Received: Added at production | Revised: Added at production | Accepted: Added at production
DOI: xxx/xxxx

RESEARCH ARTICLE

Automated Feature Extraction for Testing Deep Learning Systems through Illumination Search

Tahereh Zohdinasab[1] | Vincenzo Riccio[2] | Paolo Tonella[1]

[1]Università della Svizzera italiana, Lugano, Switzerland

[2]University of Udine, Udine, Italy

Correspondence
Corresponding author Tahereh Zohdinasab, Email: t.zohdinasab@gmail.com

Abstract

The opacity of Deep Neural Networks (DNNs) poses challenges in understanding the causes of their misbehaviours. Illumination search characterizes the inputs of a DNN by means of relevant features and explores the resulting feature map extensively. This facilitates the interpretation of misbehaviour-inducing inputs based on the regions they occupy in the feature map. However, current illumination-based approaches necessitate human expert involvement for the definition of the features, limiting broad applicability.

In this paper, we address these limitations with DeepTheia, our fully automated illumination-based test generator that automatically extracts the features, and explores the feature space using cutting-edge diffusion models. Experimental results show that DeepTheia consistently extracts highly discriminative features. Independent human assessors certified that DeepTheia is able to group misbehaviour-inducing inputs in a way that is understandable to humans in over 78% of the cases. Moreover, the inputs generated by DeepTheia were useful in significantly improving the ability of the original DL systems to handle inputs with critical feature combinations through fine-tuning.

1 | INTRODUCTION

The widespread adoption of Deep Learning (DL) systems in various domains highlights their crucial role in modern technological advancements. These systems distinguish themselves from other software by their ability to automatically learn complex tasks from patterns in training data [29]. With the growing dependence on sophisticated DL models, such as Deep Neural Networks (DNNs), availability of a solid testing framework becomes essential. Rigorous testing should not only validate the performance of these systems, but also address concerns related to their interpretability [39, 54, 6].

Various testing approaches have been proposed to deal with the unique challenges posed by DL systems [37, 58, 10, 40, 19, 3, 9, 48, 53]. These techniques leverage advanced strategies, e.g., evolutionary search or generative models, to automatically expose misbehaviours by exercising DL systems with artificially generated data beyond the datasets used during development. However, merely exposing DL misbehaviours is not sufficient to understand the input features causing them and, thus, thoroughly evaluate the system quality. In fact, a DNN model is commonly perceived by developers as a black-box, and despite its exceptional performance, it often struggles to offer meaningful explanations for specific predictions or decisions [17, 46, 57].

Recent approaches based on illumination search, such as DeepHyperion-CS [58, 59], overcome this limitation by explicitly searching for critical, misbehaviour-inducing inputs with different features. Illumination search is a family of search based algorithms that “illuminates” the input space, i.e. finds the best solution in each region of the search space, as defined by the features of interest in the target domain. Their output consists of *feature maps*, N-dimensional grids that represent the performance of generated inputs in the space of the relevant features. Testers can greatly benefit of such an approach, since it explores the feature space at large, with the feature map highlighting the most critical input feature combinations, i.e., the map cells associated with a high percentage of misbehaviour-inducing inputs. Feature maps proved to be extremely useful in several testing tasks, e.g., test selection [34], test adequacy assessment [14, 4], failure prediction [5], and misbehaviour explanation [57].

A crucial problem of illumination search algorithms is the definition of the features, as these are usually problem- and domain-specific. The authors of DeepHyperion-CS introduced a systematic methodology for defining features within a domain of interest. This methodology involves multiple human experts with domain knowledge, who identify features (i.e., map dimensions) and metrics for quantifying such features. Additionally, the approach requires human effort also for designing models of the input to be perturbed by mutation operators. Indeed, involving human experts contributes to more meaningful and understandable feature dimensions. However, the careful engineering required for defining features, metrics and input models is not trivial and may impose several limitations on the applicability of this testing approach, as sometimes it is not feasible at all to do it manually.

In this paper, we propose a novel approach, DeepTheia, that tackles the limitations of the state of the art by introducing automated feature extraction and input perturbation operators based on modern generative DL, which greatly reduce the costs of illumination search. Specifically, DeepTheia leverages the knowledge about the target domain automatically learned by a DNN to extract the features that better capture the main characteristics of test inputs. This information is easily accessible from the weights of the internal layers of a general-purpose feature extractor or the network under test.

We evaluated our proposed technique on two different popular image classification problems with increasing complexity, i.e., classification of handwritten digits and classification of real-world images. Our results show that, for both problems, DeepTheia produces feature maps that can identify the combinations of feature values that trigger misbehaviours of the DL system. In particular, DeepTheia outperforms the features manually defined by experts in problems where DeepHyperion-CS was already applied with success. Remarkably, the features automatically extracted by DeepTheia perform well also for ImageNet [43], where human-defined, measurable features could not be defined. Furthermore, we conducted a study involving independent human assessors. Results show that the automatically extracted features produce cohesive groups of inputs mapped to the same cell. This means that images with similar feature values (i.e., from the same map cell) are perceived as similar by humans, suggesting potential human interpretability of the feature map cells. Finally, we demonstrated the usefulness of DeepTheia in improving the quality of the system under test. By fine-tuning the system with inputs generated by DeepTheia, we improved its performance for feature combinations that were not initially handled (0% accuracy). This fine-tuning led to a remarkable improvement (up to 99.96% accuracy) with no significant regressions.

In comparison to the original paper describing DeepHyperion-CS, the main extensions that can be found in this paper are:

- We integrated an automated approach to extract discriminative input features with illumination search to detect diverse misbehaviour-inducing inputs.
- We proposed a novel input perturbation method for generating new images using diffusion models.
- We compared our new approach with the state-of-the-art approach DeepHyperion-CS.
- We assessed the cohesiveness of the generated feature maps by conducting a human study.

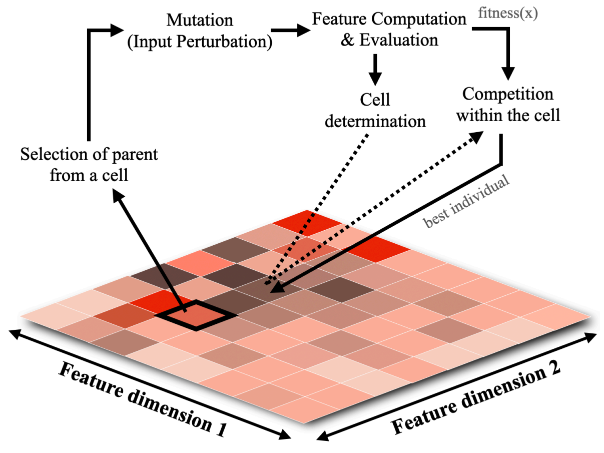

Mutation (Input Perturbation)
Feature Computation & Evaluation
fitness(x)
Cell determination
Competition within the cell
Selection of parent from a cell
best individual
Feature dimension 1
Feature dimension 2

FIGURE 1 Overview of the MAP-Elites algorithm.

- We assessed the usefulness of our approach in improving the system under the test through fine-tuning.
- We provide a comprehensive replication package to support reproduction of the results obtained in this study: https://doi.org/10.5281/zenodo.12805149

2 | ILLUMINATION SEARCH FOR TESTING DL SYSTEMS

Illumination search is a family of search based algorithms that balance *exploitation*, i.e., the mechanisms that reward the most promising inputs, with *exploration*, which allows to explore the search space at large by promoting input diversity. Illumination search has been already used for testing DL systems. Specifically, DeepHyperion-CS [58, 59] adopted MAP-Elites [31], a popular illumination search algorithm [52] that characterizes the search space as a *feature map*, whose N axes are the relevant dimensions of variation, i.e., the features. It aims to fill the feature map with the best performing individuals (i.e, inputs that expose or are close to exposing misbehaviours of the DL system, in the case of DeepHyperion-CS). Diversity among inputs is ensured by generating inputs that cover different areas of the feature space. Figure 1 illustrates the main loop of this algorithm. MAP-Elites starts by filling an empty N-dimensional feature map with an initial population to be evolved. Its evolutionary search process continues until the termination of the execution budget. In each iteration, MAP-Elites selects an individual occupying a cell of the current map and mutates it to generate a new individual. To determine the cell corresponding to the new individual, MAP-Elites computes its feature values. If the identified map cell already contains an individual, MAP-Elites places the individual with the higher fitness value in the map, thus performing a local competition. When the termination condition is satisfied, the algorithm outputs the feature map containing the highest fitness individuals. Intuitively, a suitable fitness function for testing DL systems should quantify how close the DL system is to exhibit a misbehaviour [37, 13]. Therefore, DeepHyperion-CS

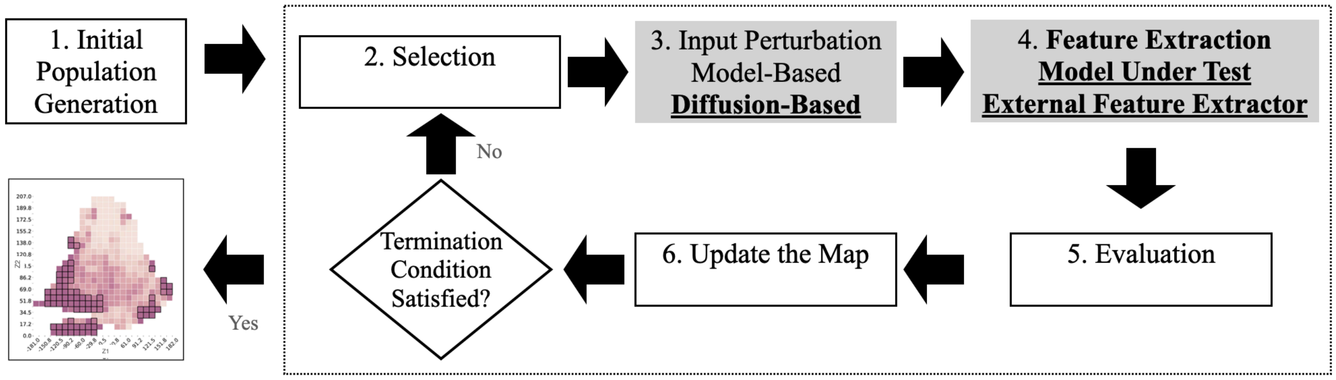


FIGURE 2 Overview of DeepTheia. The main novel contributions to illumination search for DL systems are highlighted in boldface within grey boxes.

employs a problem-specific fitness function that quantifies how close the DL system is to misbehaving when exercised with the evaluated input. By minimizing this fitness function, DeepHyperion-CS identifies inputs that are more likely to trigger misbehaviours.

$$\min \ fitness_2(x) = \min \ evaluateBehaviour(x) \tag{1}$$

The DeepHyperion-CS approach relies on human experts for feature definition and input model design, which requires domain expertise. In contrast, our approach overcomes these limitations by automating feature definition and introducing input perturbation operators using generative DL. This results in a significant reduction of the costs associated with the illumination search process.

Figure 2 presents an overview of DeepTheia's illumination search and highlights our proposed solutions for fully automating the test generation process. First, DeepTheia generates an initial population by selecting random seeds from the original dataset (step 1). At each iteration, it selects an individual from a map cell (step 2). DeepTheia can perturb inputs by leveraging diffusion models, besides input models explicitly defined by test engineers (step 3). DeepTheia applies slight perturbations to inputs with known expected labels, operating on the assumption that the perturbed image should preserve the expected label. This assumption aligns with the common practice of most test generators designed for assessing image classifiers [38]. Unlike DeepHyperion-CS, DeepTheia extracts features automatically using its feature extractor component that exploits a transfer learning approach (step 4). Then it evaluates the inputs using a fitness function that indicates the closeness to misbehaviour (step 5). Since in this work we focus on image classifiers, a misbehaviour occurs when the label predicted by the classifier under test differs from the expected label. Thus, we adopt as fitness function the *misclassification distance*, computed as the difference between the activation value of the neuron associated with the expected label and the highest incorrect activation from the DNN's softmax layer output (hence, we aim to minimize it until it becomes negative as a misclassification occurs) [37]. Following the MAP-Elites algorithm, the individuals are then placed in the map in the corresponding position based on their feature values (step 6). Finally, when the termination condition is satisfied, DeepTheia reports the final feature map. In the next sections, we detail the key contribution of the DeepTheia approach, i.e., automated feature extraction (section 3) and input perturbation strategies, which include a novel diffusion-based input perturbation technique (section 4).

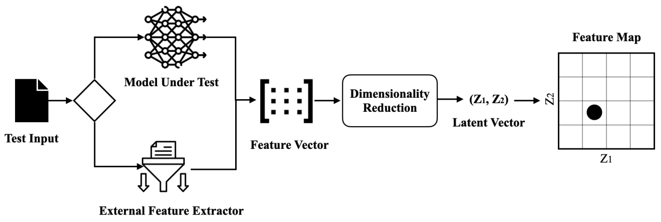


FIGURE 3 Automated feature extraction from test inputs.

3 | AUTOMATED FEATURE EXTRACTION

A crucial element of our approach is automated feature extraction, as it directly influences the ability of the test generator to produce a diverse test suite. Indeed, the extracted features define the feature map, which is progressively populated with the most promising inputs during exploration. The key characteristics of the features required by illumination search are that they must be discriminative and quantifiable.

DeepTheia leverages features automatically extracted from inner layers of DNNs, capturing patterns in the data like combinations of shapes, colors, textures. The understandability of these features is notoriously challenging due to the complex, non-linear transformations in DNNs influenced by both training data and architecture. Although the individual, automatically extracted, features could be difficult to interpret in isolation, we expect that inputs with similar values of feature combinations, i.e., inputs assigned to the same cell, are recognised as coherent and cohesive groups of inputs even by humans.

Figure 3 shows an overview of our proposed approach for extracting features to be used as dimensions of the feature map. The goal of feature extraction is to identify input characteristics that allow DeepTheia to cluster similar inputs together, hence providing a similarity based explanation for the causes of DL system's misbehaviour, when a cell contains misbehaviours. We automatically generate features using the following process: firstly, using a feature extractor we obtain feature vectors, i.e., vectors of numerical values that preserve the relevant information in the original input (subsection 3.1). Then, we project features onto the latent space through dimensionality reduction, to obtain data points with reduced dimensions, while retaining the maximum possible amount of information (i.e., data variation; see subsection 3.2).

3.1 | Feature Vector Generation

As shown in Figure 3, there are two alternative ways to generate feature vectors: (1) the model under test itself; (2) an external feature extractor.

The *Model Under Test*, which is a DNN, already includes feature extraction layers that are necessary for its processing. These may be convolutional, pooling and fully connected layers, which can be used for our purposes, to extract the features associated with a given input [16, 47]. For instance, in image processing the main building blocks of DNNs are convolutional layers that extract visual features from the input. By adding pooling layers on top of convolutional layers, the model can identify such visual features. To use the model under test for feature extraction, we need to pre-process the data to reshape the input vector and generate a vector with the size required by the model. We then feed the pre-processed vector to the model and extract features. To obtain higher level features, the output of the last feature extraction layers is collected. For instance, such output can come from the last fully connected layer before the softmax layer of an image classifier. The result is an abstract feature vector of size N_c, i.e., the size of c-th layer of the DNN. In this way, we can extract high level features from inputs without having to define and train any additional feature extraction model. On the other hand, we are bounded by the capabilities of the model under test. If this is not good at feature extraction, we will obtain poorly performing features.

Alternatively, we can apply a transfer learning based feature extraction method using an *External Feature Extractor* model. Transfer learning leverages knowledge from a general domain and applies it to a specific domain through the fine tuning of a pre-trained model. Beyond the considerable time savings associated with transfer learning, empirical evidence suggests that starting with a pre-trained model can yield superior performance compared to training from scratch, even when addressing a distinct problem [49].

There are several pre-trained models which are widely used for feature extraction in the literature [23, 1, 3, 32]. For instance, VGGNet is a convolutional neural network with multiple layers (i.e., 11 to 19 layers) for image recognition proposed by the Visual Geometry Group at the University of Oxford [44]. Its feature extraction component spans from the input layer to the final maximum pooling layer that outputs the feature matrix. In order to use an external feature extractor, the input vector must be pre-processed to fit the required input size and shape of the model. Then, the processed input is fed to the model. In the case of VGGNet, the last pooling layer returns a feature matrix, which is flattened to generate the final feature vector.

Using any of the above approaches, the output of this step is a multi-dimensional feature vector that abstracts important characteristics of the input into a numerical embedding vector.

3.2 | Dimensionality Reduction

In this step, we apply dimensionality reduction techniques to the high-dimensional feature vector generated in the previous step. The goal is to obtain a lower-dimensional vector in the *latent space*, a low-dimensional representation, inferred from the distribution of input data and preserving the similarity among inputs with similar features. In this work, each dimension of the latent space corresponds to one of the axes of DeepTheia's feature map.

The rationale behind reducing their dimensionality lies in the richness and diversity of feature vectors, which contain a plethora of information: attempting to find common characteristics among different inputs in such high dimensional space and generating discriminative feature maps is impractical without dimensionality reduction. In fact, in a high dimensional space, feature map cells tend to be sparse and scarcely populated, providing little additional information to the feature map user.

Hence, we use Principal Component Analysis (PCA) [11], a statistical method commonly used for dimensionality reduction and data visualization. It transforms high-dimensional data into a new coordinate system, where the axes are the principal components. These principal components are linear combinations of the original variables, and they are chosen to capture the space directions of maximum variance in the data. To this aim, we "train" a PCA model on each considered input dataset. PCA computes the N_p space directions (eigen-vectors) where the input domain is projected (with N_p the number of reduced dimensions, defined by the user). In this way, PCA summarizes the information content in a high dimensional dataset, by transforming a large number of attributes into a smaller one, while retaining the majority of the information (variance) present in the original attribute values.

Through PCA, we can generate a pre-defined number of feature dimensions (i.e. the number of components selected by PCA) while preserving the most important information of the inputs. Once PCA is trained, we can use it to determine the latent vector corresponding to each input feature vector.

Since the latent space is continuous, we obtain the feature map coordinate along each dimension through discretization, by scaling the latent feature value $ind.l_i$ using the scaling factors α_i:

$$x_i = \lfloor \alpha_i \cdot ind.l_i \rfloor \tag{2}$$

where $ind.l_i, \forall i \in [1 : N]$ indicates an individual's latent feature values. The map granularity, representing the number of cells in each dimension, is adjustable by changing α_i, which can be tailored for a given problem to achieve the desired level of discrimination in the feature map.

4 | INPUT PERTURBATION

Illumination search algorithms use mutation as an evolutionary operator to introduce small, random changes to individuals (i.e., candidate solutions) in the population. It mimics the concept of genetic mutation in biological evolution. In the context of testing, mutation is used to generate new and potentially better solutions, by making incremental modifications to existing test inputs.

The purpose of mutation operators is to explore the solution space by generating different (and potentially better) test inputs. DeepTheia projects the solution space to a feature space with lower dimensionality, i.e., the feature map, and applies mutation operators to perturb inputs from the map cells, so as to explore the feature map at large.

Test input generators for DL typically apply small perturbations to initial seeds, i.e., inputs with known ground truth labels, under the assumption that such perturbations will not change the label. For instance, a simple method for input perturbation applies small changes directly to the input space, e.g., by modifying pixel values for images [18]. While these techniques proved to be extremely effective in assessing the robustness of the DL systems against adversarial attacks, they are limited in testing the DNN performance in the presence of unexpected data, i.e., inputs not represented in the training set that may occur during system operation. To overcome such limitations, two main families of approaches gained popularity in DL testing, i.e., the manipulation of models of the inputs [37, 40, 58] and generative DL [10, 24, 25, 30, 50]. In the following, we describe how we applied these two input perturbation approaches to our case studies, classification of handwritten digits from the MNIST dataset and classification of photo-realistic images from the ImageNet-1K dataset.

4.1 | Model-Based Input Perturbation

Inspired by classic model driven engineering, model-based input perturbation approaches leverage a model-based representation of their input domain to manipulate existing tests. Specifically, these approaches transform an input into an input model instance, which abstracts the main characteristics that define the input domain. Then, they apply a perturbation on the model, whose space is defined by the model parameters and has typically a lower dimensionality than the input space. Finally, the perturbed model is transformed back to the original input space, resulting in a concrete input vector that can be used to exercise the DL system under test.

The input model is highly tailored to the corresponding specific application domain. This is also the main limitation of these approaches, which require the existence of a high-quality model representation for the given input domain. While such models are typically available in contexts where model-based engineering approaches are adopted, e.g. in

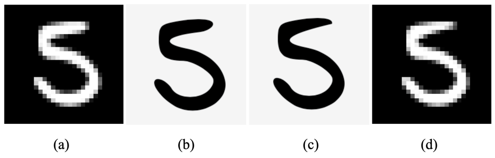


FIGURE 4 Model-based input representation and mutation. (a) original input; (b) original SVG model after vectorization; (c) SVG model mutated by moving a control point; (d) mutated input.

safety-critical domains such as automotive and avionics, we cannot assume their general availability. When available, these models proved to ensure input validity [38] and have already been successfully combined with illumination search [58, 59].

We applied a model-based input perturbation approach to the handwritten digit classification problem. As shown in Figure 4, the original format of the MNIST database consists of 28×28 images with greyscale levels that range from 0 to 255. We adopted Scalable Vector Graphics (SVG) as model representation [37]. SVG abstracts each digit as a sequence of points (start, end, and control), defining the corresponding Bézier segments. Such an abstraction is achieved using the Potrace algorithm, which executes binarization, despeckling and smoothing, to create a smooth contour composed of Bézier segments around the given image. DeepTheia's model-based mutation operator applies small displacements to the SVG model points to mutate the corresponding digit shape. To convert an SVG model back to a 28×28 grayscale image, we utilize rasterization operations through the functionalities provided by two widely used open-source graphic libraries, LibRsvg[†] and Cairo[‡].

4.2 | Diffusion-Based Input Perturbation

When the DNN input is an image, we can use generative DL to perform input perturbation. Generative DL models operate by reconstructing the underlying probability distribution of their training data as a low-dimensional latent space usually consisting of a normal multivariate probability distribution of parameters. This knowledge is then used to generate new inputs or to modify existing inputs that belong to the considered input domain.

Differently from model-based input perturbation, generative DL models do not need human effort in designing manually the input model, as the probability distribution of data is automatically extracted from the inputs used for training. For this reason, generative DL models are particularly useful when an input model is not available and is difficult to craft, e.g., for feature-rich input datasets such as real images. Recent work from the literature adopt Variational AutoEncoders (VAEs) [26] and Generative Adversarial Networks (GANs) [15]. However, these DL models may generate invalid inputs due to the lack of continuity in the

[†] https://wiki.gnome.org/Projects/LibRsvg
[‡] https://www.cairographics.org

latent space. GANs are known for their potentially unstable training and lack of diversity of generated inputs due to their adversarial training.

Overall, the quality of inputs generated by these techniques heavily relies on the quality of the training set and of the adopted generative model [38]. Hence, we prefer to adopt more recent diffusion models for generating image variations, which result in more realistic and valid images [8], by using domain-specific text prompts (i.e, a text prompt including the expected class label) that guarantee the preservation of the ground truth classification label.

4.2.1 | Diffusion Models

Diffusion models [45] are probabilistic models designed to learn a data distribution, denoted as $p(x)$, through a gradual denoising process applied to a variable sampled from a Gaussian distribution. In particular, the sampling process initiates with a noisy sample x_t and progressively generates less noisy samples $x_{t-1}, x_{t-2}, ...$ until arriving to a final sample x_0. At each time step t, there is a corresponding noise level, and x_t can be understood as a mixture of a signal x_{t-1} with some noise ϵ, where the signal-to-noise ratio is determined by the time step t. Instead of operating directly on the image, latent diffusion models (LDMs) [41] operate by repeatedly reducing noise in a latent representation space generated by a VAE. Like other categories of generative models, LDMs have the potential to characterize conditional distributions.

The training process of diffusion models follows a forward noising process and a reverse denoising process. During training, the forward process gradually adds Gaussian noise to clean data samples over T timesteps according to a predefined noise schedule $\beta_1, \beta_2, ..., \beta_T$. The model learns to reverse this process by training a neural network (typically a U-Net architecture) to predict the noise that was added at each timestep. The training objective is to minimize the variational lower bound of the negative log-likelihood, which in practice reduces to predicting the noise ϵ added to the data at timestep t:

$$\mathcal{L} = \mathbb{E}_{x_0, \epsilon \sim \mathcal{N}(0,I), t}[||\epsilon - \epsilon_\theta(x_t, t)||^2] \tag{3}$$

where ϵ_θ is the learned noise prediction network, and x_t is the noisy version of the original data x_0 at timestep t.

Text conditioned (text-to-image) LDMs [8, 35] process a text prompt fed into the noise predictor U-Net. Text-to-image models start from a random noisy image, while image-to-image latent diffusion models accept as input an image along with a textual prompt.

Fine-tuning diffusion models for specific tasks or domains involves adapting pre-trained models to new objectives while leveraging the rich representations learned during initial training. Several approaches exist for fine-tuning diffusion models. **Full fine-tuning** involves updating all model parameters on task-specific data, which can be computationally expensive but provides maximum flexibility. **Parameter-efficient fine-tuning** methods, such as LoRA (Low-Rank Adaptation) [21], update only a small subset of parameters by learning low-rank decompositions of weight updates, significantly reducing computational costs while maintaining performance. **Textual inversion** [12] learns new text embeddings for specific concepts while keeping the diffusion model frozen, enabling personalization with minimal data. **DreamBooth** [42] fine-tunes the entire model on a few images of a specific subject, binding unique identifiers to new concepts. For domain adaptation, **progressive fine-tuning** strategies can gradually adapt models from general domains to specific applications, helping to preserve general knowledge while learning domain-specific features. The choice of the fine-tuning strategy depends on factors such as available computational resources, target dataset size, and degree of domain shift from the original training data.

The model starts by encoding the input image to the latent space. During the sampling step, the noise predictor U-Net takes the latent noisy image and the textual prompt as inputs and predicts the noise in the latent space, considering the image features described within the prompt. Then, it generates a new latent vector by subtracting the noise from the input latent vector. After repeating this sampling step a predefined number of times, the VAE decodes the obtained latent vector to generate the new image. Generating realistic images while retaining the style and semantic content of the input image is challenging for image-to-image LDMs, since the latent distribution is biased in comparison to a standard Gaussian distribution. For this reason, Zhang et. al. introduced an inference pipeline called Real-world Image Variation by Alignment (RIVAL) [55] for diffusion models, which is able to generate high-quality image variations by performing adaptive cross-image attention and latent distribution alignment in the denoising steps. Using the RIVAL pipeline, it is possible to generate image variations while maintaining semantic and style consistency with the seed image (i.e. the reference image).

4.2.2 | DeepTheia's diffusion-based approach

DeepTheia generates image variations using a modern image-to-image diffusion model. This involves supplying the model with a reference image and a domain-specific textual prompt. In this way, we address our twofold objective: to use a reference image for mutation and to ensure label preservation.

In particular, we adopt a pre-trained LDM called *Stable Diffusion* [41] and fine-tune it by using images from the training set of the target class. While the pre-trained model lays a robust foundation, tailoring it to a specific dataset and task significantly enhances its efficacy, ensuring alignment with user-defined objectives and preserving style, format and other qualitative aspects of the given input domain through fine-tuning [12, 42]. The computational cost of fine-tuning the diffusion model depends on factors such as model size, dataset complexity, and hyperparameter configurations, typically requiring several hours on standard GPU hardware [30]. However, this represents a one-time setup effort per subject system, after which the fine-tuned components can be reused across multiple testing campaigns without requiring retraining, making the upfront investment cost-effective over time. By using the

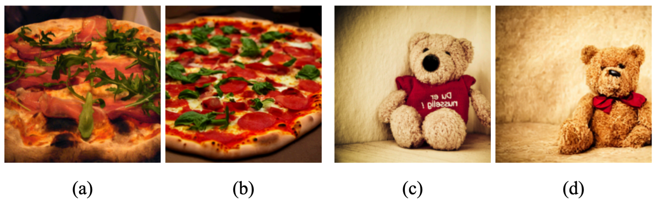


FIGURE 5 Diffusion-based input mutation. (a) original input of class "*pizza*"; (b) mutated input; (c) original input of class "*teddy*"; (d) mutated input.

RIVAL inference pipeline, we feed the LDM with the reference image (i.e., the image to be mutated) and a predefined domain-specific textual prompt. As output, the LDM generates a new input, which is a variation of the reference image. One can control how much the output image adheres to the textual prompt by determining the *Guidance Scale* parameter. Higher guidance scale means less creativity for the LDM. The *Sampling Steps* parameter can also be used to determine the number of iterations LDM performs to denoise the image. With each step, some noise is progressively eliminated, leading to an improvement in the output image quality. However, the greater the number of sampling steps, the longer it takes to produce an image. Using textual prompts beside the image as inputs for the diffusion model guarantees label preservation, as the prompt ensures the presence of the subject class in the generated image.

Figure 5 shows two example images generated by our diffusion-based mutator. Figure 5 (a) is a sample "pizza" image and (b) is the mutant image generated by LDM starting from the previous sample image and the text prompt: *"A photo of pizza, best quality, extremely detailed"*. Figure 5 (c) is a sample image labeled as "teddy", while (d) is the mutated image generated by LDM when feeding it with the sample image and the prompt: *"A photo of teddy, best quality, extremely detailed"*.

5 | EXPERIMENTAL EVALUATION

5.1 | Research Questions

The goal of our evaluation is to understand whether the features automatically extracted by DeepTheia are effective for testing DL systems through illumination search. Therefore, we seek answers for the following research questions:

RQ1 (Feature Discrimination): *How are the features automatically extracted by* DeepTheia *able to discriminate failure-inducing inputs?*

Effective features should be able to define feature maps that identify the combinations of feature values that are likely to trigger a misbehaviour of the DL system under test. This insight could offer developers a deeper understanding of the root causes of misbehaviours. In fact, the presence of regions in the feature map (i.e., one or more adjacent cells) characterized by significantly high probabilities of misbehaviours can suggest that the input data clustered into these cells are prone to causing misbehaviours. Moreover, generation or acquisition of new data that fall into these cells can be useful to obtain more evidence about the observed failures and to possibly fix them (e.g., by re-training).

Metrics: We aim to assess whether the generated feature map M, defined by the automatically extracted features, is discriminative. Moreover, we verify that the combination of DeepTheia's features with illumination search is effective by allowing a thorough exploration of the feature map. For the latter aspect, we measure the map coverage as number of *Filled Cells* (FC) in the map; for the former the *Average Cell Impurity* (ACI) of the map with respect to the behaviour of the inputs in each filled cell:

$$ACI(M) = \frac{\sum_{i=1}^{FC} 1 - (p_{misb_i}^2 + p_{correct_i}^2)}{FC} \tag{4}$$

where p_{misb_i} and $p_{correct_i}$ are the probabilities of misbehaviour and correct behaviour of the model in the i_{th} filled cell of the map M, respectively. Based on Equation 4, the ACI value is between 0 and 0.5. A low value of ACI means that DeepTheia can effectively discriminate the system's behaviour, by grouping in the same cells inputs with the same behaviour.

RQ2 (Cohesiveness): *How cohesive are the features automatically extracted by* DeepTheia*?*

Although features are automatically extracted (hence, not necessarily human interpretable), it would be useful if they were also able to group inputs in a way that is cohesive and understandable to humans.

Metrics: The comprehensive evaluation of the cohesiveness of our approach requires humans in the loop. Therefore, we performed a human study involving independent assessors to determine whether the group of images from a feature map cell are more cohesive (i.e., contain more similar images) than a group of randomly selected images. We report the cohesiveness rate for the groups of images from the same cell vs randomly selected images from different cells of the feature map.

RQ3 (Usefulness): *Can the test inputs generated by* DeepTheia *be used to improve the DL system under test?* Automatically generating pure and cohesive feature maps can be extremely useful for characterizing the behavior of the system under test, especially for datasets consisting of complex images. Improving the quality of the system by retraining it with inputs that have failure-inducing features would further confirm the usefulness of the proposed approach.

Metrics: We evaluate DeepTheia's usefulness by assessing the model's performance after fine-tuning it on DeepTheia's inputs. We measure the accuracy of the model, i.e., the ratio between the number of correct predictions to the total number of predictions, before and after fine-tuning.

5.2 | Subject Systems and Datasets

We evaluate our approach using two popular image datasets, i.e., MNIST and ImageNet. These datasets are commonly employed in the literature to assess testing techniques for DL systems [28, 24, 40, 9, 3, 51] and enable two distinct image classification tasks. In particular, ImageNet, with its 1k classes and large-size real images poses a challenging task

TABLE 1 Hyperparameters used in the experiments

Parameter	MNIST	ImageNet
class/classes	5	Pizza, Teddy
initial pop size	800	100
time budget (s)	3600	10800
input perturbation type	model-based	diffusion-based
guidance scale	-	5
sampling steps	-	50
image size	28 × 28	224 × 224
model	ConvNet	ResNet50
framework	Tensorflow	PyTorch
# of epochs for fine-tuning	6	15
learning rate for fine-tuning	0.001	0.0001

to test generators. Due to the complexity of ImageNet images, it is extremely difficult to manually define discriminative and understandable features. For each of these two subjects, we consider widely adopted, pre-trained DL models.

MNIST [27] consists of 70000 greyscale images of handwritten digits. Their size is 28 × 28 and their pixel levels range from 0 to 255. The DL model predicts which digit is represented in an input image. As DL classifier, we considered the convolutional DNN (ConvNet) architecture provided by Keras [7] and trained it on the the MNIST training set. In particular, we used its default configuration, i.e. 12 epochs, batches of size 128, and a learning rate equal to 1×10^{-3}, to train a strong model which achieved 99.11% test accuracy. Moreover, we trained a weaker model to show how DeepTheia performs as model quality varies. To obtain a weaker model we injected the "sub-optimal learning rate" fault from the taxonomy of real faults for DL [22]. Specifically, we maintained the same configuration but used a lower learning rate of 1×10^{-6}, resulting in a test accuracy of 88.12%.

ImageNet is a large and extremely popular image dataset, which has been used for the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [43]. This dataset includes images of 1000 classes, partitioned into three sets: training (1.3M images), validation (50K images), and testing (100K images with held-out class labels). For this dataset, we used the pre-trained *ResNet*50 neural network [20] provided by the timm library[§] in Pytorch[¶] with 81.17% accuracy.

5.3 | Experimental Procedure

To answer our research questions, we ran DeepTheia against the considered subjects. As a baseline, we considered the state of the art approach DeepHyperion-CS. Since it can perform only model-based input perturbations and an input model is available only for the MNIST subject, we can compare our new tool with DeepHyperion-CS only on MNIST. We consider the results achieved by DeepTheia on the two considered subjects with two different feature extraction approaches, i.e., by using (1) the same model under test or (2) an external feature extractor. For each feature extractor, we trained its PCA component by using the inputs with the label of interest from the original training set and setting the number of components to be selected (which corresponds to the number of feature dimensions) to 2. Having bi-dimensional maps allows us to directly and fairly compare with DeepHyperion-CS, which also uses 2 dimensions. We performed only one training of each PCA component for each considered subject as the output of PCA is deterministic.

Additionally, we report the results without test generation, exclusively considering inputs from the MNIST test set and the ImageNet training set. Due to the insufficient number of inputs (50 inputs) for each class in the ImageNet test set for generating the feature maps, we used inputs from the training set instead.

For MNIST, we used DeepHyperion-CS with three different combinations of the following manually defined features [58]: (1) *Luminosity (Lum)*, i.e. number of pixels whose value is above 127; (2) *Orientation (Or)*, i.e. vertical orientation of the digit, obtained by computing the angular coefficient of the linear regression of the non-black pixels; (3) *Moves* (Mov), i.e., sum of the Euclidean distances between pairs of consecutive sections of the digit. Instead, on ImageNet we do not report any results for DeepHyperion-CS as it is not applicable to this complex dataset, which is not equipped with a model of the input data, needed by DeepHyperion-CS to perform input perturbation.

To ensure a fair comparison, all the feature maps were generated with the same number of cells for each feature, i.e. 25 cells, and dimensions, i.e., 2. The discretization of the feature space into cells requires careful consideration of granularity to balance meaningful behavioral distinctions with sufficient sample populations per cell. We adopted a grid-based discretization approach with 25 cells per feature dimension, following established guidelines from the DeepHyperion-CS literature [59] that demonstrated this configuration provides reasonable granularity without creating overly sparse or dense cell populations. The feature ranges are dynamically determined based on the minimum and maximum values observed across all inputs during the experimental runs, ensuring that the discretization adapts to the actual feature distributions of each dataset. This approach maintains consistency across different feature extractors while allowing the cell boundaries to reflect the natural spread of feature values. We validated the meaningfulness of the resulting feature maps through visual inspection to ensure that nearby cells contain similar inputs, confirming that our discretization captures semantically relevant patterns in the feature space.

To account for non-determinism, we ran each tool 10 times on both MNIST and ImageNet. This allowed us to analyse the statistical significance of the differences between tools. We used the Mann-Whitney U-test and measured the effect size by means of the Vargha-Delaney's $\hat{A}_{12}$ statistic [2].

Table 1 presents the values of the hyperparameters we used for each tool. We configured DeepHyperion-CS according to the configuration suggested in the original paper. We empirically obtained the configurations for DeepTheia through some preliminary runs. For MNIST, DeepTheia randomly selects an initial population made of 800 inputs from the official MNIST test set, all belonging to the same class (i.e. digit

[§] https://timm.fast.ai
[¶] https://pytorch.org

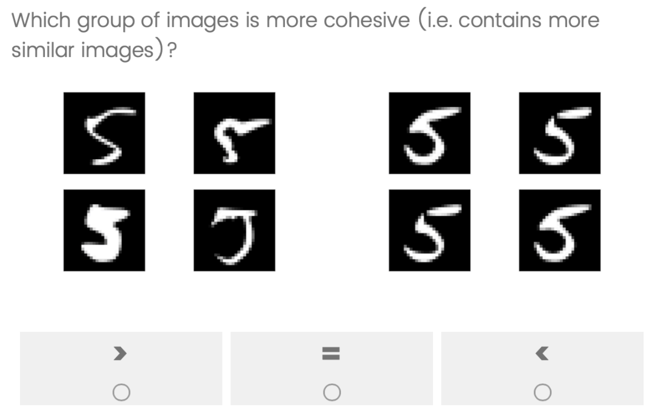

Which group of images is more cohesive (i.e. contains more similar images)?

FIGURE 6 Sample human study question for MNIST. The group of images on the right are from the same cell of a feature map. The group of images on the left are selected randomly from different cells of the same feature map.

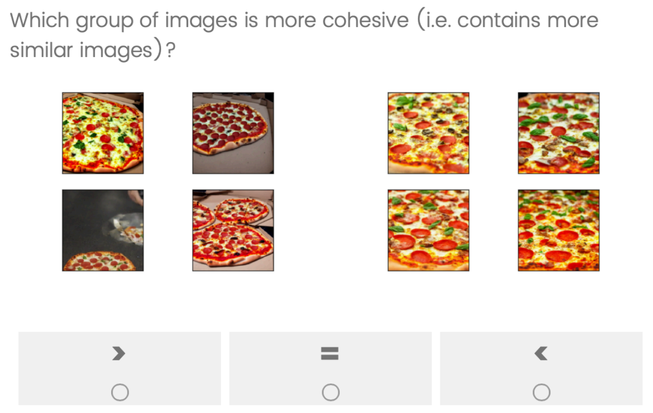

Which group of images is more cohesive (i.e. contains more similar images)?

FIGURE 7 Sample human study question for ImageNet. The group of images on the right are from the same cell of a feature map. The group of images on the left are selected randomly from different cells of the same feature map.

"5"). For ImageNet, DeepTheia randomly selects 100 inputs from the ImageNet official training set. We considered two semantically different ImageNet classes in our experiments (i.e. "pizza" and "teddy").

We used the same model-based input perturbation approach to manipulate MNIST inputs. In this way, we can effectively rule out all confounding factors and clearly compare the features automatically extracted by DeepTheia with the ones defined by experts for DeepHyperion-CS.

For ImageNet, we used our novel diffusion-based input perturbation approach, presented in subsection 4.2. In particular, we used the pre-trained Stable-Diffusion V1.5[#] model provided by Runway[||] and fine-tuned it on an NVIDIA GeForce RTX 2080 Ti GPU machine using Dreambooth [42] for 3000 training steps. For fine-tuning, we used all the inputs in the ImageNet training set belonging to the considered target class (i.e., pizza or teddy) and 200 images generated by the diffusion model itself with a domain-specific prompt (e.g. "A photo of pizza") designed to ensure label preservation. For the inference, we used the RIVAL pipeline[**] (the most recent approach for generating realistic, high-quality images at the time of writing) with guidance scale 5 and sampling steps 50.

For the human study, we published two surveys (one for each subject) using the Mechanical Turk platform provided by Amazon[††]. Each survey consists of 11 questions to be answered by human assessors: 10 assessment questions (ASQ) and 1 attention check question (ACQ). Specifically, we randomly selected 10 cells from a feature map generated by DeepTheia with the best performing feature extractor and generated plots with groups of 4 images from each cell. Then, we generated 10 groups of 4 random images from different cells (with a minimum mutual distance of 9, which was the maximum possible distance to have at least 10 different groups of random images) from the same feature map. In each ASQ, we showed the human assessor a group of images from one feature map cell and a group of random images and asked them *"Which group of images are more cohesive (i.e. contains more similar images)?"*. The assessors were provided with three possible choices: they could indicate that either the first group of images (>) or the second group of images (<) is more cohesive, or the two groups have the same level of cohesiveness (=). To prevent potential bias, we randomized the order of the choices: sometimes images selected from the same cell (resp. randomly) are presented on the left; sometimes on the right. Assessors were also provided some examples of cohesive vs random groups of images, to explain them how to carry out the task. To avoid bias, such examples come from an independent dataset (Fashion-MNIST). Figure 6 and Figure 7 show two sample questions from the human study for MNIST and ImageNet, respectively. For ACQ instead, we showed the human assessors the same groups of images, hence two groups that are equal in cohesiveness level (=). To ensure the quality of the answers we restricted the participation to the workers with approval rate above 95% and we only accepted answers from the users who passed the ACQ. We collected 80 answers from the human assessors, 40 for each case study.

To answer RQ3, we fine-tuned the original DL models by training them for more epochs at lower learning rate, by including the misbehaviour-inducing inputs generated by DeepTheia in the training set. As regards MNIST, for each run, we equally divided the inputs into two sets, i.e. $training_{DT}$ and $test_{DT}$. We used the combination of original training set and $training_{DT}$ to fine-tune the DL system. The combined training set reduces the risk of forgetting the learned task by ensuring that both the original training data and newly generated inputs are available during training. We used the original test set and $test_{DT}$ to evaluate

[#] https://huggingface.co/runwayml/stable-diffusion-v1-5
[||] https://runwayml.com
[**] https://github.com/dvlab-research/RIVAL
[††] https://www.mturk.com

the accuracy of the fine-tuned model. In this way, we assessed the accuracy improvement of the fine-tuned DL system on $test_{DT}$ and verified if it exhibited a decline (i.e., a regression) in handling inputs from the original set that were predicted correctly before fine-tuning. We repeated fine-tuning 10 times for each run of DeepTheia to enable statistical analysis. Also for ImageNet, for each run, we equally divided the inputs into two sets, i.e. $training_{DT}$ and $test_{DT}$. Due to the higher complexity and larger number of classes (i.e., 1000) in ImageNet, we needed a diversified training set representing all classes to avoid regressions. Therefore, we generated a balanced training set by combining $training_{DT}$ with an equal number of inputs from other classes of the ImageNet test set[‡‡]. For instance, if we have 10 generated inputs of class pizza in $training_{DT}$, we add 10 images from each of the other 999 classes of the ImageNet test set to our training set. We used the rest of the ImageNet test set as the original test set in each run. To provide a comprehensive comparison, we also evaluated a simple diffusion-based data augmentation baseline. We used the pre-trained Stable-Diffusion V1.5 (from Runway, without fine-tuning) and prompted it with simple class-based prompts such as "A photo of pizza" to generate 10 synthetic images. We then applied the same retraining methodology used for DeepTheia-generated inputs: combining these baseline synthetic images with the original training data and fine-tuning the ResNet model using identical hyperparameters and training procedures. This baseline allows us to assess whether DeepTheia's feature-map-guided generation provides advantages over straightforward class-conditional image synthesis in terms of model improvement effectiveness and efficiency. To enable statistical analysis, we repeated the fine-tuning process 5 times for each run of DeepTheia for class "pizza". To assess statistical significance, we again employed the Mann-Whitney U-test and the Vargha-Delaney's $\hat{A}_{12}$ statistic.

6 | RESULTS

6.1 | RQ1: Feature Discrimination

In this RQ, we investigate the discriminative capability of the features automatically extracted by DeepTheia. Table 2 and Table 3 report the results achieved by the considered tools for MNIST and ImageNet, respectively. Metric values are computed on the feature maps filled by either the original test/training sets or the inputs generated by the test generation approaches. This allows us to analyze the compared feature extractors both with and without integration with the test generators.

Columns 2 and 3 of Table 2 report the results obtained on the original MNIST test set, while columns 4 and 5 report the results by multiple runs of DeepHyperion-CS and DeepTheia. The results with and without test generation are in agreement. In particular, the external feature extractor $VGG16$ and the weak DNN generate maps that are always more significantly covered than those obtained by the human-defined

[‡‡] Given the large size of the ImageNet training set, it was not feasible to use the entire original training set for fine-tuning.

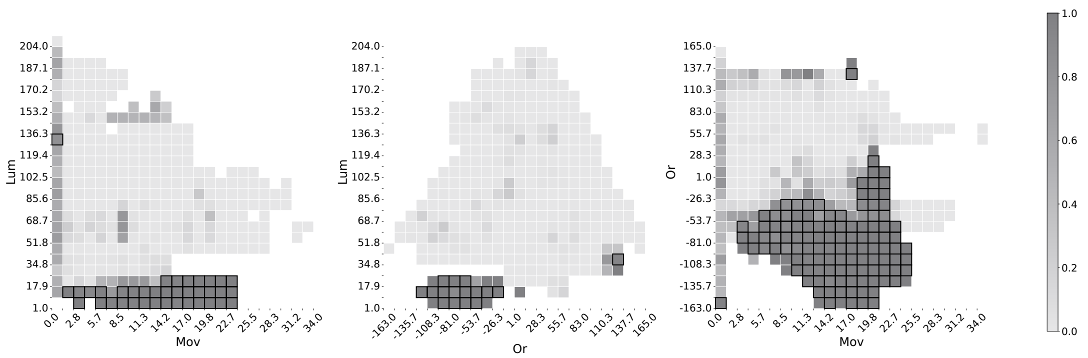
TABLE 2 RQ1 - Number of Filled Cells (FC) and Average Cell Impurity (ACI) of DeepTheia and DeepHyperion-CS for MNIST using different Feature Extractors (FE); **best** results in boldface.

	Test set		DeepHyperion-CS	
Features	FC	ACI	FC	ACI
Mov-Lum	93	0.006	269.2 ± 7.1	0.070 ± 0.007
Or-Lum	217	0.007	288.6 ± 7.6	0.031 ± 0.005
Or-Mov	88	0.005	279.2 ± 10.7	0.083 ± 0.008

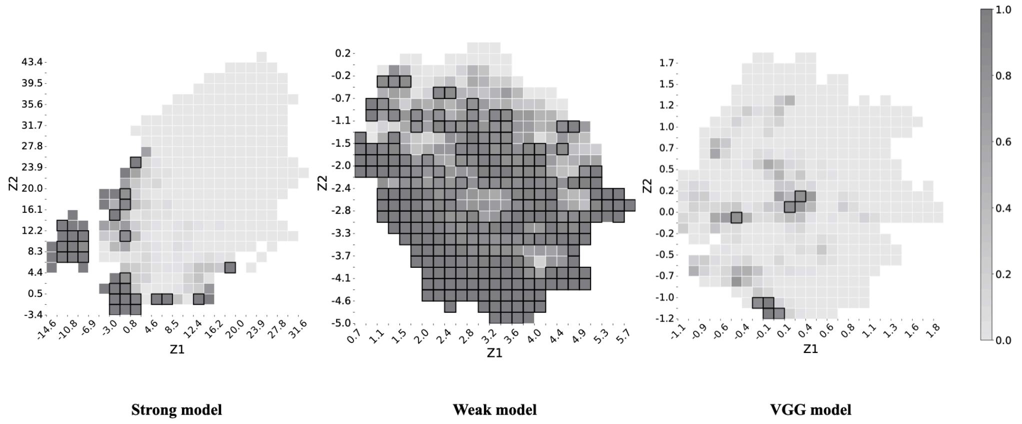
	Test set		DeepTheia	
FE	FC	ACI	FC	ACI
Strong DNN	225	**0.004**	262.5 ± 8.2	**0.028** ± 0.004
Weak DNN	**278**	0.196	**357.1** ± 5.1	0.169 ± 0.010
VGG16	272	0.008	346.3 ± 9.3	0.052 ± 0.007

features with large effect size. Moreover, the strong DNN model always achieves a significantly lower impurity with large effect size and p-value < 0.05 (ACI is 0.004 on the test set and 0.028 on the inputs generated by DeepTheia). This means DeepTheia is better at grouping inputs with the same behaviours when a strong model is used as feature extractor. Considering generated tests (columns 4 and 5), the $VGG16$ feature extractor achieves a significantly better ACI than the *Mov – Lum* and *Or – Move* feature combinations with large effect size, and has a comparable ACI with the *Or – Lum* feature combination (p-value > 0.05), while covering the map more extensively than all of them (FC = 346.3). The higher FC coverage observed with the weak DNN can be attributed to its limited representational capacity and poor feature extraction capabilities. Unlike the strong DNN and VGG16 models, which learn discriminative features that group semantically similar inputs into the same cells, the weak model produces less meaningful feature representations. This results in inputs that should logically belong to the same behavioural group being scattered across different cells in the feature map, artificially increasing the number of filled cells. However, this higher FC coverage comes at the cost of significantly higher impurity (ACI = 0.196 vs 0.004 for the strong model), indicating that the weak model fails to create coherent groupings of inputs with similar behaviors. This demonstrates that FC coverage alone is insufficient to assess feature map quality; the impurity metric is crucial for evaluating whether the increased coverage represents meaningful behavioural distinctions or merely noise in the feature space.

We further analysed the results obtained by DeepHyperion-CS and DeepTheia by comparing their Average Misbehaviour Probability (AMP) maps (see Figure 8). These are feature maps that indicate, for each cell, the average misbehavior probability observed across various test suites. AMP is calculated by dividing the number of inputs causing a misbehaviour by the total number of inputs in each cell. The shading of cells corresponds to their AMP values, with darker cells representing higher AMP values and cells with bold border having misbehaviour probability greater than 0.8 with lower bound of the confidence interval greater than 0.65. Therefore, specific regions with bold-bordered dark cells highlight areas of the feature space that are more likely to trigger failures in the system under test. Blank cells indicate feature combination values not represented in the existing test inputs. DeepTheia, similar to



(a)



(b)

FIGURE 8 Average Misbehaviour Probability (AMP) maps generated by (a) DeepHyperion-CS and (b) DeepTheia for MNIST. The axes quantify different features. The cells report the probability of exposing a misbehaviour for the corresponding feature value combinations, i.e., darker colors correspond to higher misbehaviour probabilities.

DeepHyperion-CS, produces discriminative feature maps. As shown in Figure 8 (b), maps obtained by the strong and external feature extractors have specific regions where the probability of misbehaviour is high (bold-bordered dark cells). This result is comparable with AMP maps generated by DeepHyperion-CS (see Figure 8 (a)). Instead, the weak model failed to generate discriminative feature maps as it produces multiple regions with high misbehaviour probability scattered across the feature space (i.e., most of the covered cells are dark and bold-bordered).

TABLE 3 RQ1 - The number of Filled Cells (FC) and Average Cell Impurity (ACI) of DeepTheia for ImageNet using different Feature Extractors (FE), for two different classes, i.e., “Pizza” and “Teddy”; **best** results in boldface.

		Training set		DeepTheia	
Class	FE	FC	ACI	FC	ACI
Pizza	ResNet50	**276**	0.048	146.4 ± 5.1	0.021 ± 0.004
	VGG16	272	**0.039**	$\mathbf{185.4} \pm 6.1$	$\mathbf{0.011} \pm 0.004$
Teddy	ResNet50	229	0.100	$\mathbf{163.9} \pm 7.5$	0.061 ± 0.015
	VGG16	**251**	**0.052**	159.2 ± 6.1	$\mathbf{0.047} \pm 0.009$

As for ImageNet, Table 3 shows that both automated feature extractors generated discriminative maps, with low ACI both on training set and on tests generated by DeepTheia. In particular, the features extracted by the VGG16 model showed better ACI values with statistical significance for both subjects, while achieving a map coverage higher than (p-value < 0.05 for the class *Pizza*) or comparable to (p-value > 0.05 for the class *Teddy*) the features extracted by the model under test (Table 3 columns 5 and 6).

Singletons in feature map cells artificially decrease the value of ACI, because their impurity is by definition 0. To make sure that our results are not influenced by some unbalance in the occurrence of singletons, we analysed their prevalence and found it consistently around 40% with both *ResNet*50 and *VGG*16 feature extractors.

Summary RQ1: *Automatically extracted features result in highly discriminative feature maps (ACI <* 0.04*), while enabling* DeepTheia *to cover the feature space extensively (FC > 200). External feature extractors (VGG16) achieved superior or comparable performance to internal ones.*

A major implication of this study for practitioners is that not only automated feature extraction is possible and results in discriminative maps, but also that general purpose feature extractors, independent of the model under test, can be used for feature map construction. This relieves developers from the need of a strong model as feature extractor, which might not be available in the initial development phase, when the model might be still weak.

6.2 | RQ2: Cohesiveness

TABLE 4 RQ2 - Human assessment of Feature map vs Random based on cohesiveness; **best** results in boldface.

Subject	Feature map	Random	No difference
MNIST	**78.25%**	4.00%	18.50%
ImageNet	**78.25%**	8.75%	13.00%

Table 4 reports the results of our human study on the cohesiveness of DeepTheia's feature maps. In the *Feature map* column, we report the average percentage of the crowdworkers who identified the group of images from the same cell as more cohesive. The *Random column* reports the percentage of answers where the randomly selected group of images was considered more cohesive. The last column indicates the average percentage of crowdworkers who considered a similar cohesiveness level between the two groups.

Overall, crowdworkers were able to perceive the higher cohesiveness of feature map cells (more than 78%). Despite the variety of ImageNet images, the cohesiveness of the feature map cells was clear for the large majority of the assessors.

We collected 80 responses from human assessors across two case studies: 40 responses for MNIST and 40 responses for ImageNet. Each response represents a three-way choice where assessors could select either the cohesive group (images from one cell), the random group (images from different cells), or indicate equal cohesiveness between the groups.

For statistical analysis, we applied the Mann-Whitney U-test separately for each case study, treating the 40 individual responses per case study as independent datapoints. We converted the three-way responses to ordinal values for statistical comparison, excluding ties from the analysis. A significantly higher percentage (78.25%) of assessors chose the one cell images as more cohesive than the random cell ones, with *p*-value < 0.05 and large effect size, demonstrating that human assessors can systematically distinguish between images grouped by our feature extraction approach versus randomly selected images. This indicates that our feature maps capture meaningful semantic relationships that align with human perception of image similarity.

Figure 9 shows image groups used in the human study: the images selected from one feature map cell are clearly similar among them (see Figure 9 (a) and (c)), while random images from different cells are more diverse (see Figure 9 (b) and (d)).

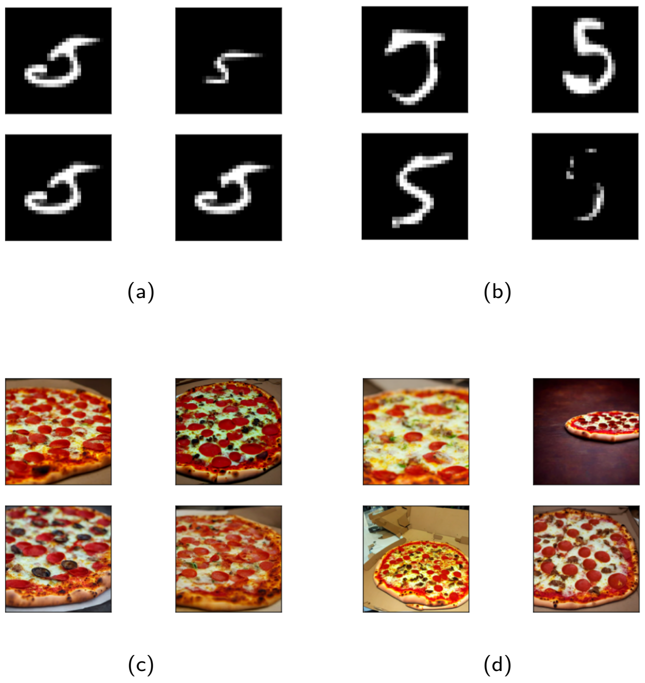


FIGURE 9 Sample groups of images used for the human study: (a) Selected from one cell of the feature map of MNIST (b) Randomly selected from different cells of MNIST (c) Selected from one cell of the feature map of ImageNet (d) Randomly selected from different cells of ImageNet.

Summary RQ2: *The features automatically extracted by* DeepTheia *are associated with a perception of high cohesiveness in human assessors. The automatically generated feature map cells contain cohesive groups of images.*

A major implication of this study for practitioners is that the presence of a high proportion of misbehaviour-inducing inputs in a given feature map cell is to some extent human-interpretable. In fact our study shows that misclassified images assigned to the same cell form a cohesive group of images that share substantial similarity. This may possibly point to some human-understandable reason for the misbehaviour, which might trigger proper corrective actions (e.g., re-training on real-world images with such features).

6.3 | RQ3: Usefulness

Table 5 shows the accuracy improvement achieved by fine-tuning the considered DL systems with inputs generated by DeepTheia. The *ACC* columns show the accuracy of the original DL systems on the original test set and DeepTheia's test set, i.e., the test partition of the inputs generated by DeepTheia. The *ACC'* columns show the accuracy values after fine-tuning the DL systems with the training partition of the inputs generated by DeepTheia. The values in the *ACC'* column (column 6) are underlined to indicate a statistically significant improvement in the accuracy after fine-tuning (*p*-value < 0.05 and large effect size). Since

TABLE 5 RQ3 - Model Accuracy (ACC) on the original test set and on the test set generated by DeepTheia, before and after fine-tuning the DL system with the training partition of the generated inputs. Results are reported for each considered Feature Extractor (FE). In each row, underline indicates values statistically significant.

	Subject	Original Test Set		DeepTheia Test Set	
	FE	ACC	ACC'	ACC	ACC'
MNIST	Strong DNN	99.25	99.27 ± 0.05	0.00	$\underline{99.59 \pm 0.67}$
	VGG16		99.27 ± 0.06		$\underline{99.96 \pm 0.14}$
ImageNet	Strong DNN	79.17	76.73 ± 0.03	0.00	$\underline{51.16 \pm 16.80}$
	VGG16	79.22	76.58 ± 0.02		$\underline{63.33 \pm 28.18}$

TABLE 6 RQ3 - Model Accuracy (ACC) on the original test set and on the test set generated by the Stable Diffusion, before and after fine-tuning the DL system with the training partition of the inputs generated for ImageNet.

Original Test Set		Generated Test Set	
ACC	ACC'	ACC	ACC'
79.19	76.68 ± 0.00	0.58 ± 0.16	0.66 ± 0.21

we used pre-trained state-of-the-art models for our experiments, the initial accuracy on the original test is fairly high, i.e. 99.25% for MNIST and 79.19% on average for ImageNet. Their accuracy on DeepTheia's test set is 0% by construction, as we used only failure-inducing inputs generated by DeepTheia. As regards MNIST, retraining using inputs generated by DeepTheia significantly improved on DeepTheia's test set by 99.59% when using the available Strong model for feature extraction and 99.96% when using the VGG16 model for feature extraction, with no regressions (i.e., no new mis-behaviours) on the original test set. As regards ImageNet, both feature extractors were able to improve the accuracy on DeepTheia's test set with no significant drop of the original test set accuracy. Specifically, DeepTheia with the VGG16 extractor significantly increased the accuracy by more than 60% on the misbehaviour-inducing inputs through fine-tuning (p-value < 0.05 and large effect size).

To evaluate the effectiveness of our feature-map-guided approach against a baseline, we conducted an additional experiment using standard diffusion-based data augmentation. The results reported in Table 6 show that while this baseline approach maintains model performance on the original test set (ACC: 79.19% to ACC': 76.68% $\pm$ 0.00), it provides minimal improvement on the generated test set (ACC: 0.58% $\pm$ 0.16 to ACC': 0.66% $\pm$ 0.21). This limited improvement occurs because simple class-conditional prompts tend to generate easily classified, prototypical images rather than challenging edge cases that expose model weaknesses. In contrast, DeepTheia's feature-map-guided generation specifically targets regions of the input space where the model exhibits poor performance, resulting in more effective model improvement through targeted augmentation of difficult cases.

Summary RQ3: *The inputs generated by* DeepTheia *are useful to improve the DL system performance through fine-tuning.*

6.4 | Threats to Validity

External Validity: A potential threat to external validity is the selection of the experimental subjects and datasets. To mitigate this threat, we chose two diverse image datasets with increasing complexity that have been widely adopted in the literature. With MNIST, we show that DeepTheia outperforms existing approaches in a relatively simple challenge, where manual feature definition is feasible. With ImageNet, we demonstrate that feature exploration is effective and meaningful also in complex tasks with 1000 labels and realistic images, thanks to recent advances in Deep Learning, i.e., transfer learning and generative AI.

External Validity: A potential threat to external validity is the selection of the experimental subjects and datasets. To mitigate this threat, we chose two diverse image datasets with increasing complexity that have been widely adopted in the literature. With MNIST, we show that DeepTheia outperforms existing approaches in a relatively simple challenge, where manual feature definition is feasible. With ImageNet, we demonstrate that feature exploration is effective and meaningful also in complex tasks with 1000 labels and realistic images, thanks to recent advances in DL, i.e., transfer learning and generative AI. While ImageNet represents a substantial large-scale benchmark with over 1.2 million training images and significant visual complexity, scalability to even larger datasets remains a consideration. Future work could explore application to domain-specific large-scale datasets (e.g., medical imaging, satellite imagery) or datasets with higher resolution images to further validate scalability and generalizability across different visual domains.

Conclusion Validity: The inherent stochasticity in DL and search-based approaches introduces variability in the results. To mitigate this, we employed a rigorous experimental methodology, running each experiment multiple times. We further applied standard statistical tests to evaluate the significance of the observed differences.

Reproducibility of our results is ensured by the online availability of source code, experimental subjects and data.

7 | RELATED WORK

Different techniques proposed in the literature for testing DL systems focus on the features of the test inputs.

O'Shaughnessy et. al [36] generate post-hoc causal explanations for classifiers by leveraging a learned low-dimensional representation of the data. Their method involves constructing a generative model (e.g. a VAE) with a disentangled representation of the data and a mapping to the data space. They use a structural causal model to formalize the relationships between independent latent factors, classifier inputs, and outputs. Our approach also relies on latent features of the input that are automatically extracted. However, our aim is different, i.e., to generate test inputs by covering the feature space while providing discriminative feature maps for further analysis of the model's behaviour.

Kang et. al [24] introduced SINVAD for testing DL systems using the latent space of VAEs. It performs optimization to find inputs close to the decision boundary of the DL system. In particular, it adds perturbations to the input latent representations to generate surprising or misbehaviour-inducing inputs. Our approach, instead of using a VAE, relies on the DNN under test or an external feature extractor to define the feature space to explore.

Dola et. al [9] extracted feature vectors using a VAE trained on the training data of the DNN under test. These feature vectors establish a coverage domain for the application of Combinatorial Interaction Testing on a partitioned latent space, facilitating the measurement of test coverage. They capture feature diversity in their test adequacy metric named Input Distribution Coverage (IDC) by computing the interaction between abstracted features. While VAEs are effective at extracting related features of the input, they can be less accurate when they encounter inputs that differ from their training set and lead to unrelated connections between features and behaviour of the model. Unlike IDC, we use either the DNN model under test or an external feature extractor such as VGGNet for feature extraction.

Attaoui et.al [3] used a pre-trained VGGNet model to extract relevant features of the misbehaviour-inducing inputs. Their tool, called SAFE, uses these features to compute root cause clusters and selects unsafe test inputs to improve the DL system through retraining. Like SAFE, DeepTheia also uses pre-trained models for feature extraction. However, we aim to explore the feature space using illumination search and cover the feature map by generating diverse inputs belonging to different areas of the feature map.

DeepAtash [56] is a focused test input generator for DL systems. It generates misbehaviour-inducing inputs with user defined feature values by targeting specific areas of the feature map. Like DeepHyperion-CS, DeepAtash relies on input features manually defined by domain experts. Instead, DeepTheia can automatically extract features for any DL system and explores the feature space at large.

Neelofar et. al [33] proposed an adequacy metric for black-box testing of autonomous vehicles considering their instance space. An instance space refers to a 2D representation of the test scenarios, defined based on the most effective features of the test scenarios. Our work is similar in providing 2D maps that indicate the diversity and coverage of test inputs. Their approach requires significant domain knowledge to extract meaningful features from a test scenario. In contrast, our proposed method requires no prior knowledge of the system under test since it can automatically extract features.

8 | CONCLUSIONS AND FUTURE WORK

This paper introduces DeepTheia, a novel test generator for DL systems based on illumination search. It automates the extraction of relevant input features using pre-trained models, overcoming limitations of existing illumination-based tools by eliminating the need for human experts to define the features. DeepTheia shows significant improvements in the discriminative power of feature maps, while preserving their cohesiveness and understandability, with respect to expert-aided illumination search. Additionally, our novel mutation operator based on diffusion models enables the generation of valid tests for complex image classification tasks, while ensuring label preservation. The inputs generated by DeepTheia are also useful for improving DL systems through fine-tuning. In future work, we aim to enhance generalizability by including a broader spectrum of DL systems in our analysis and generating controlled variations of images through more customized prompts for the diffusion-based mutation operator. We also plan to explore additional transfer learning architectures beyond VGG16 as feature extractors, including recent advances such as Inception-V3, ResNet, EfficientNet, and Vision Transformers, which could provide superior feature representations for test input generation. Additionally, we aim to validate our approach on larger and more diverse datasets, including domain-specific applications such as medical imaging and autonomous driving scenarios, to further demonstrate scalability across different domains.

ACKNOWLEDGMENTS

This work was partially supported by the SNSF Project Toposcope (n. 214989) and the M4C2 I1.3 "SEcurity and RIghts In the CyberSpace - SERICS" (PE00000014 -CUP H73C2200089001, D33C22001300002) projects, funded under the National Recovery and Resilience Plan (NRRP) funded by the European Union - NextGenerationEU.

REFERENCES

1. A. B. Ahadit and R. K. Jatoth, *A novel multi-feature fusion deep neural network using hog and vgg-face for facial expression classification*, Machine Vision and Applications **33** (2022), no. 4, 55.
2. A. Arcuri and L. Briand, *A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering*, Software Testing, Verification and Reliability **24** (2014), no. 3, 219–250. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/stvr.1486.
3. M. Attaoui, H. Fahmy, F. Pastore, and L. Briand, *Black-box safety analysis and retraining of dnns based on feature extraction and clustering*, ACM Transactions on Software Engineering and Methodology **32** (2023), no. 3, 1–40.
4. M. Biagiola, S. Klikovits, J. Peltomäki, and V. Riccio, *Sbft tool competition 2023-cyber-physical systems track*, *2023 IEEE/ACM International Workshop on Search-Based and Fuzz Testing (SBFT)*, IEEE, 2023, 45–48.
5. M. Biagiola, A. Stocco, V. Riccio, and P. Tonella, *Two is better than one: digital siblings to improve autonomous driving testing*, Empirical Software Engineering **29** (2024), no. 4, 1–33.
6. H. B. Braiek and F. Khomh, *On testing machine learning programs*, Journal of Systems and Software **164** (2020), 110542.
7. F. Chollet, *Simple mnist convnet*, https://github.com/keras-team/keras-io/blob/master/examples/vision/mnist_convnet.py (2020).
8. P. Dhariwal and A. Nichol, *Diffusion models beat gans on image synthesis*, Advances in neural information processing systems

34 (2021), 8780–8794.

9. S. Dola, M. B. Dwyer, and M. L. Soffa, *Input distribution coverage: Measuring feature interaction adequacy in neural network testing*, ACM Transactions on Software Engineering and Methodology **32** (2023), no. 3, 1–48.
10. I. Dunn, H. Pouget, D. Kroening, and T. Melham, *Exposing previously undetectable faults in deep neural networks*, *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2021, 56–66.
11. K. Fukunaga, *Introduction to statistical pattern recognition*, Elsevier, 2013.
12. R. Gal, Y. Alaluf, Y. Atzmon, O. Patashnik, A. H. Bermano, G. Chechik, and D. Cohen-Or, *An image is worth one word: Personalizing text-to-image generation using textual inversion*, arXiv preprint arXiv:2208.01618 (2022).
13. A. Gambi, M. Müller, and G. Fraser, *Automatically testing self-driving cars with search-based procedural content generation*, *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2019, Beijing, China, July 15-19, 2019*, ACM, 2019, 318–328, . URL https://doi.org/10.1145/3293882.3330566.
14. A. Gambi, G. Jahangirova, V. Riccio, and F. Zampetti, *Sbst tool competition 2022*, *Proceedings of the 15th Workshop on Search-Based Software Testing*, 2022, 25–32.
15. I. Goodfellow et al., *Generative adversarial nets*, Advances in neural information processing systems **27**.
16. J. Gu et al., *Recent advances in convolutional neural networks*, Pattern recognition **77** (2018), 354–377.
17. R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, *A survey of methods for explaining black box models*, ACM computing surveys (CSUR) **51** (2018), no. 5, 1–42.
18. J. Guo, Y. Jiang, Y. Zhao, Q. Chen, and J. Sun, *Dlfuzz: Differential fuzzing testing of deep learning systems*, *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2018, 739–743.
19. F. U. Haq, D. Shin, and L. Briand, *Efficient online testing for dnn-enabled systems using surrogate-assisted and many-objective optimization*, *Proceedings of the 44th international conference on software engineering*, 2022, 811–822.
20. K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, 770–778.
21. E. J. Hu et al., *Lora: Low-rank adaptation of large language models.*, ICLR **1** (2022), no. 2, 3.
22. N. Humbatova, G. Jahangirova, G. Bavota, V. Riccio, A. Stocco, and P. Tonella, *Taxonomy of real faults in deep learning systems*, *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, Association for Computing Machinery, New York, NY, USA, 2020, 1110–1121, . URL https://doi.org/10.1145/3377811.3380395.
23. M. Jogin, M. Madhulika, G. Divya, R. Meghana, S. Apoorva et al., *Feature extraction using convolution neural networks (cnn) and deep learning*, *2018 3rd IEEE international conference on recent trends in electronics, information & communication technology (RTEICT)*, IEEE, 2018, 2319–2323.
24. S. Kang, R. Feldt, and S. Yoo, *Sinvad: Search-based image space navigation for dnn image classifier test input generation*, *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, 2020, 521–528.
25. S. Kim and S. Yoo, *Dandi: Diffusion as normative distribution for deep neural network input*, *2025 IEEE/ACM International Workshop on Deep Learning for Testing and Testing for Deep Learning (DeepTest)*, IEEE, 2025, 9–16.
26. D. P. Kingma and M. Welling, *Auto-encoding variational bayes*, arXiv preprint arXiv:1312.6114 (2013).
27. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, *Gradient-based learning applied to document recognition*, Proceedings of the IEEE **86** (1998), no. 11, 2278–2324.
28. Z. Li, X. Ma, C. Xu, C. Cao, J. Xu, and J. Lü, *Boosting operational dnn testing efficiency through conditioning*, *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, 499–509.
29. C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*, Cambridge University Press, 2008.
30. M. Maryam, M. Biagiola, A. Stocco, and V. Riccio, *Benchmarking generative ai models for deep learning test input generation*, *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*, IEEE, 2025, 174–185.
31. J.-B. Mouret and J. Clune, *Illuminating search spaces by mapping elites* (2015).
32. D. Musleh, M. Alotaibi, F. Alhaidari, A. Rahman, and R. M. Mohammad, *Intrusion detection system using feature extraction with machine learning algorithms in iot*, Journal of Sensor and Actuator Networks **12** (2023), no. 2, 29.
33. N. Neelofar and A. Aleti, *Towards reliable ai: Adequacy metrics for ensuring the quality of system-level testing of autonomous vehicles*, arXiv preprint arXiv:2311.08049 (2023).
34. V. Nguyen, S. Huber, and A. Gambi, *Salvo: Automated generation of diversified tests for self-driving cars from existing maps*, *2021 IEEE International Conference on Artificial Intelligence Testing (AITest)*, IEEE, 2021, 128–135.
35. A. Nichol et al., *Glide: Towards photorealistic image generation and editing with text-guided diffusion models*, arXiv preprint arXiv:2112.10741 (2021).
36. M. O'Shaughnessy, G. Canal, M. Connor, C. Rozell, and M. Davenport, *Generative causal explanations of black-box classifiers*, Advances in neural information processing systems **33** (2020), 5453–5467.
37. V. Riccio and P. Tonella, *Model-based exploration of the frontier of behaviours for deep learning system testing*, *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2020, 876–888.
38. V. Riccio and P. Tonella, *When and why test generators for deep learning produce invalid inputs: an empirical study*, *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, IEEE, 2023, 1161–1173.
39. V. Riccio, G. Jahangirova, A. Stocco, N. Humbatova, M. Weiss, and P. Tonella, *Testing machine learning based systems: a systematic mapping*, Empirical Software Engineering **25** (2020), 5193–5254.
40. V. Riccio, N. Humbatova, G. Jahangirova, and P. Tonella, *Deepmetis: Augmenting a deep learning test set to increase its mutation score*, *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, IEEE, 2021, 355–367.
41. R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, *High-resolution image synthesis with latent diffusion models*, *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, 10684–10695.
42. N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, *Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation*, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, 22500–22510.
43. O. Russakovsky et al., *Imagenet large scale visual recognition challenge*, International journal of computer vision **115** (2015), 211–252.
44. K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, arXiv preprint arXiv:1409.1556 (2014).

45. J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, *Deep unsupervised learning using nonequilibrium thermodynamics*, *International conference on machine learning*, PMLR, 2015, 2256–2265.
46. C. Tantithamthavorn and J. Jiarpakdee, *Explainable AI for Software Engineering*, Monash University, 2021, . URL http://xai4se.github.io/, retrieved 2021-05-17.
47. E. Tatulli and T. Hueber, *Feature extraction using multimodal convolutional neural networks for visual speech recognition*, *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2017, 2971–2975.
48. L. Wang, X. Xie, X. Du, M. Tian, Q. Guo, Z. Yang, and C. Shen, *Distxplore: Distribution-guided testing for evaluating and enhancing deep learning systems*, *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2023, 68–80.
49. K. Weiss, T. M. Khoshgoftaar, and D. Wang, *A survey of transfer learning*, Journal of Big data **3** (2016), no. 1, 1–40.
50. O. Weißl, A. Abdellatif, X. Chen, G. Merabishvili, V. Riccio, S. Kacianka, and A. Stocco, *Targeted deep learning system boundary testing*, ACM Transactions on Software Engineering and Methodology .
51. Z. Wu, Z. Wang, J. Chen, H. You, M. Yan, and L. Wang, *Stratified random sampling for neural network test input selection*, Information and Software Technology **165** (2024), 107331.
52. Y. Xiang, H. Huang, S. Li, M. Li, C. Luo, and X. Yang, *Automated test suite generation for software product lines based on quality-diversity optimisation*, ACM Transactions on Software Engineering and Methodology (2023).
53. H. You, Z. Wang, J. Chen, S. Liu, and S. Li, *Regression fuzzing for deep learning systems*, *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*, IEEE, 2023, 82–94.
54. J. M. Zhang, M. Harman, L. Ma, and Y. Liu, *Machine learning testing: Survey, landscapes and horizons*, IEEE Transactions on Software Engineering **48** (2020), no. 1, 1–36.
55. Y. Zhang, J. Xing, E. Lo, and J. Jia, *Real-world image variation by aligning diffusion inversion chain*, arXiv preprint arXiv:2305.18729 (2023).
56. T. Zohdinasab, V. Riccio, and P. Tonella, *Deepatash: Focused test generation for deep learning systems*, *Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2023.
57. T. Zohdinasab, V. Riccio, and P. Tonella, *An empirical study on low-and high-level explanations of deep learning misbehaviours*, *2023 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, IEEE, 2023, 1–11.
58. T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, *Deephyperion: exploring the feature space of deep learning-based systems through illumination search*, *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Virtual, Denmark, 2021, 79–90.
59. T. Zohdinasab, V. Riccio, A. Gambi, and P. Tonella, *Efficient and effective feature space exploration for testing deep learning systems*, ACM Trans. Softw. Eng. Methodol. (2022). URL https://doi.org/10.1145/3544792.