Learning to Select Metamorphic Relations using Contextual Bandits

Helge Spieker
Simula Research Laboratory
Lysaker, Norway
helge@simula.no

Arnaud Gotlieb
Simula Research Laboratory
Lysaker, Norway
arnaud@simula.no

***Abstract*—Metamorphic Testing is a software testing paradigm which aims at using partial input/output relations to either check the expected outputs of a system-under-test, or to generate follow-up test cases. These metamorphic relations have been successful to replace complete test oracles and to test programs, which are considered as very difficult to validate. We propose the application of reinforcement learning to select one of multiple metamorphic relations to generate a follow-up test case per source test case. By using contextual bandits, we learn which metamorphic relation is likely to transform a source test case, such that it has more chance to fail. Our method explores the behavior of the system-under-test for a set of metamorphic relations and can exploit its weaknesses during testing. We present detailed experimental results over two major case studies in machine learning, namely image classification and object detection, and identify weaknesses and robustness boundaries.**

***Index Terms*—Metamorphic Testing, Contextual Bandits, Machine Learning**

1. Introduction

Metamorphic testing has been proposed as a paradigm to generate new test cases from an initial set of source test cases without the necessity of providing explicit test oracles for the follow-up test cases [1]. As other testing techniques, metamorphic testing cannot show that the system-under-test (SUT) is fault-free, but it can show the presence of faults, if caught by the generated test case. However, in practice, when a system has many different metamorphic relations (MRs), that is, strategies to generate and describe follow-up test cases, the number of potential test cases is large and it is resource-intensive to perform all tests.

In this paper, we propose to overcome this issue by formulating the selection of MRs as a machine learning problem, based on a *contextual bandit*. Our method, called Tetraband, defines a *test transformation bandit* which sequentially selects a MR that is expected to provide the highest pay-off, i.e., that it most likely to reveal a fault. Which MRs are likely to reveal faults is learned from 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 a Continuous Integration (CI) environment, where SUT changes are frequently integrated and tested, but also for infrequent testing, when the number of MRs is large or their execution is costly. We also discuss a second application, which is to identify robust boundaries of the MR parameters. Robust boundaries describe scenarios, where the MR can be controlled via parameters. The interest is to find parameters, that produce fault-revealing test cases with minimal changes only.

Recently, metamorphic testing has also attracted attention for the testing of learning systems [2, 3, 4], which are difficult to test with standard software testing approaches, partly because of the difficulty to provide exact test oracles. We perform an evaluation of Tetraband on two major case studies from the 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 as well as robustness boundaries discovery.

The main contribution of this paper is two-fold. First, we introduce Tetraband, an adaptive selection method for metamorphic relations and test transformations. The method is based on 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 the system is to be repeatedly tested, such as in Continuous Integration. Second, we explore the benefice of Tetraband on two major case studies coming from image analysis, namely image classification and object recognition. Our experiments show that Tetraband is highly beneficial to optimize the testing process towards the highest fault-revealing MRs.

2. Background

2.1. Metamorphic Testing

Metamorphic Testing is a testing paradigm, which can generate follow-up test cases from a set of source test cases

This work is supported by the Research Council of Norway (RCN) through the research-based innovation center Certus, under the SFI program.

without knowing the explicit expected result [1], i.e. the test oracle, of the follow-up test cases. It is effective for testing systems for which it is difficult to define exact test oracles, for example machine learning systems [2], as well as for reusing test inputs to increase the test suite size.

Central to metamorphic testing is the concept of *Metamorphic Relations* (MRs), which are high-level observable properties that must hold over inputs and outputs of the SUT.

Formally, a source test case $\mathcal{S}$ produces output $f(x)$ for the SUT f with test input x. Using the metamorphic transformation T, the follow-up test case $\mathcal{F}$ with test input $T(x)$ can be generated, and due to the metamorphic relation between $\mathcal{S}$, T and $\mathcal{F}$, the result $f(T(x))$ can be verified. Note that MRs are only partial properties, which means that only test cases which violate them, indicate the presence of bugs in the SUT. When a test suite satisfies a set of MRs, it does not guarantee the absence of faults, even though our confidence in the system correctness is increased. However, this problem is not restricted to metamorphic testing.

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 image classification does not change. In most cases, these transformations must not modify the class of these images. However, in object detection, some transformations of images which impact object location markers entail similar transformation over the outputs.

2.2. Contextual Bandits

The selection of a test transformation to apply on a source test case can be formalized as a multi-armed bandit problem with side information, also known as a contextual bandit [5]. A contextual bandit acts in discrete iterations, where each iteration corresponds to the generation and execution of one follow-up test case. The bandit is setup such that it has k arms $\mathcal{A}$, every arm representing one possible MR to generate the follow-up test case. In every iteration j, the bandit receives the source test case as context x_j. Based on earlier payoffs and the exploration strategy, the bandit chooses an arm and receives a payoff (also reward) r_{j,a_j}. As only one arm can be selected, there is also only feedback for the effect of this single arm. Afterwards, the bandit updates its strategy from the observation (x_j, a_j, r_j, a_j).

The goal of the contextual bandit is to maximize the total payoff, i.e. the cumulative reward, over all iterations $\sum_{j=1,2,3,\ldots} r_{j,a_j}$. 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.

Contextual bandits are related to the general machine learning area of reinforcement learning (RL). The main distinction between bandits and RL agents is, that bandits perceive each iteration as independent of the previous one. 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 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 [6], constraint optimization [7, 8], or real-time strategy games [9]. In this work, we apply contextual bandits to the selection of MRs in software testing.

2.3. Related Work

Metamorphic testing has been applied to a variety of domains and applications, see [10, 11] for an in-depth overview. Previous works already focused on its automation, for example by exploring algorithms to specifically identify fault-revealing inputs [12]. Other works predict the applicability of a MR for a system [13, 14]. Based on source code traces, a classification model predicts which MRs from a given set can be applied to the SUT.

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 ML systems based on software testing techniques, such as differential, multi-implementation [15] or mutation testing [16].

Because testing ML systems, due to their stochastic nature, are affected by the oracle problem in software testing [17], there has been work to especially apply metamorphic testing for this purpose. Murphy *et al.* identify a set of general MRs, that hold for a variety of machine learning algorithms [2], and are shown to be effective [18] It has also been shown, that metamorphic testing can be used for deep learning-based applications, e.g. to test the classification of biological cells [3]. Dwarakanath *et al.* identify implementation bugs in image classifiers [4]. They introduce MRs, that affect the training and test data used during model training and show how they can be applied to find implementation errors in training procedures and model architecture.

3. Test Transformation Bandit

We introduce Tetraband, a test transformation bandit, that learns to select follow-up test cases from a set of applicable metamorphic test cases. First, we present the general method and the components necessary to implement it. Afterwards, we discuss two main application scenarios in which the method can be deployed and be useful for testing.

3.1. Method

At the core of Tetraband, a contextual bandit receives a description of the source test case. Based on this context vector, the bandit selects an action, which resembles a test transformation belonging to a MR. After generating the follow-up test case, the SUT is executed and the test result evaluated according to the MR acceptance criterion.

Algorithm 1: Tetraband method for MR selection

```
B = initialize bandit
MR = load metamorphic relations
SUT = setup system under test
TS = load source test suite
while running do
    S ← draw random source test case from TS
    x ← extractContextFeatures(S)
    M ← sampleMR(B, x, MR)
    F ← transform(S, M)
    F_Result ← executeTest(F, SUT)
    verdict ← evaluate(S, F, F_Result)
    updateBandit(B, (x, M, verdict))
end
```

Algorithm 1 shows an overview of Tetraband and its steps per test iteration. The method can be directly deployed without pre-training steps, however, during first iterations the MR selection is partly random to gather initial experiences about the different MRs and their effects and 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 SUT as well as to gather additional information about the effect of MRs in different contexts.

The method is generally independent of the SUT's domain or implementation and the specific MRs that can be applied. It only requires to have access to a set of source test cases, i.e. the original test suite, the transformation functions of the available MRs, and SUT execution.

3.2. Components

The Tetraband method, as described in Algorithm 1, requires only few system-specific components for a new SUT, besides having a set of MRs and a set of source test cases on which these MRs can be applied.

Extract Context Features To be able to select an appropriate MR for a source test case, it is necessary to provide relevant information to the contextual bandit. This information is represented as a feature vector, which is usually a real-valued vector of fixed size n. The function receives the source test case as input and returns the feature vector: $extractContextFeatures : \mathcal{S} \rightarrow \mathbb{R}^n$.

Sample MR The selection is mostly handled by the internal contextual bandit algorithm and does not have to be individually implemented for a new SUT. Nevertheless, depending on the number of available MRs and the robustness of the SUT, it can be helpful to adjust the exploration parameters accordingly. This allows to focus on exploiting MR, that reveal faults in the system, or to broadly explore the effects of many different MRs.

Transform and *Evaluate* Transforming the source into a follow-up test case and evaluating the follow-up test case's result represent the main functions of the MRs and the metamorphic testing paradigm. There is no change in this functionality to comply with Tetraband.

Update Bandit After the follow-up test case has been executed and the resulted been evaluated, the bandit's policy is updated by providing it information about the initial context feature vector, the chosen MR and the test result. The update routine is internal to the contextual bandit algorithm, but choosing the appropriate reward for a failed test case has to be done while adjusting the system for a new SUT. In most scenarios, where the goal is to find the most fault-revealing MRs, as described below in Sec. 3.3.1, 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 Sec. 3.3.2.

3.3. Applications

Contextual bandits are powerful to explore the effects of the MRs in different contexts, but can also exploit the experience gathered during exploration to subsequently focus on those relations, that are most likely to reveal faults. From this properties, we identify two application scenarios for Tetraband, that we are going to discuss further and evaluate as part of the case studies.

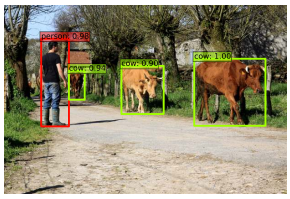
3.3.1. Fault-Revealing Test Cases. The first application is the pure selection of metamorphic relations to generate follow-up test cases, which are reveal faults in the SUT. This application, we refer to it as fault-revealing selection, is close to the typical application of metamorphic testing. All MRs are independent of each other and do not need to have any similarity to each other. As the goal is to find any faults in the SUT, each MR has the same (hidden) meaning to the selecting bandit. Accordingly, the achievable reward received for revealing a fault will be identical for all MRs.

3.3.2. Robustness Boundaries. The second application uses the exploration/exploitation trade-off of contextual bandits to identify *robustness boundaries* of the SUT. With robust boundaries, we focus on single metamorphic relations whose effect can be adjusted by external parameters. As an example taken from the case studies, while testing an image classification algorithm, a transformation is to rotate the image. The degree of rotation is here the external parameter. If the algorithm is susceptible to misclassify rotated images, it is likely, that large rotations, e.g. by 90 degrees, are more likely to be wrongly classified than smaller rotations.

4. Case Studies

We describe two case studies of testing machine learning systems, especially image analysis tasks (see Figure 1). In the first case study, an image classification system is tested. The second case study focuses on object detection.

(a) Image Classification Example (from [19]). The goal is to assign a single class label to the image.



(b) Object Detection Example (from [20]). The goal is to identify and categorize objects in the image by drawing a bounding box and assigning a class label.

Figure 1: Image classification and object detection tasks.

4.1. Image Classification

An image classification task, also image recognition, has the goal to identify the object shown on an image, e.g. assign the image to one of a fixed number of classes. The state-of-the-art method for image classification, among other image analysis tasks, are deep neural networks, such as residual neural networks (ResNets) [21] or SqueezeNet [22].

In our case study, we test a SqueezeNet model, which has been initially trained on the ImageNet corpus [19] and then fine-tuned for the 10 classes of the CIFAR-10 dataset [23]. For testing the model, we use the CIFAR-10 test set, which consists of 10,000 labeled images.

As metamorphic relations, we consider 9 image transformations: 1) blur the image by the average value of neighbour pixels, 2) flip left $\leftrightarrow$ right, 3) flip up $\leftrightarrow$ down, 4) convert to grayscale, 5) invert the image, 6) rotate by -30 degrees, 7) rotate by +30 degrees, 8) shear by -20 degrees, 9) shear by +20 degrees. The expectation, and underlying metamorphic relation between input and output, is, that, for an optimal implementation of the image analysis tasks, any of these transformations should not change the classification result.

Following this metamorphic relation between source and follow-up test case, we consider a test as filed, 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 has no knowledge about 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. 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.

4.2. 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 and 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 SUT in this case study is a pretrained object detection model, based on the open source TensorFlow Object Detection API[1] [24]. In particular, we test an implementation of a single shot multibox detector (SSD) [20] with a feature pyramid network (FPN) [25], based on a ResNet-50 network [21]. We refer the reader to the given references for an in-depth overview of the models and see the SUT as a black box for our testing purposes. 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 a classification scores at the same time. By reducing the complexity to a single model, it is a fast model for real-time object detection, that achieves state-of-the-art performance. The used model has been trained on Microsoft COCO dataset [26] and is available within the Object Detection API.

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. Because each image annotation consists of an additional bounding box per object in the image, the metamorphic relations are extended to transformation also the bounding boxes. 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 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. 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 according to different IoU thresholds, for 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 failed.

5. Experiments

For each of the case studies, we perform two sets of experiments. These experiments reflect the two applications of Tetraband, as described in Section 3.3. For both experiments, we describe how Tetraband is implemented.

5.1. Implementation

We have implemented Tetraband in Python, using Vowpal Wabbit[2] for contextual bandit functionality. Metamorphic transformations are realized with the imgaug library

1. Tensorflow Object Detection API: github.com/tensorflow/models/tree/master/research/object_detection

2. Vowpal Wabbit: github.com/VowpalWabbit/vowpal_wabbit

for image augmentation.[3] Our implementation is available at http://github.com/helges/tetraband (available before camera-ready). All experiments are implemented as environments for OpenAI Gym,[4] a toolkit for comparing reinforcement learning algorithms. This allows to reuse the experiments for other selection strategies.

At each iteration, the source test case consists of an input image and the annotations for this image from the data set. The context feature vector is extracted through an additional neural network, that processes the image and returns a feature vector of 512 float numbers. The network is a pretrained ResNet-18 network [21] from the PyTorch Torchvision model zoo.[5] The final layer, that 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.

The contextual bandit uses the doubly robust policy evaluation algorithm for learning and action selection [27]. 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 [28] with five policies. The policy itself is approximated by a feed-forward artificial neural network with a single hidden layer with 16 nodes. We choose a comparatively high exploration rate, because we do not only want the bandit 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. This is helpful for re-using the bandit to test the SUT repeatedly, e.g. in Continuous Integration, because it can adapt to changing behaviours in the SUT.

5.2. 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 nine transformations described in Sec. 4.1. If the transformed reveals a fault, i.e. the transformed image is classified differently than the original image, the bandit receives a positive reward of 1, otherwise it receives reward 0.

5.3. Rotation Robustness

The second experiment takes a specific transformation, i.e. image rotation, and learns the robustness boundaries of the SUT for this transformation. As the robustness boundary, we describe the parametrization of the transformation which changes the source test case as little as possible, but is most likely to reveal a fault.

3. imgaug: https://imgaug.readthedocs.io/
4. OpenAI Gym: https://gym.openai.com/
5. PyTorch: https://pytorch.org / https://github.com/pytorch/vision

The transformation in this experiment is the image rotation, which is parameterized by the degree of rotation. We include 36 parameter values in steps of 5 degrees in the degree range $[-90; 90]$, excluding rotations of 0 degrees. The reward is set up such that the smallest rotation 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 rotation has always higher payoff than a larger rotation, if successful.

6. Experimental Results

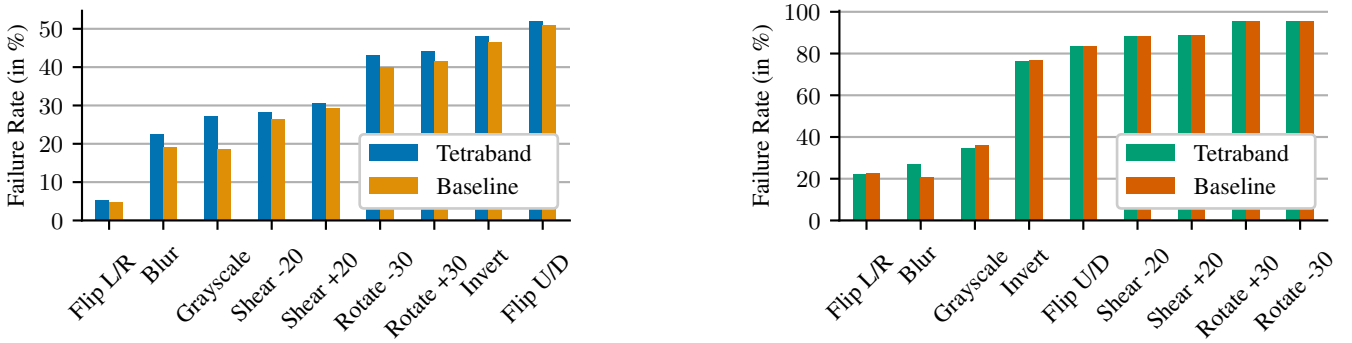
We have performed each two experiments, that is identifying fault-revealing MRs and rotational robustness boundaries. Each run consists of one pass over the full training set, i.e. 10000 iterations for image classification and 5000 iterations for object detection. For each of the experiments, we show the mean result of 10 runs. Our findings underline the effectiveness of Tetraband for selecting MR in software testing, especially for testing machine learning systems.

6.1. Fault-Revealing Metamorphic Relations

6.1.1. Case Study 1: Image Classification. In the first case study, image classification, 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. For the nine basic transformations, Figure 2b shows the distribution of the failure rate over the all available MRs. We compare the failure rate of follow-up test cases generated via Tetraband to the true failure rate for the set of source test cases. These baseline ground truth results are determined by applying all available MRs to all source test cases and recording the results.

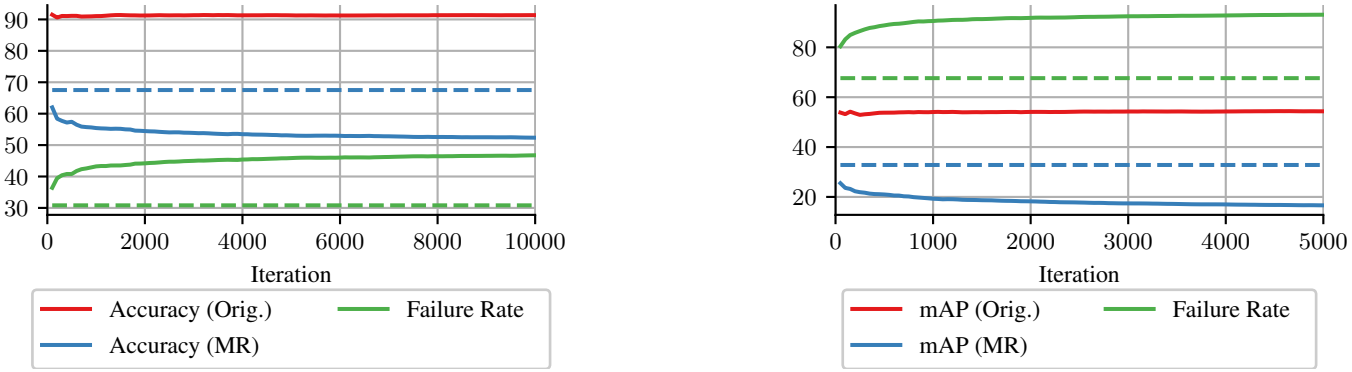
In general, there is an impact distribution over the several MRs. The least fault-revealing MR is to mirror the image over the horizontal axis, i.e. flip left and right. This transformation is most likely to be found in the training data for the image classifier. Often the image is either is symmetric by itself, e.g. like a face, or it is likely to be included from both direction, e.g. like a photo of a car from both sides. 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, while, at the same time, they represent transformations that could be expected in real-world applications and should be covered by a robust image classification system. The most fault-revealing MR showed to be flipping the image upside-down, i.e. mirroring on the vertical 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 failure rates are also observed for less invasive MRs, such as rotating the image or inverting it, and these MRs are either likely to be encountered in practical applications (rotation) or preserve many of the distinctive features (invert). Having this statistic not only allows to identify the weaknesses of the classifier, but it can



(a) Image Classification (b) Object Detection

Figure 2: Fault-Revealing MRs: Action and Failure Rate Distribution: For both case studies does Tetraband result in a close approximation of the true error distribution.



(a) Image Classification (b) Object Detection

Figure 3: Fault-Revealing MRs: Failure Rate and system performance during runtime (all in %). Dashed lines show the reference value from the full baseline. During its execution, Tetraband causes a higher failure rate than the baseline.

also be used as a based to configure image augmentation techniques to train new version of the image classification model. With image augmentation, the training set of images is extended by including modified versions of the original image, e.g. 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.

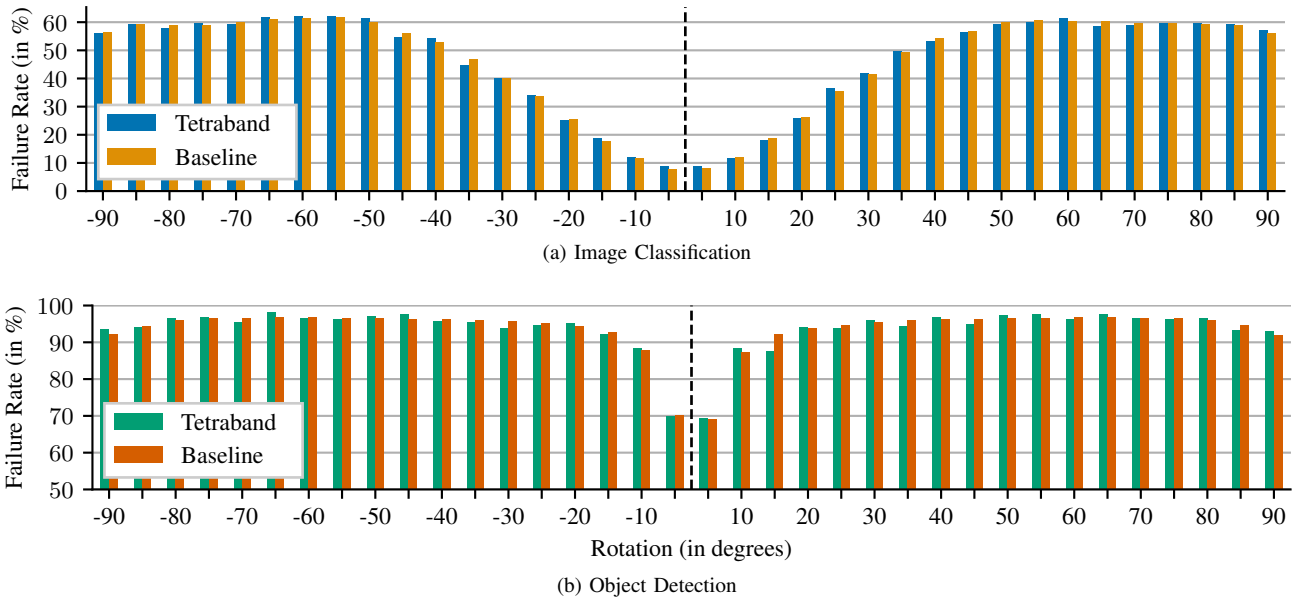
To better understand the learning process of the underlying contextual bandit in Tetraband, we plot the failure rate as well as accuracy for original and transformed image over time in Figure 3b as a running average. First, there is a big difference in accuracy, in terms of whether the classifier predicted the correct class label for the image, between the original and transformed images. This underlines the effectiveness of Tetraband to select the most fault-revealing MRs, but at the same time emphasizes the need for more robust classifiers, that are not as susceptible to modified images. The progress over time itself shows an initial learning phase of a few 100 iterations, where the bandit explores and adapts to the distribution of faults in the SUT, before finding a stable distribution of fault-revealing MRs to exploit.

The dashed lines in Figure 3b (and Figure 3b) show the mean accuracy and failure rate in every iteration, if all available MRs were applied to the original image. That means, instead of applying only one generating one follow-up test case for the selected MR, nine follow-up test cases are generated. As the results show both a lower accuracy and higher failure rate, they indicate the ability of Tetraband to prioritize fault-revealing MRs.

6.1.2. Case Study 2: Object Detection. Similar to the presentation for the image classification case study, Figure 2b and 3b show the results of the object detection case study. As a first main difference, the results show a higher failure rate for object detection, with over 80% failure rate for four of nine MRs. The ranking of MRs is similar. The MRs Flip L/R, Blur and Grayscale have the lowest failure rate, i.e. the model is most robust to these changes.

6.2. Rotation Robustness

As a second experiment, we aim to find the robustness boundaries of the case study systems against different



(a) Image Classification

(b) Object Detection

Figure 4: Rotation robustness: Average failure rate per degree step. The approximated failure rate closely approximates the true failure distribution of the ground truth baseline.

degrees of image rotations. We show the results for both case study applications in Figure 4. 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 failure 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.

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 Figure 4a). 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 baseline search. 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 easily happen due to tilt or shifts in either the camera or due to external influences on the actual object.

For object detection, an 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 original data set than the image classification SUT, as can be seen in Figure 3. When considering the system to be more imprecise for unmodified data, than it is also likely to be more fragile for modified data. Second, the evaluation metric used, mean average precision, is more fragile than the the metric used for image classification. The results show a high 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.

6.3. Discussion

For both case studies and both 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 take two major conclusions. First, we see a confirmation for the applicability of Tetraband for selecting metamorphic relations using contextual bandits, 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 area of active research [29, 30]. The research under the area of *adversarial examples* focuses on finding input perturbations that lead to misbehaviour of the model with only minimal or hard-to-detect changes in the input. This approach is different to the setting of our experiment. We select distinct and known image transformations to modify the image without the goal to hide the transformation, which often is the intent of an adversarial example.

7. Conclusion

This paper introduces a method to select metamorphic relations for source test cases by using a contextual bandit. Our method, Tetraband, 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, Tetraband exploits non-optimal actions to learn about before unknown weaknesses and adopt to changing behaviour in repeated testing of the same system.

We have presented the applicability of our method on two case studies with each two applications. For both case studies, our results showed that Tetraband is able to approximate the true distribution of faults in the SUT with only 11% respectively 3% of the executions compared to an exhaustive search, depending on the experiment. 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 robust boundaries, that is exploring the parametrization of MRs, which have different impacts on the SUT. Our experiment explored how different degrees of rotating the original image affected the classification result of the transformed image.

In conclusion, Tetraband is effective for selecting MRs and is more time-efficient than exhaustive testing. 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 explore the simultaneous selection of both a MR and its parameters.

References

[1] T. Chen, S. Cheung, and S. Yiu, "Metamorphic Testing: A New Approach for Generating Next Test Cases," Department of Computer Science, The Hong Kong University of Science and Technology, Tech. Rep., 1998.

[2] C. Murphy, G. Kaiser, L. Hu, and L. Wu, "Properties of Machine Learning Applications for Use in Metamorphic Testing," *International Conference on Software Engineering and Knowledge Engineering (SEKE)*, pp. 867–872, 2008.

[3] J. Ding, X.-H. Hu, and V. Gudivada, "A Machine Learning Based Framework for Verification and Validation of Massive Scale Image Data," *IEEE Transactions on Big Data*, vol. 26, no. 3, 2017.

[4] A. Dwarakanath, M. Ahuja, S. Sikand, R. M. Rao, R. P. J. C. Bose, N. Dubash, and S. Podder, "Identifying implementation bugs in machine learning based image classifiers using metamorphic testing," in *International Symposium on Software Testing and Analysis (ISSTA)*, 2018, pp. 118–128.

[5] J. Langford and T. Zhang, "The Epoch-Greedy Algorithm for Multi-armed Bandits with Side Information," in *Advances in Neural Information Processing Systems (NIPS)*, 2007.

[6] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *International Conference on World Wide Web (WWW)*, 2010.

[7] M. Loth, Y. Hamadi, and M. Schoenauer, "Bandit-based Search for Constraint Programming," in *International Conference on Principles and Practice of Constraint Programming (CP)*, 2013.

[8] A. Balafrej, C. Bessiere, and A. Paparrizou, "Multi-armed bandits for adaptive constraint propagation," in *International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[9] S. Ontañón, "Combinatorial Multi-armed Bandits for Real-Time Strategy Games," *Journal of Artificial Intelligence Research*, vol. 58, no. 1, pp. 665–702, 2017.

[10] S. Segura, G. Fraser, A. B. Sanchez, and A. Ruiz-Cortes, "A Survey on Metamorphic Testing," *IEEE Transactions on Software Engineering*, vol. 42, no. 9, pp. 805–824, 2016.

[11] T. Y. Chen, F.-C. Kuo, H. Liu, P.-L. Poon, D. Towey, T. H. Tse, and Z. Q. Zhou, "Metamorphic Testing: A Review of Challenges and Opportunities," *ACM Computing Surveys*, vol. 51, no. 1, pp. 1–27, 2018.

[12] A. Gotlieb and B. Botella, "Automated Metamorphic Testing," in *International Computer Software and Applications Conference (COMPAC)*, 2003, pp. 34–40.

[13] U. Kanewala and J. M. Bieman, "Using machine learning techniques to detect metamorphic relations for programs without test oracles," in *International Symposium on Software Reliability Engineering (ISSRE)*, 2013.

[14] U. Kanewala, J. M. Bieman, and A. Ben-Hur, "Predicting metamorphic relations for testing scientific software: a machine learning approach using graph kernels," *Software Testing, Verification and Reliability (STVR)*, vol. 26, no. 3, pp. 245–269, 2016.

[15] K. Pei, Y. Cao, J. Yang, and S. Jana, "DeepXplore: Automated Whitebox Testing of Deep Learning Systems," in *Symposium on Operating Systems Principles (SOSP)*, 2017.

[16] L. Ma, F. Zhang, J. Sun, M. Xue, B. Li, F. Juefei-Xu, C. Xie, L. Li, Y. Liu, J. Zhao, and Y. Wang, "DeepMutation: Mutation Testing of Deep Learning Systems," in *International Symposium on Software Reliability Engineering (ISSRE)*, 2018, pp. 100–111.

[17] E. T. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo, "The Oracle Problem in Software Testing: A Survey," *IEEE Transactions on Software Engineering*, vol. 41, no. 5, pp. 507–525, 2015.

[18] X. Xie, J. W. Ho, C. Murphy, G. Kaiser, B. Xu, and T. Y. Chen, "Testing and validating machine learning classifiers by metamorphic testing," *Journal of Systems and Software*, vol. 84, no. 4, pp. 544–558, 2011.

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

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision (ECCV)*, 2016, pp. 630–645.

[22] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size," 2016. arXiv: 1602.07360.

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

[24] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy, "Speed/Accuracy Trade-Offs for Modern Convolutional Object Detectors," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296–3297.

[25] T.-y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for Object Detection," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 936–944.

[26] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 740–755.

[27] M. Dud and J. Langford, "Doubly Robust Policy Evaluation and Learning," in *International Conference on Machine Learning (ICML)*, 2011.

[28] A. Agarwal, D. Hsu, S. Kale, J. Langford, L. Li, and L. G. Oct, "Taming the Monster : A Fast and Simple Algorithm for Contextual Bandits," in *International Conference on Machine Learning (ICML)*, 2014, pp. 1638–1646.

[29] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng, "Measuring Invariances in Deep Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2009, pp. 646–654.

[30] N. Carlini and D. Wagner, "Towards Evaluating the Robustness of Neural Networks," in *IEEE Symposium on Security and Privacy*, 2017, pp. 39–57.