Adaptive metamorphic testing with contextual bandits

Helge Spieker*, Arnaud Gotlieb

Simula Research Laboratory, P.O. Box 134, Lysaker 1325, Norway

ABSTRACT

Metamorphic Testing is a software testing paradigm which aims at using necessary properties of a system under test, called metamorphic relations, to either check its expected outputs, or to generate new test cases. Metamorphic Testing has been successful to test programs for which a full oracle is not available or to test programs for which there are uncertainties on expected outputs such as learning systems. In this article, we propose *Adaptive Metamorphic Testing* as a generalization of a simple yet powerful reinforcement learning technique, namely contextual bandits, to select one of the multiple metamorphic relations available for a program. By using contextual bandits, Adaptive Metamorphic Testing learns which metamorphic relations are likely to transform a source test case, such that it has higher chance to discover faults. We present experimental results over two major case studies in machine learning, namely image classification and object detection, and identify weaknesses and robustness boundaries. Adaptive Metamorphic Testing efficiently identifies weaknesses of the tested systems in context of the source test case.

© 2020 Elsevier Inc. All rights reserved.

1. Introduction

Metamorphic Testing (MT) is a software testing paradigm that aims at using necessary properties of a software under test to either check its expected outputs or to generate new test cases (Chen et al., 1998; 2018). More precisely, MT tackles the so-called *oracle problem* which occurs whenever predicting the expected outputs of a system is just too difficult or even impossible. Typical examples include machine learning models used for classification tasks, for which only stochastic behaviors can be specified (Barr et al., 2015). Indeed, these models are often initially trained with existing datasets and then exploited to classify new data samples. However, the expected class of any new data sample is unknown and thus, these samples cannot be used for testing the trained models. Fortunately, transformations over the data samples which do not change their (unknown) class, are usually available. By applying these transformations, called Metamorphic Relations (MRs) in MT, it becomes possible to effectively test trained machine learning models (Murphy et al., 2008; Ding et al., 2017; Dwarakanath et al., 2018).

MT has been very successful to address testing issues in various application domains, e.g., driverless cars (Zhou and Sun, 2019), search engines (Zhou et al., 2016), or bioinformatics (Shahri et al., 2019) just to name a few (see Section 2.3 for more references).

However, generally speaking, applying MT in practice requires to address two issues: the *MR identification* and the *MR selection* problems (Chen et al., 2018). The former occurs when trying to identify MRs for a specific system, i.e., to formalize input transformation properties which lead to a known transformation of the outputs. Finding such relations may be difficult when there is no obvious symmetries in the input data, or obvious system invariant, or else when the functional behavior of the system is unknown. The second occurs when several MRs have been identified, but determining which ones are best suited to discover faults in the system under test is hard. It is important to select appropriate MRs for testing the system to avoid redundancies in test cases and thus to avoid slack in the test execution process. This problem is especially critical when testing is part of a continuous integration process where there is usually a limit on the time allocated for testing in an integration cycle.

This paper addresses exclusively the latter problem, i.e., MRs selection, by formulating the effective selection of MRs as a reinforcement learning problem, based on *contextual bandits*. Our method, called *Adaptive Metamorphic Testing* (AMT), defines a *test transformation bandit* which sequentially selects a MR that is expected to provide the highest payoff, i.e., that is most likely to reveal faults. Which MRs are likely to reveal faults is learned from successive exploration trials. The bandit explores the different available MRs and evaluates the fault landscape of the system under test, thereby providing valuable information to the tester.

Learning the selection of MRs can be useful when testing under resource-constraints, for example in cases where the system

* Corresponding author.
  E-mail addresses: helge@simula.no (H. Spieker), arnaud@simula.no (A. Gotlieb).

under test changes are frequently integrated and tested, but also for infrequent testing when the number of MRs is large or their checking is costly. We also discuss a second application which is to identify robust boundaries of the MR parameters. Robust boundaries describe related scenarios but focus on a single MR that can be controlled via parameters. The interest is to find parameters that produce fault-revealing test cases with minimal changes only.

In this paper, we evaluate Adaptive Metamorphic Testing on two case study applications for image analysis, namely, *image classification* and *object detection*. As implementations of these case studies, we test freely available and pre-trained deep learning systems that can be used as black-box components in other software systems. For each system, we explore both the general fault-revealing capabilities of metamorphic relations and the discovery of robustness boundaries.

The main contributions of this paper are three-fold. First, we introduce Adaptive Metamorphic Testing as a general adaptive selection method for metamorphic relations and test transformations. The method is based on reinforcement learning with contextual bandits and learns to identify those relations which are likely to reveal faults in the system under test. This method is useful in a context where a source test case can be modified by several different metamorphic relations and the system is to be repeatedly tested. To the best of our knowledge, it is the first time that reinforcement learning is applied to select MRs and embedded into a general methodology for MT.

Second, we provide an implementation of Adaptive Metamorphic Testing in a tool called Tetraband, dedicated to testing machine learning models for image analysis. Our tool facilitates metamorphic relations based on image augmentation functions and provides dedicated environments for adaptive metamorphic testing that can be integrated with other implementations.

Third, we explore the benefits of Tetraband on two case studies coming from image analysis, namely image classification and object recognition. Both of these case studies are relevant subsystems in a wide number of applications, such as autonomous cars, robot navigation, or industrial automation (LeCun et al., 2015; Pouyanfar et al., 2018). For these applications, high-quality standards are essential and rigorous testing is a requirement. Our experiments show that Tetraband is highly beneficial to optimize the testing process towards fault-revealing MRs.

The remainder of the paper is structured as follows. We review the background on metamorphic testing and contextual bandits and related work in the area in Section 2. Section 3 introduces Adaptive Metamorphic Testing and its components. We discuss general application scenarios in Section 4 before introducing the experimental setup, consisting of our implementation of AMT, Tetraband, and the two case studies in Section 5. The results are presented in Section 6 and Section 7 finally concludes the paper.

2. Background

2.1. Metamorphic testing

Metamorphic Testing (MT) aims at using necessary properties of a software under test to either check its expected outputs or to generate new test cases (Chen et al., 1998; 2018). Central to MT is the concept of *Metamorphic Relations* (MRs) which are high-level observable properties that must hold over inputs and outputs of the system under test.

In the following, we formalize the definition of a metamorphic relation, the transformation from source test cases to follow-up test cases, and metamorphic testing. Our definitions follow the formalization by Chen et al. (2018), except for the transformation

from a source to a follow-up test case. We interpret the transformation function to apply to the whole test case, which includes the test input, and formalize it accordingly in this more general, but compatible, way:

Definition 1 (Metamorphic Relation (MR)). Let f be a target function or algorithm. A metamorphic relation (MR) is a necessary property over a sequence of multiple inputs $\langle x_1, x_2, \ldots, x_n \rangle$ ($n \geq 2$) and their corresponding outputs $\langle f(x_1), f(x_2), \ldots, f(x_n) \rangle$. It can be expressed as a relation $\mathcal{R} \subseteq X^n \times Y^n$, with $\subseteq$ being the subset relation, and X^n and Y^n being the Cartesian products of n input and output spaces.

Definition 2 (Transformation from Source Test Cases to Follow-up Test Cases). Consider a MR $\mathcal{R}(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), f(x_3))$. The sequence of inputs and their corresponding outputs defines the set of source test cases. For each source test case $\mathcal{S}(x) \in \mathcal{R}$, a follow-up test case $\mathcal{F}$ is derived by applying a possibly non-deterministic transformation function T to the input of the source test case x: $\mathcal{F}(x) = f(T(x))$. The transformation function T is constructed such that the follow-up test case $\mathcal{F}$ fulfills the necessary property of $\mathcal{R}$.

Definition 3 (Metamorphic Testing (MT)). Let P be an implementation of a target algorithm f. For an MR $\mathcal{R}$, suppose that we have $\mathcal{R}(x_1, x_2, \ldots, x_n, f(x_1), f(x_2), f(x_3))$. Metamorphic testing (MT) based on this MR for P involves the following steps:

1. Define $\mathcal{R}'$ by replacing f by P in $\mathcal{R}$.
2. Given a sequence of source test cases $\langle x_1, x_2, \ldots, x_k \rangle$, execute them to obtain their respective outputs $\langle P(x_1), P(x_2), \ldots, P(x_k) \rangle$. Construct and execute a sequence of follow-up test cases $\langle x_{k+1}, x_{k+2}, \ldots, x_n \rangle$ according to $\mathcal{R}'$ and obtain their respective outputs $\langle P(x_{k+1}), P(x_{k+2}), \ldots, P(x_n) \rangle$.
3. Examine the results with reference to $\mathcal{R}'$. If $\mathcal{R}'$ is not satisfied, then this MR has revealed that P is faulty.

For the remainder of this paper, we refer in general to a metamorphic relation R as the tuple of $\langle \mathcal{R}, T \rangle$, the combination of necessary properties over the outputs for a specific MR and the transformation function T to generate follow-up test cases from source test cases. Therefore, a source test case $\mathcal{S}$ produces output $P(x)$ for the system P with test input x. Using a metamorphic relation R, the follow-up test case $\mathcal{F}$ with test input $T(x)$, where T is a transformation over the input x, can be generated. Due to the metamorphic relation over R (including T), $\mathcal{S}$, and $\mathcal{F}$, the result $P(T(x))$ can be verified. Note that MRs are only partial properties, which means that only test cases that violate them indicate the presence of faults in the system under test. Showing that the system satisfies a MR on any input or a test suite does not guarantee the absence of faults but increases our confidence in the system correctness. However, this issue concerns any software testing method, not only MT.

The transformation function T of a metamorphic relation R does not have to be a deterministic function, but is usually parameterized and can result in several different follow-up test cases from one source test case, depending on a parameter ϕ which specifies the exact transformation to be applied. In many cases and the simplest implementation of metamorphic testing, ϕ is chosen from a random distribution and T produces a random follow-up test case. We use R_ϕ to denote a metamorphic relation configured by ϕ, which makes it deterministic, and R for the general metamorphic relation, which might be non-stochastic if no additional configuration is possible.

2.2. Contextual bandits

The selection of a test transformation to apply on a source test case is formalized as a multi-armed bandit problem with con-

text information, also known as a contextual bandit (Langford and Zhang, 2007; Zhou, 2016). A contextual bandit acts in discrete iterations, where each iteration corresponds to the generation and execution of one follow-up test case. A bandit $\mathcal{B}$ has k arms, where every arm corresponds to the selection of one possible MR to generate the follow-up test case. In every iteration i, the bandit receives the context vector c_i which in our case describes the source test case.

The bandit then acts according to one or multiple policies π which formalize the action selection, i.e. the decision making strategy. These policies are trained from external feedback about the success of previously made decisions. A policy is often realized by function approximation techniques, for example, multiple linear regression or neural networks. Additionally, an exploration strategy is used to try previously unexplored actions instead of following the policy only. The bandit chooses an arm ($a_i = \pi(c_i)$) and receives a reward, also called payoff, r_i which is the external feedback for this decision. As only 1 arm can be selected, there is also only feedback for the effect of this single arm a_i. Afterwards, the bandit updates its policy from the observation (c_i, a_i, r_i). Updating the bandit works by adjusting a set of weights, such that the new set of weights better fits the previously made experiences and minimizes regret for historical decisions. The actual implementation of the weight update is dependent on the specific contextual bandits algorithm (Agarwal et al., 2014) and the learner used to approximate the policy, e.g. whether it is a form of linear regression or a non-linear neural network.

Our goal for the contextual bandit is to maximize the total payoff, i.e., the cumulative undiscounted reward over all iterations $\sum_{i=1}^{T} r_i$. To achieve this goal and to identify highly rewarded actions, contextual bandit algorithms are designed to also minimize the regret. The regret of a bandit is the gap between the expected reward over a number of iterations when following one policy and the cumulative reward the agent actually receives over the same number of iterations (Lattimore and Szepesvari, 2019). As a smaller regret means to choose actions more closely to the highest possible payoff, minimizing the regret implies the maximization of the total payoff, but by maintaining the concept of regret also badly rewarded actions contribute to the improvement of the policy.

A challenge in the design of contextual bandits is to find a balance between *exploration*, i.e., evaluating the effect of rarely used actions, and *exploitation*, i.e., repeating those actions that showed to be effective before. This process is called the *exploration-exploitation trade-off*. To this extent, several exploration techniques have been developed as part of bandit algorithms. We introduce here two techniques that are useful in Adaptive Metamorphic Testing. *Epsilon-greedy* (Sutton and Barto, 2018) decides to explore a random action with probability ϵ and with probability $1 - \epsilon$ the current learned policy is used to select an action. The parameter $\epsilon \in [0, 1)$ is chosen by the user. A more advanced exploration strategy is *online cover* (Agarwal et al., 2014). Instead of training a single policy, m different polices are trained to produce diverse behaviors. The exploration algorithm then chooses from those actions which have not been learned to perform bad, i.e., having high regret, in the current context. Again, m is a parameter to be chosen by the user.

Contextual bandits are related to Reinforcement Learning (RL) (Sutton and Barto, 2018). The main distinction between bandits and RL agents is that bandits perceive each iteration as independent of the previous one (Lattimore and Szepesvari, 2019), i.e., the selected action does not affect the next context that is observed. In our scenario, the next test case is independent of the test transformation chosen for the previous test case. RL agents, however, are designed to operate over multiple subsequent iterations, where a chosen action influences the context in the next iteration. General RL agents could be applied by reducing the length of each scenario to one step, but our early experiments found contextual bandits to be more efficient.

Bandit algorithms have been successfully applied in a variety of domains, such as news article recommendation (Li et al., 2010), advertisement selection (Lu et al., 2010; Tang et al., 2013), statistical software testing (Baskiotis et al., 2006), constraint optimization (Loth et al., 2013; Balafrej et al., 2015), or real-time strategy games (Ontañón, 2017). In this work, we apply contextual bandits to the selection and configuration of metamorphic relations in software testing.

2.3. Related work

Metamorphic Testing (MT) has been applied to a variety of domains and applications, see (Segura et al., 2016; Chen et al., 2018) for an in-depth overview. Successful application domains include testing of driverless cars (Zhou and Sun, 2019), search engines (Zhou et al., 2016), machine translation systems (Sun and Zhou, 2018), performance testing (Segura et al., 2018; Johnston et al., 2019), constraint solvers (Akgün et al., 2018) or bioinformatics (Shahri et al., 2019). Previous works already focused on its automation, for example by exploring algorithms to specifically identify fault-revealing inputs (Gotlieb and Botella, 2003) or performing an empirical study on selecting good Metamorphic Relations (MRs) (Mayer and Guderlei, 2006a). Other works predict the applicability of a MR for a system (Kanewala and Bieman, 2013; Kanewala et al., 2016). Based on source code traces, a classification model predicts which MRs of a given set can be applied.

Due to the emergent success and usage of machine learning in different application areas, the verification and validation of these systems has received increasing attention. Several works approach testing machine learning systems based on software testing techniques, such as differential, multi-implementation (Pei et al., 2017) or mutation testing (Ma et al., 2018).

Because testing machine learning systems, due to their stochastic nature, is affected by the oracle problem (Barr et al., 2015), there has been work to especially apply MT for this purpose. Murphy *et al.* identify a set of general MRs, that hold for a variety of machine learning algorithms (Murphy et al., 2008), and are shown to be effective (Xie et al., 2011). Chan et al further use MT to identify violations of MRs from passed test cases of classification models (Chan et al., 2010). It has also been shown that MT can be used for deep learning-based applications, e.g., to test the classification of biological cells (Ding et al., 2017). Dwarakanath et al identify implementation faults in image classifiers (Dwarakanath et al., 2018). They introduce MRs that affect the training and test data used during model training and demonstrate how these MRs can be applied to find implementation errors in training procedures and model architecture. Yang *et al.* propose to test unsupervised clustering methods (Yang et al., 2019) and Mekala et al explore the application of MRs to detect adversarial examples for deep learning models (Mekala et al., 2019). However, Saha and Kanewala (Saha and Kanewala, 2019) recently evaluated the effectiveness of MRs for testing supervised classifiers based on mutations of the system under test. They found that the detection rates for the used MRs of previous studies are limited when generating a large set of mutants.

Previous work also considered the adaptive control of software testing through feedback while testing (Cai et al., 2007; Zhou et al., 2018). Similar to our method, these works exploit the behavior of the system during the test execution and adjust the testing strategy when the understanding of the software changes. In these works, the adjustment of the testing strategy focuses on test case prioritization and selection, whereas our method focuses on the generation of follow-up test cases using metamorphic testing and under consideration of many repeated testing cycles.

Algorithm 1 Adaptive Metamorphic Testing with Contextual Bandits..

Input: M: set of MRs; SUT: system under test P; TS: set of test cases; Iter: Number of iterations

Output: $\mathcal{B}$: trained bandit

1: $i \leftarrow 0$
2: $\mathcal{B} \leftarrow$ Load *existing* or initialize *new* Bandit
3: **while** $i <$ Iter **do**
4:     **Select** $\mathcal{S} \in TS$ ▷ Draw source test case from test suite
5:     $c \leftarrow \mathcal{B}.\text{ExtractContextFeatures}(\mathcal{S})$ ▷ Generate feature vector for source test case $\mathcal{S}$
6:     $R_\phi \leftarrow \mathcal{B}.\text{SelectBanditArm}(c, M)$ ▷ Select one MR R_ϕ using the bandit
7:     $v \leftarrow \text{Apply}(SUT, R_\phi(\mathcal{F}))$ ▷ Execute SUT with transformed test $\mathcal{F}$, get a verdict v
8:     $\mathcal{B} \leftarrow \mathcal{B}.\text{UpdateBandit}(R_\phi, c, v)$ ▷ Train the bandit with the feedback
9:     $i \leftarrow i + 1$
10: **end while**
11: **return** $\mathcal{B}$ ▷ Return updated bandit for future test cycle

---

3. Adaptive metamorphic testing

3.1. Overview

In this section, we introduce Adaptive Metamorphic Testing (AMT) with contextual bandits. Our method is based on a test transformation bandit that learns to select follow-up test cases from a set of applicable metamorphic test cases. Algorithm 1 shows an overview of the main steps of AMT. At the core of AMT, a contextual bandit receives a description of the source test case. Based on this context vector, the bandit selects an action, which resembles a MR, and the configuration of this transformation. Both are applied to generate a follow-up test case. After generating the follow-up test case, the system under test is executed and the test result evaluated according to the MR acceptance criterion. The method can be directly deployed without any pre-training step. However, during the first iterations, MR selection is partly random to gather initial experiences about the different MRs effectiveness and their potential payoffs, when applied to the available source test cases. After several iterations have been performed, the bandit learns to focus on MRs which are most likely to reveal faults. Nevertheless, the bandit continues to explore among the MRs, i.e., it sometimes chooses MRs which do not promise the highest payoff. This is important to adjust to changes in the system under test as well as to gather additional information about the effect of MRs in different contexts.

Definition 4 (Adaptive Metamorphic Testing). Let P be an implementation of a target algorithm f with TS being its test suite; let M be a set of metamorphic relations applicable on TS and let $\mathcal{B}$ be a contextual bandit. Adaptive Metamorphic Testing (AMT) is an iterative variant of metamorphic testing and involves the following steps at each iteration:

1. A test case $\mathcal{S}$ is (randomly) selected from the test suite TS and executed to obtain its output $P(\mathcal{S})$.
2. The bandit $\mathcal{B}$ selects a MR R based on the context features of $\mathcal{S}$.
3. Construct one or more follow-up test cases $\langle \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k \rangle$ according to R and obtain their respective outputs $\langle P(\mathcal{F}_1), P(\mathcal{F}_2), \ldots, P(\mathcal{F}_k) \rangle$.
4. Examine the results with reference to R. If R is not satisfied, then this MR has revealed that P is faulty.
5. Report the results of the execution back to $\mathcal{B}$ for adaptation of the learning algorithm.

The arm selection of the bandit, i.e., the selection of a metamorphic relation R with its parameters is handled by a hierarchy of contextual bandits using the context features c at each iteration of the algorithm.

On the highest level, the main contextual bandit $\mathcal{B}$ selects one MR R from the set of supported MRs. Afterwards another action-specific contextual bandit $\mathcal{B}_R$ is queried, using the same context information as the main bandit, for the configuration parameter ϕ: $\mathcal{B}_R(c_i) \rightarrow \phi$. If the metamorphic relation does not require additional configuration, the second step is skipped. The MR R_ϕ can then be used to generate a follow-up test case for the current iteration. After test execution, both bandits $\mathcal{B}$ and $\mathcal{B}_R$ are trained from the received feedback. Tetraband consists of one main contextual bandit plus one additional contextual bandit for each configurable MR.

Adaptive Metamorphic Testing is independent of the application domain or the implementation or else the specific MRs that can be applied. It only takes as inputs a set of MRs, the system under test, a set of test cases, and a user-defined parameter corresponding to the maximum number of iterations to run. As output, the method returns a trained bandit $\mathcal{B}$ which has learned to select the MRs which have the greatest chances to detect faults in the system under test.

3.2. Components

Adaptive Metamorphic Testing requires only a few system-specific components. In the following, we discuss each of these components.

3.2.1. Extract context features

In order to select an appropriate MR which is likely to reveal a fault, for a source test case, it is mandatory to feed the contextual bandit with relevant context information about the source test case. This context is captured with a feature vector, which is a real-valued vector of fixed size n. The function receives the source test case as input and returns the feature vector: $ExtractContextFeatures : \mathcal{S} \rightarrow \mathbb{R}^n$.

By using representative features, the contextual bandit can learn a mapping from the source test cases, which are described through the features, to the MRs. For that, it has to include test characteristics that can be affected by the metamorphic relations. Therefore, the features need to capture the necessary details distinctive about the individual test case, especially those that relate to the effect of the MR.

How the feature vector is formed, is a domain-specific problem and requires some degree of domain knowledge. For example, when testing scientific software for matrix calculations (Mayer and Guderlei, 2006a), the features should describe the characteristics of the original input matrix in order to select which transformation is applied. In our experiments, which are based on testing computer vision problems, we rely on common feature modeling used in machine learning. For instance, using a pre-trained neural network to extract image features is common in computer vision. A similar approach could be used for text processing, where textual features,

e.g., used vocabulary, text sentiment, or sentence structure, can be derived using pre-trained networks.

3.2.2. Metamorphic relations

The most relevant component to acquire for applying Adaptive Metamorphic Testing is the set of MRs, which is highly domain-dependent. The automatic and systematic identification of MRs for a system is an ongoing research topic (Chen et al., 2018), but in many cases, MRs can be extracted by using domain knowledge about the system or reviewing the existing literature from the Software Testing community. In the special case of testing machine learning systems, a starting point to uncover MRs is to exploit data augmentation methods that are commonly used. As shown in our experiments, these augmentations can serve as a basis for MRs and help to identify weaknesses in ML systems.

3.2.3. Select bandit arm

The selection of an appropriate MR is mostly handled by the internal contextual bandit algorithm and does not have to be individually implemented for a new system under test. The main bandit selects the MR and, if necessary, the action-specific bandit for this MR selects the parameter to configure the MR. Nevertheless, the configuration of the contextual bandit influences the performance of the system and should be adjusted, depending on the number of available MRs and the robustness of the SUT. This allows us to focus on exploiting MR that reveal faults in the system or to broadly explore the effects of many different MRs.

The most important configuration parameter to adjust is the exploration rate, i.e., how often does the bandit choose a different action than the most promising one. Using a high exploration rate allows us to examine different combinations of test cases and systems, which is relevant to detect new faults in the system and extend the coverage of different tests. Conversely, a lower exploration rate exploits combinations of tests and MRs that have often fail previously. Traditionally, Metamorphic Testing often creates follow-up test cases at random, which corresponds to a maximal exploration rate here.

Exploitation is relevant when repeatedly testing the system, e.g., in continuous integration settings, or when trying to understand the weaknesses of the system for a certain group of MRs. Still, it is not only desired to exploit known weaknesses, but broad coverage of the system behavior is desired for higher test confidence. If the behavior of the system changes, because it is becoming more robust to previously effective MRs, the bandit learns this and can adjust its selection for future iterations. Conclusively, compared to other applications of contextual bandits, where especially the exploitation of known good actions is in focus, exploration is more prominent in Adaptive Metamorphic Testing to broadly test the system behavior.

3.2.4. Transform, execute and evaluate

The selected MR and its configuration transform the source test case into a follow-up test case. That test case is then executed and the test verdict is evaluated according to the MR. These core steps of Adaptive Metamorphic Testing are similar to those required in traditional MT.

3.2.5. Update bandit

After the follow-up test case has been executed and the results have been evaluated, the bandit's policy is updated. This requires information about the initial context feature vector, the chosen MR and its configuration, and the test verdict. The *update routine* updates the expected reward for this MR. The exact update routine is specific to the contextual bandit algorithm and its configuration and we refer to the corresponding literature for its description (Langford and Zhang, 2007; Lattimore and Szepesvari, 2019).

Nevertheless, choosing the appropriate reward for a failed test case has to be done while adjusting the bandit for a new system to test. In most scenarios, where the goal is to find the most fault-revealing MRs, as described below in Section 4.1, the reward is the same for every failing test case. However, if the bandit has the goal to identify certain properties of the SUT, it can be necessary to propose a different reward structure that depends on the selected MR. This second scenario is further described in Section 5.3.

4. Application scenarios of adaptive metamorphic testing

Contextual bandits are powerful to explore the effects of the MRs in different contexts, but can also exploit the gathered experiences to subsequently focus on those relations that are most likely to reveal faults. From these properties, we identify two application scenarios of Adaptive Metamorphic Testing that we discuss further and evaluate as part of the case studies.

4.1. Fault-Revealing MR selection

The first application of Adaptive Metamorphic Testing is the selection of metamorphic relations which are prone to reveal faults in the system under test. In cases where the MR has parameters, an additional contextual bandit is responsible to select these parameters, as described before. This application, which we refer to as *fault-revealing MR selection*, steers MT towards greater effectiveness when there are many MRs available and not sufficient resources to apply them all. In this application, all MRs are considered distinct from each other. Accordingly, the achievable reward received for revealing a fault is identical for all MRs.

4.2. Robustness boundaries

The second application uses the exploration/exploitation trade-off of contextual bandits to identify *robustness boundaries* of the software under test. With robust boundaries, we focus on MRs whose effect can be adjusted by user-defined parameters, especially those with continuous or a range of discrete values that control the distance between source and follow-up test cases. As an example taken from the case studies, while testing an image analysis system, one possible transformation is to rotate the image, where the degree of rotation is a user-defined parameter. If the system is susceptible to treat wrongly rotated images, it is likely that large rotations, e.g. by 90 degrees, are more likely to cause mistakes than smaller rotations. By identifying the robust boundaries, information can be inferred about the trade-off between acceptable transformations and exceedingly strong manipulations. This result yields both a robustness characteristic and a starting point for a more curated set of requirements on the system under test.

5. Experimental evaluation

5.1. Case studies

We consider two case studies to evaluate our tool Tetraband. Both case studies come from the field of digital image processing, where deep learning methods commonly represent the state-of-the-art approaches (Pouyanfar et al., 2018): *image classification* and *object detection*. For each of the case studies, we consider both previously introduced application scenarios (see Section 4) and identify fault-revealing MRs as well as robustness boundaries against configurable image transformations. We have formulated the following three research questions as a guideline for our experiments:

(a) Image Classification Example (from ImageNet [45]). The goal is to assign a single class label to the image, e.g. *toucan* here.
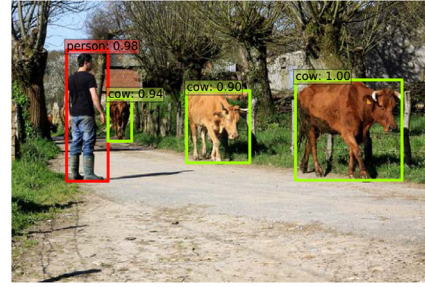
(b) Object Detection Example (from [46]). The goal is to identify and categorize objects by drawing a bounding box and assigning a class label.

Fig. 1. Image classification and object detection examples.

- **RQ1** Does Adaptive Metamorphic Testing, implemented as Tetraband, learn to select MRs whose follow-up test cases reveal faults?
- **RQ2** Is AMT effective to approximate the distribution of faults in the system under test?
- **RQ3** Is AMT computation- and data-efficient compared to random sampling of MRs and exhaustive search of all follow-up test cases?

Previous work for testing image processing applications has considered random and metamorphic testing (Mayer and Guderlei, 2006b; Guderlei and Mayer, 2007), but focused on the evaluation of handcrafted image processing applications, whereas we focus in our experiments especially on machine learning-based computer vision systems. The existing studies focus on testing the basic functionality of the system by generating random images and transforming them using a set of transformations, different from our approach where we base on an existing dataset of images from the domain that the ML model has been trained on. We furthermore especially consider the selection of good MRs. Xu *et al.* recently presented another use case for metamorphic relations in image classification applications beyond testing (Xu et al., 2018). Their work uses metamorphic relations, based on separation and occlusion, to augment the training data and fine-tune the model.

In the following, we present the two case studies, image classification and object detection, with their setup and the considered MRs. We further discuss the configuration of Tetraband, and finally, present the experimental results and our findings.

5.2. Image processing applications

We describe two case studies where testing image processing systems is necessary (see Fig. 1). In the first case study, an image classification system is tested. The second case study focuses on an object detection application.

5.2.1. Image classification

An image classification task, also image recognition, has the goal to identify the object shown in an image, e.g., assign the image to one of a fixed number of classes. Since 2012, the state-of-the-art method for image classification, among other image analysis tasks, are deep neural networks, such as residual neural networks (ResNets) (He et al., 2016) or SqueezeNet (Iandola et al., 2016).

In our case study, we test a SqueezeNet model that has been initially trained on the ImageNet dataset (Russakovsky et al., 2015) and then fine-tuned for the 10 classes of the CIFAR-10 dataset

(Krizhevsky et al., 2014). For testing the model, we use the CIFAR-10 test set, consisting of 10,000 labeled images.

Following the metamorphic relation between source and follow-up test cases, we consider a test as failed, if the transformed image leads to a different classification result than the original image. The correctness of the original class prediction does not influence the test result, because the bandit does not know the initial model performance. Instead, the bandit aims to select transformations that affect the outcome in a fault-revealing manner compared to the original output. Testing the difference in outputs between source and follow-up test cases removes the dependency from having to use labeled data, i.e. data where the ground-truth class is known and allows the integration of other data sources. Nevertheless, for testing the system, we monitor also the accuracy of the system for correctly classifying the images, but we do not use this information as feedback for the test transformation bandit, although that would also be a viable setup.

5.2.2. Object detection

Object detection is a generalization of the image classification task in the sense that there can be multiple objects on a single image. Besides assigning classes to these objects, it is also necessary to provide bounding boxes around the location of each object. This means that the output of an object detection model consists of a class label and four coordinates for the bounding box for each detected object. Object detection systems employ deep neural networks of a similar, but extended, architecture compared to image classification systems.

The system to test in this case study is a pre-trained object detection model, based on the open-source TensorFlow Object Detection API[1] (Huang et al., 2017). In particular, we test an implementation of a single shot multibox detector (SSD) (Liu et al., 2016) with a feature pyramid network (FPN) (Lin et al., 2017), based on a ResNet-50 network (He et al., 2016). We refer the reader to the given references for an in-depth overview of the models. We see the system to test here as a black box. However, briefly said, the model detects objects in images with a single neural network by assigning one of multiple predefined box sizes, their size adjustment and classification scores at the same time. By reducing the complexity to a single neural network, it is a fast model for real-time object detection that achieves state-of-the-art performance. The used model was trained on Microsoft COCO dataset (Lin et al., 2014) and is available within the Object Detection API[2].

---

[1] TensorFlow Object Detection API: http://github.com/tensorflow/models/tree/master/research/object_detection.

[2] The exact name in the object detection model zoo is *ssd_resnet50_v1_fpn_shared_box_predictor_640x640_coco14_sync_2018_07_03*

For testing, we use 5,000 images from the validation set of the *MS COCO challenge 2017* as source test cases. We apply the same input transformations on the images as in the image classification case study and as described in Section 5.3. Because each image annotation consists of an additional bounding box per object in the image, the metamorphic relations are extended to transformation also on the bounding boxes. For example, rotating the image rotates the bounding box of the object to match the rotated object, and flipping the image from left to right also flips the positions of the bounding boxes in the image (see the next Section 5.3 for an introduction of the applied transformations).

Furthermore, we consider the different evaluation metrics for object detection tasks. In image classification, the result is easily verified by comparing the estimated class with the ground truth class. In object detection, it is necessary to evaluate the overlapping regions between the estimated bounding boxes and the ground truth, which is called the intersection-over-union (IoU), in addition to the class label of each box:

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

where *A* are the proposed pixels from the object detection model and *B* is the ground-truth from the dataset. If the IoU value exceeds a certain threshold, the object is counted as correctly detected. We follow the evaluation guidelines from the MS COCO challenge and compare the results for the mean average precision (mAP) which is calculated over all objects in an image and the average of different IoU thresholds $0.5, 0.55, 0.6, \ldots, 0.95$, for both the original and the transformed image. If the mAP of the transformed image is below the original mAP minus a performance reduction of 0.05 which is close to 10% of the average model performance, we interpret the test case to be violating the MR and therefore as failed.

5.3. Metamorphic relations and rewards

In the case studies of this paper, we use tests where the input is an image and the output is a classification of this image to recognize certain objects. The considered MRs are all related to image transformations, e.g. mirroring or rotating, under the property that the results of image classification and object detection do not change, i.e., the MR defines equality of the outputs, while the input is transformed.

In most cases, these transformations must only not modify the class of these images. However, in object detection, some transformations of images that impact object location markers entail similar transformation over the outputs. Here, the MR defines a relation between the outputs that is similar to the transformation of the input.

We select seven common image transformations among all possible transformations as MRs, some of which have been used in previous work on metamorphic testing for image analysis methods (Mayer and Guderlei, 2006b; Mekala et al., 2019). Among these MRs, two are configurable by an additional parameter ϕ. These transformations are the following:

1. Blur the image by the average value of neighbor pixels (Blur)
2. Flip the image from the left to the right (Flip L/R)
3. Flip the image upside down (Flip U/D)
4. Convert a colored image to grayscale (Grayscale)
5. Invert the colors of the image (Invert)
6. Rotate the image by *x* degrees (Rotation)
7. Shear the image by *x* degrees (Shear)

The effects of these transformations are shown in Fig. 2. The MRs Rotation and Shear expect a parameter to define the transformation effect. For Rotation, we consider 36 distinct values in steps of 5 degrees in the degree range $[-90; 90]$, excluding rotations of 0 degrees. For shear, we include 18 values in the range $[-45; 45]$ in steps of 5 degrees, again excluding 0 degrees.

The reward for the main MRs is set up such that a failed test case is rewarded with 1 and a passed test case with 0, independent of which MR was selected. For the action-specific bandits, which select the MR parameters for Rotation and Shear, the reward structure is designed to encourage the selection of the smallest failing parameters. Therefore, the smallest parameter of -5 respectively 5 degrees receives a reward of 10000, if revealing a fault. For each additional step, the reward is divided by two. Thereby, choosing a smaller parameter value has always higher payoff than a larger rotation, if successful.

5.4. Implementation

We implemented Adaptive Metamorphic Testing in a tool called Tetraband. Our implementation package is available at http://github.com/helges/tetraband. The software is implemented in Python 2.7 and it is structured into two main components, as shown in Fig. 3.

(a) Original Image   (b) Blur   (c) Flip left ↔ right   (d) Flip up ↔ down   (e) Convert to grayscale

(f) Invert the image   (g) Rotate -30 degrees   (h) Rotate +30 degrees   (i) Shear -20 degrees   (j) Shear +20 degrees
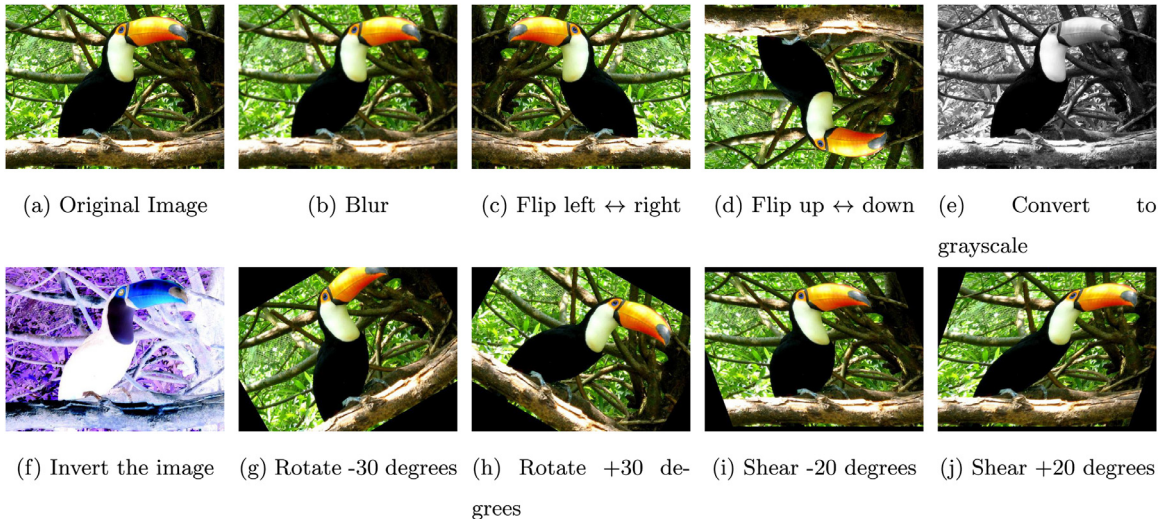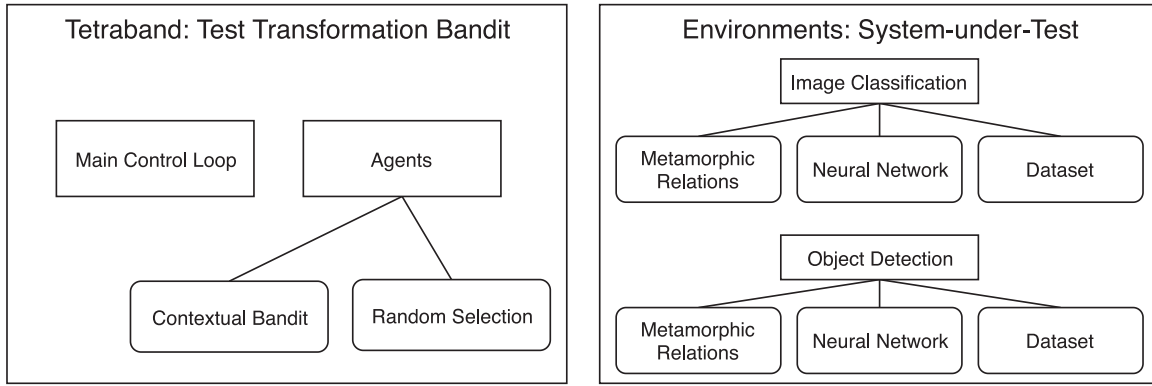
Fig. 2. Metamorphic relations, i.e. image transformations, and their effects. Image taken from the ImageNet dataset (Russakovsky et al., 2015).

Fig. 3. Overview of Tetraband, our implementation of Adaptive Metamorphic Testing. It is a light-weight implementation that allows extension and adaptation to other environments and settings.

One component provides the SUTs used in our case studies, i.e. the image classification and object detection systems. These SUTs are encapsulated via the OpenAI Gym[3] interface (Brockman et al., 2016). Having separate, standardized environments allows easier reproduction of the experiments and their usage in other work. Their functionality includes the feature extraction for each SUT, as well as the application of the available MRs. The metamorphic relations for image manipulation are realized with the imgaug library (version 0.2.6) for image augmentation[4]. Further details for the setup of the SUTs are given in the description of the case studies in Section 5.2.

The second component is Tetraband itself, our implementation of AMT. It is mostly an adaptation of a contextual bandit as the main actor, using the machine learning library Vowpal Wabbit 8.6.1[5]. Additionally, for comparison, we include a random agent that uniformly picks an arbitrary MR and configuration.

The contextual bandits use the doubly robust policy evaluation algorithm for learning and action selection (Dud and Langford, 2011). Exploration is performed through a combination of epsilon-greedy exploration, where it chooses a random action in 10% of the iterations, and online cover exploration (Agarwal et al., 2014) with three policies. The policy itself is approximated by a feed-forward artificial neural network with a single hidden layer with 16 neurons. We choose a moderately high exploration rate, because we not only want the bandits to converge on few single actions that repeatedly provide high payoff, but we also want to learn about the effectiveness of other actions. An important aspect of contextual bandits is that the exploration is not reduced or disabled after training, but stays active although self-adjusted to a lesser extent than in the initial iterations. This helps to re-use the bandit to test the SUT repeatedly, e.g. in Continuous Integration, because it can adapt to changing behaviors in the SUT.

5.5. Experimental setup

At each iteration, the source test case is formed by an input image and its annotations from the data set. The context feature vector is extracted through an additional neural network (Sharif Razavian et al., 2014) which processes the image and returns a feature vector of 512 floating-point numbers. The network is a pre-trained ResNet-18 network (He et al., 2016) from the PyTorch Torchvision model zoo[6]. The final layer, that usually outputs the identified image class, has been removed and the output of the previous layer

is used as the feature vector. We have also experimented with perceptual image hashing for feature extraction but did not find the hash value expressive enough.

5.6. Fault-Revealing metamorphic relations

The first experiment focuses on identifying general weaknesses in the system, i.e. identifying which metamorphic relations are fault-revealing for the system under test.

The bandit can freely choose from the seven transformations described in Section 5.3 and their reward structure. A follow-up test case fails, i.e. it violates the MR, if the transformed image is classified differently than the source image: $SUT(\mathcal{S}) \neq SUT(\mathcal{F})$. We define the MR to produce equal outputs for the source and follow-up test cases, but we could instead also consider the annotated ground-truth labels for comparison. Using the difference in outputs between source and follow-up test cases reduces the dependency on this labeling, and is thereby applicable to unlabeled datasets.

5.7. Robustness boundaries

The second experiment takes two specific transformations and learns the robustness boundaries of the SUT for these transformations. As the robustness boundary, we describe the parameterization of the transformation which changes the source test case as little as possible but is most likely to reveal a fault.

For evaluation, we use the two parameterized MRs that are already used in the other experiments, i.e. Rotation and Shear transformation. For both MRs the same parameter space as described in Section 5.3 is kept. The main difference in this experiment is that the focus lies on a single MR and its parameterization. This allows us to specifically examine the weaknesses of the SUT towards one transformation and learn about its robustness.

6. Experimental results

For each case study, we have performed two experiments: identifying fault-revealing MRs and robustness boundaries. Each run consists of one pass over the full training set, i.e. 10,000 iterations for image classification and 5000 iterations for object detection. For each of the experiments, we show the mean result of 10 runs with different random seeds. Our findings underline the effectiveness of Tetraband for controlling the metamorphic testing process in software testing, especially for testing machine learning systems.

6.1. Effectiveness of metamorphic relations on image classification

Before the evaluation of Tetraband, we first analyze the effectiveness of the selected MRs for our experiments on the image

---

[3] OpenAI Gym: https://gym.openai.com/.

[4] imgaug: https://imgaug.readthedocs.io/.

[5] Vowpal Wabbit: http://github.com/VowpalWabbit/vowpal_wabbit.

[6] PyTorch: https://pytorch.org / https://github.com/pytorch/vision.

Table 1

CIFAR-10 dataset: Effects of MRs by the true class of the image. Each cell value shows the percentage of images in the class, which are wrongly classified after applying the MR. Every class contains 1000 images. Rotation and Shear are parameterized by 30 degrees. The values are determined using the baseline method, i.e. exhaustive search over all MRs and all images. The rightmost column (Avg.) corresponds to the baseline results in the image classification case study (see Fig. 4a).

	Airplane	Automobile	Bird	Cat	Deer	Dog	Frog	Horse	Ship	Truck	Avg.
Blur	10.60	11.40	13.10	9.81	7.30	13.50	17.70	9.00	6.00	6.20	10.46
Flip L/R	2.90	1.00	4.10	6.71	2.20	6.80	1.30	2.40	0.90	2.40	3.07
Flip U/D	14.90	74.60	37.80	33.13	59.10	53.90	29.30	92.80	72.20	43.30	51.06
Grayscale	4.70	5.40	28.10	7.91	18.10	26.00	14.30	6.70	4.80	5.30	12.13
Invert	16.50	29.40	29.50	33.13	41.40	70.30	41.80	38.30	27.30	35.70	36.33
Rotation	25.49	37.09	35.43	17.70	69.00	46.10	20.63	60.44	42.44	50.01	40.43
Shear	11.22	4.99	26.69	35.79	45.45	51.97	15.63	40.24	19.78	55.24	30.70
Avg.	12.33	23.41	24.96	20.60	34.65	38.37	20.10	35.64	24.77	28.31	26.31

classification dataset, i.e. CIFAR-10. We aim to understand whether there are different effects of different MRs and how they affect certain classes of images. To this end, all MRs were applied to all images of the dataset, which we refer to in the other experiments as the *baseline* reference and the changes in the predicted classes are observed. The baseline results reported for image classification correspond thereby to the average violation rate over all classes as shown in the rightmost column of the table.

Table 1 shows the percentage of images in a class that is affected by a MR, such that they are wrongly classified afterward. CIFAR-10 consists of ten classes of images, printed as column names. The effectiveness of the image transformations varies between both MRs and image classes. The MR Flip U/D is an example of a particularly effective transformation that affects over 50% of the images in the test set. However, when noting the different classes in the dataset, all images have a clear vertical orientation and the flipped image of the objects is unlikely to occur in the dataset, for example for automobiles where 74.6% of the images are misclassified. Images of frogs or airplanes, where the perspectives of the images vary more naturally are less affected, but still to a moderately high degree compared to other MRs. This is different for the MR Flip L/R where the horizontal orientation is reversed, which corresponds to a variation that is already included in the training data and therefore is the least effective MR. Other MRs identify stronger differences between classes. Converting the image to grayscale has little effect on airplanes or ships, which commonly have few distinct features related to color, but large effects for birds or dogs, where colors are more distinctive characteristics.

Conclusively, from the experiment using the image classification dataset, we see different effects per MR and image, which underlines the motivation to learn which MRs are effective for a particular image.

6.2. Fault-Revealing metamorphic relations
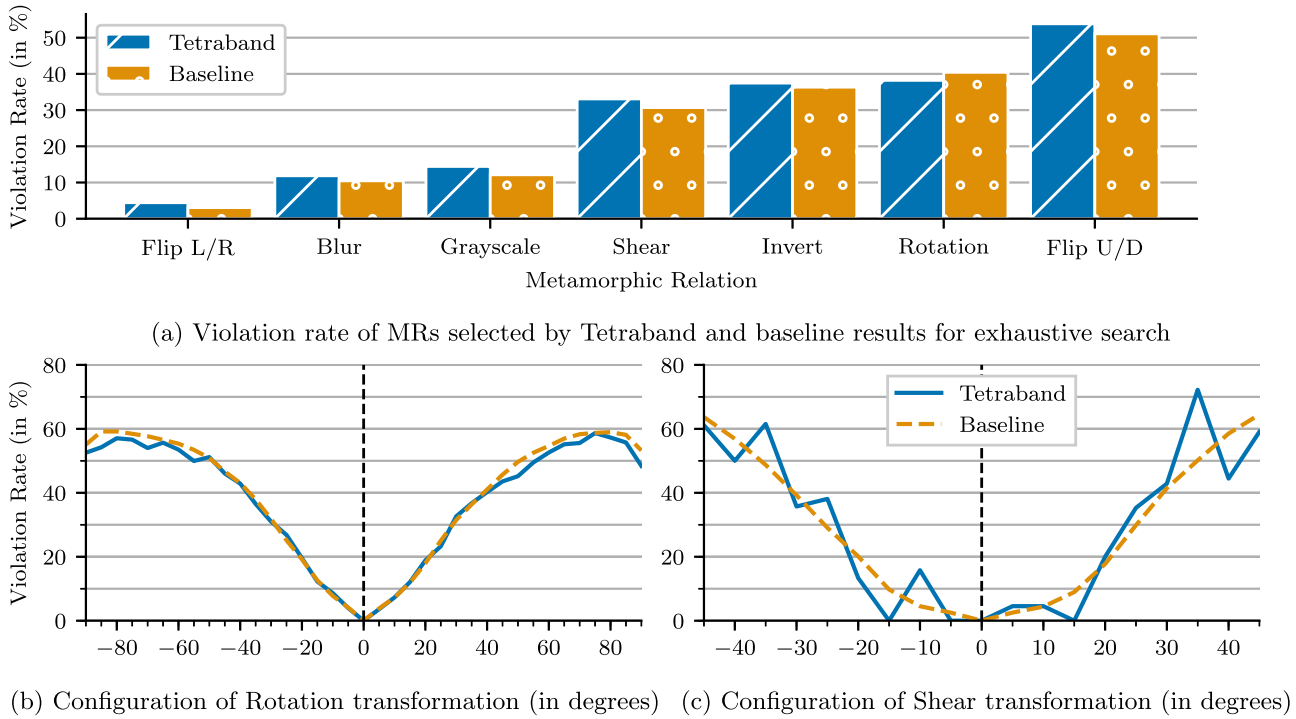
6.2.1. Case study 1: Image classification

In our first case study, which is an image classification ML model, the goal of the bandit is to select a MR which leads to a different classification of the transformed image compared to the original image of the source test case. While we looked at the true classes of the source images in the previous initial experiment, we only consider the change of the predicted class from source to a follow-up test case in this experiment. This approach focuses on the consistency of outputs for different variants of the same input image, i.e., the follow-up test cases. The actual true label is not as relevant in this case, as it is unlikely to find one MR which transforms the image consistently in a way that the correct label is predicted. Additionally, the accuracy, i.e., the correctness of the outputs is usually already tested during or after the training of the model. Focusing on the difference between outputs for source and follow-up test cases furthermore allows to extend the set of source

test cases from unlabelled datasets, making it easier to enlarge the system's test suite.

For the seven main transformations, Fig. 4a shows the distribution of the violation rate for each follow-up test case generated by a selected MR. We compare this violation rate of MRs selected by Tetraband to the true violation rate for the set of source test cases, which corresponds to the rightmost column (Avg.) in Table 1. These ground-truth results form our baseline and are determined by applying all available MRs to all source test cases in an exhaustive search. While this exhaustive search covers all possible follow-up test cases, it is time- and resource-intense compared to adaptive metamorphic testing and not ideally suited for repeated testing. We have therefore considered the random selection of a MR, such as it is common in traditional metamorphic testing, as the comparison method for computational cost and resource consumption. Conclusively, to discuss the quality of the selected MRs with Tetraband, we use exhaustive search as the baseline. The evaluation of computational cost, including a comparison to the common random selection, is discussed separately in Section 6.2.3.

The *violation rate* estimates how often the image classifier changes its prediction after the MR was applied to the source image. This can be different from the *failure rate* in cases where the prediction on the original image was wrong, but the transformation made the model predict the correct output. However, as we do not expect the test data to be labeled, we do not require and consider this information for our experiments and instead focus on the robustness and consistency of the model for the predictions. Therefore, we mostly report the violation rate. Ideally, a perfectly-trained image classification model should not show any change and the violation rates should be zero independently of the selected MR.

Generally speaking, the violation rate distribution shows a different impact for different MRs. The least fault-revealing MR is the one that mirrors images over the middle vertical axis, i.e., flip left and right. This transformation is most likely to be found in the training datasets for the image classifier. Often the image is either symmetric by itself, like a human face or body, or included together with another image of the same object but taken from a different angle, showing a symmetric profile. Other MRs are more effective to reveal faults in the image classifier, which is related to the disturbance impact they have on the original image. They often represent transformations that could be expected in real-world applications and should be covered by a robust image classification system. As seen from the exhaustive search baseline experiment, the most fault-revealing MR identified by Tetraband showed to be flipping the image upside-down, i.e., mirroring on the middle of the horizontal axis. This is within expectations, as it results in an image, which is unlikely to be represented in the distribution of training images, e.g., an image of a car is likely to be shown with its wheels on the ground. Nevertheless, high violation rates are also observed for less invasive MRs, such as rotating the image

(a) Violation rate of MRs selected by Tetraband and baseline results for exhaustive search

(b) Configuration of Rotation transformation (in degrees)   (c) Configuration of Shear transformation (in degrees)

Fig. 4. Fault-Revealing MRs for image classification: Violation rate and configuration of parameterized MR transformations. Tetraband approximates the true error distribution and select fault-revealing MRs and their parameters.

or inverting it, and these MRs are either likely to be encountered in practical applications or preserve many of the distinctive features. Having this statistic not only allows us to identify the weaknesses of the classifier (model testing), but it can also be used as a basis to configure image augmentation techniques to train a new version of the image classification model (model training). With image augmentation, the training set of images is extended by including modified versions of the original image, through small perturbations or affinity scaling, while preserving the original label. By knowing the MRs that fail the old classifier, the necessary transformations to include in future image augmentation are known and can help to improve the performance of new models. However, not all MRs are necessarily suitable image augmentations at training time, as they might produce images that are not within the distribution of inputs, i.e. images, for which the model is trained.

The performance of Tetraband closely approximates the violation distribution for the baseline results and exceeds them for all MRs, except Rotation where the violation rate is close to the baseline violation rate. The reason for the lower violation rate is related to the necessary exploration to select an appropriate parameter to rotate the source image. Due to the large number of parameters from −90 to 90 the bandit exploration had to take ineffective actions to learn, which leads to an initially lower violation rate.

However, we also observe that longer runtime and more iterations over the dataset further increase the averaged violation rate as the bandit algorithm can focus more on exploitation than exploration. This can be explained as initial iterations do not have knowledge about the MRs effectiveness and the selection is more random, which includes selecting ineffective MRs. While these actions are valuable for exploration, they do not contribute to the violation rate. Later, when the effect of MRs has been sufficiently explored, the focus changes on also exploiting the MRs and selecting fault-revealing MRs. These actions then contribute to increasing the violation rate. Accordingly, if the system under test does not rapidly change, more iterations will increase the amount of ex-
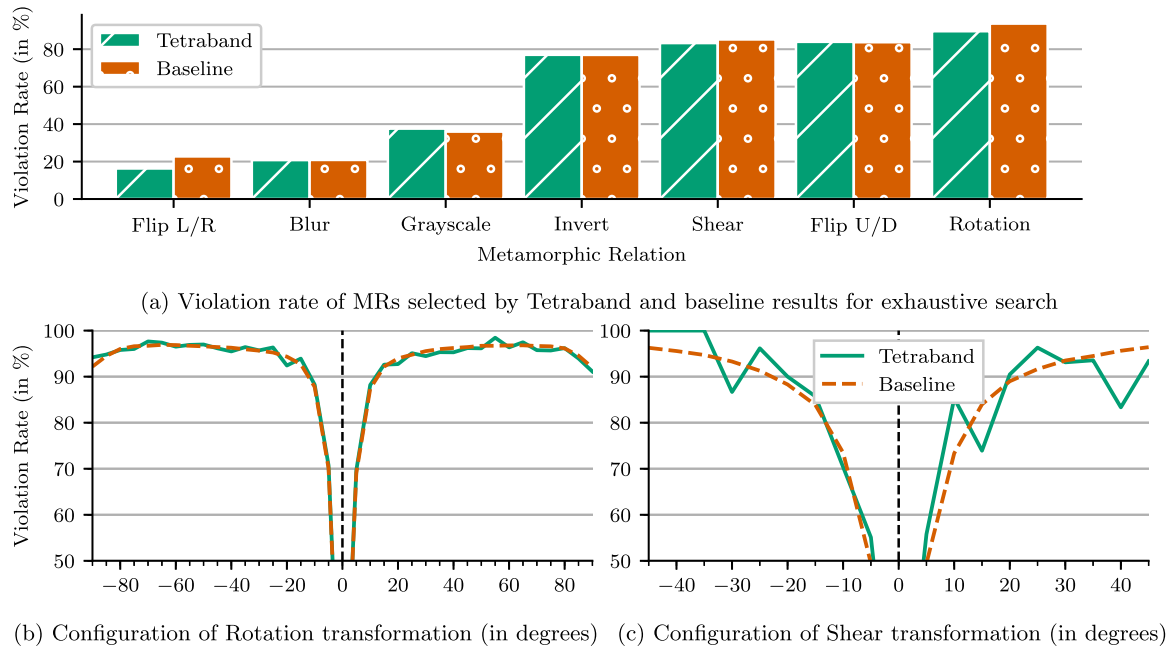
ploitation with high violation rates and the impact of the initial exploration on the violation rate decreases.

The violation distribution for the different parameters of the MRs Rotation and Shear are shown in Fig. 4b and c. The bandit effectively picks the appropriate degree of rotation to closely resemble the true error distribution. The only exceptions are the largest degrees of rotation, where the selection does not completely approximate the true distribution. However, due to our reward structure, the agent is encouraged to focus on minimal rotations in the images that lead to some misclassification. Especially in the range between −45 and 45 degrees the parameter selection is appropriate, which indicates the successful convergence towards the most revealing parameter for image rotation.

In the Shear transformation, Tetraband was less effective to smoothly approximate the true error distribution but broadly follows its shape. Here, we observe that due to the lower general violation rate of the MR, there are fewer chances for successful exploration of the parameter space than for the Rotation MR. Accordingly, the approximation of the parameter distribution for Shear is not as close as for Rotation, but still follows the general distribution.

6.2.2. Case study 2: Object detection

The second case study application is the object detection neural network. Similar to the presentation of the first case study, Fig. 5 shows the results of the object detection case study. As a first main difference, the results show a higher violation rate for object detection, with over 70% violation rate for four of the seven MRs. The ranking of MRs is similar but applying the Shear MR with parameter selection is more efficient here than in the image classification case study, where the Invert MR showed a higher violation rate, and Rotation is the most effective MR. The MRs Flip L/R, Blur and Grayscale have the lowest violation rate, i.e., the model is most robust to these changes. However, while the total violation rate is higher, the case study itself is more difficult as the dataset on which we apply Tetraband is smaller and only consists of 5000 im-

(a) Violation rate of MRs selected by Tetraband and baseline results for exhaustive search

(b) Configuration of Rotation transformation (in degrees)

(c) Configuration of Shear transformation (in degrees)

Fig. 5. Fault-Revealing MRs for object detection: Violation rate and configuration of parameterized MR transformations. Tetraband approximates the true error distribution and select fault-revealing MRs and their parameters.

ages, which means the overall process takes half of the iterations of the image classification case study. This difference explains the approximation difference for some of the main MRs in comparison to the baseline violation rates.

For the two MRs Rotation and Shear, the approximation quality for the additional parameter is similar to the image classification results. For Rotation, the distribution is close to the true distribution of the exhaustive search baseline results, related to the high overall violation rate for this MR and its corresponding higher selection and thereby better exploration opportunities. The Shear parameters match the true distribution less closely in this case study, which is also related to the higher general effectiveness of this MR for the object detection dataset.

In general, the results of the object detection case study confirm the results of the image classification case study while at the same time respecting the higher difficulty of fewer iterations, due to which the violation rate of the main MRs is close to the true violation rate, but does not exceed it after the given number of iterations.

6.2.3. Computational cost and random selection

From the previous experiments, we have evaluated the effectiveness of Tetraband for selecting fault-revealing MRs in image classification and object detection systems. However, we did not consider the computational cost of introducing machine learning in the MT process or compared Tetraband to the commonly used approach of randomly selecting MRs and their parameters. In this experiment, to answer RQ3, we analyze common characteristics for both case studies, which consider the computational cost of introduced contextual bandits and the additional learning step in the MR process. We further briefly discuss another comparison method that more closely resembles the state-of-practice in MT, which is to randomly select MRs and their parameters. A summary of the results is given in Table 2.

Running Tetraband is computationally cheaper and more sample-efficient than the exhaustive search that we consider as a baseline. While the exhaustive search considers all MRs with all different parameters for Rotation and Shear, in total 59 different transformations, Tetraband selects one MR and one parameter per

iteration. In addition to the application of the MR and the execution of the SUT, Tetraband has a small computational overhead for selecting the MR and its parameter and learning from the received feedback. The average duration per iteration in the image classification case study is 0.1s using Tetraband and 0.06s when using a random MR and not learning from feedback. While this overhead increases the execution time, it is still faster and more efficient than running all possible transformations, as we will see below. For object detection, the average duration is 0.63s with learning and 0.56s without, here the main computation lies in the object detection neural network. The overhead for training the contextual bandit could be further reduced by moving the learning step outside the main processing loop; however, we argue that in a practical application the overhead is negligible due to the lower number of executions and the availability of highly optimized contextual bandit implementations.

We also considered a random selection of MRs and their parameters. At each iteration, a random MR is sampled uniformly instead of using the bandit selection. This selection is less efficient than Tetraband. Using Tetraband, the accuracy of the image classifier is reduced from 96.6% for the unmodified images to 54.0% for the images modified by the selected MRs. With random selection, the accuracy for the modified images remains at 72.9%, which is a substantial reduction, but not as high as Tetraband. In object detection, the precision is reduced from 54.3% to 23.1% with Tetraband and to 36.3% with random selection. Due to the lower general effectiveness of random selection, we do not further discuss its results in detail.
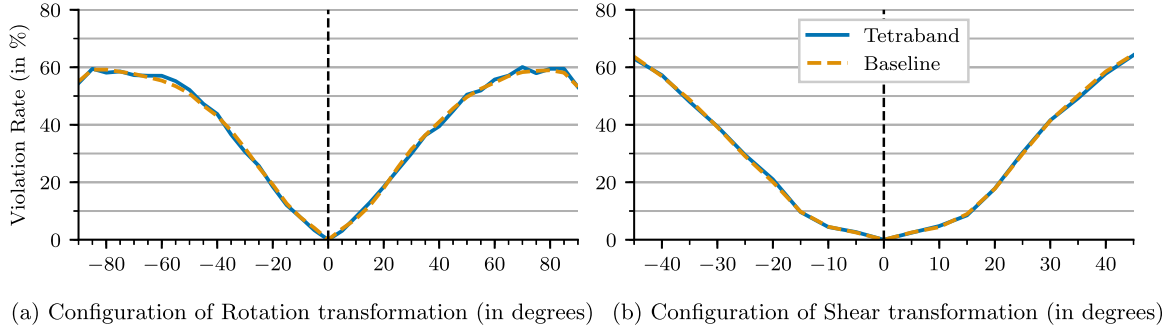
6.3. Robustness boundaries

As a second experiment and application of Tetraband to learning MR selection, we aim to find the robustness boundaries of the case study systems against different degrees of image rotations and shearing. We show the results for both case study applications in Fig. 6 for image classification and Fig. 7 for object detection. The experimental results mostly confirm the inherent hypothesis, also following the previous results from the first experiments that larger modifications of the source test case lead to a higher viola-
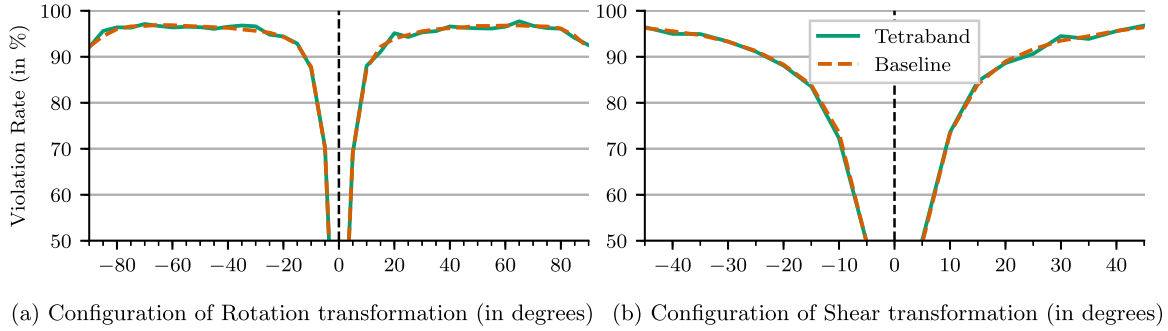
Table 2

Computational cost and accuracy of Tetraband and random selection for selecting fault-revealing MRs. The average runtime per image shows that the overhead introduced by the ML model is relatively small, especially for object detection with a more costly test execution. Tetraband selects MRs more effectively and the accuracy is smaller than with random selection.

Case Study	Runtime (in s)		Accuracy (in %)		
	Tetraband	Random	Unmodified	Tetraband	Random
Image Classification	0.1	0.06	96.6	54.0	72.9
Object Detection	0.63	0.56	54.3	23.1	36.3

(a) Configuration of Rotation transformation (in degrees) (b) Configuration of Shear transformation (in degrees)

Fig. 6. Image classification: Average violation rate per degree step. The approximated violation rate closely approximates the true violation distribution of the ground truth baseline, i.e. exhaustive search.

(a) Configuration of Rotation transformation (in degrees) (b) Configuration of Shear transformation (in degrees)

Fig. 7. Object detection: Average violation rate per degree step. The approximated violation rate closely approximates the true violation distribution of the ground truth baseline, i.e. exhaustive search.

tion rate of the follow-up test case. This is true for both case study applications, but the extent to which the effect applies varies with the object detection system being much more susceptible to images rotated even only by small degrees. At the same time, we confirm that a larger number of iterations allows better exploration and approximation of the true error distribution. Where the parameter distribution in the previous experiments showed divergence, mostly for larger parameter values, the focus on the specific MRs in this experiment allows sufficient exploration and good approximation.

The results for image classification clearly show the effect that larger rotations of the original image are more likely to cause a different classification (see Fig. 6a). Here, the bandit does not only learn to select the largest rotation but does accurately approximate the distribution of true faults as given by the exhaustive search baseline. Our results show that for more than 10% of the source test cases a rotation of at least 10 degrees leads to a different image classification. When considering real-world scenarios for the application of image classification systems, a rotation of 10 degrees can likely occur due to tilt or shifts in either the camera or due to external influences on the actual object.

For object detection, the interpretation of the results needs to consider two aspects, which lead to the conclusion that the results can not be directly compared to the image classification case study. First, the original SUT already has lower performance for the orig-

inal data set than the image classification SUT. When considering the system to be more imprecise for unmodified data, then it is also likely to be more fragile for modified data. Second, the evaluation metric used, mean average precision, is more fragile than the metric used for image classification. The results show the fragility of the object detection system, as well as the capability to learn to approximate this error distribution over transforming each of the 5000 source test case images only once.

6.4. Discussion

For both case studies in our experiments, our results showed the effectiveness of contextual bandits, as part of Tetraband, to adapt to a prior unknown error distribution in two different case study applications, based on two different neural network architectures and tasks.

From the results, we draw two major conclusions. First, we see a confirmation for the applicability of Tetraband for selecting metamorphic relations using contextual bandits (i.e., Adaptive Metamorphic Testing), as is shown by the close approximation of the true error distribution with limited iterations. Second, our tests reveal robustness weaknesses in the two systems-under-test. Weaknesses in neural networks have been addressed before and are an active research area (Goodfellow et al., 2009; Carlini and Wagner, 2017; Biggio and Roli, 2018). The research under the area of *adversarial examples* focuses on finding input perturbations that lead to misbe-

7. Conclusion

This paper introduces Adaptive Metamorphic Testing (AMT), a method to control metamorphic testing using contextual bandits. AMT receives a feature vector representing the source test case and selects a MR to generate a follow-up test case. From the result of evaluating the follow-up test case, whether it reveals a fault, the bandit learns which MRs can be used to exploit weaknesses in the system. At the same time, using state-of-the-art algorithms for contextual bandits, AMT explores non-optimal actions to identify previously unknown weaknesses and to adapt to changing behavior in repeated testing of the same system.

We have evaluated the applicability of AMT using our implementation Tetraband on two image analysis case studies in two distinct tasks. For both case studies, our results showed that Tetraband approximates the true distribution of faults in the SUT with fewer iterations and executions of the SUT than an exhaustive search and more efficiently than random sampling of MRs and their parameters, which is the common best practice. Tetraband learns to select fault-revealing MRs in relation to the source test case while ignoring non-relevant MRs. Furthermore, in the second experiment, Tetraband proved to be effective for the identification of robustness boundaries, that is, exploring the parameterization of individual MRs, which have different impacts on the SUT. Our experiment explored how different degrees of rotation and shear affected the classification result of the transformed image.

In conclusion, AMT is effective for selecting MRs and is more time-efficient than exhaustive testing and more effective than the standard approach of pure random sampling. We see the method to be useful in repeated testing scenarios, such as continuous integration, where regression of the SUT can be tested from an initial knowledge about previous fault characteristics. For future work, we plan to investigate the combination of multiple MRs to create follow-up test cases instead of selecting only one MR per iteration.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Helge Spieker: Conceptualization, Data curation, Methodology, Validation, Visualization, Software, Writing - original draft, Writing - review & editing. **Arnaud Gotlieb:** Methodology, Writing - original draft, Writing - review & editing, Supervision, Funding acquisition.

Acknowledgements

References

Agarwal, A., Hsu, D., Kale, S., Langford, J., Li, L., Oct, L.G., 2014. Taming the monster: a fast and simple algorithm for contextual bandits. In: International Conference on Machine Learning, pp. 1638–1646.

Akgün, Ö., Gent, I.P., Jefferson, C., Miguel, I., Nightingale, P., 2018. Metamorphic testing of constraint solvers. In: Hooker, J. (Ed.), Principles and Practice of Constraint Programming. Vol. 11008 of LNCS, pp. 727–736. doi:10.1007/978-3-319-98334-9_46.

Balafrej, A., Bessiere, C., Paparrizou, A., 2015. Multi-armed bandits for adaptive constraint propagation. Int. Joint Conf. Artif. Intell. 290–296.

Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S., 2015. The oracle problem in software testing: a survey. IEEE Trans. Softw. Eng. 41 (5), 507–525. doi:10.1109/TSE.2014.2372785.

Baskiotis, N., Sebag, M., Gaudel, M.-C., Gouraud, S.-D., 2006. EXIST: exploitation/exploration inference for statistical software testing. On-Line Trading of Exploration and Exploitation, NeurIPS Workshop.

Biggio, B., Roli, F., 2018. Wild patterns: ten years after the rise of adversarial machine learning. Pattern Recognit. 84, 317–331. doi:10.1016/j.patcog.2018.07.023.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., Zaremba, W., 2016. Openai gym. arXiv:1606.01540 [cs].

Cai, K.-Y., Gu, B., Hu, H., Li, Y.-C., 2007. Adaptive software testing with fixed-memory feedback. J. Syst. Softw. 80 (8), 1328–1348. doi:10.1016/j.jss.2006.11.008.

Carlini, N., Wagner, D., 2017. Towards evaluating the robustness of neural networks. In: IEEE Symposium on Security and Privacy, pp. 39–57. doi:10.1109/SP.2017.49.

Chan, W.K., Ho, J.C.F., Tse, T.H., 2010. Finding failures from passed test cases: improving the pattern classification approach to the testing of mesh simplification programs. Softw. Test. Verificat. Reliab. 20 (2), 89–120. doi:10.1002/stvr.408.

Chen, T., Cheung, S., Yiu, S., 1998. Metamorphic testing: a new approach for generating next test cases. Technical Report. Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong.

Chen, T.Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T.H., Zhou, Z.Q., 2018. Metamorphic testing: a review of challenges and opportunities. ACM Comput. Surv. 51 (1). doi:10.1145/3143561.

Ding, J., Hu, X.-H., Gudivada, V., 2017. A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data. IEEE Transactions on Big Data 26 (3). doi:10.1109/TBDATA.2017.2680460. 1–1

Dud, M., Langford, J., 2011. Doubly Robust Policy Evaluation and Learning. In: Proceedings of the 28th International Conference on Machine Learning, pp. 1097–1104.

Dwarakanath, A., Ahuja, M., Sikand, S., Rao, R.M., Bose, R.P.J.C., Dubash, N., Podder, S., 2018. 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 (ISSTA), pp. 118–128. doi:10.1145/3213846.3213858.

Goodfellow, I., Lee, H., Le, Q.V., Saxe, A., Ng, A.Y., 2009. Measuring invariances in deep networks. In: Advances in Neural Information Processing Systems, 22, pp. 646–654.

Gotlieb, A., Botella, B., 2003. Automated metamorphic testing. Proc. 27th Ann. Int. Comput. Softw. Appl. Conf. 34–40. doi:10.1109/CMPSAC.2003.1245319.

Guderlei, R., Mayer, J., 2007. Towards automatic testing of imaging software by means of random and metamorphic testing. Int. J. Softw. Eng. Knowl. Eng. 17 (6), 757–781. doi:10.1142/S0218194007003471.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Identity mappings in deep residual networks. In: European Conference on Computer Vision. In: LNCS, 9908, pp. 630–645. doi:10.1007/978-3-319-46493-0_38.

Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S., Murphy, K., 2017. Speed/accuracy trade-offs for modern convolutional object detectors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3296–3297. doi:10.1109/CVPR.2017.351.

Iandola, F.N., Han, S., Moskewicz, M.W., Ashraf, K., Dally, W.J., Keutzer, K., 2016. Squeezenet: alexnet-level accuracy with 50x fewer parameters and < 0.5MB model size. arXiv preprint arXiv:1602.07360.

Johnston, O., Jarman, D., Berry, J., Zhou, Z.Q., Chen, T.Y., 2019. Metamorphic relations for detection of performance anomalies. In: 2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET), pp. 63–69. doi:10.1109/MET.2019.00017.

Kanewala, U., Bieman, J.M., 2013. Using machine learning techniques to detect metamorphic relations for programs without test oracles. 2013 IEEE 24th Int. Sympos. Softw. Reliab. Eng. (ISSRE) doi:10.1109/ISSRE.2013.6698899.

Kanewala, U., Bieman, J.M., Ben-Hur, A., 2016. Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels. Softw. Test. Verificat. Reliab. 26 (3), 245–269. doi:10.1002/stvr.1594.

Krizhevsky, A., Nair, V., Hinton, G., 2014. The CIFAR-10 dataset. online: http://www.cs.toronto.edu/kriz/cifar.html.

Langford, J., Zhang, T., 2007. The epoch-greedy algorithm for multi-armed bandits with side information. In: Advances in Neural Information Processing Systems 20 (NeurIPS 2007), pp. 817–824.

Lattimore, T., Szepesvari, C., 2019. Bandit algorithms. Vol. Revision: 8b22b8b6131c37e388d5e3b2eecf0b4ff5d7db92. https://banditalgs.com/.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521 (7553), 436–444. doi:10.1038/nature14539.

Li, L., Chu, W., Langford, J., Schapire, R.E., 2010. A contextual-bandit approach to personalized news article recommendation. In: International Conference on World Wide Web (WWW), pp. 661–670. doi:10.1145/1772690.1772758.

Lin, T.-Y., Dollar, P., Girshick, R., He, K., Hariharan, B., Belongie, S., 2017. Feature pyramid networks for object detection. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, pp. 936–944. doi:10.1109/CVPR.2017.106.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L., 2014. Microsoft COCO: common objects in context. In: European Conference on Computer Vision. In: LNCS, 8693, pp. 740–755. doi:10.1007/978-3-319-10602-1_48.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-y., Berg, A.C., 2016. SSD: single shot multibox detector. In: European Conference on Computer Vision. In: LNCS, 9905, pp. 21–37. doi:10.1007/978-3-319-46448-0_2.

Loth, M., Sebag, M., Hamadi, Y., Schoenauer, M., 2013. Bandit-based search for constraint programming. In: International Conference on Principles and Practice of Constraint Programming, pp. 464–480.

Lu, T., Pal, D., Pal, M., 2010. Contextual multi-armed bandits. In: Proceedings of the 13th International Conferenceon Artificial Intelligence and Statistics (AISTATS), pp. 485–492.

Ma, L., Liu, Y., Zhao, J., Wang, Y., Juefei-Xu, F., Zhang, F., Sun, J., Xue, M., Li, B., Chen, C., Su, T., Li, L., 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering - ASE 2018, pp. 120–131. doi:10.1145/3238147.3238202.

Mayer, J., Guderlei, R., 2006. An empirical study on the selection of good metamorphic relations. In: 30th Annual International Computer Software and Applications Conference, pp. 475–484. doi:10.1109/COMPSAC.2006.24.

Mayer, J., Guderlei, R., 2006. On random testing of image processing applications. In: 2006 Sixth International Conference on Quality Software (QSIC'06), pp. 85–92. doi:10.1109/QSIC.2006.45.

Mekala, R.R., Magnusson, G.E., Porter, A., Lindvall, M., Diep, M., 2019. Metamorphic detection of adversarial examples in deep learning models with affine transformations. In: Proceedings of the 4th International Workshop on Metamorphic Testing, pp. 55–62. doi:10.1109/MET.2019.00016.

Murphy, C., Kaiser, G., Hu, L., Wu, L., 2008. Properties of machine learning applications for use in metamorphic testing. Proc. 20th Int. Conf. Softw. Eng. Knowl. Eng. (SEKE) 867–872. doi:10.7916/D8XK8PFD.

Ontañón, S., 2017. Combinatorial multi-armed bandits for real-time strategy games. J. Artif. Intell. Res. 58 (1), 665–702. doi:10.1613/jair.5398.

Pei, K., Cao, Y., Yang, J., Jana, S., 2017. DeepXplore: automated whitebox testing of deep learning systems. In: Proceedings of the 26th Symposium on Operating Systems Principles. ACM Press, pp. 1–18. doi:10.1145/3132747.3132785.

Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M.P., Shyu, M.-L., Chen, S.-C., Iyengar, S.S., 2018. A survey on deep learning: algorithms, techniques, and applications. ACM Comput. Surv. 51 (5), 1–36. doi:10.1145/3234150.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015. Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. 115 (3), 211–252. doi:10.1007/s11263-015-0816-y.

Saha, P., Kanewala, U., 2019. Fault detection effectiveness of metamorphic relations developed for testing supervised classifiers. In: 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), pp. 157–164. doi:10.1109/AITest.2019.00019.

Segura, S., Fraser, G., Sanchez, A.B., Ruiz-Cortes, A., 2016. A survey on metamorphic testing. IEEE Trans. Softw. Eng. 42 (9), 805–824. doi:10.1109/TSE.2016.2532875.

Segura, S., Troya, J., Durán, A., Ruiz-Cortés, A., 2018. Performance metamorphic testing: a proof of concept. Inf. Softw. Technol. 98, 1–4. doi:10.1016/j.infsof.2018.01.013.

Shahri, M.P., Srinivasan, M., Reynolds, G., Bimczok, D., Kahanda, I., Kanewala, U., 2019. Metamorphic testing for quality assurance of protein function prediction tools. In: 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), pp. 140–148. doi:10.1109/AITest.2019.00017.

Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S., 2014. cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 806–813.

Sun, L., Zhou, Z.Q., 2018. Metamorphic testing for machine translations: MT4MT. In: 2018 25th Australasian Software Engineering Conference (ASWEC), pp. 96–100. doi:10.1109/ASWEC.2018.00021.

Sutton, R.S., Barto, A.G., 2018. Reinforcement Learning: An Introduction, 2nd MIT Press.

Tang, L., Rosales, R., Singh, A., Agarwal, D., 2013. Automatic ad format selection via contextual bandits. In: Proceedings of the 22nd ACM International Conference on Conference on Information & Knowledge Management, pp. 1587–1594. doi:10.1145/2505515.2514700.

Xie, X., Ho, J.W., Murphy, C., Kaiser, G., Xu, B., Chen, T.Y., 2011. Testing and validating machine learning classifiers by metamorphic testing. J. Syst. Softw. 84 (4), 544–558. doi:10.1016/j.jss.2010.11.920.

Xu, L., Towey, D., French, A.P., Benford, S., Zhou, Z.Q., Chen, T.Y., 2018. Enhancing supervised classifications with metamorphic relations. In: Proceedings of the 3rd International Workshop on Metamorphic Testing - MET '18, pp. 46–53. doi:10.1145/3193977.3193978.

Yang, S., Towey, D., Zhou, Z.Q., 2019. Metamorphic exploration of an unsupervised clustering program. In: Proceedings of the 4th International Workshop on Metamorphic Testing, pp. 48–54. doi:10.1109/MET.2019.00015.

Zhou, L., 2016. A survey on contextual multi-armed bandits. arXiv preprint arXiv:1508.03326.

Zhou, Z.Q., Sinaga, A., Susilo, W., Zhao, L., Cai, K.-Y., 2018. A cost-effective software testing strategy employing online feedback information. Inf. Sci. 422, 318–335. doi:10.1016/j.ins.2017.08.088.

Zhou, Z.Q., Sun, L., 2019. Metamorphic testing of driverless cars. Commun. ACM 62 (3), 61–67. doi:10.1145/3241979.

Zhou, Z.Q., Xiang, S., Chen, T.Y., 2016. Metamorphic testing for software quality assessment: a study of search engines. IEEE Trans. Softw. Eng. 42 (3), 264–284. doi:10.1109/TSE.2015.2478001.

Helge Spieker is a PhD student at Simula Research Laboratory in Oslo, Norway. His research interests are in the application of machine learning and artificial intelligence techniques to optimize the software testing process. Contact him at helge@simula.no.

Arnaud Gotlieb is Chief Research Scientist at Simula Research Laboratory in Norway. He is an expert on the application of Artificial Intelligence to the validation of software-intensive systems, cyber-physical systems including industrial robotics and autonomous systems. Dr. Gotlieb has co-authored more than 120 publications in Artificial Intelligence and Software Engineering and developed several tools for testing critical software systems. He was the scientific coordinator of the French ANR-CAVERN project (2008–2011) for Inria, led the Research-Based Innovation Center Certus dedicated to Software Validation and Verification (2011–2019) at Simula. He was awarded with a prestigious Norwegian grant for the T-LARGO project on testing learning robots (2018–2022). He leads the industrial pilots experiments of the H2020 AI4EU Project (2019–2022), dedicated to the construction of the European AI-on-demand platform. Dr. Gotlieb has served in many PCs including ICSE-SEIP, ICST, ISSRE, IJCAI, AAAI, CP, etc. He co-chaired the scientific program of QSIC 2013, the SEIP track of ICSE 2014, the Testing and Verification track of CP from 2016 to 2019. He co-chaired the First IEEE Artificial Intelligence Testing Conference in April 2019 and he is a member of the editorial board of Wileys Software Testing, Verification and Reliability Journal. Contact him at arnaud@simula.no.