^{th} image in a dataset D and o_i be the foreground object in $\mathbf{x}_i$. $\mathbf{M}$ be an ML model trained to recognise or classify o_i from $\mathbf{x}_i$. In general, the invariance quality of $\mathbf{M}$ characterizes the ability of $\mathbf{M}$ to perform consistently when a type of

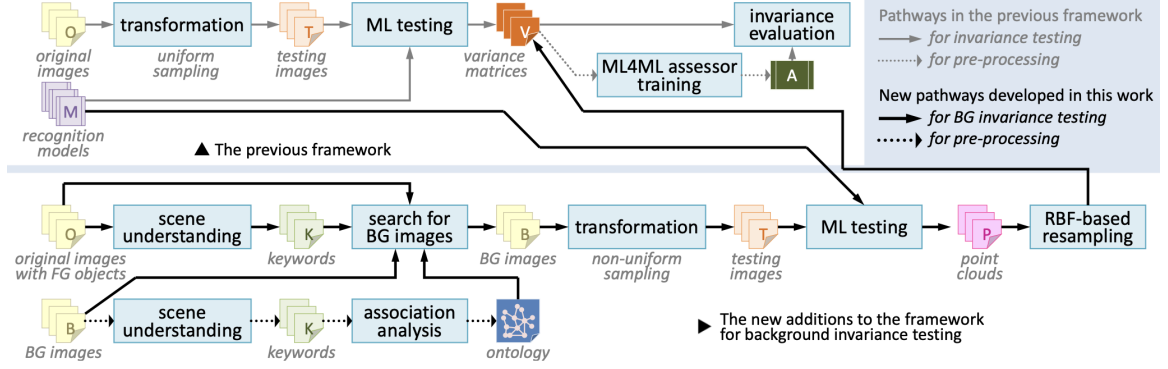


Figure 4.3: The upper part of the figure shows the invariance testing framework for simple data attributes, e.g., rotation, where the transformations for invariance testing are uniformly sampled. As the transformations for background invariance testing cannot be easily sampled in a consistent way, we introduce a new sub-workflow (lower part) with an additional set of technical components to enable non-uniform sampling of such transformations. This sub-workflow is detailed in Section 4.4.4 Methodology.

transformation is applied to $\mathbf{x}_i$. For example, one may apply a sequence of rotation transformations $\mathbf{y}_{i,j} = R(\mathbf{x}, j^\circ)$, $j = 0, 1, \dots$, and test $\mathbf{M}$ with the set of testing images of $\mathbf{y}_{i,j}$. As reported by [11], when the testing results are organised into a variance matrix (Figure 1a), the visual patterns in the variance matrix can be analyzed using an ML4ML assessor to evaluate the invariance quality of $\mathbf{M}$.

The background invariance quality characterizes the ability of $\mathbf{M}$ in recognizing o_i when it is with different backgrounds. Hence the transformations of $\mathbf{x}_i$ involve the replacement of the original background in $\mathbf{x}_i$ with different background scenes $\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n$. The transformations:

$$\mathbf{y}_{i,j} = B(o_i, \mathbf{b}_j), j = 1, 2, \dots, n \quad (4.1)$$

generate n testing images. Similar to other invariance testing, the testing results can be summarized as a variance matrix. However, as these background scenes may be ordered according to their locations in a database, different orderings may yield different variance matrices (Figure 1b). The visual patterns in such a variance matrix are not as meaningful as those resulting from rotation transformations (Figure 1a).

If we can find a way to consistently produce variance matrices for background transformations, we can adopt the ML4ML invariance testing framework proposed by [11] for background invariance testing. This motivates us to address the following challenges:

1. We need to introduce a metric for measuring the semantic distance between each background scene and the corresponding target image $\mathbf{x}_i$.

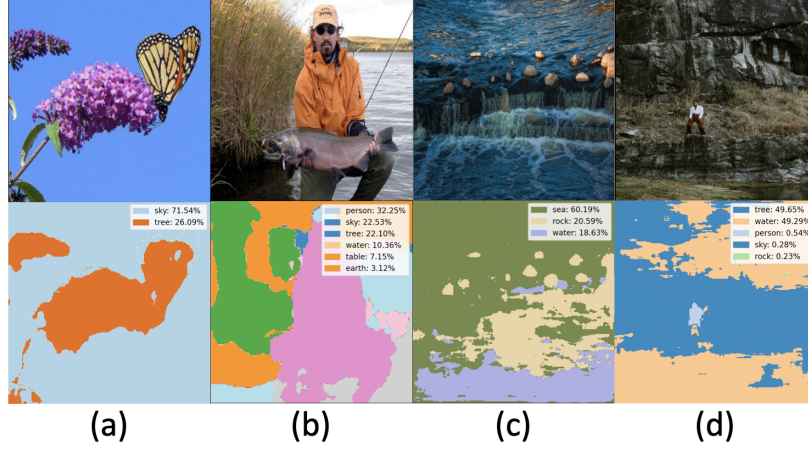


Figure 4.4: (a), (b) are target images. (c), (d) are background candidates. (a) and (c) have only a few detected keywords.

2. We need to produce a variance matrix as a uniform data representation from non-uniform sampling of background scenes, as sampling background scenes will not be as uniform as sampling sizes, rotation angles, and many other types of variables in invariance testing.
3. We need to have an effective way to search for background scenes that will be distributed appropriately for producing variance matrices.

As illustrated in Figure 4.3, the upper part of the figure shows the previous framework [11] for invariance testing involving uniform sampling of transformations. In this work, we introduce a number of new technical components (the lower part of the figure) to address the aforementioned challenges. Once these challenges are addressed, background invariance testing can be integrated into the ML4ML invariance testing framework.

4.4 Methodology

In this section, we follow the pathways in the lower part of Figure 4.3 to describe a series of technical solutions for enabling background invariance testing with non-uniform sampling of the transformations of the original images.

4.4.1 To Obtain Detected Keywords

Research on scene understanding aims to extract different semantic information from images. In this work, we represent the semantics of each image with the keywords

extracted by employing existing scene understanding techniques to process the original images to be tested and background scenes to be used for transformation. From each image, $\mathbf{a}$ ($\mathbf{a} = \mathbf{x}_i$ or $\mathbf{b}_j$), a scene understanding model identifies a set of objects that are recorded as a set K_a of keywords. When $\mathbf{a}$ is a target image, i.e., $\mathbf{a} = \mathbf{x}_i$, we assume that K_a contains a keyword for the foreground object o_i .

In this work, we use two models pre-trained on two different databases. In the ADE20k dataset [374], 150 object classes in 20,000 background scenes from the SUN and Places database are annotated. Some labelled objects can be a whole object or a part of another object, e.g., a door can be an indoor picture or a part of a car. Place365 dataset consists of 1,803,460 training images with 365 different scene classes [373]. And the number of images per class varies from 3,068 to 5,000. **For preparing models** to be tested: we use a smaller version of ImageNet, namely IN9 which consists of images from ImageNet but labelled with nine classes [340]. There are 45,405 images on the training set (5045 images per class) and 4185 images on the testing set (465 images per class). They also provide segmented foreground masks for some images on the testing set. Finally, **For background candidates**: we use BG20k which consists of 20,000 background scenes with no foreground images [160].

Figure 4.4 shows two examples of foreground images and two examples of background scenes. For some images, scene understanding may result in many keywords, but in other cases, only 1-3 keywords (e.g., the 1st and 3rd images in Figure 4.4). Therefore it is desirable to consider not only the keywords extracted from each image, but also the keywords related to the extracted keywords.

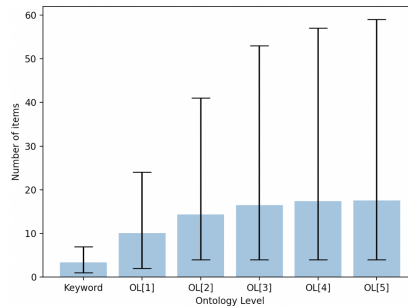


Figure 4.5: Number (maximum, mean, minimum) of enriched / expanded items after each ontology level.

4.4.2 To Obtain Expanded Keywords

In prior works, background candidates were sampled randomly, leading to inconsistent visual representations. In this work, we introduce a systematic sampling approach to

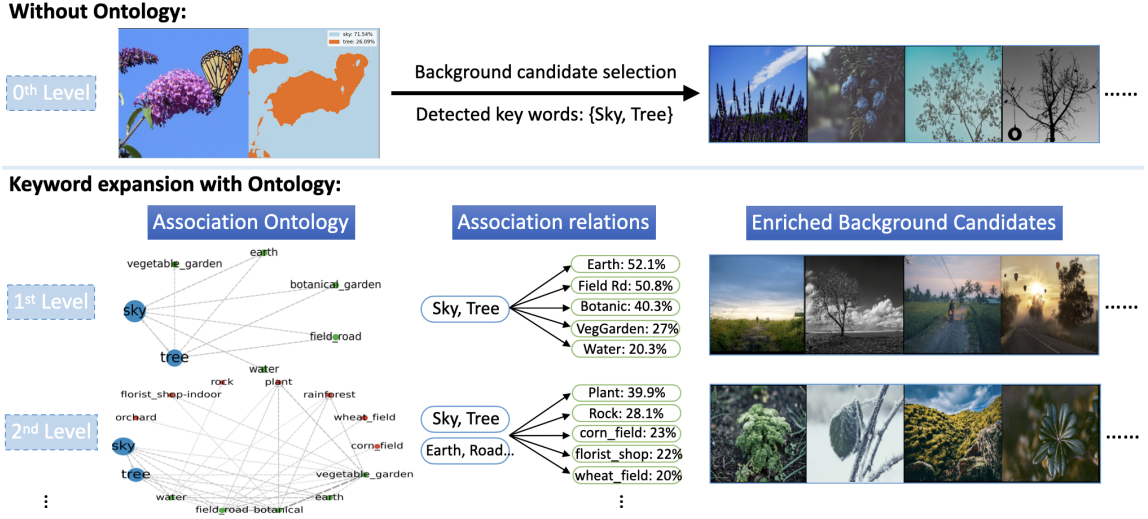


Figure 4.6: An example image has only two keywords detected by a pre-trained scene understanding model, namely $\{sky, tree\}$. Using an ontology, more keywords can be discovered iteratively, increasing the number and diversity of background scenes.

obtain background testing candidates. Firstly, we use a scene understanding model and association analysis to build an ontology. We then use the ontology to retrieve indirectly relevant keywords to the original images. Finally, with a set of detected and expanded keywords, we do keyword-based sampling to obtain background testing candidates.

From each testing image, a scene understanding model identifies a set of objects that are recorded as a set K_a of keywords. Multiple keywords can be identified for some images, but in many other cases, less than 3 keywords were detected. To select a suitable (non-random) testing suite that contains rich semantics, it is desirable to consider not only the keywords extracted from each image, but also other keywords that are related to the detected keywords.

Ontology is a graph-based knowledge representation, which is used to store the relationships among different keywords in this work. As illustrated in Figure 4.6, nodes represent keywords, and an edge between two nodes indicates that two keywords have been extracted from the same image. The weight on the edge indicates how strong is the association between the two keywords. The ontology is typically constructed in a pre-processing step by training association rules using the extracted keywords for all images in the repository $\mathbb{B}$.

To consider indirectly relevant keywords to the original testing image, we built an ontology using association analysis, i.e., Apriori algorithm [5] and Frequent Pattern Growth algorithm [103]. The Apriori algorithm [5] is widely used for association

analysis. When the size of the dataset is great, the Frequent Pattern Growth algorithm [103] can run more efficiently. Given a set of all possible keywords K_{all} that can be extracted from all images in a dataset $\mathbb{B}$, the level of association between two keywords k_a and k_b can be described by *support* and *confidence*. For three itemsets: $s_a = \{k_a\}$, $s_b = \{k_b\}$, and $s_{ab} = \{k_a, k_b\}$, the *support* for the itemset s_{ab} is defined as:

$$\text{support}(s_{ab}) = \frac{\text{number of images where } s_{ab} \text{ is present}}{\text{total number of images}}$$

which indicates the co-occurrence rate of k_a and k_b .

An association rule from one itemset to another, denoted as $\exists s_a \rightarrow \exists s_b$, is defined as *confidence*:

$$\text{confidence}(\exists s_a \rightarrow \exists s_b) = \frac{\text{support}(s_a \cup s_b)}{\text{support}(s_a)} \quad (4.2)$$

which indicates the confidence level about the inference that if the object of keyword k_a appears in a scene, the object of keyword k_b could also appear in such a scene. Similarly, we can compute $\text{confidence}(\exists s_b \rightarrow \exists s_a)$. For a large image repository, the value of $\text{support}(s_{1,2})$ is usually tiny, and is more easily changed by the increase of images in the repository, the introduction of more keywords, and the improvement of scene understanding techniques. Hence, it is difficult to use *support* consistently. We therefore use *confidence* as the weights (directed edges) in our ontology.

In the ontology, the shortest path between two keywords indicates the level of association between two keywords, typically facilitating two measures, (i) the number of edges along the path (i.e., hops) and (ii) an aggregated weight, e.g., $\prod_i w_{i=1}$ or $\min(0, w_1 - \sum_{i=2} (1 - w_i)^{\alpha_i} (\alpha_i \geq 1))$. As illustrated in Figure 4.6, nodes represent keywords, and an edge between two nodes indicates that two keywords have been detected from the same image at least once. The weight on the edge indicates how strong is the association between the two keywords. The ontology is typically constructed in a pre-processing step by training association rules using the extracted keywords for all images in a dataset.

A set of keywords K_x extracted by a scene understanding model can be used to search for background scenes with at least one of the matching keywords $k \in K_x$. When there are many keywords in K_x , semantic search can work very well. However, as exemplified in Figure 4.6(top), when an image has only two keywords, the search will likely yield a small number of background scenes, undermining the statistical significance of the test.

To address this issue, we expand the keyword set K_x by using the ontology that has acquired knowledge about keyword relationships in the preprocessing stage. As

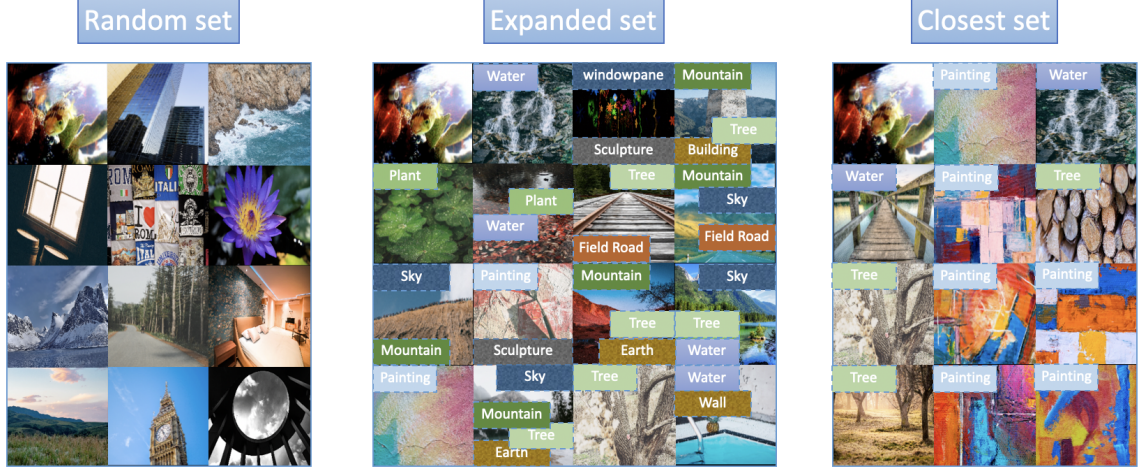


Figure 4.7: Three sets of example background scenes discovered for a target image of fish (the top-left of each set). The random set includes mostly unsuitable images. The closest set includes those discovered using only the original keywords $K_x = \{\text{painting, water, tree}\}$. The expanded set includes those discovered using the ontology, showing more suitable background scenes.

illustrated in Figure 4.6, the initial set K_x has keywords [sky, tree]. The ontology shows that {Sky, Tree} are connected to {Earth, Field Road, Botanic Garden, Vegetable Garden, Water}, which form the level 1 expansion set $E_{1,x}$. Similarly, from E_1 , the ontology helps us to find the level 2 expansion set $E_{2,x}$, and so on. The set of all keywords after i -th expansion is:

$$\text{OL}_x[i] = K_x \cup \left(\bigcup_{j=1}^i E_{j,x} \right) \quad (4.3)$$

Figure 4.7 shows three sets of example background scenes discovered for a targeting image (i.e., the fish image on the top-left corner of each set). The background scenes in the first set (left) are discovered by searching the image repository randomly. Those in the last set (right) are discovered using the initial set of keywords K_x . Those in the second set (middle) are discovered using an expanded set of keywords, OL_4 , after the 4th expansion. In Section 4.5, we quantitatively show that keyword-based sampling methods achieve the best balance between comprehensive testing suites and reliable/consistent testing results (human judgements).

4.4.3 To Synthesize/Generate Testing Images

For an original image, with a set of keywords (detected and expanded), one can synthesize testing images by 1) generative blending, or 2) simple background replacement eq.4.1. In this work, we use the latter to avoid unwanted foreground objects.

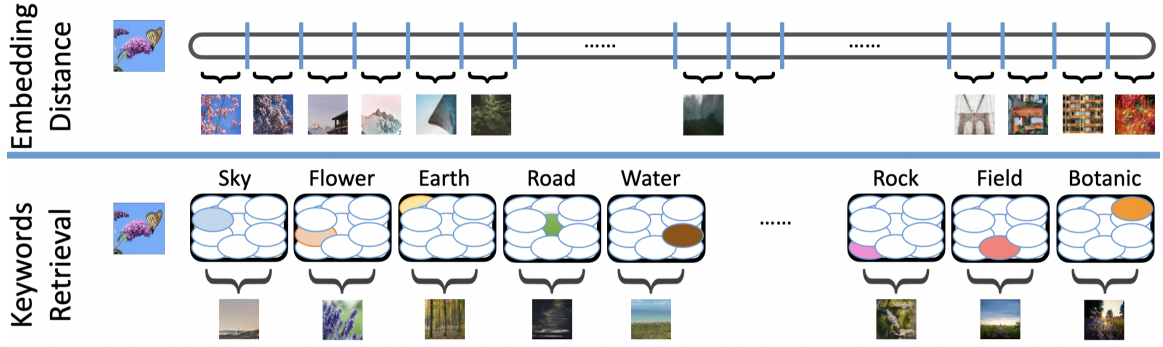


Figure 4.8: We can sample background scenes based on: 1) similarity/distance metrics, or 2) keywords. When sampling one instance from each subspace, we choose the first one leads to a higher realness score defined by [85] than a threshold.

We guarantee that no foreground objects will appear in any background scene by carefully selecting the dataset of the background candidates.

4.4.3.1 Background Scenes Sampling

Given a target image $\mathbf{x}$, to test if an ML model is background-invariant, we would like to sample a set of background scenes that can be used to replace the background in $\mathbf{x}$ while maintaining the foreground object o . As shown in Figure 4.8, to replace random sampling, we can sample background scenes based on: 1) cosine/l2 distance (metric learning), or 2) keywords. When sampling one instance from a subspace defined by 1) a distance interval (bin), or 2) background scenes containing a certain keyword, we run the Dreamsim [85] model to select the first background scene that leads to a testing image with a higher realness score than a threshold. In Section 4.5, we quantitatively show that keyword-based sampling is the most balanced between diversity and reliability.

4.4.3.2 Simple Background Replacement

The transformation process for sampling the background variations is more complex than other commonly-examined invariance qualities, requiring the use of a segmentation tool to separate the foreground object o_i from each target image $\mathbf{x}_i$, and then superimpose o_i into individual background scenes discovered in the previous step. As defined in Eq. 4.1, for n different background scenes $\mathbf{b}_1, \mathbf{b}_1, \dots, \mathbf{b}_n$, the transformation process produces n testing images $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \dots, \mathbf{y}_{i,n}$

To test ML models that were trained to recognise or classify foreground objects, any background scenes containing any of the foreground objects should not be used.

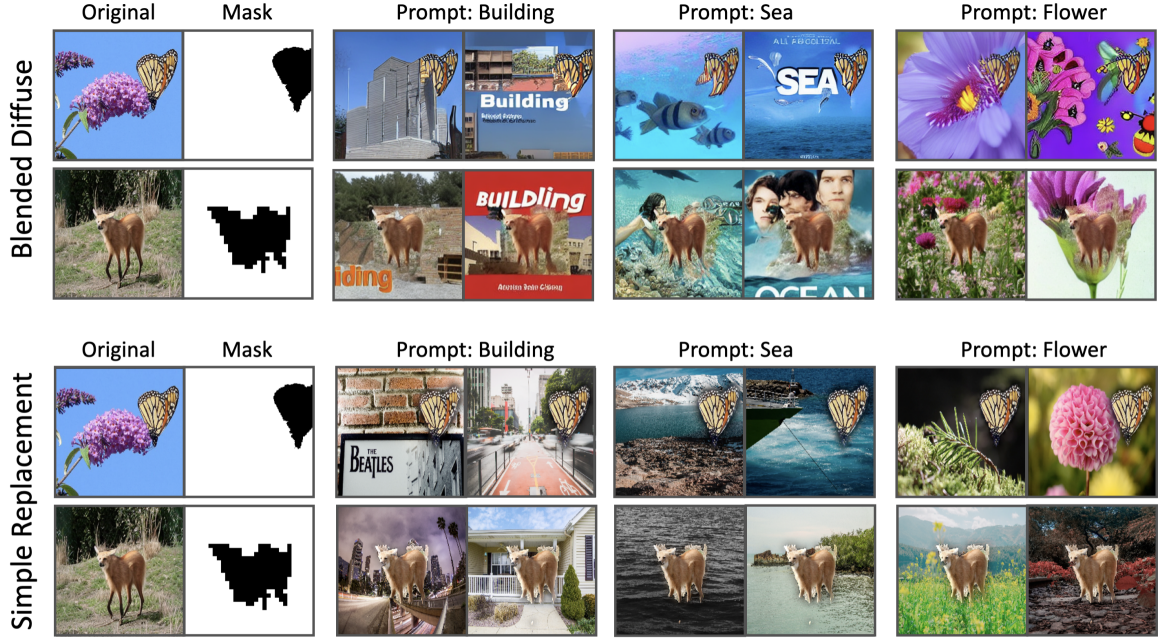


Figure 4.9: Background invariance testing images generated by the blended diffuse model [14] (first row), and synthesized by simple background replacement (a segmentation model [374], second row). The generated images can be more realistic, i.e., smooth blending. However, generative models can include unwanted biases, e.g., fish in the sea and insect on a flower, both of which are one of the nine foreground objects that the original models (to be tested) were trained to classify.

For this reason, we did not use generative algorithms to do the background replacement. We show some generated testing images using the latest generative model (blended latent diffusion [14]) in Figure 4.9. Some generated images include foreground objects due to unwanted biases, e.g., fish in the sea. This will affect the models’ behaviors in an uncontrolled way. Furthermore, running large generative models can be computationally expensive, taking hours to generate one testing image. For these two reasons, we use simple background replacement eq.(4.1) together with image blending using Laplacian pyramids [233] to remove some artifacts.

As shown in Figure 4.10, the distances are Gaussian distributed. As expected, compared with nearest neighbor, the distribution (of association ontology) suggests that our sampled scenes are more diverse. Compared with random sampling, the distribution shows that the sampled scenes prioritize scenes with a stronger association with the original images.

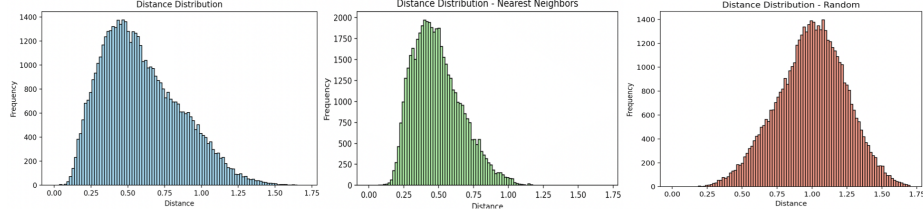


Figure 4.10: Distribution of distances between original images and synthesized testing images. Left (blue): Association Ontology, middle (green): nearest neighbors, right (red): random.

4.4.4 Point Clouds from ML Testing.

Unlike the previous framework where the transformation process samples variations uniformly (e.g., $0^\circ, 1^\circ, 2^\circ, \dots$ for rotation angles), the testing images $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \dots, \mathbf{y}_{i,n}$ are a set of non-uniform sampling points. When we test an ML model $\mathbf{M}$ against the testing images, we can measure the results and intermediate results of $\mathbf{M}$ in many different ways. In [11], the different measurements are controlled by the notions of *position* and *modality*. The position indicates where in $\mathbf{M}$ the signal may be captured, e.g., Max@CONF (the confidence vector of the final predictions) and Max@CONV-1 (the map of the last convolutional layer). The modality indicates what mathematical function is used to abstract the signal vector or map at a position to a numerical measure, e.g., Mean or Max. Hence, within the context of an ML model $\mathbf{M}$, a fixed position (ps) and a fixed modality (md) for each testing image $\mathbf{y}_{i,j}$, ML testing results in a numerical measure $S(\mathbf{M}, ps, md, \mathbf{y}_{i,j})$.

In addition, we can measure the *semantic distance* between each testing image $\mathbf{y}_{i,j}$ and the target image $\mathbf{x}_i$. As shown in Eq. 4.2, the confidence concept in association analysis is non-commutative. We therefore always use the semantic distance starting from $\mathbf{x}_i$. This assigns the value $m(\mathbf{y}_{i,j})$ with a position away from $\mathbf{x}_i$.

Consider two different testing images $\mathbf{y}_{i,j}$ and $\mathbf{y}_{i,k}$ and their corresponding semantic distances to $\mathbf{x}_i$ as $d_{i,j}$ and $d_{i,k}$. The difference between their numerical measures

$$v_{j,k} = \text{dif}(S(\mathbf{M}, ps, md, \mathbf{y}_{i,j}), S(\mathbf{M}, ps, md, \mathbf{y}_{i,k})) \quad (4.4)$$

indicates the variation between the two testing results. As the variation corresponds to positions $d_{i,j}$ and $d_{i,k}$, this gives us a 2D data point at coordinates $p_{j,k} = (d_{i,j}, d_{i,k})$ with data value $v_{j,k}$. When we consider all the testing results for all $\mathbf{y}_{i,1}, \mathbf{y}_{i,2}, \dots, \mathbf{y}_{i,n}$ as well as $\mathbf{x}_i$, there is point cloud with $n(n+1)$ data points in the context of $\mathbf{x}_i$.

When we combine the testing results for all l targeting images, we have a point cloud with $ln(n+1)$ data points, which can be visualized as scatter plots. The first

column in Figure 4.11 shows five examples of such point clouds. Because the number and the distribution of these points depend on the set of background scenes, we can observe that when the level of expansion $OL[i]$ (see Eq. 4.3) increases, the sampling has more data points and better distribution.

4.4.5 Variance Matrices from RBF-based Resampling.

Because the sampling of background transformation is not uniform (i.e., challenge (2) in Section 3), unlike the previous framework in Figure 4.3, we have to consider the options of training an ML4ML assessor with point clouds or converting point clouds to variance matrices. We select the latter option primarily because, in the previous framework, ML4ML assessors are trained with ML experts' annotation of invariance quality based on their observation of variance matrices. Replacing variance matrices with scatter plots in the annotation process would introduce an inconsistency in the framework in general and annotation in particular. In the short term, this would not be desirable, but in the longer term, one should not rule out the possibility of training ML4ML assessors with scatter plots.

We use the common approach of radial basis functions (RBFs) to transform a point cloud into a variance matrix. For each element e in a variance matrix, an RBF defines a circular area in 2D, facilitating the identification of all data points in the circle. Let these data points be $p_1, p_2, \dots, p_c$ and their corresponding values are $v_1, v_2, \dots, v_c$. As discussed earlier, the coordinates of each data point are determined by the semantic distances from the target image to two testing images. A Gaussian kernel ϕ is then applied to these data points, and produces an interpolated value for element e as

$$value(e) = \frac{\sum_{i=1}^c (\phi(\|e - p_i\|) \cdot v_i)}{\sum_{i=1}^c \phi(\|e - p_i\|)}$$

However, when the RBF has a large radius, the computation can be costly. When the radius is small, there can be cases of no point in a circle. In order to apply the same radius consistently, we define a new data point at each element e and use K nearest neighbors algorithm to obtain its value $u(e)$. The above interpolation function thus becomes:

$$value(e) = \frac{\phi(0) \cdot u(e) + \sum_{i=1}^c (\phi(\|e - p_i\|) \cdot v_i)}{\phi(0) + \sum_{i=1}^c \phi(\|e - p_i\|)}$$

In Figure 4.11, we show the application of three different RBFs. The mixed green and yellow patterns in row $OL[1]$ gradually become more coherent towards $OL[5]$. We can clearly see a green square at the centre and yellow areas towards the top and right edges.

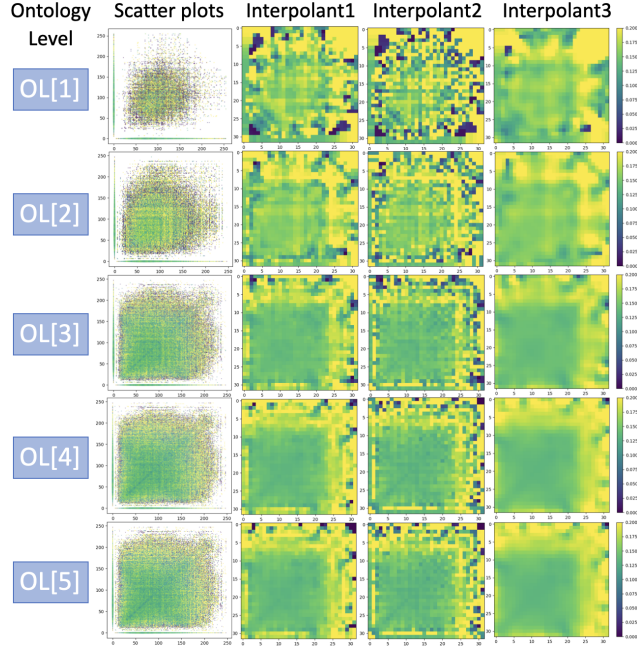


Figure 4.11: Original scatter plots (first column) and interpolated variance matrices (second to fourth column). The scatter plots are generated using the ontology level one to five, i.e., $OL[1], OL[2], \dots, OL[5]$ for a randomly selected model M_{61} . Interpolant 1 - 3: (kernel size 16 $\sigma=5$), (kernel size 32 $\sigma=2$), and (kernel size 32 $\sigma=10$).

4.4.6 Professional Annotations

We survey the three professionals who provide the annotations for background invariance testing. We ask them the following questions:

- Q1: Which locations are you interested in?
- Q2: Are the interpolated scatter plots helpful? Which interpolant is the most helpful?
- Q3: Are there any common visual patterns?
- Q4: Are the annotations aligned with the worst-case accuracy?
- Q5: Would you say your decisions are consistent? Are you confident about your decisions?
- Q6: Any interesting findings about the models?

Q1: Testing locations All the three coders agree that the final predictions and outputs from the last layer before the final fully connected are important to be tested.

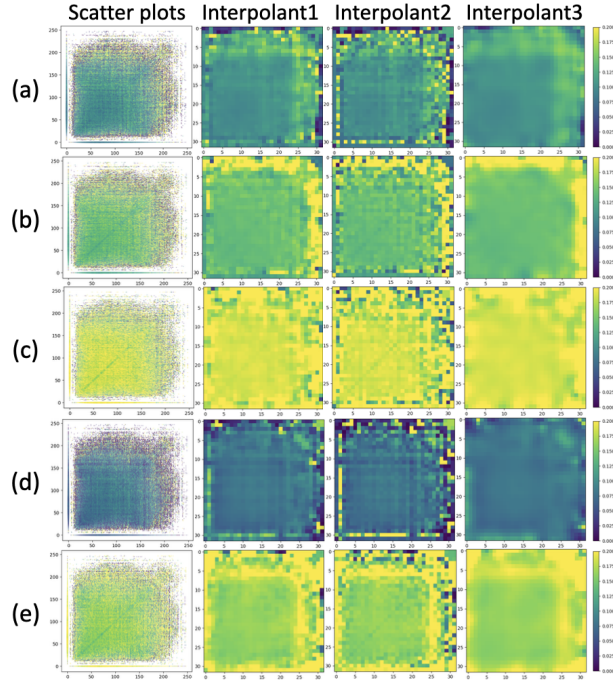


Figure 4.12: Examples of original scatter plots and interpolated / accumulated scatter plots. From (a) to (e) are model M_1 , M_7 , M_{16} , M_{68} , M_{71} respectively.

Q2: Helpful interpolated Coder 1 and 2 mention all of the interpolated plots are helpful and there is no particular interpolator that is superior to the others. Coder 3 mentions they check all of the interpolated plots and find the one with kernel size 32 and $\sigma = 10$ helps their judgements the most because it shows the most straightforward global shape of the plots.

Q3: Common visual patterns Coder 1 and 3 mention that the most common patterns are those with a green area (which indicates a lower error rate) at the bottom left, and some yellowish patterns (which indicates a higher error rate) at either the edges or corners at the top right. Coder 2 notices it is common that there is a green line along the primary diagonals (from bottom left to top right) of the plots. We show more different patterns in Figure 4.12.

Q4: Annotations versus worst-case accuracy All three coders mention their annotations are not completely aligned with worst-case accuracy. However, they all find their annotations and the traditional metric are related to each other to some extent.

Q5: Consistent annotations All three coders are confident about their decisions for the majority of the models. However, they are less confident for those labelled as “borderline”.

Q6: Interesting findings Coder 1: vision transformers seem to be naturally more robust against background permutations. Even when the final predictions do not appear to be robust at all, their last layer before the head module can still be robust (dark green). Coder 2: models trained using images without background (black pixels as their background) appear to be more robust than those trained with natural images. Coder 3: metric learning, e.g., triplet loss, does not seem to be helpful even for the convolutional layers that are used to form the triplet loss function.

4.4.7 ML4ML Assessor Training and Deployment.

With the variance matrices, we can use the same processes to train ML4ML assessors and deploy them to evaluate the background invariance quality of ML models in the same way as the previous framework. The same technical approaches in [11] can be adopted, including: (i) collecting variance matrices and ML models, (ii) splitting the model repository into a training and testing set and provide expert annotations of invariance quality based on variance matrices, (iii) engineering of imagery features for variance matrices, (iv) training ML4ML assessors using different ML techniques, and (v) testing and comparing ML4ML assessors.

4.5 Experiments

To prepare ML models to be tested, we train a small repository of 250 models on the IN9 database (a smaller ImageNet with only nine classes) [340]. To guarantee no foreground objects would appear in our testing images, we use the BG-20k [160] database with 20,000 background scenes as all the background candidates. We select $N = 32$ background scenes for each target image, and retrieve background scenes using the keyword-based sampling or distance-based sampling methods described in Section 4.2.14.4.3.1. We show that similar results can be found when $N = 50$ or $N = 100$ in the ablation study at the end of this section.

For each model, we measure the signals at two positions, including the final predictions (confidence score) and the embedding layer (after the final pooling layer, or the last layer before the final MLP for Vision Transformers), as these two positions are considered the most important and interesting by our annotators. For the

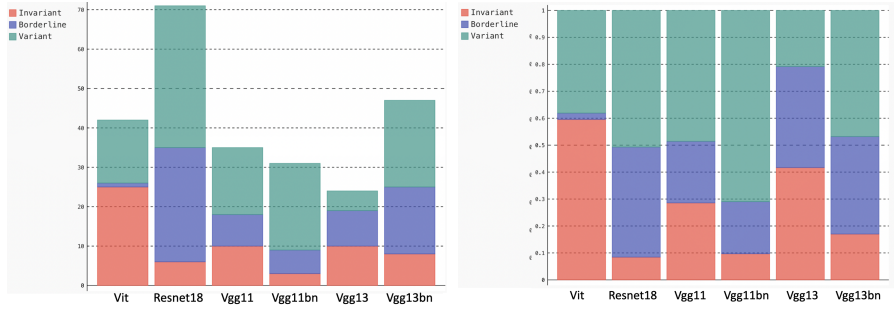


Figure 4.13: Statistics of model architectures in our model repository. Left: number of models with each architecture in our model repository. Right: percentage of invariant / borderline / variant models for each architecture in our model repository.

embedding layer, we choose 1) the average value of the top 3 neurons triggered by foreground-only images, 2) the maximum value, and 3) the average value. This leads to one scatter plot for the confidence score and three scatter plots from the embedding layer. For each model, we provide professional annotations based on the visualized testing results with three quality levels, namely, 1) not invariant, 2) borderline, and 3) invariant.

4.5.1 Model Repository

In this section, we show some statistics of our model repository (250 models). We trained the models under different settings, and provide professional annotations for each of them as being background-invariant, borderline or background-variant.

Table 4.1: Statistics of the annotations of the model repository

Invariant	Borderline	Variant	Total
62 (24.8%)	70 (28.0%)	118 (47.2%)	250 (100%)

As shown in Table 4.1, the model repository is not balanced. There are more background variant models than background invariant models, which is expected as most of the models were trained using no special technique to boost their background invariance qualities; Unlike other types of transformations, e.g., rotation, background augmentation is not commonly used.

Table 4.2: Statistics of the accuracy of the model repository

	mean	max	min	std
Accuracy	91.88%	99.42%	43.40%	9.11%
Worst-case	28.76%	65.36%	0.18%	11.92%

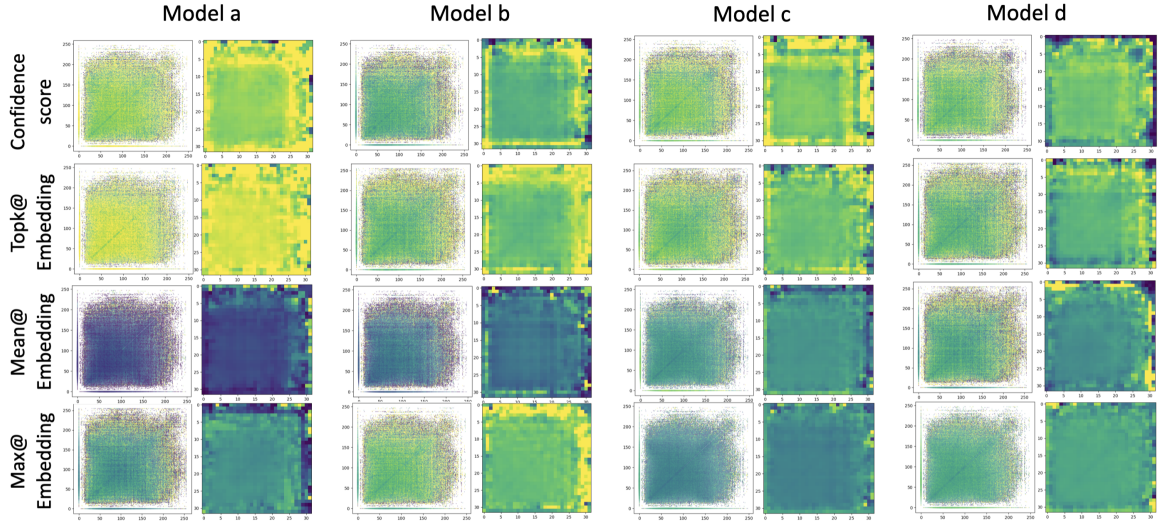


Figure 4.14: We show four trained ML models on IN9 dataset have different visualized testing results even though they have the same accuracy score and worst-case accuracy score (averaged). We analyse these patterns in Section 4.5.2.

Table 4.3: Consistency, Reliability and Comprehensiveness Level of Different Background Invariance Testing Approaches

	Random Sampling	Distance-based Sampling		Keyword-based Sampling	
		Nearest Top K	Interval (bin)	CLIP	Ontology (Ours)
Neuron Coverage (recall) [239]	0.681	0.133	0.667	0.591	0.652
Fleiss' reliability (precision)	0.384	0.906	0.531	0.640	0.649
F1 score	0.491	0.232	0.591	0.615	0.650

In Table 4.2 we show that the average accuracy of the models in the collection is around 91.9%. We did not intentionally make any model not sufficiently trained. However, for some models that have not been pre-trained on ImageNet, the accuracy can drop to lower than 50%.

In Figure 4.13, we show that in our model repository, we have a wide range of different architectures, namely vision transformer patch16-224, resnet 18, and vgg variants. We also show the statistics for their background variance qualities. However, the size of our model collection is limited and we leave the studies of the architectures' impact on invariance qualities to future work.

4.5.2 Worst-Case Accuracy versus Variance Matrices

We show the visualized performance (variance matrices) of four models can be very different even though they have the same (worst-case) accuracy score in Figure 4.14. The patterns of $\mathbf{M}_a$ suggest that $\mathbf{M}_a$ gives wrongly predicted testing images high

confidence scores, and the outputs of the top-3 neurons triggered by foreground-only images are not consistent, which indicates that $\mathbf{M}_a$ might rely more on the background to make its decisions instead of the foreground objects. The abrupt green pattern at the top-left for $\mathbf{M}_c$ might be a signal of data leakage which needs further examination before being deployed. The yellow pattern on the edges of $\mathbf{M}_b$ suggests that $\mathbf{M}_b$ treats a few of testing images differently from the others. This could suggest that $\mathbf{M}_b$ might be sensitive to the appearance of certain objects. For these reasons, our three annotators labelled $\mathbf{M}_a$ and $\mathbf{M}_c$ as *failed*, $\mathbf{M}_b$ as *borderline*, and $\mathbf{M}_d$ as *passed*. These four models have the same statistical evaluation scores (worst-case accuracy) whilst their visualized testing results can yield different judgements on their invariance qualities. To consolidate if we can use such visualization-based testing framework for background invariance testing, we evaluate 1) how diverse the testing suites are, and 2) how consistent the human annotations are.

4.5.3 Comparisons Between Testing Methods

Our human annotations on background invariance qualities are based on the visualized testing results of selected testing suites, and we evaluate whether our tests are desired based on: 1) neuron coverage rate (*recall*), and 2) inter-rater reliability (IRR) score (*precision*) as discussed in Section 4.3. Neuron coverage rates are often used to evaluate whether a selected testing suite is diverse enough to cover as many scenarios as possible [123, 239], and IRR scores are often used to evaluate the consistency and reliability of professional annotations [71] between different practitioners. We report f1 score (between diversity - recall, and reliability - precision) when comparing difference testing methods.

4.5.3.1 Prior Works: Random/Nearest Sampling

In Table 4.3, we show that although the random sampling testing approach resulted in the best neuron coverage rate, the IRR score is relatively lower because of the randomness. Meanwhile, selecting the nearest top-k background scenes resulted in the highest inter-rate reliability score, however, the neuron coverage rate (diversity level) is the lowest. Therefore, these two testing methods might not be the most suitable for background invariance testing.

4.5.3.2 Distance and Keyword-based Sampling

As shown in Table 4.3, although distance-based sampling (interval) can achieve a higher neuron coverage rate, the resultant human judgements are not consistent or reliable, whereas keyword-based sampling methods are the most balanced between diversity and reliability. Among keyword-based sampling methods, CLIP [272] tends to find similar (matched) items for the target image, which leads to less diverse testing suites. Meanwhile, our ontology expands the originally detected keywords using association analysis and thus becomes the most balanced between diversity and consistency (i.e., the best f1 score).

4.5.4 Training ML4ML Assessors

We build a small model repository of 250 models for object classification. The models were trained under different settings:

- Architectures: VGG13bn, VGG13, VGG11bn, VGG11 [287], ResNet18 [117], and Vision Transformer [67]
- Hyper-parameters: learning rate, batch size, epochs
- Augmentation: rotation, brightness, scaling, using images with only foreground (black pixels as background)
- Optimizers: SGD, Adam, RMSprop
- Loss: cross-entropy loss, triplet loss, adversarial loss

We use our sub-workflow in Figure 4.3 to generate testing images using the target images in the IN9 dataset, and the background scenes in the BG-20K database. We apply these models to the generated testing images. With the two measuring positions per model, we generate two variance matrices using the results from the ML Testing process.

For each model, we then provide professional annotations based on the original scatter plots and interpolated variance matrices with three quality levels, namely, 1) not invariant, 2) borderline, and 3) invariant.

Finally, to automate the testing process, we extract the same set of statistical features as [11], e.g., mean / standard deviation, from the interpolated variance matrices. After that, we use the statistical features and the professional annotations of

Table 4.4: Experiment results: the automation accuracy using random forest as the assessor is around 80% and the inter-rater reliability score with majority votes is around 0.65.

	Automation Accuracy	IRR Score
Random Forest	79.7 + 7.5%	0.649+0.091
AdaBoost	74.8 + 9.1%	0.599+0.102
Worst-case Acc	64.4%	0.387

Table 4.5: Experiment results: the automation accuracy using random forest as the assessor is around 80%. Meanwhile, no significant differences are found using different settings (i.e., kernel size, sigma value) for RBF interpolation.

Interpolator Assessor	Automation Accuracy		
	Size: 32, $\sigma=2$	Size: 16, $\sigma=5$	Size: 32, $\sigma=10$
Random Forest	79.6 $\pm$ 8.1%	78.7 $\pm$ 7.7%	79.7$\pm$7.5%
Adaboost	54.5 $\pm$ 7.5%	70.9 $\pm$ 10.2%	74.8 $\pm$ 9.1%
Worst-case accuracy	64.4%		

the training set of the model repository to train an ML4ML assessor, e.g., a random forest / adaboost in our case.

To evaluate the feasibility of the automation process using ML4ML assessors, we split the model repository into a training set (2/3 of the models) and a testing set (1/3 of the models). We do not tune any hyper-parameters for the ML4ML assessors, therefore we do not further split the training set into training and validation set. To make the results more statistically significant, we randomly split the data, repeat the experiments ten times and report the averaged results and the standard deviation.

4.5.5 Ablation Studies

To generate the scatter plots for a given model, we use the testing set of IN9 dataset [340]. However, among all the 4185 images in the testing set, only 1712 images are provided with a foreground mask. Therefore, we use the 1712 images to generate the scatter plots. In Figure 4.15, we show that using different percentages of the 1712 images will not significantly affect the qualities of the generated scatter plots (neither the original nor interpolated). However, when the number of images becomes too low, e.g., $\leq 25\%$ (less than 430 images), the generated scatter plots are starting to be heavily affected.

For RBF-based interpolation, we use the following parameters: ($r=32$, $\sigma=10$, $K=32$). For each model, we therefore obtain two variance matrices. We conduct the ablation study for some parameters of the interpolation, i.e., the kernel size and sigma

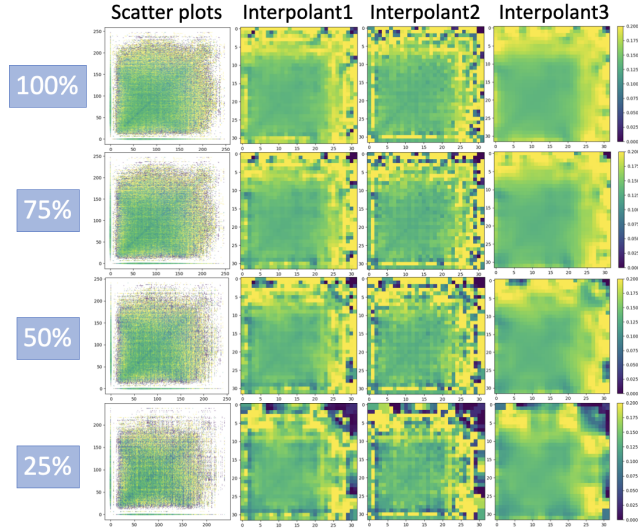


Figure 4.15: We use different numbers (percentages) of the testing images to generate scatter plots. The number of images will not significantly affect the quality of the generated plots unless the number becomes too small, e.g., $\leq 25\%$. The example we show here is model M_{61}

Table 4.6: Experiment results: the inter-rater reliability score between predictions from our ML4ML assessors and majority votes is around 0.65. Meanwhile, no significant differences are found using different settings (i.e., kernel size, sigma value) for RBF interpolation.

Interpolator Assessor	Inter-rater Reliability with Majority Votes		
	Size: 32, $\sigma=2$	Size: 16, $\sigma=5$	Size: 32, $\sigma=10$
Random Forest	0.651 ± 0.103	0.635 ± 0.116	0.649 ± 0.091
Adaboost	0.358 ± 0.134	0.548 ± 0.159	0.599 ± 0.102
Worst-case accuracy	0.387		

values. In Figure 4.15, the three interpolants are (kernel size: 16, sigma: 5, K : 32), (kernel size: 32, sigma: 2, K : 32) and (kernel size: 32, sigma: 10, K : 32) respectively. The K values only affect a few points at the corners and we barely see any difference. As shown in Figure 4.16, kernel size and sigma value affect how blurry the interpolated scatter plots are. Based on the quality of the interpolated scatter plots, we tested these three interpolants and reported the automation results in table 4.5 and table 4.6 where we show that using different parameters for the interpolation did not heavily affect the automation accuracy (the performance of ML4ML assessors).

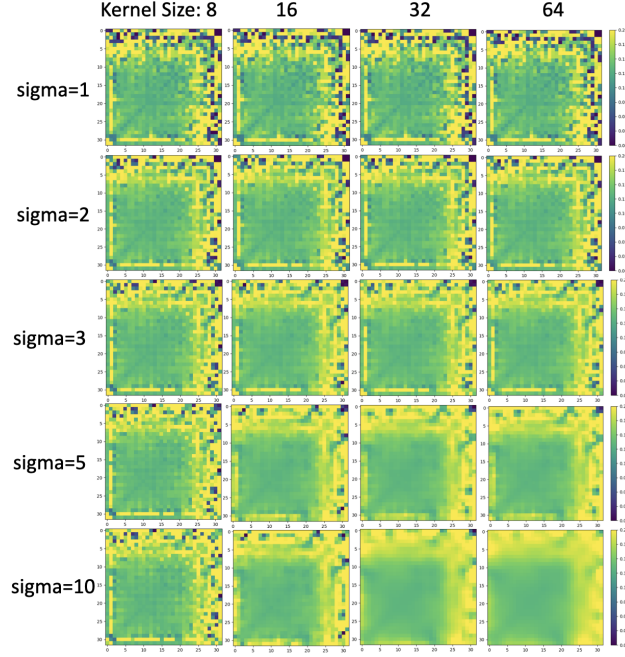


Figure 4.16: Different interpolant parameters for M_{61} . We use (Kernel size 16, sigma 5), (Kernel size 32, sigma 2), and (Kernel size 32, sigma 10) respectively in Figure 4.11.

4.5.6 Results and Analysis

In Table 4.4 we show that the majority votes of the professional annotations have around 0.4 IRR score with worst-case accuracy, which shows that the annotations are still aligned with the traditional accuracy metric. However, the professional labels are provided by considering the variance matrices and they are the decisions from many different factors instead of relying only on one single metric. Furthermore, Table 4.7 shows that the inter-rater reliability scores among the three professionals are around 0.65 which indicates that the annotations are consistent compared with many NLP tasks reported by [71]. Therefore, we believe that such annotations are both desirable and reliable.

In Table 4.4, we also show that by using ML4ML assessors [11], we could achieve around 70-80% automation accuracy which shows that the automation method (using ML4ML assessors) can achieve a satisfactory accuracy ($\sim 80\%$). Furthermore, the

Table 4.7: IRR: Cohen’s and Fleiss’ kappa scores

Cohen’s	Coder 1	Coder 2	Coder 3	Fleiss’
Coder 1	1	0.651	0.643	0.628
Coder 2	0.651	1	0.591	
Coder 3	0.643	0.591	1	

IRR scores between the predictions from ML4ML assessors and the majority votes are similar to those of the three coders (~ 0.6). Therefore it shows the proposed framework can work as a fully automated background testing mechanism with sufficient accuracy.

Shapley values (a game theoretic approach) [186] are commonly used to explain outputs or predictions of machine learning models. It isolates each input feature and averages its expected marginal contribution. In this section, we report the usage of Shapley values to explain the predictions from our ML4ML assessors.

As shown in Figure 4.17, for an arbitrarily selected model M_5 , the ML4ML assessor’s prediction is “background invariant”. And the top factor contributing to the predictions is the mean value of the variance matrix generated at pos_0 (final prediction). We also show that, in general, the ML4ML assessor will consider the mean value of the variance matrix and the error rate (defined as the percentage of the pixels $\geq thres$ [11]). This way, we can have a testing report for the ML4ML assessor for both the overview analysis and case studies for any interested model(s).

4.6 Conclusion

In this work, we propose a technical solution to address a major limitation of an invariance testing framework recently reported by [11]. The limitation is that background invariance testing cannot be incorporated into the framework as many other invariance problems. Our technical solution brings several non-trivial techniques together to overcome the challenges. The introduction of ontology is both novel and vital in making background invariance tests as meaningful as other commonly-seen invariance tests, e.g., rotation invariance. In this way, the previous framework has been expanded and improved significantly, paving the critical path for introducing other invariance tests with transformations that are not uniformly sampled, e.g., variations of clothing or hairstyles.

While selecting and sampling background scene candidates is the most important part for background invariance testing, it is also crucial to order the testing data according to semantics rather than visual similarity. In the next chapter, we introduce the semantic image similarity method that we used for background testing.

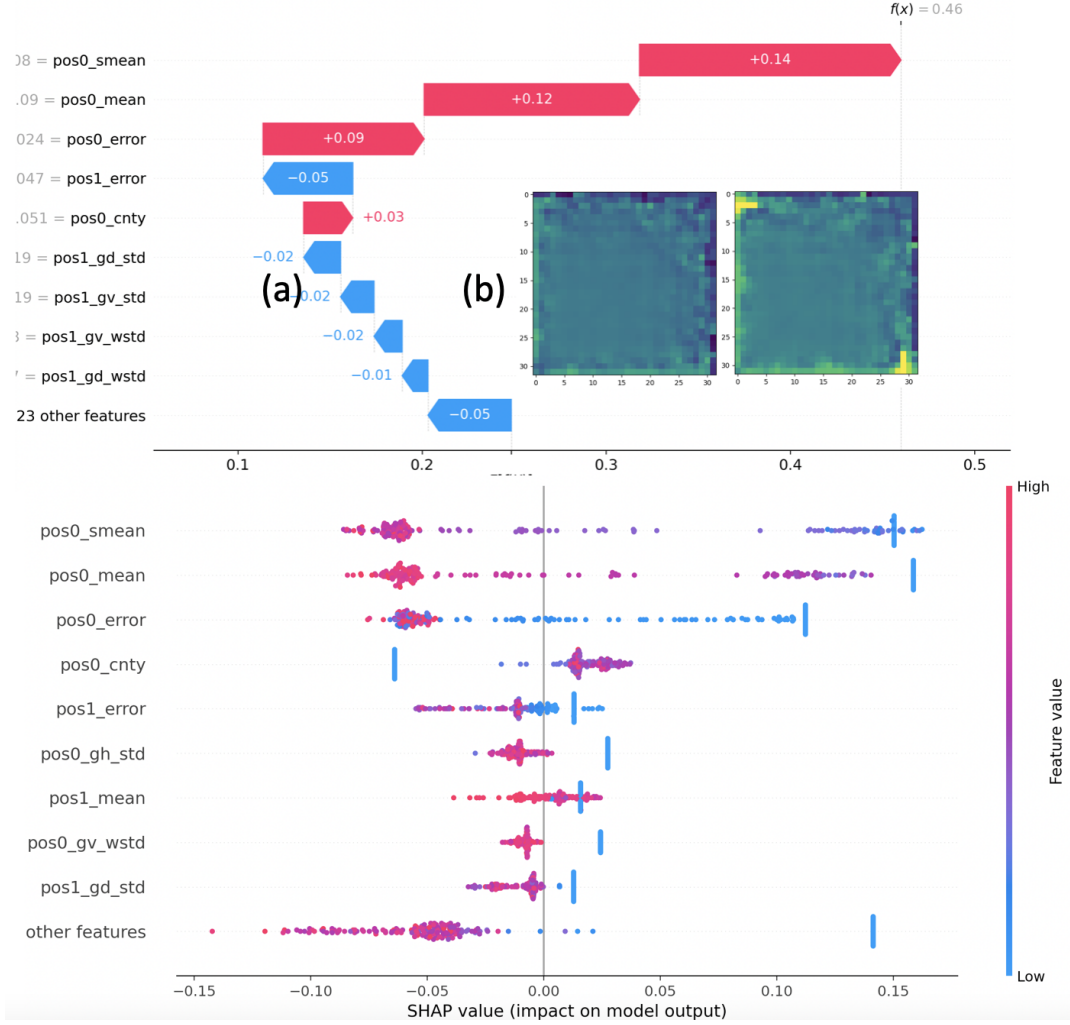


Figure 4.17: Shapley values of a randomly selected ML4ML assessor (a random forest). Left: given a model M_5 , and its variance matrices generated at (a) the final prediction (pos_0) and (b) the final convolutional layer (pos_1), the ML4ML assessor predicts the model to be “background invariant”. And the top reasons for this prediction is: the mean value of the variance matrix (pos_0) and the error rate (defined in [11]) of the variance matrix (pos_0). Note that there are yellowish areas on the corners of the variance matrices at pos_1 , therefore the continuity score and the gradient scores at these two positions have a negative impact on the decision. Right: In general, the ML4ML assessor’s decisions for being “background-invariant” depend on the mean value of the variance matrix (pos_0) and the error rate of the variance matrix (pos_0), which is aligned with the analysis of M_5 on the left.

Chapter 5

Ordering Invariance Testing Data: Image Similarity using An Ensemble of Context-Sensitive Models

5.1 Introduction

Similarity between images, which has been studied for decades, is crucial for various computer vision tasks, e.g., content-based retrieval [187] and image recognition [277]. In recent years, deep embeddings have become available in metrics or models for estimating image similarity, especially those machine-learned (ML) models, e.g., through metric learning [141] or contrastive self-supervised learning [131]. However, as shown in Figure 5.1, deep embeddings are not always aligned with human annotations in terms of judging semantic similarity. Moreover, while similarity scores are typically of numerical values (e.g., 0.45), there is no easy way to obtain such values as ground truth data for training or testing, making it difficult to improve or compare the performance on the task of semantic similarity between images for those deep models.

Humans' perception of image similarity is often context-sensitive (CS). In some labelling processes, binary scores were assigned to image pairs in relation to a reference image (i.e., a context), i.e., is A more similar to R than B . Such labelling processes have been shown to be more consistent and objective, and have been used in image retrieval [349], face recognition [271], and evaluation of synthesized images [367]. In these fields, the existing databases are typically used to train models that can identify closely related images, i.e., either images (R, A) or (R, B) are very similar. For the general problem of image similarity, A and B can both be unrelated to R , as exemplified in Figure 5.1. Ideally, one might wish to have a vast number of image

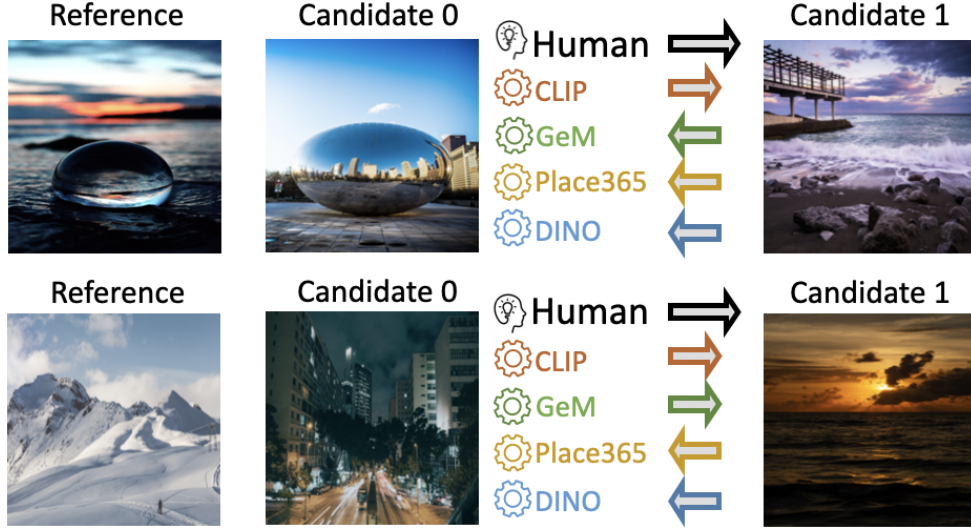


Figure 5.1: Arrows point at the candidate which is considered closer to the reference by the model(s) or human annotation. Visual similarity scores computed by deep models are not always aligned with human annotations.

triples (R, A, B) randomly selected from an image domain $\mathbb{D}$. However, it would be costly to label these triples. Therefore, due to the gigantic data space and the limited amount of labelled data, directly fine-tuning deep models might not be effective (see Section 5.5.2).

In this work, we considered an alternative approach, with which we selected only a small set of K reference images and the labelling effort ensured adequate sampling of (A, B) in the context of each selected reference image R_i , $i = 1..K$, as illustrated on the left of Figure 5.2. Each R_i group of labelled triples is referred to as a context-sensitive (CS) data cluster. We then obtained K context-sensitive (CS) models each of which was fine-tuned on one CS data cluster (w.r.t. a reference image R_i). Our experiments show that these CS models are able to improve the performance only when unseen triples contain reference images that are similar to R_i , e.g., when they are both flowers. We refer to such improvement as **local improvement**, whilst the improvement on the entire dataset as to **global improvement**. We show how the performance of these CS models gradually improves locally but not globally during fine-tuning in Section 5.5.4, and we also showcase the local improvement of each CS model in Section 5.5.5.4 using some randomly selected triples. To fully utilize the advantage of each CS model and improve the performance on the entire dataset, we introduce two different ways to build ensemble models. To consolidate our proposed method, we compared our ensemble models with 1) existing deep embeddings, e.g., CLIP [272] and DINO [34], 2) individual CS models, 3) models directly fine-tuned on

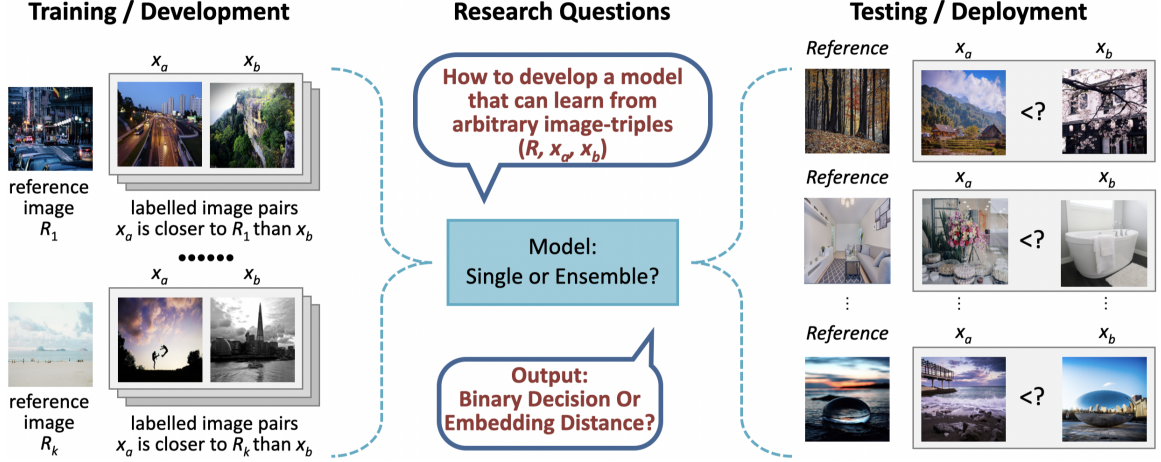


Figure 5.2: Given a training set of random triples that are annotated which candidate is semantically closer to the reference, can a model learn from the training data and predict correctly for unseen triples (i.e., unseen reference images and unseen candidates)?

the entire dataset where all labelled triples (R, A, B) are amalgamated, and 4) the elementary ensemble models, e.g., majority voting. Our testing demonstrates that it is feasible to use CS data to develop models with little or very low context sensitivity, providing an efficient and effective approach for sampling image triples in the vast data space $\mathbb{D}^3$.

Contributions. In summary, (1) we revisit the problem of semantic similarity between images, and introduce a dataset with 30k labelled triples, facilitating the improvement and comparisons on the task of image semantic similarity, (2) we evaluate and compare the performance of existing deep embeddings, e.g., ViT or CLIP, and image retrieval models/algorithms, e.g., CVNet [159], on the collected dataset, (3) we found fine-tuning directly on the collected dataset not effective due to the huge data space and limited amount of labelled data, (4) we found by fixing the reference image R , our CS models are able to improve the performance locally (when unseen reference images are similar to R , e.g., when they are both mountains), but not globally, (5) we provide two novel methods for constructing ensemble models using our CS models to improve the performance globally, and (6) we conduct extensive experiments to compare the proposed approach with existing methods and some more conventional solutions, and we show that the proposed method is efficient and effective when data sampling is sparse and labelling resource is limited.

Table 5.1: Comparison with existing datasets of triples

Dataset	Input Type	Size	Annotators	Data Source	Candidate Restriction	Random Candidates
Yoon et al. [349]	Images	1,752	5.7	Visual Genome [150], MS-COCO [177]	Similar to the reference	✗
BAPPS (real-algo) [367]	64x64 Patches	26.9k	2.6	MIT-Adobe5k [26], RAISE1k [56]	Distorted from the reference	✗
NIGHTS [85]	Images	20k	7.1	Diffusion [254]-synthesized	Synthesized from the reference	✗
CoSIS (Ours)	Images	30k	3	BG20k [160]	No Restriction	✓

5.2 Related Work

In the literature, prevalent feature extractors, such as histograms of gradient/color or local binary patterns mainly focus on visual attributes of images with semantic information often being overlooked. For this reason, Wang et al. [317] introduced SDML that utilized geometric mean with normalized divergences to balance inter-class divergence. [82] used a combination of different distance measures, e.g., wordNet [203], Google similarity [50] and tested their method on 520 random pairs collected from Flickr. Similarly, Deselaers et al. [63] studied the relationship between visual and semantic similarity on ImageNet and they introduced a new distance metric which was shown effective on image classification tasks. Zhang et al. [357] introduced a differential earth mover’s distance (DeepEMD) and their method was proven effective on various image classification tasks under a k-shot setting. However, unlike context-based similarity, traditional scores are not consistent among different metrics and do not always have physical explanations. Additionally, to our best knowledge, existing datasets containing triples where two candidates can be both different from the reference are all relatively small, e.g., 520 labelled pairs [82], or 1.7k labelled triples [349]. Therefore, it is necessary to revisit the context-based similarity problem and provide a relatively larger dataset.

The BAPPS dataset [367] consists of 26.9k triples of reference and two distorted candidates (64x64 patches). They provide the two alternative forced choice (2AFC) similarity annotations for the triples. Similarly, DreamSim [85] provided 20k triples of reference and two synthesized candidates (images). D’Innocente et al. [64] provided 10,805 triples of women’s dress images. Yoon et al. [349] ordered 1,752 triples of random images and defined a metric to evaluate the performance of image retrieval models. However, all the existing triples are carefully selected or synthesized. Therefore, at least one of the two candidates is fundamentally similar to or almost the same as the reference image. In this work, we extend the study of image similarity to arbitrarily sampled candidates.

For image similarity, the data space is gigantic. Therefore, Wray et al. [332] used proxies to largely reduce the labour cost of annotation. Similarly, Movshovitz-Attias et al. [208] used static and dynamic proxies to improve models’ performance on image

retrieval and clustering tasks. Given an anchor image and a smaller subset of data points (candidates), they defined the proxy as the one with minimum distance to the anchor image. This way, they showed that the loss over proxies is a tight upper bound of the original one. Aziere et al. [15] trained an ensemble of CNNs using hard proxies to compute manifold similarity between images and their method was proven effective for image retrieval tasks. Similarly, Sanakoyeu et al. [265] introduced a divide and conquer training strategy which divided the embedding space into multiple sub-spaces and then assigned each training data object a learner according to which sub-space the training data object was located.

5.3 Dataset

As part of this work, we provide a new image similarity dataset (CoSIS), which currently consists of 30k labelled triples. The CoSIS dataset has 8k context-sensitive (CS) triples, which are divided into eight CS training sets (1k each) namely *Indoor*, *City*, *Ocean*, *Field*, *Mountain*, *Forest*, *Flower*, and *Abstract*. The CoSIS also contains 22k context-convolute (CC) triples, which are divided into two subsets, a validation set (12k) and a testing set (10k) for evaluating all models in an unbiased manner. As shown in Table 5.1, unlike existing datasets of triples in the literature, e.g., BAPPS [367], in both CS and CC portions of CoSIS, the two candidates x_a and x_b are selected randomly. Therefore, there is no guarantee that any of the candidates is semantically similar to or the same as the reference image x_r . The images of the triples are from the BG20k dataset [160], which consists of 20k background images. Hence the data space of the triples is of the size of $\|\mathbb{D}^3\| = (20k)^3$.

Two-Alternative Forced-Choice (2AFC). For each of the 30k triples (x_r, x_a, x_b) , we provide binary annotation: -1 if x_a is considered closer to x_r , and 1 otherwise.

Each triple is labelled by three annotators. Among the three annotators, our inter-rater reliability score is 0.947 which is higher than most cognitive tasks, e.g., emotion detection and many NLP tasks [71]. In some cases, we discarded triples when: (i) annotators considered two candidates x_a and x_b were very similar and equally distanced from the reference (e.g., they are both snowy mountains), and (ii) when both x_a and x_b are totally irrelevant to the reference image x_r (e.g., a desert and an ocean are both almost completely irrelevant to a kitchen). With random selection, cases of (i) are relatively rare ($\leq 4\%$), while cases of (ii) are more common (around 14%).

Context-Sensitive (CS) Training Sets. For each representative reference image R_i , we collect 1,000 triples where the two candidates x_a and x_b are randomly selected. We denote it as a CS dataset T_{CS_i} . CoSIS currently has eight CS datasets based on the eight categories defined by [76]. The eight representative reference images are: Indoor (#715), City (#2723), Ocean (#389), Field (#1673), Mountain (#1006), Forest (#254), Flower (#2352), and Abstract (#667). When training and evaluating each CS model on a CS dataset D_{CS_i} , we split the 1,000 triples into 667 for training and 333 for validation.

Context-Convolute (CC) Data. These randomly selected triples were labelled for aiding the analytical ensemble strategies and evaluating fairly the performance of all models concerned. The CC data set has 22,008 triples with 2,330 unique reference images. Three images in each triple are randomly selected, and each unique reference image has at least 9 labelled triples. Therefore, the testing results for each single reference image are reasonably statistically significant. We further split the 22k random triples into a validation set (12,006 triples with 1,320 unique reference images) and a testing set (10,002 triples with 1,010 unique reference images). The validation set is used to construct ensemble models and directly fine-tune deep models (which is not effective) for comparisons, while the testing set is used for comparing the global performance of all models/algorithms concerned. Note that the testing set does not overlap with either the validation set or any CS training set, in terms of both reference images and candidate images.

Data Cleaning. Among the three annotators, when there were disagreements, we used majority votes as the final labels. In the original labelled triples, there were some loops (e.g., with reference R , A is closer than B , B is closer than C , C is closer than A). We found only 0.11% triples that were in at least a loop, and the longest loop involved four candidates. These triples were manually removed.

5.4 Methodology

As discussed earlier, the data space of triples (R, A, B) is huge. Since human intelligence is typically developed in a context-sensitive manner (e.g., most people grew up in one small region), we explore a methodology for developing similarity models based on context-sensitive (CS) learning. As illustrated in Figure 5.3, we selected several representative reference images, $R_i, i = 1..K$, and for each R_i , we labelled N_i image pairs (A, B) in relation to R_i . We fine-tune each CS model on one of the CS clusters. We then use the validation set of the context-convolute (CC) dataset (triples with

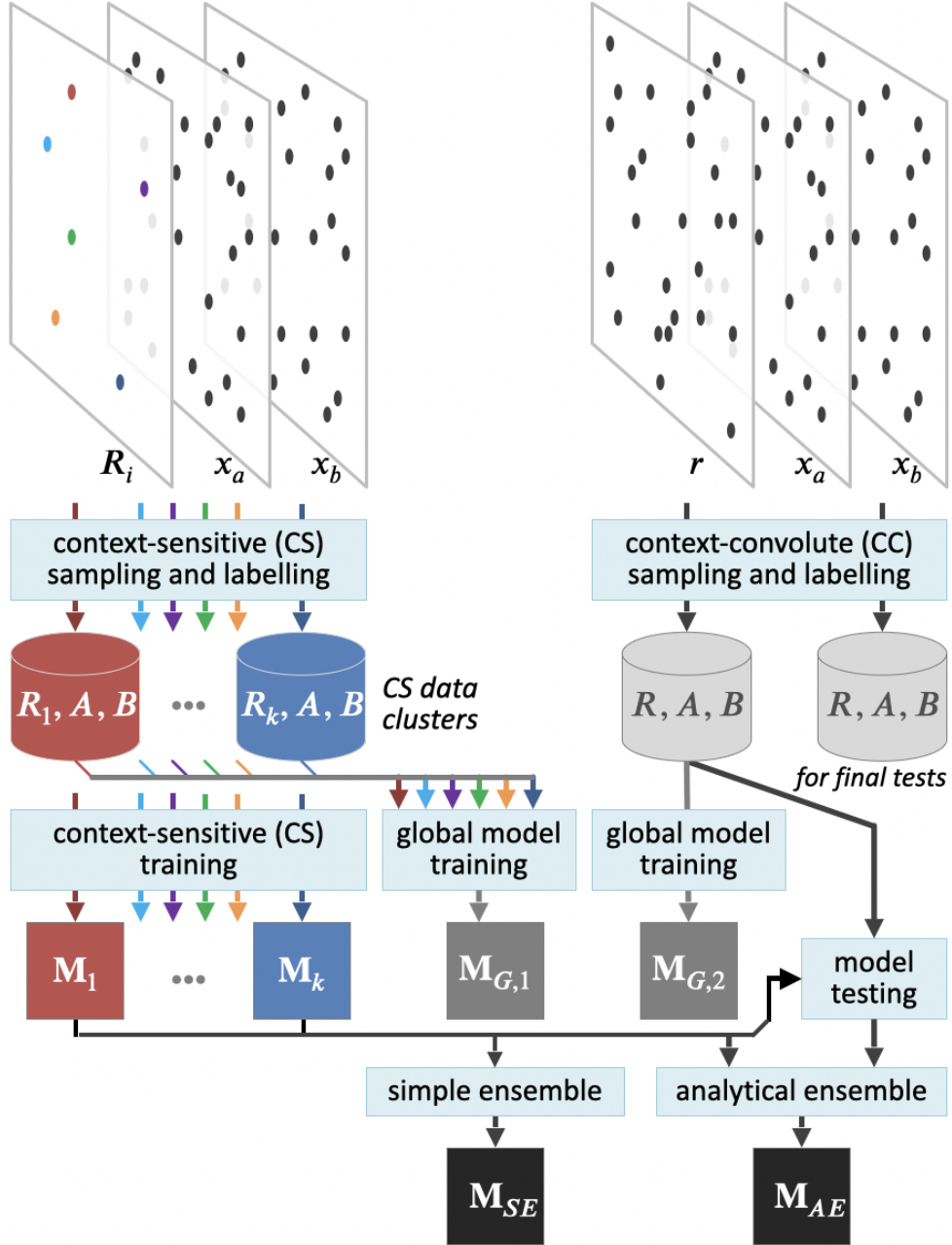


Figure 5.3: Overview workflow: each CS-model is trained on each CS data cluster. An analytical ensemble model is obtained based on the performance of each CS-model on the validation set. We also train global models using amalgamated data from the validation set and CS clusters for comparisons.

random reference images) to conduct testing/analysis and construct ensemble models from the K CS models. Finally, we test and compare all models concerned on the testing set of the CC dataset.

Due to the limited size of the annotated dataset, in this work, we focus on 1) showing that directly fine-tuning on the entire dataset is not effective (see Section 5.5.2), 2) demonstrating that by fixing one reference image (R_i) to form a CS data cluster, our CS models are able to improve local performance, i.e., when unseen reference images are similar to R_i , and 3) addressing the problem of lack of labelled data by constructing ensemble of our CS models to improve global performance. We leave the studies on more advanced triplet loss and more advanced deep metric training paradigms for future work when more annotated data are available. In the following subsections, we detail the fine-tuning of CS models and the construction of ensemble models.

5.4.1 Fine-tuning Models

We used a simple paradigm to fine-tune our CS models on each CS data cluster. And for the purpose of comparative evaluation, we also used the same approach to fine-tune two global models, $M_{G,1}$ and $M_{G,2}$. The former is trained on a mixture of CS data (fixing reference images), while the latter is on the validation set of the CC data (triples with random reference images) as shown in Figure 5.3.

Training Paradigm. As shown in Figure 5.4, we use a standard training procedure with both triplet loss and cross-entropy loss for the ranking block (binary classifier). The ranking block is helpful when the data is sparse or when the backbone is not a large model, e.g., resnet18. When the backbone is large, e.g., ViT, Lora [122] is used to reduce the number of trainable weights.

Context-based Triplet Loss. The (R, A, B) triples that we used in the work are conceptually similar to the traditional contrastive / triplet loss setting (anchor, positive, negative) in deep metric learning. However, instead of pulling positive examples closer to the anchor whilst pushing negative examples away from the anchor, the selection of the two candidates A and B is random. One can switch A and B or flip the annotation for similarity augmentation. Therefore, it is not always appropriate to push A or pull B . Formally, we define the triplet loss function as:

$$L_{\text{diff}} := (d(f(x_{\text{ref}}), f(x_a)) - d(f(x_{\text{ref}}), f(x_b))) \times y$$

$$L_{\text{triplet}} := \max(\text{margin} - L_{\text{diff}}, 0)$$

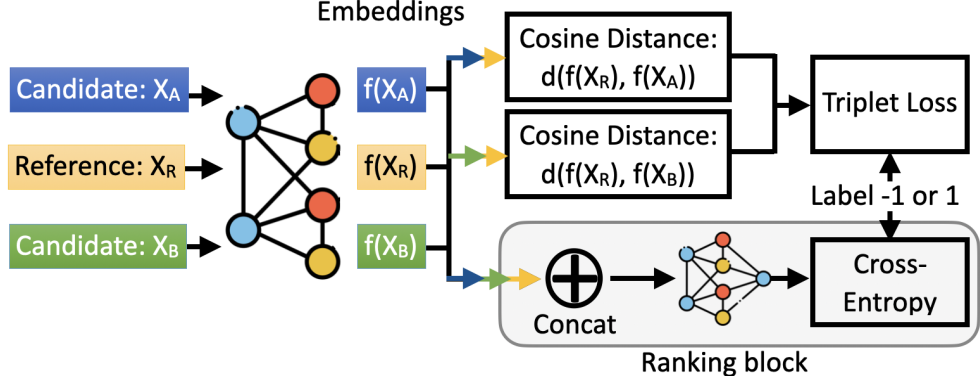


Figure 5.4: To train each CS model, we concatenate the embeddings and train a small ranking block to conduct binary classification. The cross-entropy loss of the ranking block, triplet loss, and LoRA [122] are used to assist in fine-tuning the backbone.

where f represents the backbone of an ML model (M), $f(x)$ denotes the embedding of an input x to M , d is a traditional distance function between two embeddings (e.g., cosine distance), and y is the annotated similarity label of the triple (x_r, x_a, x_b) that also controls the sign of the loss function.

5.4.2 Ensemble Strategies

Our Ensemble Strategies are constructed based on the performance of each CS model. In this section, we first detail how we analyse the testing results of each model on the validation set of our CC dataset (triples with random reference images), as well as how we construct ensemble models based on the analysis.

Context-Convolute (CC) Testing. While one may train a set of CS models, one would like to use these models to construct an ensemble that can be applied to other contexts, i.e., when the testing triples include unseen reference images. In our CC dataset, we include multiple triples with $\|\mathbb{T}_V\|$ random reference images, and for each random reference image, there are at least nine labelled triples (see Section 5.3). Therefore, in addition to a global accuracy score, we can report the testing results on the CC dataset based on individual reference images. For all triples (R_i, x_a, x_b) sharing the same reference image R_i , and each CS model $M_i, i = 1..K$, the testing yields a correctness indicator. The total number of such indicators is $\|\mathbb{T}_V\| \times K$. The testing can also result in additional information, such as confusion matrix and uncertainty or confidence values. The CC testing results can inform the construction of ensemble models.

Feature-based Analysis and Specialization. Given a large set of images, one can extract l features, which define an l -D feature space of the images. These features

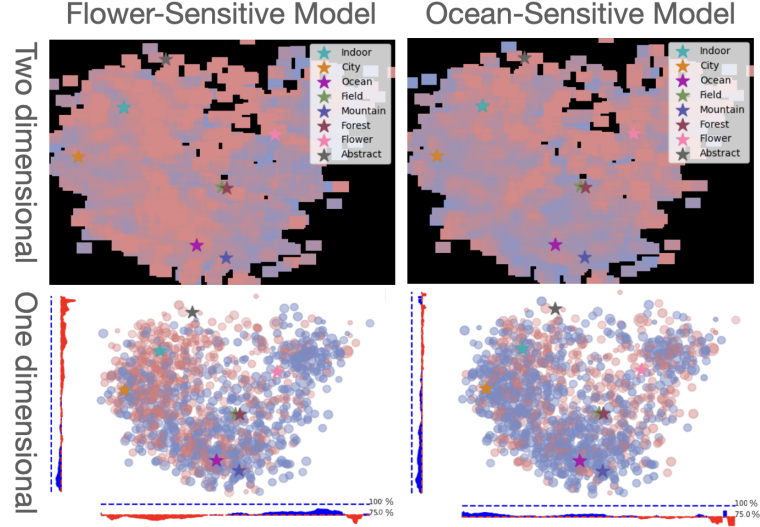


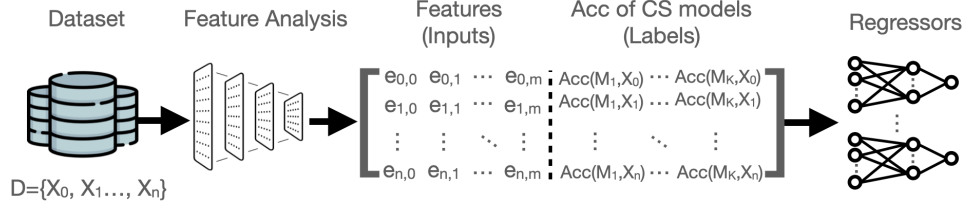
Figure 5.5: Ensemble Approach (PCA): for all triples (R_i, x_a, x_b) sharing the same reference image R_i , we compute an accuracy score from each model. We visualize the accuracy scores of the $|\mathbb{T}_V|$ reference images in our validation set using PCA or tSNE. Different models perform well in different areas. An ensemble method can be obtained based on the scatter plots.

can be the results of dimensionality reduction techniques such as PCA and t-SNE as well as hand-crafted feature extraction algorithms. Each image can thus be encoded as an l -D feature vector $\Theta = [\theta_1, \theta_2, \dots, \theta_l]$.

For an arbitrary reference image R_i , its feature vector Θ_r determines its l -D coordinates in the feature space. When all triples (R_i, x_a, x_b) sharing the same reference image R_i are tested against a CS model M_i , the correctness indicator can be considered as a sample of a correctness manifold at position Θ . With K CS models, we have K such manifolds based on correctness indicators. The testing of a CS model M_i on the validation set provides us with a way to establish an approximate model of the manifold that can be used to predict the correctness of applying M_i to a previously-unseen image triple as shown in Figure 5.5.

Ensemble based on Credibility Maps. An l -D *credibility map* of a model M_i is a discrete partition of the l -D feature space into a number of l -D cells, and each cell stores a value (or values) indicating the probability of M_i to be correct when it applies to any image triples (R_i, x_a, x_b) where the feature vector of R_i falls into the cell. Figure 5.5 illustrates such credibility maps in 1D and 2D sub-spaces. The two plots above show the 2D credibility maps of two CS models fine-tuned on the flower data cluster and ocean data cluster respectively. Each of the 2D manifolds is sampled on 12k triples with 1,320 different reference images, and the 2D feature subspace is

Development / Training



Deployment / Testing

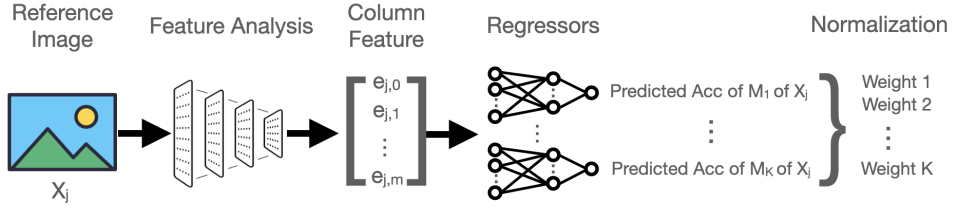


Figure 5.6: Ensemble Approach (MLP): The input of the MLP regressors is the features of a reference image R_i , and the outputs are the predicted accuracy score of each CS model on all triples (R_i, x_a, x_b) sharing the same reference image R_i . For any previously unseen triple, the outputs can be used as the tailored ensemble weights of the CS models.

partitioned into 200^2 cells. The two images below include four line plots representing four 1D credibility maps. Each line plot results from the projection of a set of testing results. From Figure 5.5, we can observe that these credibility maps can provide useful information about the past and potential performance of different CS models in different parts of the feature space.

One ensemble strategy is to determine, for any image triple (R_i, x_a, x_b) , how much each CS model should contribute to the decision. The feature vector of the reference image R_i is used to look up relevant cells in one or more credibility maps. Likely partitioning a high-dimensional feature space will result in many empty cells. A practical solution is for an ensemble algorithm to consult several low-dimensional credibility maps for each CS model and aggregate the credibility values into a single credibility score per CS model. The scores for different CS models can then be used to determine the contribution of each CS Model in the final decision specifically for the image triple(s) (R_i, x_a, x_b) .

ML-Based Ensemble Strategy. The weights of the CS models can also be produced by another ML model, which is trained using the feature vector of R_i as the input and the accuracy scores on all triples (R_i, x_a, x_b) (sharing the same reference images R_i) as the correctness label. Theoretically, the weights can be jointly

Table 5.2: Local Performance of existing supervised and self-supervised models on different Context-Sensitive testing clusters.

<i>Traditional Similarity Metrics and Deep Image Retrieval Models</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
CVNet [159]	67.4%	58.8%	71.6%	52.8%	62.3%	56.9%	58.9%	64.6%	61.6%
HesAff SIFT+SP [240]	66.8%	60.0%	55.1%	41.8%	53.3%	37.7%	54.4%	59.2%	53.5%
GeM-ResNet50 [247]	73.7%	77.6%	75.1%	71.0%	68.0%	77.2%	73.9%	55.9%	71.6%
SfM-ResNet50 [246]	81.9%	81.6%	85.8%	69.5%	83.3%	74.8%	73.6%	69.8%	77.5%
<i>Large Self-Supervised Models Trained on Large Datasets</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
DINO [34]-ResNet50	69.2%	81.2%	76.6%	70.7%	81.7%	74.6%	77.8%	49.2%	72.6%
DINO [34]-ViT-B16	74.6%	83.3%	83.8%	69.0%	87.7%	72.5%	75.4%	70.3%	77.1%
CLIP [272]-ViT-B32	70.1%	78.8%	82.9%	68.7%	85.6%	79.0%	77.2%	80.8%	77.9%
<i>Embedding Distances Based on Backbones of Supervised Models Trained on Large Datasets</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
ResNet18-Place365 [373]	61.1%	78.2%	78.1%	69.6%	83.8%	77.5%	72.1%	72.1%	74.1%
ResNet18-ImageNet [117]	76.9%	82.4%	81.1%	66.0%	87.4%	72.2%	66.1%	67.6%	75.0%
VGG16-ImageNet [288]	81.4%	82.7%	88.9%	62.4%	88.6%	75.4%	70.6%	67.0%	77.1%
ViT-ImageNet [67]	82.3%	83.1%	83.0%	76.0%	87.4%	83.8%	86.8%	72.9%	81.9%

Table 5.3: Local performance of different CS models (trained on the CS training dataset) on the corresponding testing dataset.

<i>Our Context-Sensitive Models - Different Pre-trained Architectures</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
VGG16-ImageNet	86.2%	86.6%	86.1%	90.1%	87.8%	76.9%	72.3%	74.5%	82.6%
ResNet18-ImageNet	88.0%	86.0%	88.0%	78.8%	90.4%	82.9%	85.5%	79.3%	84.8%
ResNet18-Place365	86.5%	87.5%	88.6%	88.1%	90.4%	77.8%	70.6%	83.8%	84.2%
ViT-Lora	80.2%	82.7%	89.5%	83.6%	90.7%	86.2%	86.8%	78.4%	84.8%

learned by a large ML model with a larger dataset. In this work, we show that, with a relatively small validation set (12k triples with 1,320 unique reference images), we can still train another simple ML model to predict the performance of each CS model. As shown in Figure 5.6, firstly, we extract features of the reference images in the validation set using a neural net (e.g., ViT), and we then use a dimensionality reduction method (e.g., PCA) to counteract the sparseness of the annotated data. In our main implementation, we used 64 dimensions. The feature of a reference image R_i is fed to several multi-layer perceptrons (MLPs), each of which is trained to predict the accuracy score of one CS model on all triples with R_i . In the deployment process, given a random triple, the MLPs estimate the likelihood score of each CS model making a correct decision. The normalized scores are used as the weights for determining the contribution of each CS model for the specific triple.

5.5 Experiment

To consolidate our methodology, we conducted extensive tests. In this section, we use experimental results to show:

Local improvement of CS Models. We fine-tuned eight CS models $M_i, i = 1..8$, each of which was fine-tuned on the training-split of one CS data cluster (667 triples with fixed reference image R_i , see Section 5.3). We show that our CS models are able to improve performance only when unseen triples contain reference images that are similar to R_i (local improvement), but not on the entire CC testing set of 10k random triples.

Ineffective Global Models. We directly fine-tuned two global models, one on the CC validation set (12k triples with random reference images), and the other on the amalgamated CS training sets (8k in total). Due to the limited amount of labelled data, these two global models do not perform well on the CC testing set (10k).

Effective Ensemble Models of CS models. We analyze the performance of our CS models on the CC validation set (12k), and construct ensemble models to improve the performance on the CC testing set (10k),

5.5.1 Performance of Context-Sensitive Models

Performance on the CS Training Sets. As shown in Table 5.2, using embedding distances from image retrieval models achieved around 60%~78%, and large self-supervised models, e.g., CLIP/DINO, and supervised models, e.g., ViT, were able to achieve 78%~82%. To investigate whether we can improve the performance, we fine-tuned several models of different architectures. As shown in Table 5.3, as expected, the fine-tuned CS models outperform all existing methods on their corresponding CS datasets. Due to the limited number of labelled data, compared with ResNet18 and VGGNets, larger architectures might not be effective, e.g., ResNet34 and ResNet50 reached around 60%~65% whilst ResNet18 and VGGNets reached around 81%~84% on average. Therefore, we fine-tuned larger and deeper models with LoRA [122] to boost the performance. ViT with Lora (denoted as ViT-Lora) achieved the best averaged accuracy (around 85%), which is aligned with the most recent studies on evaluating synthesized images using labelled triples [85, 367].

In Table 5.3, we highlight the best CS model fine-tuned on each CS data cluster. These eight CS models achieved 84%~91%, which is significantly better than all of the existing methods in Table 5.2. The results suggest that our CS models can

Table 5.4: No significant global improvement for CS models

Model	Indoor	City	Ocean	Field	Mountain	Forest	Flower	Abstract
Validation Set	77.3%	75.4%	78.3%	73.3%	79.5%	75.7%	76.6%	79.3%
Testing Set	76.9%	74.3%	78.4%	72.7%	79.1%	73.9%	73.9%	77.2%

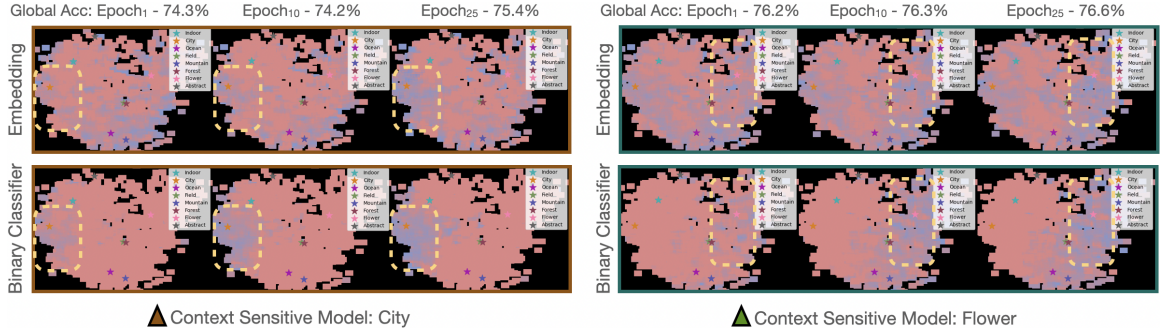


Figure 5.7: Visualized CS training: the local performance (highlighted areas) is improved gradually during training, whilst the global accuracy remains stable. The second row shows the changes in binary classifiers’ performance from scratch. It is more noticeable that the performance in the highlighted areas is constantly improving during training. The first row shows that we can also see the same improvement in the highlighted areas when using embeddings, especially when comparing the results at the beginning and end of CS training. This shows that CS training improves local performance for both binary classifiers and embeddings. The bluer, the more accurate the CS model that is being trained, whilst red indicates accuracy $\leq 75\%$.

Table 5.5: Performance of global models on the testing set (10k)

Training Set Architecture	No Training Embedding Distance	Context Training Set (8k)	Validation Set (12k)
VGG16	78.3%	71.4% $\pm$ 4.2%	77.3% $\pm$ 2.0%
ResNet18	77.7%	72.1% $\pm$ 1.9%	80.2% $\pm$ 1.6%
ViT-Lora	79.9%	68.9% $\pm$ 3.6%	79.6% $\pm$ 1.3%

outperform existing models locally: when triples contain seen reference image R_i and unseen candidates x_a and x_b . In Section 5.5.4 and Figure 5.7, we visualize the CS training procedure, and the results also show that our CS models are able to improve performance locally: when the triples contain reference images that are similar to R_i . **Performance on the CC Validation (12k) / Testing Set (10k):** To evaluate how the selected CS models perform on triples with random unseen reference images, we run each CS model M_i on the CC validation and testing set. As shown in Table 5.4, the CS models achieved 73%~79.5%, which is lower than the results they achieved on the CS clusters. And the results are also slightly worse or similar to the existing models, e.g., ViT, as shown in Table 5.6. This shows that our CS models can only improve performance locally (on triples with similar reference images) but not globally (on triples with random reference images).

Table 5.6: Global performance comparisons on testing set: 10k random triples

Model Description	Global Performance on Testing Set
Best Existing Model (not fine-tuned): ViT [67]	79.9%
Best global model (directly and globally fine-tuned)	80.2%
Best single CS model (locally fine-tuned)	79.1%
Ensemble - Majority Voting of CS models	81.9%
Ensemble PCA - Ours	83.3%
Ensemble MLP - Ours	84.7%±0.3%

5.5.2 Performance of Global Models

One straightforward potential solution to improve global performance is to fine-tune deep models on triples with random reference images. Therefore, we fine-tuned two global models, M_{G1} and M_{G2} . The former was fine-tuned with an amalgamation of the eight CS data clusters (8k triples in total), and the latter with the CC validation set (12k triples with 1,320 random reference images). Three architectures were used. Table 5.5 shows the results of testing these models on the CC testing set (10k triples with 1,010 unique and random reference images).

The global models trained on the CC validation set (M_{G2}) achieved 77%~80% on average, which is similar to existing models, e.g., ViT: ~80% and ResNet18: ~77%, which are not trained on any of our labelled data. Additionally, fine-tuning on all of our context training sets (8k) (M_{G1}) does not improve the performance on the testing set either. Moreover, fine-tuning one single model on all of the amalgamated context datasets led to a decrease in accuracy on the testing set (69%~72%) compared with their untrained counterparts (77%~80%). This might be caused by the sparsity of the context training set which only contains eight reference images.

The worse performance (compared with CLIP or DINO) of these two global models shows that: directly fine-tuning on random triples does not improve the performance due to the huge data space and limited amount of labelled data. By contrast, fixing the reference image R_i hugely reduces the size of data needed to fine-tune the neural nets. Therefore, each single CS model is able to outperform existing algorithms on triples with seen reference images that are similar to R_i (Figure 5.7). To boost the global performance (on random triples), one plausible approach is to construct ensemble models to utilize the local improvement of each CS model.

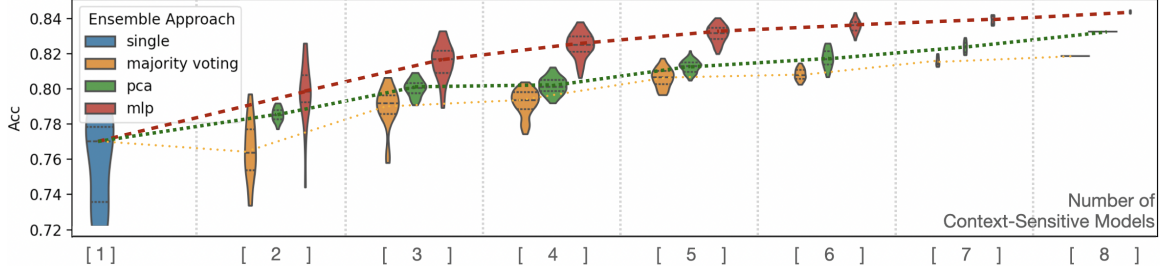


Figure 5.8: Y-axis: accuracy of the ensemble methods on the testing set (10k). X-axis: the number of CS models used to form the ensemble model. Experiments are run on all the combinations, e.g., when choosing two CS models, we run experiments on all of the $C_2^8 = \frac{8!}{2! \cdot 6!} = 28$ combinations. For MLP, we repeat the same experiment three times for one combination, e.g., for choosing two CS models, we run $3 \cdot C_2^8 = 84$ experiments. MLP-based approach consistently performs the best. Dashed lines inside the blobs are the quartiles of the data.

5.5.3 Performance of Ensemble Models

Experiments on the Validation Set (12k). Based on each CS model’s performance (e.g., visualized in Figure 5.5 and 5.7) on the validation set, we obtain the weights of the ensemble models using two methods: PCA, and MLP as specified in Section 5.4. As shown in Table 5.6, both of our ensemble models perform 8%~10% higher than existing models, the best CS model, the global models and the simple ensemble approach (majority voting) on the validation set. The improvement is expected as the ensemble weights are constructed based on the performance of CS models on the validation set.

Experiments on the Testing Set (10k). To show the performance of our ensemble models on random unseen triples, we run the ensemble models on the CC testing set (10k triples with 1,010 random and unique reference images) which has no overlapping with the CC validation set or the CS training set (as stated in Section 5.3). Table 5.6 shows that both of our ensemble approaches perform ~5% better than existing models, the best CS model, the global model, and majority voting. The results show that our ensemble model is able to improve global performance (i.e., on random triples), and perform the best on the task of semantic similarity between images.

5.5.4 Result Analysis and Training Visualization

Number of Context-Sensitive Models Figure 5.8 shows the accuracy on the testing set of the three ensemble approaches (Majority Voting, PCA, and MLP) when using different numbers of CS models to form the ensemble models. For a

number of selected CS models, we run experiments on all possible combinations. To be specific, when selecting r from the $n = 8$ CS models where $r = \{1, 2, \dots, n\}$, we run experiments on all of the $C_r^n = \frac{n!}{(n-r)!*r!}$ combinations. For the MLP-based ensemble approach, we repeat the same experiments three times for one given combination, which leads to $3 * C_r^n$ runs of experiments for the MLP-based ensemble approach. The results show that the MLP-based approach consistently performs the best, and both of our proposed approaches (MLP-based and PCA-based) perform constantly better than the simple ensemble method, e.g., majority voting. In addition, the results indicate that the accuracy scores on the testing set start to saturate when we use more than six CS models. This might be the reason that the field-sensitive model, forest-sensitive model and mountain-sensitive model learn similar rules and perform similarly on the testing set. Therefore, assembling these similar CS models might not lead to a significant increase in global accuracy on the testing set.

Visualization of CS Fine-tuning: We visualize the CS fine-tuning process by showing testing results (on the validation set, 12k random triples), as well as reporting a global accuracy score (on the validation set) at the end of each training epoch. Each scatter point represents the accuracy score of a CS model on all triples (R_i, x_a, x_b) sharing the same reference image R_i , and the scatter point is located based on the feature vector of R_i (as stated in Section 5.4). As shown in Figure 5.7, we highlight the area where we see the local improvement (around R_i). Whilst the local performance on the highlighted area has improved, the global accuracy (on the entire validation set) almost remains unchanged, i.e., $\sim 74.5\%$ and $\sim 76\%$ for the city- and flower-sensitive model. This shows that CS fine-tuning is able to improve local performance but not global performance.

5.5.5 Ablation Studies

To complement Table 5.3 in Section 5.2, we also train CS models using other different architectures as shown in Table 5.14. The results show that training CS models using larger architectures might not be efficient, e.g., ResNet50 achieved only 61.7% on average whilst VGG11 and VGG13 achieved over 81%. Therefore, in the earlier part of this chapter, we report our best CS models with a smaller architectures, e.g., ResNet18, or a larger architectures trained with LORA [122]. Similarly to Figure 5.7 in section 5.5, we also show the CS training process for more CS models in Figure 5.10. All the CS models are able to acquire knowledge to improve local performance whilst the global performance remains stable during CS training.

Table 5.7: Performance of different context-sensitive models (trained on the context training dataset) on the corresponding testing dataset using ranking block (binary classifier)

<i>Keywords Extraction and Linear Regression</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
ResNet18-Place365	67.4%	71.3%	72.7%	70.1%	69.5%	70.9%	55.8%	62.2%	60.2%
ResNet18-ImageNet	64.2%	68.2%	64.8%	58.3%	68.6%	61.8%	50.2%	55.9%	56.1%
ViT-ImageNet	75.3%	74.5%	60.6%	58.4%	57.9%	54.5%	52.5%	63.5%	61.9%

<i>Our Models (Ranking Blocks) - Different Pre-trained Architectures</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
VGG11-ImageNet	87.0%	86.0%	86.8%	87.7%	88.1%	89.3%	87.3%	81.2%	86.7%
VGG13-ImageNet	88.0%	83.6%	88.4%	86.6%	87.6%	89.3%	88.8%	83.8%	87.0%
VGG16-ImageNet	88.3%	84.9%	85.4%	89.3%	90.1%	86.9%	89.1%	79.8%	86.7%
ResNet50-ImageNet	60.6%	61.4%	66.7%	68.3%	62.9%	64.8%	61.1%	55.8%	62.7%
ResNet34-ImageNet	72.2%	71.0%	73.4%	68.4%	75.2%	74.8%	73.9%	63.6%	71.6%
ResNet18-ImageNet	87.7%	83.0%	89.8%	88.4%	88.3%	89.8%	87.9%	80.8%	87.0%
ResNet18-Place365	88.6%	88.4%	91.0%	91.3%	92.8%	90.1%	90.1%	85.6%	89.7%
Lora-ViT	87.1%	82.7%	83.8%	80.6%	85.3%	90.1%	85.9%	79.6%	84.4%

Training CS models: when training each CS model, we use the following setting: 1) learning rate: 10^{-4} for ViT with LoRA, and 10^{-5} for others architectures, 2) number for epochs: 25, 3) loss: cross entropy + 0.1 * triplet loss, 4) batch size: 8, 5) optimizer: adam, 6) single image augmentation: random resized crop and horizontal flip, 7) triples augmentation: randomly swap candidate A and B, 8) normalization: (0.485, 0.456, 0.406), (0.229, 0.224, 0.225), 9) all images resized to: 224×224 . We did not carefully tune the hyper-parameters. Due to the small size of each CS cluster, we are able to train all of our CS models on one single laptop with Apple M1 Chip. The training time of each CS model varies from one day to three days (with ViT and LoRA), whilst the training time of the global models can be as long as two weeks (with ViT and LoRA). With the limited resources and limited amount of labelled data, we are able to improve the performance on the problem of context-sensitive image similarity using the proposed methodology.

5.5.5.1 Evaluation using Ranking Block (Binary Classifier)

In Figure 5.4, we showed two possible mechanisms for training a model. We can use embeddings to compute the cosine distance between two images. The results reported in the main body of this chapter (i.e., Table 5.4 in Section 5.2) are CS models built with this mechanism. Alternatively, also shown in Figure 5.4 (bottom-right), we can use a ranking block to derive a binary classifier. Here we report additional experiments for this alternative mechanism.

Using ranking block is beneficial for training the CS models, and it is necessary when data is sparse. Additionally, we can also use the ranking block to conduct

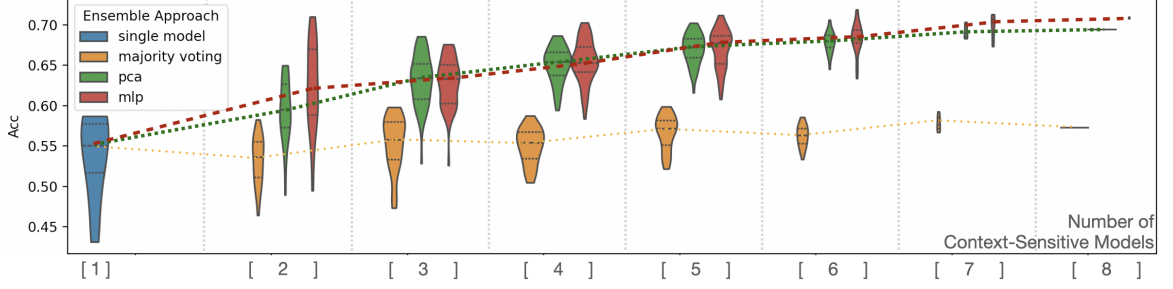


Figure 5.9: Y-axis: accuracy of the ensemble methods on the testing set (10k). X-axis: the number of CS models used to form the ensemble model. Experiments are run on all the combinations, e.g., when choosing two CS models, we run experiments on all of the $C_2^8 = \frac{8!}{2! \cdot 6!} = 28$ combinations. For MLP, we repeat the same experiment three times for one combination, e.g., for choosing two CS models, we run $3 \cdot C_2^8 = 84$ experiments. MLP-based approach consistently performs the best. Dashed lines inside the blobs are the quartiles of the data.

testing and inference. As shown in Table 5.7, The performance of the ranking blocks on their corresponding CS clusters is better than using the embeddings (cf. Table 5.3). The ranking blocks fine-tuned from ResNet18 (pretrained on Place365) consistently perform the best on all of the data clusters (highlighted). More specifically, the ranking blocks achieved 89.7% on average, which is 5.5% higher than using distances between embeddings as shown in Table 5.3. Due to the better performance on the CS clusters than using embeddings, it is worthy studying the performance of the ranking blocks.

To evaluate the performance of the ranking block (binary classifier O) and the trained backbone model (f), we use three metrics: 1) accuracy, 2) symmetry accuracy (swapping the two candidates and flipping the label), and 3) symmetry score defined as:

$$|O(f(R), f(A), f(B)) \times M(R, A, B) + O(f(R), f(B), f(A)) \times M(R, B, A)|$$

where $O(f(R), f(A), f(B))$ is the confidence score (output after softmax function) of the ranking block given a triples (Reference, Candidate A, Candidate B), and $f(x)$ represents the embedding of an input image x . We denote the final binary prediction of the whole model M (including the backbone f and the ranking block O) as $M(R, A, B)$. Note that, if the binary classifier is able to predict both (R, A, B) and (R, B, A) correctly, then $M(R, A, B) = -M(R, B, A)$.

Ambiguous Triples: Despite of the satisfactory performance on the CS clusters, using the ranking block for inference can bring in other problems. We define a triple

Table 5.8: Averaged Symmetric Scores of Ranking Blocks

CS Binary Classifier Testing Cluster	Indoor	City	Ocean	Field	Mountain	Forest	Flower	Abstract	Average
The same cluster	0.069	0.062	0.048	0.037	0.053	0.055	0.060	0.108	0.072
All other 7 different clusters	0.100	0.116	0.091	0.080	0.066	0.046	0.094	0.091	0.086

Table 5.9: Performance of Ensemble Models (Ranking Block) on Validation Set: 12k triples, and Testing Set: 10k triples.

	Best Single	Majority Vote	Ensemble(PCA)	Ensemble(MLP)
Validation Set	58.8%	57.1%	84.1%	78.7%± 0.6%
Testing Set	58.7%	57.3%	69.5%	71.5% ± 1.0%

Table 5.10: Cross Validation: performance of each context sensitive models, global model and ensemble models on all of our context datasets.

<i>Performance of each context-sensitive models on each context dataset</i>									
Dataset Model	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
Model: Indoor	88.0%	85.7%	84.7%	71.0%	88.9%	79.9%	74.8%	68.8%	80.2%
Model: City	55.4%	87.5%	79.0%	70.4%	82.3%	81.7%	77.8%	69.4%	75.4%
Model: Ocean	71.9%	82.1%	89.5%	76.7%	90.4%	65.3%	70.0%	70.9%	77.1%
Model: Field	79.0%	77.6%	87.1%	90.1%	86.2%	81.4%	54.7%	70.9%	78.4%
Model: Mountain	76.0%	85.1%	85.0%	75.5%	90.7%	81.4%	82.9%	75.4%	81.5%
Model: Forest	57.8%	76.4%	75.4%	83.3%	87.7%	86.2%	86.2%	58.3%	76.4%
Model: Flower	59.9%	70.4%	76.0%	76.1%	82.6%	79.0%	86.8%	52.3%	72.9%
Model: Abstract	57.2%	77.6%	81.7%	77.3%	90.6%	84.7%	72.4%	83.8%	78.2%
Average	68.2%	80.3%	82.3%	77.6%	87.4%	80.0%	75.7%	68.7%	-
Average wo diagonal	65.3%	79.3%	81.3%	75.7%	87.0%	79.1%	74.1%	66.6%	-
<i>Performance of Our Global, and Ensemble Models</i>									
Model: Global	57.5%	76.4%	82.9%	74.6%	84.1%	74.9%	64.0%	67.3%	72.7%
Ensemble (PCA)	76.0%	86.9%	88.6%	85.1%	94.0%	84.4%	86.2%	80.5%	85.2%
Ensemble (MLP)	79.7%	85.1%	87.5%	89.5%	94.0%	86.8%	86.2%	81.2%	86.3%

as ambiguous for the model M if the ranking block outputs the same prediction when we swap the two candidates, i.e., $M(R, A, B) = M(R, B, A)$. When using the binary classifier, to deal with ambiguous triples, we first check the confidence scores of the triples, i.e., if $O(f(R), f(A), f(B))$ is greater than $O(f(R), f(B), f(A))$, then we use the prediction of the former as the final prediction. If the confidence scores are the same, we can compare the distance between the embeddings obtained from the backbone f .

Symmetry Scores: We investigate whether the ranking blocks can produce consistent predictions when swapping the two candidates. We show the averaged symmetry scores of each ranking block in Table 5.8. We test each ranking block on the same CS cluster that it was trained on, as well as all other seven different CS clusters. And the results show that the trained ranking blocks are more symmetric when the reference image has been seen.

Ensemble Models: We also construct ensemble models using the binary classifiers, and test the ensembles on the randomly collected triples, i.e., our validation

Table 5.11: Ensemble (MLP-based) models with one missing CS model, performance on each context dataset.

Cluster Ensemble	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
No Indoor Model	76.0%	83.8%	86.8%	89.2%	93.7%	86.2%	87.1%	81.2%	85.5%
No City Model	81.7%	83.5%	87.4%	88.3%	93.4%	86.2%	87.7%	80.8%	86.1%
No Ocean Model	78.7%	85.1%	85.6%	90.7%	93.4%	87.7%	87.6%	81.3%	86.3%
No Field Model	73.6%	84.7%	86.2%	86.6%	93.7%	85.0%	87.9%	80.5%	84.8%
No Mountain Model	79.3%	84.4%	87.4%	90.1%	92.8%	86.2%	86.2%	80.5%	85.9%
No Forest Model	78.1%	86.5%	88.6%	89.8%	93.4%	85.9%	85.2%	80.8%	86.0%
No Flower Model	80.2%	85.6%	88.0%	89.5%	93.7%	85.9%	85.5%	81.7%	86.3%
No Abstract Model	80.2%	85.6%	87.4%	87.7%	93.1%	86.2%	86.5%	78.9%	85.7%
Average	78.5%	84.9%	87.2%	89.0%	93.5%	86.2%	86.7%	80.7%	-

Table 5.12: Performance of Ensemble Models with one context-sensitive model being left out on Testing Set (10k)

Missing Model:	<u>Indoor</u>	<u>City</u>	<u>Ocean</u>	<u>Field</u>	<u>Mountain</u>	<u>Forest</u>	<u>Flower</u>	<u>Abstract</u>	No Missing
Ensemble PCA	81.9%	82.0%	82.1%	82.4%	82.4%	82.7%	82.6%	82.9%	83.3%
Ensemble MLP	83.9%	83.8%	84.2%	84.5%	83.7%	84.1%	84.2%	83.9%	84.7%

set (12k triples) and testing set (10k triples). As shown in Table 5.9, the ensemble model of ranking blocks achieved around 78%~84% on the validation set, and ~70% on the testing set. The results are significantly better than any of the single CS model (binary classifier) and majority voting. However, the results are 10~15% worse than using embeddings as shown in Table 5.6 (which is expected as the ranking blocks are trained from scratch using 667 triples only). Similarly to Figure 5.8 in section 5.5, we also show the accuracy scores of the ensemble models increase when the number of the CS models (using binary classifiers) increases in Figure 5.9. Due to the worse performance on random unseen triples, we focused on the ensemble models constructed using those CS models based the embedding distances, rather than these binary classifiers.

5.5.5.2 Cross Validation of CS Models

To investigate how the eight selected CS models perform on the other types of unseen reference images, we test each CS model M_i on all CS datasets $\{D_1, D_2, \dots, D_k\}$. As shown in Table 5.10, each CS model perform the best when the reference image is the same as the ones they are trained for, i.e., the accuracy scores on the diagonal are the highest for each CS cluster. Comparing Table 5.10 with Table 5.2, we can make the following observations:

1. Trained with small CS datasets, the fine-tuned CS models performed well on their corresponding datasets (84%~91%)., suggesting some possible advantages

Table 5.13: Performance of meta-CS Models (ResNet18-Place365) trained on combined CS clusters, tested on each CS cluster. Nature: Ocean / Field / Mountain / Forest, Object: Flower / Abstract.

CS Cluster CS Model	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
City/Indoor	84.4%	82.1%	63.2%	49.3%	80.8%	73.7%	72.7%	56.2%	70.3%
Nature	78.4%	77.0%	68.9%	59.7%	86.8%	76.9%	63.1%	77.2%	73.5%
Object	60.2%	77.9%	74.3%	67.8%	76.6%	76.9%	68.2%	71.5%	71.7%

Table 5.14: Performance of different context-sensitive models (trained on the context training dataset) on the corresponding testing dataset.

<i>Our Context-Sensitive Models - Different Pre-trained Architectures</i>									
Reference Image	#Indoor	#City	#Ocean	#Field	#Mountain	#Forest	#Flower	#Abstract	Average
VGG11-ImageNet	86.5%	84.2%	85.9%	88.3%	86.9%	76.4%	67.2%	76.7%	81.5%
VGG13-ImageNet	85.9%	86.9%	84.5%	89.0%	87.1%	77.5%	68.3%	76.9%	82.0%
ResNet50-ImageNet	41.6%	66.8%	58.6%	74.3%	80.1%	60.0%	62.6%	49.6%	61.7%
ResNet34-ImageNet	50.3%	71.0%	60.2%	72.8%	83.9%	63.3%	62.6%	53.4%	64.7%

of context-sensitive training.

2. As shown in the last column of Table 5.10, the fine-tuned CS models achieved 73%~81% across different CS datasets on average. Compared with existing models (Table 5.2), e.g., 75% for ResNet18 and 81.9% for ViT, individual CS models (with much smaller CS training datasets) achieve comparable results on different CS clusters on average.
3. Close examination shows that some CS models perform reasonably well on some other CS data clusters (e.g., the Indoor Model on #City, #Ocean, and #Mountain datasets), but not so well on other CS data clusters (e.g., the Forest Model on #Indoor and #Abstract datasets). This suggests that (i) our CS models can be used on the data that they have not seen in some cases, and (ii) If we can predict statistically how our CS models will perform on unseen reference images via testing and analysis, we are able to produce a stronger ensemble model as we show earlier in this chapter.

5.5.5.3 Ablation Study on Ensemble Models

To compare how eight CS models contribute to the ensemble model, we construct ensemble models using seven CS models with one CS model being left out. We run each ensemble model on the CS clusters. The results are shown in Table 5.11. For labelling each ensemble model in the table, “No X Model” denotes an ensemble that is composed of seven CS models without the X CS model. For example, “No Indoor

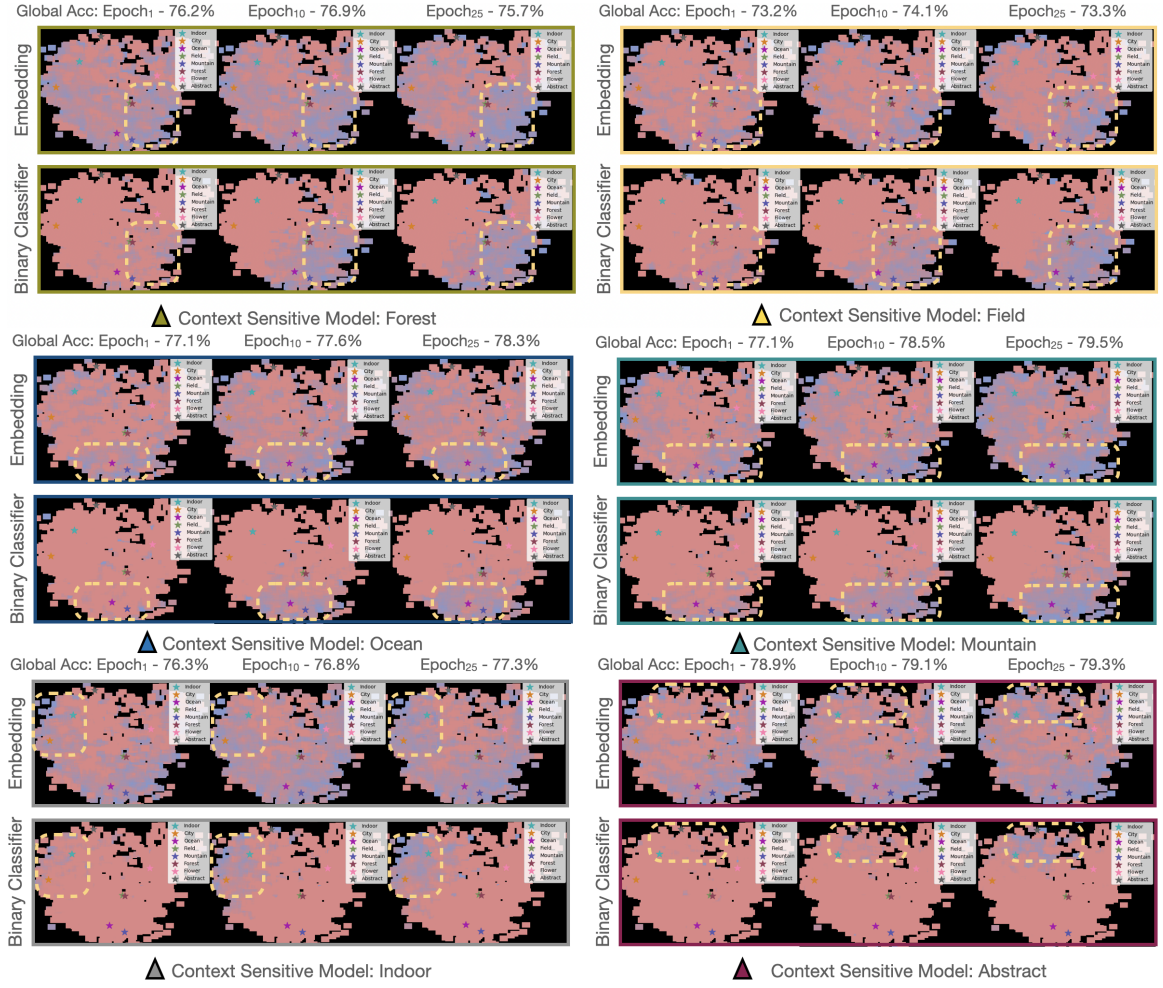


Figure 5.10: More visualized CS training process: all of the CS models are able to improve local performance (highlighted areas) but not global accuracy. The local improvement of abstract-sensitive model (on the right) is less noticeable because 1) there are not too many “abstract” reference images in the validation set, and 2) the “abstract” images might not be grouped together when applying dimension reduction, e.g., tSNE or PCA.

Model” denotes that the CS model trained with #Indoor dataset was left out in the ensemble model.

When comparing each column, the ensemble without the X CS model often performs the worst on the X data cluster. For example, the ensemble “No City Model” performs the worst on the #City data cluster in comparison with other ensemble models with the “City” CS model as shown in the #City column. Occasionally, there are exceptions, e.g., the ensemble “No Indoor Model” is the second worst in the #Indoor column, and “No Flower Model” is the second worst in the #Flower column.

Nevertheless, in general, the performance difference in the table is much smaller than those in Table 5.10, and the average values in the last column also improves significantly. This indicates that (i) an ensemble of CS model can improve the overall performance of those component CS models, (ii) an ensemble of CS model can work reasonably well on a CS dataset that none of the component CS models saw previously. One interesting observation is that all ensemble models perform well ($\geq 93\%$) on the #Mountain data cluster, including the ensemble “No Mountain Model”. This suggests that the knowledge of image similarity in the context of mountains might also be learned from other CS data clusters.

We also test the eight “No X models” on the 10k context-convolute testing set, and the results are shown in Table 5.12. For example, the cell “Ensemble PCA”-“~~Indoor~~” shows the result of applying the an ensemble model that consists of seven CS models excluding the Indoor Model and is constructed based on credibility maps (Section 3.1). We highlight the best and the worst results for each ensemble approach. The results show CS models have different impacts on different ensemble strategies, e.g., the mountain-sensitive model is considered the most important for the MLP-based ensemble whilst the PCA-based ensemble might consider the indoor-sensitive model the most important.

All of the ensemble models perform relatively satisfactory, even on the left-out clusters. Compared with results of the ensemble model using all eight CS models (the last row in Table 5.10), the ensemble models with seven CS models only perform slightly worse on average.

5.5.5.4 Meta Context-sensitive models

In this chapter, we show that training a global model on the mixtures of all CS clusters does not improve the performance. As inspired by [15] where randomized meta-proxies are shown to be more effective, we also run experiments on meta-CS

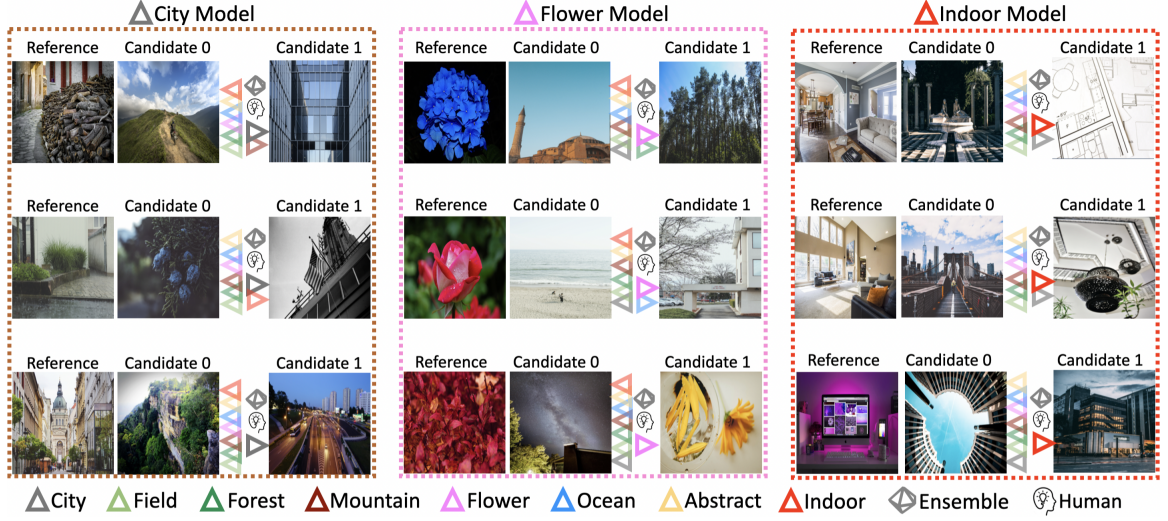


Figure 5.11: Arrow points at the candidate which is considered closer to the reference by the model(s) or human annotation. A CS model, e.g., city model, flower model and indoor model, is specifically designed and fine-tuned for one reference image R_i . Therefore, each CS model is able to achieve satisfactory results when the unseen reference images are close to R_i , e.g., when they are all flowers, whilst it cannot generalize well to other reference images that are not in the same cluster of R_i . Our ensemble approach is able to utilize each CS model’s advantages to the max and is able to achieve satisfactory results on the CoSIS dataset we collected that consists of multiple random triples.

models trained on meta-CS clusters. In Table 5.13, we show performance of meta-CS models trained with two or three CS clusters. To be more specific, we train: 1) City/Indoor-sensitive model with the City and Indoor clusters, 2) Nature-sensitive model with the Mountain, Forest, Ocean, and Field clusters, 3) Object-sensitive model with the Abstract and Flower clusters. The three meta-CS models achieved 70%~73% on average when applied to different CS clusters, which is similar to the performance of the global models (Table 5.5). Additionally, compared with individual CS models (as shown in Table 5.10), the meta-CS models do not perform well on any of the individual CS clusters. The results provides more evidence supporting the observation that training a global model on the mixtures of all CS clusters does not improve the performance. Therefore, when assembling the CS-models, we consider only the CS models, each of which is trained with only one CS cluster.

Local Improvement from CS Training (Examples) In Figure 5.11, we show some examples in which deep embeddings, e.g., CLIP and DINO, are not aligned with human annotations. Moreover, some examples can only be correctly predicted by one

or two CS models. For example, in the middle of Figure 5.11, when the reference images contain flowers, only the Flower CS Model is able to predict the triples correctly and consistently. Similarly, each CS model is able to improve performance when the reference images are close to its CS data cluster. As a result, our ensemble approach is able to utilize the advantage of each CS model and improve global performance on the CoSIS dataset.

5.6 Conclusions

In this chapter, we addressed a new problem of developing image similarity models based on context-sensitive (CS) training datasets that contain images triples (R, A, B) focusing only on a few reference images. We trained a set of CS models, and our tests showed their ability to improve performance locally in their corresponding contexts but not globally when being applied to other contexts. We introduced a new approach to estimate a correctness manifold for each CS model based on imagery features and the testing results of the CS model. The estimated manifolds of CS models enable an ensemble strategy that predicts the correctness probability of each CS model dynamically for each input triple (R, A, B) and determines the contribution of CS models accordingly. Our extensive experiments showed that our proposed methods performed the best in comparison with all existing models, simple ensemble models, and trained CS and global models.

In addition, we have collected a dataset of 30k labelled triples, including eight CS training data sets (1k each) and a context-convolute (CC) dataset (22k). We are in the process of making this CoSIS dataset available in the public domain.

In future work, we will further explore the paradigm of constructing ensemble models using CS models, which in many ways bears some similarity to human learning.

Chapter 6

Conclusions

As training techniques for deep learning and ML models are advancing rapidly, testing for ML models has been largely overlooked by the community. To fill in this gap, this thesis set out to investigate a more systematic and advanced framework to evaluate invariance qualities of machine learning (ML) models, focusing on both simple / traditional variables (e.g., rotation- and brightness invariance qualities of ML models) and more complex variables, e.g., background invariance qualities. In this thesis, several key contributions (C_0 to C_3) were made towards enhancing invariance testing for ML models, together with the following major findings, reflections on the research journey, and discussions on the broader implications of invariance testing for both academic research and industrial deployment of ML models. Below, we summarize the key successes and limitations of each contribution:

- **C_0 The First Comprehensive Survey on Invariance Testing:** In the literature, many prior works were directly and strongly related to invariance testing whilst they were named differently, e.g., “resilient in hidden factors”. In our Chapter 2, we summarised these studies and grouped them by three types of testing methodology, i.e., formula-based, visualization-based, and theory-based. In addition, we also included existing works that could be useful for invariance testing, e.g., controls of input permutations and existing datasets that contained different types of variations. This comprehensive survey was the first survey on invariance testing, which included related works that were named differently. This could facilitate future studies in invariance testing. Through this literature research, the concept of invariance testing has been extended in several aspects:

- bringing a number of ML testing problems, e.g., robustness against affine transformations, adversarial attacks, and out-of-distribution testing, into a common framework, i.e., invariance testing.
 - allowing visualization and ML techniques to be considered as testing methods in addition to statistical measures.
 - grouping and categorising different existing testing methods into three major testing frameworks, e.g., metric-based statistical testing, visualization-based testing, and theory-guided testing.
- **C_1 The First Automatic Invariance Testing Framework based on Visualization:** We re-framed the problem of invariance testing from a unique and novel perspective: using visualised testing patterns to demonstrate detailed performance of ML models. In Chapter 3, we focused on simple data attributes (e.g., rotation), because of the simplicity of the control of the permutations (e.g., based on how many degrees the images were rotated). For these simple and traditional variables, we introduced a generic invariance testing framework for the first time that was:
 - ***Informative***: our testing framework converted judgements on invariance qualities of ML models from comparing statistical measures to analyzing detailed visualized performance for the first time. And we showed complex visual patterns can provide deeper insights on the invariance qualities of ML models.
 - ***Consistent and Reliable***: model decisions based on our testing framework were significantly more consistent than manual examination.
 - ***Automatic***: whilst the professional analysis based on visualized patterns demanded extensive manual labour especially for a large number of interested ML models, our framework resolved the problem by introducing ML4ML assessors, which automated the entire testing procedure.
 - ***Efficient and Cost-saving***: As manual examination of visualized testing patterns is not yet a common practice, the cost for training experts to perform such a task would not be small. The ML4ML approach allows us to reduce such training needs and cost significantly.

Limitation: Because of resource constraints of a DPhil project, our framework was only evaluated using a few simple transformations (i.e., rotation, brightness

and size), and only one type of complex variation (i.e., background). Furthermore, automation through ML-based assessors requires more extensive validation across different ML model architectures and datasets.

- **C_2 Significant Improvement on Invariance Testing for Complex Variables:** In Chapter 4, we added extra technical components to extend our invariance testing framework to a more complex variable: background invariance testing. Due to the gigantic data space for background invariance testing and limited computational resources, one of the biggest challenges was to select a suitable testing suite that was diverse, yet related and connected to the original context. We showed that our chosen testing images could trigger a high neuron coverage rate (which indicates diversity) and the resultant visualised testing patterns lead to a consistent and reliable decision and judgement on the invariance quality. The experimental results showed that our testing framework could be extended to more complex variables with the assistance of extra technical components, e.g., selection of testing data. Through this extensive research, the concept of invariance testing has been extended in several aspects:

- Our invariance testing was not limited to simple variables and it could be tailored and applied to complex variables, e.g., background, by introducing additional technical components, e.g., association ontology.
- When it was less straightforward to control the variations, e.g., for complex variables, we identified two major metrics to evaluate the plausibility of the tests for the first time, i.e., neuron coverage (comprehensiveness) and inter-rater reliability (consistency).

Limitation: The proposed method involves different technical components, and relies heavily on the ordering of testing data. The metrics that are used to evaluate the testing framework can also be further improve, e.g., avoid using neuron coverage to suggest diversity of the selected background candidates.

- **C_3 Significantly Improved the Ordering of Invariance Testing Data:** Finally, we introduced a novel approach to order the testing data using an ensemble of context-sensitive models. We found that directly and globally fine-tuning a deep model could not improve global performance. By contrast, via invariance testing using visualised patterns, we found that, although global improvement could not be straightforwardly achieved, locally fine-tuning models (or context-sensitive models) could not improve local performance. We then

introduced our ensemble strategies to utilise the local advantages of these locally fine-tuned context-sensitive models, which led to global improvement. In Chapter 5, we introduced a novel approach to order invariance testing data, and we showed that visualised invariance testing could lead to new findings and new strategies to improve ML systems’ performance.

Limitation: The proposed method targets sparse data, and therefore the MoE mechanism remains simple. The selection of context-sensitive models involves heuristic elements that may require further refinement.

Future Work

To expand the regime of invariance testing, some hard problems in invariance testing can be considered in the future, for example, 3D object rotation within the same scene / similar scenes, deformable objects, or weather conditions. As generative techniques are advancing rapidly in the recent years, we expect invariance testing could become maturer, and more complex variables can be taken into consideration. To control the variations in input data, in addition to generative models, for each subarea of interest, researchers might need to consider tailored additional technical components. For example, we proposed using association ontology to improve the selection of background invariance testing data, which made the tests more comprehensive and more reliable. Another potential future direction is realness invariance testing. Whilst we can generate realistic testing images using advanced generative models, we still cannot easily score the realness of the generated images using a well-defined measure. Once this problem is considered solved, it will open a new research direction on realness invariance testing. Additionally, ML models can be re-trained with augmented background images. One hypothesis for the re-trained models is: (a) the models might learn whether the background and the foreground agree with each other for the classification task, and (b) when foreground and background suggest different predictions, the models might prioritize the decision made from foreground solely. It could be insightful to dive into such a hypothesis and to verify it using suitable techniques, e.g., formal methods or visualization.

Additionally, for the generic invariance testing framework, the analysis and conclusions are drawn based on the experiments on our model repository of 600 ML models that were trained on a relatively small dataset (CIFAR or MNIST). A larger repository of models that are trained on a larger dataset (e.g., ImageNet) can be collected to further confirm the feasibility of the framework. The scalability should

also be thoroughly examined; when both the size of model repository and the size of dataset (used for training ML models) increase, the computational costs for testing can increase heavily. To overcome the increasing need for computational resources, an advanced sampling technique (similar to the case of background invariance testing) can be developed.

For the context-sensitive learning that we used to improve the ordering of invariance testing data, in the future, to further improve the problem of semantic gaps, we suggest to add more contexts. We also suggest to thoroughly study the selection of contexts. When the amount of available (labelled) data increases, the problem of semantic image similarity can be approached and addressed by many more different techniques, e.g., self-supervised, semi-supervised learning, or more advanced techniques in fine-tuning. We look forward to seeing large ensemble vision models using context-sensitive learning in the future. Interested researchers might also find the need to understand the shared knowledge among different local context-sensitive models for better accountability and interpretability of the vision models.

To conclude, in this thesis, we summarised directly and indirectly related works in the literature, utilised visualised patterns to make invariance testing more informative, introduced an automated invariance testing framework, extended our testing framework to more complex variables (background), and studied the selection and ordering of testing data. We found that visualisation-based invariance testing was consistent and reliable, and it could lead to actionable items to further improve ML models' performance. We hope in the future, the ML community will focus more on testing ML models, including invariance testing. And we look forward to seeing more invariance testing approaches for different complex variables using our proposed visualization-based framework.

Acknowledgement

I would like to express my deepest gratitude to my PhD supervisor, Professor Min Chen, for his patient guidance and encouragement throughout my doctoral journey. Your expertise, insightful feedback, and dedication have been invaluable in shaping his dissertation. I am deeply thankful for my friend and ex-colleague at Tencent Video, Pengfei Zhang. Thank you for supporting my research ideas and letting me run all the experiments on your personal GPUs. My sincere appreciation goes to Michael Cheung (at Oracle) for helping me run experiments for our project on your personal cloud service. I am also grateful for my family's unconditional support.

Bibliography

- [1] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- [2] Yossi Adi, Carsten Baum, Moustapha Cisse, Benny Pinkas, and Joseph Keshet. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1615–1631, 2018.
- [3] Namita Agarwal, Amit Kumar Singh, and Pradeep Kumar Singh. Survey of robust and imperceptible watermarking. *Multimedia Tools and Applications*, 78(7):8603–8633, 2019.
- [4] Vedika Agarwal, Rakshith Shetty, and Mario Fritz. Towards causal vqa: Revealing and reducing spurious correlations by invariant and covariant semantic editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9690–9698, 2020.
- [5] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499. Citeseer, 1994.
- [6] Shamima Ahmed, Muneer M Alshater, Anis El Ammari, and Helmi Hammami. Artificial intelligence and machine learning in finance: A bibliometric review. *Research in International Business and Finance*, 61:101646, 2022.
- [7] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.
- [8] Naveed Akhtar and Ajmal Mian. Threat of adversarial attacks on deep learning in computer vision: A survey. *IEEE Access*, 6:14410–14430, 2018.

- [9] Ignacio Algreto-Badillo, Brenda Sánchez-Juárez, Kelsey A Ramírez-Gutiérrez, Claudia Feregrino-Urbe, Francisco López-Huerta, and Johan J Estrada-López. Analysis and hardware architecture on fpga of a robust audio fingerprinting method using ssm. *Technologies*, 10(4):86, 2022.
- [10] Eid M Alhaisoni, Khalid M Al-Zuoud, and Daya Ram Gaudel. Analysis of spelling errors of saudi beginner learners of english enrolled in an intensive english language program. *English Language Teaching*, 8(3):185–192, 2015.
- [11] Anonymised Authors. Anonymised title. In *Proc. Anonymised Conference Name*, 2022. The paper is included in the supplementary materials.
- [12] Saeed Anwar, Salman Khan, and Nick Barnes. A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.
- [13] Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv:1907.02893*, 1, 2019.
- [14] Omri Avrahami, Ohad Fried, and Dani Lischinski. Blended latent diffusion. *ACM Trans. Graph.*, 42(4), jul 2023.
- [15] Nicolas Azierie and Sinisa Todorovic. Ensemble deep manifold similarity learning using hard proxies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7299–7307, 2019.
- [16] Aharon Azulay and Yair Weiss. Why do deep convolutional networks generalize so poorly to small image transformations? *arXiv preprint arXiv:1805.12177*, 2018.
- [17] Tao Bai, Jinqi Luo, Jun Zhao, Bihan Wen, and Qian Wang. Recent advances in adversarial training for adversarial robustness, 2021.
- [18] Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443, 2018.
- [19] Ajay Bandi, Pydi Venkata Satya Ramesh Adapa, and Yudu Eswar Vinay Pratap Kumar Kuchi. The power of generative ai: A review of requirements, models, input–output formats, evaluation metrics, and challenges. *Future Internet*, 15(8):260, 2023.

- [20] E. Begelfor and M. Werman. Affine invariance revisited. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 2087–2094, 2006.
- [21] Ronald N Bracewell. The fourier transform. *Scientific American*, 260(6):86–95, 1989.
- [22] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.
- [23] Antonio Brunetti, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey. *Neurocomputing*, 300:17–33, 2018.
- [24] Peter Bühlmann. Invariance, causality and robustness. *Statistical Science*, 35(3):404–426, 2020.
- [25] Saikiran Bulusu, Bhavya Kailkhura, Bo Li, Pramod K Varshney, and Dawn Song. Anomalous example detection in deep learning: A survey. *IEEE Access*, 8:132330–132347, 2020.
- [26] Vladimir Bychkovsky, Sylvain Paris, Eric Chan, and Frédo Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *The Twenty-Fourth IEEE Conference on Computer Vision and Pattern Recognition*, 2011.
- [27] Taejoon Byun, Vaibhav Sharma, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. Input prioritization for testing neural networks. In *2019 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 63–70. IEEE, 2019.
- [28] Peter Bühlmann. Invariance, causality and robustness, 2018.
- [29] Adnane Cabani, Karim Hammoudi, Halim Benhabiles, and Mahmoud Melkemi. Maskedface-net—a dataset of correctly/incorrectly masked face images in the context of covid-19. *Smart Health*, 19:100144, 2021.
- [30] Alexander Camuto, Matthew Willetts, Umut Simsekli, Stephen J Roberts, and Chris C Holmes. Explicit regularisation in gaussian noise injections. *Advances in neural information processing systems*, 33:16603–16614, 2020.

- [31] Qiong Cao, Li Shen, Weidi Xie, Omkar M Parkhi, and Andrew Zisserman. Vggface2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face & gesture recognition (FG 2018)*, pages 67–74. IEEE, 2018.
- [32] Marc-André Carboneau, Julian Zaidi, Jonathan Boilard, and Ghyslain Gagnon. Measuring disentanglement: A review of metrics. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.
- [33] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2017.
- [34] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.
- [35] Gaffari Celik. Covidcoughnet: A new method based on convolutional neural networks and deep feature extraction using pitch-shifting data augmentation for covid-19 detection from cough, breath, and voice signals. *Computers in Biology and Medicine*, page 107153, 2023.
- [36] Mayank Arya Chandra and SS Bedi. Survey on svm and their application in image classification. *International Journal of Information Technology*, 13(5):1–11, 2021.
- [37] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.
- [38] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. Zoo. *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, Nov 2017.
- [39] Shuxiao Chen, Edgar Dobriban, and Jane H Lee. A group-theoretic framework for data augmentation. *The Journal of Machine Learning Research*, 21(1):9885–9955, 2020.
- [40] Xinyun Chen, Chang Liu, Bo Li, Kimberly Lu, and Dawn Song. Targeted backdoor attacks on deep learning systems using data poisoning. *arXiv preprint arXiv:1712.05526*, 2017.

- [41] Xinyun Chen, Wenxiao Wang, Chris Bender, Yiming Ding, Ruoxi Jia, Bo Li, and Dawn Song. Refit: a unified watermark removal framework for deep learning systems with limited data. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 321–335, 2021.
- [42] Yun-Chun Chen, Yu-Jhe Li, Xiaofei Du, and Yu-Chiang Frank Wang. Learning resolution-invariant deep representations for person re-identification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 8215–8222, 2019.
- [43] Danni Cheng, Xiang Li, Wei-Hong Li, Chan Lu, Fake Li, Hua Zhao, and Wei-Shi Zheng. Large-scale visible watermark detection and removal with deep convolutional networks. In *Pattern Recognition and Computer Vision: First Chinese Conference, PRCV 2018, Guangzhou, China, November 23-26, 2018, Proceedings, Part III 1*, pages 27–40. Springer, 2018.
- [44] Gong Cheng, Junwei Han, Peicheng Zhou, and Dong Xu. Learning rotation-invariant and fisher discriminative convolutional neural networks for object detection. *IEEE Transactions on Image Processing*, 28(1):265–278, 2018.
- [45] Yu Cheng, Bo Yang, Bo Wang, and Robby T Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10631–10638, 2020.
- [46] Cheng Chi, Shifeng Zhang, Junliang Xing, Zhen Lei, Stan Z Li, and Xudong Zou. Pedhunter: Occlusion robust pedestrian detector in crowded scenes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10639–10646, 2020.
- [47] Benjamin Chidester, Minh N Do, and Jian Ma. Rotation equivariance and invariance in convolutional neural networks. *arXiv preprint arXiv:1805.12301*, 2018.
- [48] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Convolutional recurrent neural networks for music classification. In *2017 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pages 2392–2396. IEEE, 2017.

- [49] Chang-Min Chou and Din-Chang Tseng. Affine-transformation-invariant public fragile watermarking for 3d model authentication. *IEEE Computer Graphics and Applications*, 29(2):72–79, 2009.
- [50] Rudi L Cilibrasi and Paul MB Vitanyi. The google similarity distance. *IEEE Transactions on knowledge and data engineering*, 19(3):370–383, 2007.
- [51] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.
- [52] Gordon V Cormack and Aleksander Kolcz. Spam filter evaluation with imprecise ground truth. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 604–611, 2009.
- [53] Yandre MG Costa, Luiz S Oliveira, and Carlos N Silla Jr. An evaluation of convolutional neural networks for music classification using spectrograms. *Applied soft computing*, 52:28–38, 2017.
- [54] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3412–3420, 2019.
- [55] Elliot Creager, Jörn-Henrik Jacobsen, and Richard Zemel. Environment inference for invariant learning. In *International Conference on Machine Learning*, pages 2189–2200. PMLR, 2021.
- [56] Duc-Tien Dang-Nguyen, Cecilia Pasquini, Valentina Conotter, and Giulia Boato. Raise: A raw images dataset for digital image forensics. In *Proceedings of the 6th ACM multimedia systems conference*, pages 219–224, 2015.
- [57] Tri Dao, Albert Gu, Alexander Ratner, Virginia Smith, Chris De Sa, and Christopher Ré. A kernel theory of modern data augmentation. In *International Conference on Machine Learning*, pages 1528–1537. PMLR, 2019.
- [58] Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.
- [59] Rodrigo Caye Daudt, Adrien Chan-Hon-Tong, Bertrand Le Saux, and Alexandre Boulch. Learning to understand earth observation images with weak and unreliable ground truth. In *IGARSS IEEE International Geoscience and Remote Sensing Symposium*, pages 5602–5605. IEEE, 2019.

- [60] Jodi L Davenport and Mary C Potter. Scene consistency in object and background perception. *Psychological science*, 15(8):559–564, 2004.
- [61] Jodi L Davenport and Mary C Potter. Scene consistency in object and background perception. *Psychological science*, 15(8):559–564, 2004.
- [62] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014.
- [63] Thomas Deselaers and Vittorio Ferrari. Visual and semantic similarity in imagenet. In *CVPR 2011*, pages 1777–1784. IEEE, 2011.
- [64] Antonio D’Innocente, Nikhil Garg, Yuan Zhang, Loris Bazzani, and Michael Donoser. Localized triplet loss for fine-grained fashion image retrieval. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3910–3915, 2021.
- [65] Robert Dorfman. A formula for the gini coefficient. *The review of economics and statistics*, pages 146–149, 1979.
- [66] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, pages 0210–0215. IEEE, 2018.
- [67] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 1, 2020.
- [68] Anurag Dwarakanath, Manish Ahuja, Samarth Sikand, Raghotham M Rao, RP Jagadeesh Chandra Bose, Neville Dubash, and Sanjay Podder. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 118–128, 2018.
- [69] Ville-Veikko Eklund. Data augmentation techniques for robust audio analysis. Master’s thesis, 2019.

- [70] N El Dehaibi and EF MacDonald. Investigating inter-rater reliability of qualitative text annotations in machine learning datasets. In *Proceedings of the Design Society: DESIGN Conference*, volume 1, pages 21–30. Cambridge University Press, 2020.
- [71] N El Dehaibi and EF MacDonald. Investigating inter-rater reliability of qualitative text annotations in machine learning datasets. In *Proceedings of the Design Society: DESIGN Conference*, volume 1, pages 21–30. Cambridge University Press, 2020.
- [72] Gamaleldin F. Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. Adversarial reprogramming of neural networks, 2018.
- [73] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Alexander Madry. Exploring the landscape of spatial robustness. In *International conference on machine learning*, pages 1802–1811. PMLR, 2019.
- [74] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Alexander Madry. Exploring the landscape of spatial robustness. In *International Conference on Machine Learning*, pages 1802–1811. PMLR, 2019.
- [75] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111:98–136, 2015.
- [76] Hui Fang, Gary Kwok-Leung Tam, Rita Borgo, Andrew J Aubrey, Philip W Grant, Paul L Rosin, Christian Wallraven, Douglas Cunningham, David Marshall, and Min Chen. Visualizing natural image statistics. *IEEE Transactions on Visualization and Computer Graphics*, 19(7):1228–1241, 2012.
- [77] Ronald Aylmer Fisher. *Statistical methods for research workers*. Springer, 1992.
- [78] Peter Flach and Meelis Kull. Precision-recall-gain curves: Pr analysis done right. *Advances in neural information processing systems*, 28, 2015.
- [79] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971.
- [80] Jonathan T Foote and Matthew L Cooper. Media segmentation using self-similarity decomposition. In *Storage and Retrieval for Media Databases 2003*, volume 5021, pages 167–175. SPIE, 2003.

- [81] Mohammad Mahdi Forootan, Iman Larki, Rahim Zahedi, and Abolfazl Ahmadi. Machine learning and deep learning in energy systems: A review. *Sustainability*, 14(8):4832, 2022.
- [82] Valentina Franzoni, Alfredo Milani, Simonetta Pallottelli, Clement HC Leung, and Yuanxi Li. Context-based image semantic similarity. In *2015 12th international conference on fuzzy systems and knowledge discovery (fskd)*, pages 1280–1284. IEEE, 2015.
- [83] Benoît Frénay, Ata Kabán, et al. A comprehensive introduction to label noise. In *ESANN*. Citeseer, 2014.
- [84] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2013.
- [85] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *arXiv preprint arXiv:2306.09344*, 2023.
- [86] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.
- [87] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1147–1158. IEEE, 2020.
- [88] Xiang Gao, Ripon K Saha, Mukul R Prasad, and Abhik Roychoudhury. Fuzz testing based data augmentation to improve robustness of deep neural networks. In *IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*, pages 1147–1158. IEEE, 2020.
- [89] Xifeng Gao, Caiming Zhang, Yan Huang, and Zhigang Deng. A robust high-capacity affine-transformation-invariant scheme for watermarking 3d geometric models. *ACM Trans. Multimedia Comput. Commun. Appl.*, 8(2S), sep 2012.

- [90] Sourav Garg, Niko Suenderhauf, and Michael Milford. Don't look back: Robustifying place categorization for viewpoint-and condition-invariant place recognition. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3645–3652. IEEE, 2018.
- [91] Sourav Garg, Niko Suenderhauf, and Michael Milford. Lost? appearance-invariant place recognition for opposite viewpoints using visual semantics. *arXiv preprint arXiv:1804.05526*, 2018.
- [92] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021.
- [93] Timon Gehr, Matthew Mirman, Dana Drachler-Cohen, Petar Tsankov, Swarat Chaudhuri, and Martin Vechev. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE, 2018.
- [94] Athinodoros S. Georghiades, Peter N. Belhumeur, and David J. Kriegman. From few to many: Illumination cone models for face recognition under variable lighting and pose. *IEEE transactions on pattern analysis and machine intelligence*, 23(6):643–660, 2001.
- [95] Juliana Freitas Santos Gomes and Fabiana Rodrigues Leta. Applications of computer vision techniques in the agriculture and food industry: a review. *European Food Research and Technology*, 235(6):989–1000, 2012.
- [96] Ian Goodfellow, Honglak Lee, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. *Advances in neural information processing systems*, 22, 2009.
- [97] Ian Goodfellow, Honglak Lee, Quoc Le, Andrew Saxe, and Andrew Ng. Measuring invariances in deep networks. *Advances in neural information processing systems*, 22:646–654, 2009.
- [98] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

- [99] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.
- [100] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.
- [101] Divya Gopinath, Kaiyuan Wang, Mengshi Zhang, Corina S Pasareanu, and Sarfraz Khurshid. Symbolic execution for deep neural networks. *arXiv preprint arXiv:1807.10439*, 2018.
- [102] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *European conference on information retrieval*, pages 345–359. Springer, 2005.
- [103] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering*, 17(10):1347–1362, 2005.
- [104] Jason M Grant and Patrick J Flynn. Crowd scene understanding from video: a survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 13(2):1–23, 2017.
- [105] Douglas Gray, Shane Brennan, and Hai Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *Proc. IEEE international workshop on performance evaluation for tracking and surveillance (PETS)*, volume 3, pages 1–7, 2007.
- [106] Robert C Green, Jonathan S Berg, Gerard T Berry, Leslie G Biesecker, David P Dimmock, James P Evans, Wayne W Grody, Madhuri R Hegde, Sarah Kalia, Bruce R Korf, et al. Exploring concordance and discordance for return of incidental findings from clinical sequencing. *Genetics in Medicine*, 14(4):405–410, 2012.
- [107] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples, 2015.
- [108] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv preprint arXiv:1708.06733*, 2017.

- [109] Riccardo Guidotti. Counterfactual explanations and how to find them: literature review and benchmarking. *Data Mining and Knowledge Discovery*, pages 1–55, 2022.
- [110] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of wasserstein gans. *Advances in neural information processing systems*, 30, 2017.
- [111] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 739–743, 2018.
- [112] Jianmin Guo, Yu Jiang, Yue Zhao, Quan Chen, and Jiaguang Sun. Dlfuzz: differential fuzzing testing of deep learning systems. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 739–743, 2018.
- [113] Masato Hagiwara and Masato Mita. Github typo corpus: A large-scale multilingual dataset of misspellings and grammatical errors. *arXiv preprint arXiv:1911.12893*, 2019.
- [114] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on visual transformer. *arXiv preprint arXiv:2012.12556*, 2(4), 2020.
- [115] Albert Haque, Boya Peng, Zelun Luo, Alexandre Alahi, Serena Yeung, and Li Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*, pages 160–177. Springer, 2016.
- [116] Fabrice Harel-Canada, Lingxiao Wang, Muhammad Ali Gulzar, Quanquan Gu, and Miryung Kim. Is neuron coverage a meaningful measure for testing deep neural networks? In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 851–862, 2020.
- [117] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

- [118] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021.
- [119] Arthur Hinsvark, Natalie Delworth, Miguel Del Rio, Quinten McNamara, Joshua Dong, Ryan Westerman, Michelle Huang, Joseph Palakapilly, Jennifer Drexler, Ilya Pirkin, et al. Accented speech recognition: A survey. *arXiv preprint arXiv:2104.10747*, 2021.
- [120] Joanna Hong, Minsu Kim, Jeongsoo Choi, and Yong Man Ro. Watch or listen: Robust audio-visual speech recognition with visual corruption modeling and reliability scoring. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18783–18794, 2023.
- [121] Md Altab Hossain, Yasushi Makihara, Junqiu Wang, and Yasushi Yagi. Clothing-invariant gait identification using part-based clothing categorization and adaptive weight control. *Pattern Recognition*, 43(6):2281–2291, 2010.
- [122] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [123] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinpeng Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020.
- [124] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 3277–3285, 2017.
- [125] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

- [126] Nasif Imtiaz, Justin Middleton, Peter Girouard, and Emerson Murphy-Hill. Sentiment and politeness analysis tools on developer discussions are unreliable, but so are people. In *IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, pages 55–61. IEEE, 2018.
- [127] Muhammad Mobaidul Islam, Abdullah Al Redwan Newaz, Balakrishna Gokaraju, and Ali Karimoddini. Pedestrian detection for autonomous cars: Occlusion handling by classifying body parts. In *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1433–1438. IEEE, 2020.
- [128] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [129] Jörn-Henrik Jacobsen, Jens Behrmann, Nicholas Carlini, Florian Tramèr, and Nicolas Papernot. Exploiting excessive invariance caused by norm-bounded adversarial robustness, 2019.
- [130] Arjit Jain, Pranay Reddy Samala, Deepak Mittal, Preethi Jyoti, and Maneesh Singh. Spliceout: A simple and efficient audio augmentation method. *arXiv preprint arXiv:2110.00046*, 2021.
- [131] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.
- [132] Sai Muralidhar Jayanthi, Danish Pruthi, and Graham Neubig. Neuspell: A neural spelling correction toolkit. *arXiv preprint arXiv:2010.11085*, 2020.
- [133] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020.
- [134] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review. *IEEE transactions on visualization and computer graphics*, 26(11):3365–3385, 2019.
- [135] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26:315–323, 2013.

- [136] Aranzazu Jurio, Miguel Pagola, Mikel Galar, Carlos Lopez-Molina, and Daniel Paternain. A comparison study of different color spaces in clustering based image segmentation. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications: 13th International Conference, IPMU 2010, Dortmund, Germany, June 28–July 2, 2010. Proceedings, Part II 13*, pages 532–541. Springer, 2010.
- [137] Sandesh Kamath, Amit Deshpande, and KV Subrahmanyam. Invariance vs robustness of neural networks. 2020.
- [138] Andreas Kamilaris and Francesc X Prenafeta-Boldú. Deep learning in agriculture: A survey. *Computers and electronics in agriculture*, 147:70–90, 2018.
- [139] Kamel Karoui, Fakher Ben Ftima, and Henda Ben Ghezala. Formal specification, verification and correction of security policies based on the decision tree approach. *International Journal of Data & Network Security*, 3(3):92–111, 2013.
- [140] Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. Wroning a right: Generating better errors to improve grammatical error detection. *arXiv preprint arXiv:1810.00668*, 2018.
- [141] Mahmut Kaya and Hasan Şakir Bilge. Deep metric learning: A survey. *Symmetry*, 11(9):1066, 2019.
- [142] Mourad A Kenk and Mahmoud Hassaballah. Dawn: vehicle detection in adverse weather nature dataset. *arXiv preprint arXiv:2008.05402*, 2020.
- [143] Syed Safwan Khalid, Muhammad Awais, Zhen-Hua Feng, Chi-Ho Chan, Ammarah Farooq, Ali Akbari, and Josef Kittler. Resolution invariant face recognition using a distillation approach. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 2(4):410–420, 2020.
- [144] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.
- [145] Jinhan Kim, Robert Feldt, and Shin Yoo. Guiding deep learning system testing using surprise adequacy. In *IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1039–1049. IEEE, 2019.

- [146] Taewoo Kim, Chaeyeon Chung, Sunghyun Park, Gyojung Gu, Keonmin Nam, Wonzo Choe, Jaesung Lee, and Jaegul Choo. K-hairstyle: A large-scale korean hairstyle dataset for virtual hair editing and hairstyle classification. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1299–1303. IEEE, 2021.
- [147] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth annual conference of the international speech communication association*, 2015.
- [148] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. Audio augmentation for speech recognition. In *Sixteenth annual conference of the international speech communication association*, 2015.
- [149] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.
- [150] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A Shamma, et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, 123:32–73, 2017.
- [151] Anoop Krishnan, Ali Almadan, and Ajita Rattani. Understanding fairness of gender classification algorithms across gender-race groups. In *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1028–1035. IEEE, 2020.
- [152] Brian Kulis et al. Metric learning: A survey. *Foundations and trends in machine learning*, 5(4):287–364, 2012.
- [153] Ashnil Kumar, Jinman Kim, Weidong Cai, Michael Fulham, and Dagan Feng. Content-based medical image retrieval: a survey of applications to multidimensional and multimodality data. *Journal of digital imaging*, 26(6):1025–1039, 2013.
- [154] Hyuk-Ju Kwon and Sung-Hak Lee. A two-step learning model for the diagnosis of coronavirus disease-19 based on chest x-ray images with 3d rotational augmentation. *Applied Sciences*, 12(17):8668, 2022.

- [155] Madan Lal, Kamlesh Kumar, Rafaqat Hussain Arain, Abdullah Maitlo, Sadaquat Ali Ruk, and Hidayatullah Shaikh. Study of face recognition techniques: A survey. *International Journal of Advanced Computer Science and Applications*, 9(6), 2018.
- [156] Tim Lauer, Tim HW Cornelissen, Dejan Draschkow, Verena Willenbockel, and Melissa L-H Vö. The role of scene summary statistics in object recognition. *Scientific reports*, 8(1):1–12, 2018.
- [157] Tim Lauer, Tim HW Cornelissen, Dejan Draschkow, Verena Willenbockel, and Melissa L-H Vö. The role of scene summary statistics in object recognition. *Scientific reports*, 8(1):1–12, 2018.
- [158] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. Adversarial frontier stitching for remote neural network watermarking. *Neural Computing and Applications*, 32(13):9233–9244, 2020.
- [159] Seongwon Lee, Hongje Seong, Suhyeon Lee, and Euntai Kim. Correlation verification for image retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5374–5384, June 2022.
- [160] Jizhizi Li, Jing Zhang, Stephen J Maybank, and Dacheng Tao. Bridging composite and real: towards end-to-end deep image matting. *International Journal of Computer Vision*, 130:246–266, 2022.
- [161] Jun Li, Bodong Zhao, and Chao Zhang. Fuzzing: a survey. *Cybersecurity*, 1(1):1–13, 2018.
- [162] Wei Li, Rui Zhao, Tong Xiao, and Xiaogang Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 152–159, 2014.
- [163] Yao Li, Minhao Cheng, Cho-Jui Hsieh, and Thomas C. M. Lee. A review of adversarial attack and defense for classification methods. *CoRR*, abs/2111.09961, 2021.
- [164] Zenan Li, Xiaoxing Ma, Chang Xu, and Chun Cao. Structural coverage criteria for neural networks could be misleading. In *IEEE/ACM 41st International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)*, pages 89–92. IEEE, 2019.

- [165] Zukang Liao. ML4ML invariance testing. <https://github.com/Zukang-Liao/ML4ML-invariance-testing>.
- [166] Zukang Liao. Simultaneous adversarial training-learn from others' mistakes. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–7. IEEE, 2019.
- [167] Zukang Liao. Simultaneous adversarial training-learn from others' mistakes. In *14th IEEE International Conference on Automatic Face & Gesture Recognition*, pages 1–7. IEEE, 2019.
- [168] Zukang Liao and Min Chen. Image similarity using an ensemble of context-sensitive models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1758–1769, 2024.
- [169] Zukang Liao and Min Chen. Background invariance testing according to semantic proximity. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2025.
- [170] Zukang Liao and Michael Cheung. Automated invariance testing for machine learning models using sparse linear layers. In *ICML 2022: Workshop on Spurious Correlations, Invariance and Stability*, 2022.
- [171] Zukang Liao and Michael Cheung. Invariance testing and feature selection using sparse linear layers. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 4219–4223, 2022.
- [172] Zukang Liao, Stavros Petridis, and Maja Pantic. Local deep neural networks for age and gender classification. *arXiv preprint arXiv:1703.08497*, 2017.
- [173] Zukang Liao, Pengfei Zhang, and Min Chen. Ml4ml: Automated invariance testing for machine learning models. In *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 34–41. IEEE, 2022.
- [174] Yen Khye Lim, Zukang Liao, Stavros Petridis, and Maja Pantic. Transfer learning for action unit recognition. *arXiv:1807.07556*, 2018.
- [175] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. *arXiv preprint arXiv:1908.06281*, 2019.

- [176] Shanchuan Lin, Linjie Yang, Imran Saleemi, and Soumyadip Sengupta. Robust high-resolution video matting with temporal guidance. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 238–247, 2022.
- [177] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.
- [178] Huan Ling, Karsten Kreis, Daiqing Li, Seung Wook Kim, Antonio Torralba, and Sanja Fidler. Editgan: High-precision semantic image editing. *Advances in Neural Information Processing Systems*, 34:16331–16345, 2021.
- [179] Guiqing Liu and Jinzhao Wu. Unsupervised person re-identification by intra-inter camera affinity domain adaptation. *Journal of Visual Communication and Image Representation*, 80:103310, 2021.
- [180] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.
- [181] Renting Liu, Zhaorong Li, and Jiaya Jia. Image partial blur detection and classification. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–8. IEEE, 2008.
- [182] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.
- [183] Yang Liu, Zhen Zhu, and Xiang Bai. Wdnet: Watermark-decomposition network for visible watermark removal. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3685–3693, 2021.
- [184] Yunfan Liu, Qi Li, and Zhenan Sun. Attribute-aware face aging with wavelet-based generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11877–11886, 2019.

- [185] Stephanie Lowry and Henrik Andreasson. Lightweight, viewpoint-invariant visual place recognition in changing environments. *IEEE Robotics and Automation Letters*, 3(2):957–964, 2018.
- [186] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 4765–4774. Curran Associates, Inc., 2017.
- [187] Jiayi Ma, Xingyu Jiang, Aoxiang Fan, Junjun Jiang, and Junchi Yan. Image matching from handcrafted to deep features: A survey. *International Journal of Computer Vision*, 129:23–79, 2021.
- [188] Lei Ma, Felix Juefei-Xu, Minhui Xue, Bo Li, Li Li, Yang Liu, and Jianjun Zhao. Deepct: Tomographic combinatorial testing for deep learning systems. In *IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 614–618. IEEE, 2019.
- [189] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.
- [190] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.
- [191] Lei Ma, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Felix Juefei-Xu, Chao Xie, Li Li, Yang Liu, Jianjun Zhao, et al. Deepmutation: Mutation testing of deep learning systems. In *IEEE 29th International Symposium on Software Reliability Engineering (ISSRE)*, pages 100–111. IEEE, 2018.
- [192] James MacLean and John Tsotsos. Fast pattern recognition using gradient-descent search in an image pyramid. In *Proceedings 15th International Conference on Pattern Recognition. ICPR-2000*, volume 2, pages 873–877. IEEE, 2000.

- [193] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 year, 1000 km: The oxford robotcar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.
- [194] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2019.
- [195] Shunan Mao, Shiliang Zhang, and Ming Yang. Resolution-invariant person re-identification. *arXiv preprint arXiv:1906.09748*, 2019.
- [196] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2012–2017. IEEE, 2016.
- [197] Frank J Massey Jr. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46(253):68–78, 1951.
- [198] Brian McFee and Dan Ellis. Analyzing song structure with spectral clustering. In *ISMIR*, pages 405–410. Citeseer, 2014.
- [199] Brian McFee, Eric J Humphrey, and Juan Pablo Bello. A software framework for musical data augmentation. In *ISMIR*, volume 2015, pages 248–254. Citeseer, 2015.
- [200] Brian McFee, Oriol Nieto, Morwaread M Farbood, and Juan Pablo Bello. Evaluating hierarchical structure in music annotations. *Frontiers in psychology*, 8:1337, 2017.
- [201] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35, 2021.
- [202] Mark Huasong Meng, Guangdong Bai, Sin Gee Teo, Zhe Hou, Yan Xiao, Yun Lin, and Jin Song Dong. Adversarial robustness of deep neural networks: A survey from a formal verification perspective. *IEEE Transactions on Dependable and Secure Computing*, 2022.
- [203] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

- [204] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3523–3542, 2021.
- [205] Shervin Minaee, Yuri Y Boykov, Fatih Porikli, Antonio J Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.
- [206] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.
- [207] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deep-fool: a simple and accurate method to fool deep neural networks, 2016.
- [208] Yair Movshovitz-Attias, Alexander Toshev, Thomas K Leung, Sergey Ioffe, and Saurabh Singh. No fuss distance metric learning using proxies. In *Proceedings of the IEEE international conference on computer vision*, pages 360–368, 2017.
- [209] Alhassan Mumuni and Fuseini Mumuni. Cnn architectures for geometric transformation-invariant feature representation in computer vision: a review. *SN Computer Science*, 2:1–23, 2021.
- [210] Joseph L. Mundy, Andrew Zisserman, and David Forsyth, editors. *Proc. Second Joint European-US Workshop on Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*. Springer, 1993.
- [211] Lukas Murmann, Michael Gharbi, Miika Aittala, and Fredo Durand. A dataset of multi-illumination images in the wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4080–4089, 2019.
- [212] Ghulam Murtaza, Liyana Shuib, Ainuddin Wahid Abdul Wahab, Ghulam Mujtaba, Henry Friday Nweke, Mohammed Ali Al-garadi, Fariha Zulfiqar, Ghulam Raza, and Nor Aniza Azmi. Deep learning-based breast cancer classification through medical imaging modalities: state of the art and research challenges. *Artificial Intelligence Review*, 53(3):1655–1720, 2020.

- [213] Muhammad Ferjad Naeem, Seong Joon Oh, Youngjung Uh, Yunjey Choi, and Jaejun Yoo. Reliable fidelity and diversity metrics for generative models. In *International conference on machine learning*, pages 7176–7185. PMLR, 2020.
- [214] Hyeonseob Nam, HyunJae Lee, Jongchan Park, Wonjun Yoon, and Donggeun Yoo. Reducing domain gap by reducing style bias. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8690–8699, 2021.
- [215] Hyeonuk Nam, Seong-Hu Kim, and Yong-Hwa Park. Filteraugment: An acoustic environmental data augmentation method. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4308–4312. IEEE, 2022.
- [216] Ryota Namba and Jun Sakuma. Robust watermarking of neural network with exponential weighting. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, pages 228–240, 2019.
- [217] Loris Nanni, Gianluca Maguolo, and Michelangelo Paci. Data augmentation approaches for improving animal audio classification. *Ecological Informatics*, 57:101084, 2020.
- [218] Muzammal Naseer, Salman Khan, and Fatih Porikli. Indoor scene understanding in 2.5/3d for autonomous agents: A survey. *IEEE access*, 7:1859–1887, 2018.
- [219] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.
- [220] Roger Newson. Confidence intervals for rank statistics: Somers’ d and extensions. *The Stata Journal*, 6(3):309–334, 2006.
- [221] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.
- [222] Thai-Son Nguyen, Sebastian Stueker, Jan Niehues, and Alex Waibel. Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7689–7693. IEEE, 2020.

- [223] Thekke Madam Nimisha, Akash Kumar Singh, and Ambasamudram N Rajagopalan. Blur-invariant deep learning for blind-deblurring. In *Proceedings of the IEEE international conference on computer vision*, pages 4752–4760, 2017.
- [224] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*, pages 4901–4911. PMLR, 2019.
- [225] Augustus Odena, Catherine Olsson, David Andersen, and Ian Goodfellow. Tensorfuzz: Debugging neural networks with coverage-guided fuzzing. In *International Conference on Machine Learning*, pages 4901–4911. PMLR, 2019.
- [226] Sabine Öhlschläger and Melissa Le-Hoa Võ. Scegram: An image database for semantic and syntactic inconsistencies in scenes. *Behavior research methods*, 49(5):1780–1791, 2017.
- [227] Achraf Oussidi and Azeddine Elhassouny. Deep generative models: Survey. In *2018 International conference on intelligent systems and computer vision (ISCV)*, pages 1–8. IEEE, 2018.
- [228] Roxanne A Pagaduan, Ma Christina R Aragon, and Ruji P Medina. iblurdetect: Image blur detection techniques assessment and evaluation study. In *Proceedings of the International Conference on Culture Heritage, Education, Sustainable Tourism, and Innovation Technologies-CESIT*, pages 286–291, 2021.
- [229] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.
- [230] Xingang Pan, Ayush Tewari, Thomas Leimkühler, Lingjie Liu, Abhimitra Meka, and Christian Theobalt. Drag your gan: Interactive point-based manipulation on the generative image manifold. *arXiv preprint arXiv:2305.10973*, 2023.
- [231] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE access*, 7:36322–36333, 2019.
- [232] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015.

- [233] Sylvain Paris, Samuel W Hasinoff, and Jan Kautz. Local laplacian filters: Edge-aware image processing with a laplacian pyramid. *ACM Trans. Graph.*, 30(4):68, 2011.
- [234] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. <https://openreview.net/pdf?id=BJJsrmfCZ>, 2017.
- [235] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.
- [236] William Paul and Philippe Burlina. Robustness and adaptation to hidden factors of variation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 123–130, 2022.
- [237] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
- [238] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.
- [239] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.
- [240] Michal Perd’och, Ondrej Chum, and Jiri Matas. Efficient representation of local geometry for large scale object retrieval. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 9–16. IEEE, 2009.
- [241] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

- [242] C.I. Podilchuk and E.J. Delp. Digital watermarking: algorithms and applications. *IEEE Signal Processing Magazine*, 18(4):33–46, 2001.
- [243] Javier Portilla and Eero P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *International journal of computer vision*, 40(1):49–70, 2000.
- [244] V.M. Potdar, S. Han, and E. Chang. A survey of digital image watermarking techniques. In *INDIN '05. 2005 3rd IEEE International Conference on Industrial Informatics, 2005.*, pages 709–716, 2005.
- [245] Facundo Quiroga, Franco Ronchetti, Laura Lanzarini, and Aurelio F Bariviera. Revisiting data augmentation for rotational invariance in convolutional neural networks. In *International Conference on Modelling and Simulation in Management Sciences*, pages 127–141. Springer, 2018.
- [246] F. Radenović, G. Tolias, and O. Chum. CNN image retrieval learns from BoW: Unsupervised fine-tuning with hard examples. In *ECCV*, 2016.
- [247] F. Radenović, G. Tolias, and O. Chum. Fine-tuning CNN image retrieval with no human annotation. *TPAMI*, 2018.
- [248] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.
- [249] Jayaram Raghuram, Varun Chandrasekaran, Somesh Jha, and Suman Banerjee. Detecting anomalous inputs to dnn classifiers by joint statistical testing at the layers. *arXiv preprint arXiv:2007.15147*, 338:339–341, 2020.
- [250] Zhidan Ran and Xiaobo Lu. Camera domain adaptation based on cross-patch transformers for person re-identification. *Pattern Recognition Letters*, 159:84–90, 2022.
- [251] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

- [252] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.
- [253] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. ” why should i trust you?” explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.
- [254] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.
- [255] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M Lopez. The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3234–3243, 2016.
- [256] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.
- [257] Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv:1808.03305*, 1, 2018.
- [258] Kevin Roth, Yannic Kilcher, and Thomas Hofmann. The odds are odd: A statistical test for detecting adversarial examples. In *International Conference on Machine Learning*, pages 5498–5507. PMLR, 2019.
- [259] Dennis W Ruck, Steven K Rogers, and Matthew Kabrisky. Feature selection using a multilayer perceptron. *Journal of Neural Network Computing*, 2(2):40–48, 1990.
- [260] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.
- [261] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the*

- IEEE Conference on Computer Vision and Pattern Recognition*, pages 4691–4699, 2018.
- [262] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.
 - [263] Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv:1911.08731*, 1, 2019.
 - [264] Justin Salamon, Oriol Nieto, and Nicholas J Bryan. Deep embeddings and section fusion improve music segmentation. In *ISMIR*, pages 594–601, 2021.
 - [265] Artsiom Sanakoyeu, Vadim Tschernezki, Uta Buchler, and Bjorn Ommer. Divide and conquer the embedding space for metric learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 471–480, 2019.
 - [266] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding*, 152:1–20, 2016.
 - [267] Chandramouli Shama Sastry and Sageev Oore. Detecting out-of-distribution examples with gram matrices. In *International Conference on Machine Learning*, pages 8491–8501. PMLR, 2020.
 - [268] L. Scharf and D. Lytle. Signal detection in gaussian noise of unknown level: An invariance application. *IEEE Transactions on Information Theory*, 17(4):404–411, 1971.
 - [269] Jan Schlüter and Thomas Grill. Exploring data augmentation for improved singing voice detection with neural networks. In *ISMIR*, pages 121–126, 2015.
 - [270] Allen Schmaltz and Andrew L Beam. Sharpening the resolution on data matters: a brief roadmap for understanding deep learning for medical data. *The Spine Journal*, 21(10):1606–1609, 2021.

- [271] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.
- [272] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
- [273] Björn Schuller, Martin Wöllmer, Tobias Moosmayr, and Gerhard Rigoll. Recognition of noisy speech: A comparative survey of robust model architecture and feature enhancement. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:1–17, 2009.
- [274] Gesina Schwalbe and Martin Schels. A survey on methods for the safety assurance of machine learning based systems. In *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*, 2020.
- [275] Sergio Segura, Dave Towey, Zhi Quan Zhou, and Tsong Yueh Chen. Metamorphic testing: Testing the untestable. *IEEE Software*, 37(3):46–53, 2018.
- [276] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.
- [277] Muhammad Shafiq and Zhaoquan Gu. Deep residual learning for image recognition: A survey. *Applied Sciences*, 12(18):8972, 2022.
- [278] Nima Shahbazi, Yin Lin, Abolfazl Asudeh, and HV Jagadish. A survey on techniques for identifying and resolving representation bias in data. *arXiv preprint arXiv:2203.11852*, 2022.
- [279] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew Wallace. Radiate: A radar dataset for automotive

- perception in bad weather. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7. IEEE, 2021.
- [280] Weijun Shen, Jun Wan, and Zhenyu Chen. Munn: Mutation analysis of neural networks. In *IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 108–115. IEEE, 2018.
- [281] Xian Shi, Fan Yu, Yizhou Lu, Yuhao Liang, Qiangze Feng, Daliang Wang, Yanmin Qian, and Lei Xie. The accented english speech recognition challenge 2020: open datasets, tracks, baselines, results and methods. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6918–6922. IEEE, 2021.
- [282] Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3d photography using context-aware layered depth inpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8028–8038, 2020.
- [283] Noam Shoresh and Bang Wong. Data exploration, 2011.
- [284] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.
- [285] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6:1–48, 2019.
- [286] David Silver, Satinder Singh, Doina Precup, and Richard S Sutton. Reward is enough. *Artificial Intelligence*, 299:103535, 2021.
- [287] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556*, 1, 2014.
- [288] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [289] Bharat Singh and Larry S Davis. An analysis of scale invariance in object detection snip. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3578–3587, 2018.
- [290] G Sreenu and Saleem Durai. Intelligent video surveillance: a review through deep learning techniques for crowd analysis. *Journal of Big Data*, 6(1):1–27, 2019.

- [291] Megha Srivastava, Tatsunori Hashimoto, and Percy Liang. Robustness to spurious correlations via human annotations. In *International Conference on Machine Learning*, pages 9109–9119. PMLR, 2020.
- [292] MS Srivastava and KJ Keen. Estimation of the interclass correlation coefficient. *Biometrika*, 75(4):731–739, 1988.
- [293] Carlo Strapparava, Alessandro Valitutti, et al. Wordnet affect: an affective extension of wordnet. In *Lrec*, volume 4, page 40. Lisbon, Portugal, 2004.
- [294] Ting-Wei Su, Jen-Yu Liu, and Yi-Hsuan Yang. Weakly-supervised audio event detection using event-specific gaussian filters and fully convolutional networks. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 791–795. IEEE, 2017.
- [295] Youcheng Sun, Xiaowei Huang, Daniel Kroening, James Sharp, Matthew Hill, and Rob Ashmore. Testing deep neural networks. *arXiv:1803.04792*, 2018.
- [296] Youcheng Sun, Min Wu, Wenjie Ruan, Xiaowei Huang, Marta Kwiatkowska, and Daniel Kroening. Concolic testing for deep neural networks. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 109–119, 2018.
- [297] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.
- [298] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv:1312.6199*, 2013.
- [299] Edgar Talavera, Guillermo Iglesias, Ángel González-Prieto, Alberto Mozo, and Sandra Gómez-Canaval. Data augmentation techniques in time series domain: A survey and taxonomy. *arXiv preprint arXiv:2206.13508*, 2022.
- [300] Huan Tang and Ning Chen. Combining cnn and broad learning for music classification. *IEICE TRANSACTIONS on Information and Systems*, 103(3):695–701, 2020.

- [301] Ryo Terashima, Ryuichi Yamamoto, Eunwoo Song, Yuma Shirahata, Hyun-Wook Yoon, Jae-Min Kim, and Kentaro Tachibana. Cross-speaker emotion transfer for low-resource text-to-speech using non-parallel voice conversion with pitch-shift data augmentation. *arXiv preprint arXiv:2204.10020*, 2022.
- [302] Lucas Theis, Aäron van den Oord, and Matthias Bethge. A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*, 2015.
- [303] Amirsina Torfi, Rouzbeh A Shirvani, Yaser Keneshloo, Nader Tavaf, and Edward A Fox. Natural language processing advancements by deep learning: A survey. *arXiv preprint arXiv:2003.01200*, 2020.
- [304] Kristina Toutanova and Robert C Moore. Pronunciation modeling for improved spelling correction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 144–151, 2002.
- [305] Christopher J Tralie and Brian McFee. Enhanced hierarchical music structure annotations via feature level similarity fusion. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 201–205. IEEE, 2019.
- [306] Florian Tramer, Jens Behrmann, Nicholas Carlini, Nicolas Papernot, and Joern-Henrik Jacobsen. Fundamental tradeoffs between invariance and sensitivity to adversarial perturbations. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 9561–9571. PMLR, 13–18 Jul 2020.
- [307] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2020.
- [308] Gangesh Trivedi and Nitin N Pise. Gender classification and age estimation using neural networks: a survey. *International Journal of Computer Applications*, 975:8887, 2020.
- [309] Caterina Urban and Antoine Miné. A review of formal methods applied to machine learning. *arXiv preprint arXiv:2104.02466*, 2021.

- [310] Bhavik Vachhani, Chitralekha Bhat, and Sunil Kumar Kopparapu. Data augmentation using healthy speech for dysarthric speech recognition. In *Inter-speech*, pages 471–475, 2018.
- [311] R.G. van Schyndel, A.Z. Tirkel, and C.F. Osborne. A digital watermark. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 86–90 vol.2, 1994.
- [312] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.
- [313] James R Voss, Luis Rademacher, and Mikhail Belkin. Fast algorithms for gaussian noise invariant independent component analysis. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.
- [314] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.
- [315] Sanna Wager, George Tzanetakis, Cheng-i Wang, and Minje Kim. Deep auto-tuner: A pitch correcting network for singing performances. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 246–250. IEEE, 2020.
- [316] Angtian Wang, Yihong Sun, Adam Kortylewski, and Alan L Yuille. Robust object detection under occlusion with context-aware compositionalnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12645–12654, 2020.
- [317] Huibing Wang, Lin Feng, Jing Zhang, and Yang Liu. Semantic discriminative metric learning for image similarity measurement. *IEEE Transactions on Multimedia*, 18(8):1579–1589, 2016.
- [318] Jianren Wang, Zhaoyuan Fang, and Hang Zhao. Alignnet: A unifying approach to audio-visual alignment. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3309–3317, 2020.
- [319] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains:

- A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 1, 2022.
- [320] Kai Wang, Zhi Ma, Shiyan Chen, Jinni Yang, Keke Zhou, and Tao Li. A benchmark for clothes variation in person re-identification. *International Journal of Intelligent Systems*, 35(12):1881–1898, 2020.
- [321] Kehan Wang, Seth Z Zhao, David Chan, Avidesh Zakhori, and John Canny. Multimodal semantic mismatch detection in social media posts. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2022.
- [322] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neurocomputing*, 312:135–153, 2018.
- [323] Mei Wang, Weihong Deng, Jiani Hu, Xunqiang Tao, and Yaohai Huang. Racial faces in the wild: Reducing racial bias by information maximization adaptation network. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 692–702, 2019.
- [324] Mingyang Wang, Zhe Wang, Huiyong Sun, Jike Wang, Chao Shen, Gaoqi Weng, Xin Chai, Honglin Li, Dongsheng Cao, and Tingjun Hou. Deep learning approaches for de novo drug design: An overview. *Current Opinion in Structural Biology*, 72:135–144, 2022.
- [325] Qi Wang, Yue Ma, Kun Zhao, and Yingjie Tian. A comprehensive survey of loss functions in machine learning. *Annals of Data Science*, 9(2):187–212, 2022.
- [326] Tengfei Wang, Jiaxin Xie, Wenxiu Sun, Qiong Yan, and Qifeng Chen. Dual-camera super-resolution with aligned attention modules. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2001–2010, 2021.
- [327] Zhao Wang and Aron Culotta. Robustness to spurious correlations in text classification via automatically generated counterfactuals. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 14024–14031, 2021.
- [328] Shengyun Wei, Kele Xu, Dezhi Wang, Feifan Liao, Huaimin Wang, and Qiuqiang Kong. Sample mixed-based data augmentation for domestic audio tagging. *arXiv preprint arXiv:1808.03883*, 2018.

- [329] Xiyan Wei, Xin Lu, and Hong Sun. Fast view of mass remote sensing images based-on image pyramid. In *2008 First International Conference on Intelligent Networks and Intelligent Systems*, pages 461–464. IEEE, 2008.
- [330] Roman Werpachowski, András György, and Csaba Szepesvári. Detecting overfitting via adversarial examples. *arXiv:1903.02380*, 2019.
- [331] Maciej Wiatrak, Stefano V Albrecht, and Andrew Nystrom. Stabilizing generative adversarial networks: A survey. *arXiv preprint arXiv:1910.00927*, 2019.
- [332] Michael Wray, Hazel Doughty, and Dima Damen. On semantic similarity in video retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3650–3660, 2021.
- [333] Hanzhou Wu, Gen Liu, Yuwei Yao, and Xinpeng Zhang. Watermarking neural networks with watermarked images. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(7):2591–2601, 2020.
- [334] Yue Wu and Qiang Ji. Facial landmark detection: A literature survey. *International Journal of Computer Vision*, 127:115–142, 2019.
- [335] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.
- [336] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.
- [337] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th international conference on world wide web*, pages 1391–1399, 2017.
- [338] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *ArXiv preprint arXiv:2006.09994*, 2020.
- [339] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

- [340] Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv:2006.09994*, 1, 2020.
- [341] Cihang Xie, Zhishuai Zhang, Yuyin Zhou, Song Bai, Jianyu Wang, Zhou Ren, and Alan L. Yuille. Improving transferability of adversarial examples with input diversity. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [342] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 146–157, 2019.
- [343] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, Minhui Xue, Hongxu Chen, Yang Liu, Jianjun Zhao, Bo Li, Jianxiong Yin, and Simon See. Deephunter: a coverage-guided fuzz testing framework for deep neural networks. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 146–157, 2019.
- [344] Zhenda Xie, Yutong Lin, Zheng Zhang, Yue Cao, Stephen Lin, and Han Hu. Propagate yourself: Exploring pixel-level consistency for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16684–16693, 2021.
- [345] Eddie Yan and Yanping Huang. Do cnns encode data augmentations? *arXiv:2003.08773*, 2020.
- [346] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.
- [347] Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv:2110.11334*, 2021.
- [348] Mang Ye, Jianbing Shen, Gaojie Lin, Tao Xiang, Ling Shao, and Steven CH Hoi. Deep learning for person re-identification: A survey and outlook. *IEEE transactions on pattern analysis and machine intelligence*, 44(6):2872–2893, 2021.

- [349] Sangwoong Yoon, Woo Young Kang, Sungwook Jeon, SeongEun Lee, Changjin Han, Jonghun Park, and Eun-Sol Kim. Image-to-image retrieval by learning similarity between scene graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10718–10726, 2021.
- [350] Quanzeng You, Jiebo Luo, Hailin Jin, and Jianchao Yang. Robust image sentiment analysis using progressively trained and domain transferred deep networks. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 29, 2015.
- [351] Guoshen Yu and Jean-Michel Morel. ASIFT: An Algorithm for Fully Affine Invariant Comparison. *Image Processing On Line*, 1:11–38, 2011. <https://doi.org/10.5201/ipol.2011.my-asift>.
- [352] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 271–278, 2007.
- [353] Hyeongu Yun, Taegwan Kang, and Kyomin Jung. Analyzing and controlling inter-head diversity in multi-head attention. *Applied Sciences*, 11(4):1548, 2021.
- [354] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.
- [355] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.
- [356] Hasan FM Zaki, Faisal Shafait, and Ajmal Mian. Viewpoint invariant semantic object and scene categorization with rgb-d sensors. *Autonomous Robots*, 43(4):1005–1022, 2019.
- [357] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover’s distance and structured classifiers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12203–12213, 2020.

- [358] Jialong Zhang, Zhongshu Gu, Jiyong Jang, Hui Wu, Marc Ph Stoecklin, Heqing Huang, and Ian Molloy. Protecting intellectual property of deep neural networks with watermarking. In *Proceedings of the 2018 on Asia conference on computer and communications security*, pages 159–172, 2018.
- [359] Jie Zhang, Dongdong Chen, Jing Liao, Han Fang, Weiming Zhang, Wenbo Zhou, Hao Cui, and Nenghai Yu. Model watermarking for image processing networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 12805–12812, 2020.
- [360] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.
- [361] Kaihao Zhang, Wenqi Ren, Wenhan Luo, Wei-Sheng Lai, Björn Stenger, Ming-Hsuan Yang, and Hongdong Li. Deep image deblurring: A survey. *International Journal of Computer Vision*, 130(9):2103–2130, 2022.
- [362] Ligang Zhang, Brijesh Verma, Dian Tjondronegoro, and Vinod Chandran. Facial expression analysis under partial occlusion: A survey. *ACM Computing Surveys (CSUR)*, 51(2):1–49, 2018.
- [363] Long Zhang, Xuechao Sun, Yong Li, and Zhenyu Zhang. A noise-sensitivity-analysis-based test prioritization technique for deep neural networks. *arXiv preprint arXiv:1901.00054*, 2019.
- [364] Mabel M Zhang, Jean Choi, Kostas Daniilidis, Michael T Wolf, and Christopher Kanan. Vais: A dataset for recognizing maritime imagery in the visible and infrared spectrums. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 10–16, 2015.
- [365] Richard Zhang. Making convolutional networks shift-invariant again. In *International conference on machine learning*, pages 7324–7334. PMLR, 2019.
- [366] Richard Zhang. Making convolutional networks shift-invariant again. In *International Conference on Machine Learning*, pages 7324–7334. PMLR, 2019.
- [367] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

- [368] Shanghang Zhang, Xiaohui Shen, Zhe Lin, Radomír Měch, Joao P Costeira, and José MF Moura. Learning to understand image blur. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6586–6595, 2018.
- [369] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Proceedings of the IEEE international conference on computer vision*, pages 1116–1124, 2015.
- [370] Zhilin Zheng and Li Sun. Disentangling latent space for vae by label relevant/irrelevant dimensions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12192–12201, 2019.
- [371] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020.
- [372] Zhun Zhong, Liang Zheng, Zhiming Luo, Shaozi Li, and Yi Yang. Invariance matters: Exemplar memory for domain adaptive person re-identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 598–607, 2019.
- [373] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40:1452–1464, 2017.
- [374] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017.
- [375] William Zhu, Clark Thomborson, and Fei-Yue Wang. A survey of software watermarking. In Paul Kantor, Gheorghe Muresan, Fred Roberts, Daniel D. Zeng, Fei-Yue Wang, Hsinchun Chen, and Ralph C. Merkle, editors, *Intelligence and Security Informatics*, pages 454–458, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [376] Daniel Zoran and Yair Weiss. Scale invariance and noise in natural images. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2209–2216, 2009.

- [377] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the royal statistical society: series B (statistical methodology)*, 67(2):301–320, 2005.