Deceiving Humans and Machines Alike: Search-based Test Input Generation for DNNs using Variational Autoencoders

SUNGMIN KANG, Korea Advanced Institute of Science and Technology, Republic of Korea
ROBERT FELDT, Chalmers University of Technology, Sweden
SHIN YOO, Korea Advanced Institute of Science and Technology, Republic of Korea

Due to the rapid adoption of Deep Neural Networks (DNNs) into larger software systems, testing of DNN based systems has received much attention recently. While many different test adequacy criteria have been suggested, we lack effective test input generation techniques. Inputs such as images of real world objects and scenes are not only expensive to collect but also difficult to randomly sample. Consequently, current testing techniques for DNNs tend to apply small local perturbations to existing inputs to generate new inputs. We propose SINVAD, a way to sample from, and navigate over, a space of realistic inputs that resembles the true distribution in the training data. Our input space is constructed using Variational AutoEncoders (VAEs), and navigated through their latent vector space. Our analysis shows that the VAE-based input space is well-aligned with human perception of what constitutes realistic inputs. Further, we show that this space can be effectively searched to achieve various testing scenarios, such as boundary testing of two different DNNs or analyzing class labels that are difficult for the given DNN to distinguish. Guidelines on how to design VAE architectures are presented as well. Our results have the potential to open the field to meaningful exploration through the space of highly structured images.

CCS Concepts: • **Software and its engineering → Search-based software engineering**.

Additional Key Words and Phrases: Test Data Generation, Deep Neural Network, Search-based Software Engineering

ACM Reference Format:
Sungmin Kang, Robert Feldt, and Shin Yoo. 2018. Deceiving Humans and Machines Alike: Search-based Test Input Generation for DNNs using Variational Autoencoders. 1, 1 (December 2018), 25 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Over the past decade, Deep Neural Networks (DNNs) have risen as the dominant scalable function approximators, starting with image classification [23]. As such, they were employed over a diverse range of disciplines, from music generation [1] to board game play [25]. The seeming accuracy of DNNs in perception problems in turn led to use in safety-critical domains: one prominent example being autonomous driving [7, 8].

However, one has reason to be concerned about widespread use of DNNs in safety-critical systems: their properties are difficult to determine, and even if a fault is uncovered, it is unclear what one can do to fix them. Adversarial examples [6, 13, 24, 29] provide a good demonstration of this: sophisticated noise is added to inputs, inducing incorrect

Authors' addresses: Sungmin Kang, sungmin.kang@kaist.ac.kr, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea; Robert Feldt, shin.yoo@kaist.ac.kr, Chalmers University of Technology, Gothenburg, Sweden; Shin Yoo, shin.yoo@kaist.ac.kr, Korea Advanced Institute of Science and Technology, Daejeon, Republic of Korea.

behavior in DNNs. Adversarial examples are both widespread (appearing in multiple domains, including speech recognition [36] and natural language processing [17]) as well as physically feasible [3].

This has made it increasingly clear that neural networks must be thoroughly tested; indeed, much work has been dedicated to propose adequacy criteria for DNN testing. Pei et al. [30] check the activation status of neurons over a test suite; they argue that a test suite should activate as many of the neurons as possible. Ma et al. [27] suggest that a test suite should elicit a diverse range of activations in the neurons. Kim et al. [20] approximate the distribution of activation values in the training dataset, and posit that a test suite should have a balanced amount of inputs that elicit in-distribution activations and out-distribution activations.

While some of the aforementioned work also suggests how to generate new tests based on their adequacy criteria, a problem that has seen less light is that of generating *realistic tests*. We argue that for DNNs, the utility of testing hinges on the realism of the generated tests. Recalling the adversarial examples from above, there are two issues that are in need of answers. One issue is that it is unclear what one can do to fix discovered erroneous behavior. The other, more important issue is whether such falsifying inputs are important for DNN testing at all; it is unlikely the carefully crafted noise that is used in adversarial images would appear in practice. The ultimate goal of testing is to find realistic situations in which the system under test fails, which is a fundamental principle that should also apply to DNNs. This is why we believe that when making new test inputs, we should specifically search over the space of realistic images, instead of over a space that encourages small perturbations. Such a search technique would allow us to focus our testing resources on plausible usage scenarios that reflect the actual usage of our system.

To this end, we present SINVAD (Search-based Input space Navigation using Variational AutoencoDers). SINVAD first *learns* the space of realistic images through Variational Autoencoders (VAEs) [21], an approach that can approximate the distribution of training images. Once trained, we use the latent vector space of the trained VAE to navigate within the space of realistic images. The aim of doing so is to be able to generate test data that is realistic to the human eye, and by virtue of being so relevant to the task on hand. To motivate our approach, we perform qualitative experiments demonstrating that optimization over the entire image space yields uninterpretable images that look like static noise; meanwhile optimization with SINVAD results in realistic images, which are thus more relevant to the target dataset.

The fact that SINVAD searches a space of realistic images is central to our academic contribution. Therefore, we perform a large-scale human study to verify that SINVAD is capable of searching the space of realistic images. To demonstrate that SINVAD's search space is aligned with the realistic image space throughout, we ask human evaluators to rate images that were specifically designed to be confusing to a DNN, and check whether they are also confusing to the human eye. Specifically, we devise a questionnaire that asks whether the presented image looks like a pair of categories. This is to verify that SINVAD encodes a realistic space and is not simply memorizing images. The questionnaire's results indicate that humans rate SINVAD-generated images as more confusing than other image generation techniques. Specifically, they are identifiable as more than one class of a given classification task, and are realistic in the sense that they would be identified with both classes simultaneously. Furthermore, humans would take a longer time to process SINVAD-generated images, as another indicator of their confusion. These results support the hypothesis that SINVAD is indeed well-aligned with the realistic image space.

Knowing that SINVAD is capable of generating realistic images, more so than other image generation techniques, we demonstrate the flexibility of SINVAD by searching for inputs proximate to the decision boundary of the DNN. Our results establish that SINVAD can aid the boundary testing of DNNs by generating images on the semantic boundary of labels.

We discuss a specific case that demonstrates the usefulness of modelling realistic images. Noting that abstract classes have a "perceptual distance" in practice (for example, one may have a difficult time drawing a 3 that looks like a 4, but it is easier to draw a 4 that looks like a 9), and that search using SINVAD can capitalize on such characteristics, a test generation technique that finds pairs of labels that are prone to confusion by the DNN under test is presented. Our experiments verify that the pairs that it finds roughly match the perceptual similarity between categories. Further, we quantitatively find that when training data labels are mixed, the provided technique using SINVAD can identify such issues more quickly than the default test dataset.

Additionally, we present SINVAD-generated results when the VAE architecture is varied, and provide practical guidelines for how to construct VAEs when doing test generation for DNNs. We conclude our paper by providing a discussion of threats to validity and limitations of our work.

The contribution of the paper is as follows:

- We introduce SINVAD, a novel method to search through the space of realistic images using a VAE-based representation, and show it to be a valid way of searching for new inputs with desirable properties while maintaining realism;
- We perform a large-scale human study to show that images made by SINVAD are realistic and semantically confusing to people, indicating that the space SINVAD searches is close to the realistic image space;
- We demonstrate the capability of SINVAD to perform boundary testing, illustrating the flexibility of SINVAD as a space to help test generation;
- We present a technique using SINVAD that produces tests capable of identifying training problems earlier than the provided test set;
- We propose guidelines on VAE architecture for test generation use.

This paper is an extension of our workshop paper [18] that introduced SINVAD. It has been extended with the following contributions:

- We have performed a large-scale human study to verify that the images generated by SINVAD are indeed realistic and meaningful to human perception, evaluating 800 images with more than 500 human evaluators and 18,000 evaluations.
- We extend the existing results under different settings to verify that SINVAD can reliably perform, in particular by performing experiments on the complex ImageNet-32 dataset of down-scaled real-world images.
- We perform a study of how different VAE architectures can influence image generation results.

The remainder of this paper is organized as follows. Section 2 introduces the related literature, providing background to this paper. Section 3 presents nomenclature, and defines the problem of searching the space of realistic inputs in a formal way. How SINVAD performs search in the realistic input space is subsequently explained. Section 4 presents the research questions we aim to answer, with experiments that are described in Section 5. Section 6 presents threats to validity, Section 7 provides limitations and future directions of our work, and Section 8 concludes.

2 BACKGROUND

2.1 Testing of DNNs

As emphasized in the introduction, the increasingly widespread use of DNNs in safety-critical domains has necessitated testing and verification algorithms that handle them. Much work has focused on providing adequacy criteria, and identifying which inputs are likely to reveal erroneous behavior. The first work in this line introduced Neuron

Coverage [30], in which the activation status of each neuron in a DNN is measured. The authors go on to introduce various test generation strategies that can hopefully increase the neuron coverage. While neuron coverage used a single threshold over all neurons, Ma et al. [27] argue that a good test suite would elicit diverse activation values for each neuron; this is measured using their kMNC measure. Moving away from pre-set threshold values, Kim et al. [20] suggest using the activation distribution caused by the training dataset, and to use it to evaluate test suites. Specifically, the argument is that a test suite should contain inputs of diverse levels of *surprise*, defined by the likelihood in the training activation distribution. All such work is different from ours, as they primarily use the raw pixel space to search over images. Meanwhile, Zhang et al. [43] introduce DeepRoad, in which style transfer techniques are used to generate new test images. Their approach requires separate datasets for each situation to model. Another branch of work aims to augment inputs based on a certain set of rules to enhance the training process. One example is Cubuk et al. [10] who describe AutoAugment, which uses reinforcement learning and a set of image augmentation rules to discover which set of image rules leads to the best validation performance. A key difficulty with applying AutoAugment is that a (potentially large) network needs to be trained hundreds to thousands of times to yield good data augmentation results. Furthermore, for AutoAugment, a set of manually defined image transformations must be provided, from which a reinforcement learning algorithm will choose which transformations are most effective in improving neural network performance, unlike SINVAD, which does not require such manual components.

More recently, Byun et al. [4, 5] propose a VAE-based framework to facilitate the testing of DNN systems. While their work and ours share many components, there are key differences. Most importantly, while Byun et al. [5] focus on generating tests using the VAE space and Byun et al. [4] focus on how we can use the characteristics of the vector space, our focus is on how we can use the VAE latent space as a search space. For example, Byun et al. [5] use conditional VAEs (like Song et al. [38]'s AC-GAN use) that generate images of a specific class; thus it is difficult to make semantically confusing images as SINVAD does. Further, Riccio and Tonella [32] use Bezier curves to model image space and evaluate how well neural networks perform. While they also attempt to model the realistic image space, we implement that space with VAEs and they implement it with Bezier curves. We argue that SINVAD does not require complex rule-based models of the underlying data, and hence has the potential to be more general - e.g. it would be difficult to model complex datasets such as ImageNet with manually specified models, whereas SINVAD can operate on such datasets with the appropriate selection of generative model.

In addition to the dynamic techniques introduced above, there are several methods that aim to verify properties of neural networks. Katz et al. [19] introduce ReluPlex, which utilizes SMT solvers to provide theoretic guarantees for DNN properties. Huan et al. [16] suggest a technique to prove that all images within an ϵ-ball of a certain image are classified in the same manner. More general discussion of DNN verification techniques can be found in Zhang et al. [42].

2.2 Difficulties of Input Search for DNNs

Among the greatest obstacles to DNN test generation is the semantic manifold problem [41]. In short, the problem is due to the relatively miniscule valid input space when compared to the possible image space: if one samples an image from random from the full image space, the probability that it will be a recognizable image of a 2, for example, is exceedingly small. This issue is further exacerbated by the difficulty of defining a "boundary" of acceptable inputs in a rule-based manner, making it difficult to use search-based techniques to find inputs with desirable properties. Then how does existing work search for novel test inputs? By utilizing the metamorphic principle: most test generation work starts from a *seed image*, and adds a certain perturbation that is assumed to keep the semantic meaning of the image as

is [20, 30, 39, 43]. In SINVAD, we take a different path: by training and using a VAE that approximates the distribution of realistic images, we aim to navigate the space of realistic images directly.

2.3 Generative Models

As universal function approximators, DNNs can be used to approximate distributions, i.e. act as *generative models*. Generative models allow one to calculate the probability of each input, and further sample from the distribution. In our case, we can use the sampling to generate realistic images. One example is Generative Adversarial Networks [14], in which a generator network tries to generate samples as close to the target distribution as possible, while a discriminator network tries to find the discrepancies between the generator-made distribution and the true distribution. Such a process guides the generator to generate samples as close to the true distribution as possible. Variational Autoencoders (VAEs) [21] maximize the evidence lower bound (ELBO), which is the lower bound of the odds of an image, to approximate the target distribution. The family of deep generative models is too large to describe in full here; Foster [12] provides many examples. For our purpose, any generative model with latent variables (e.g. GANs or VAEs) that can function as a condensed search space may be used.

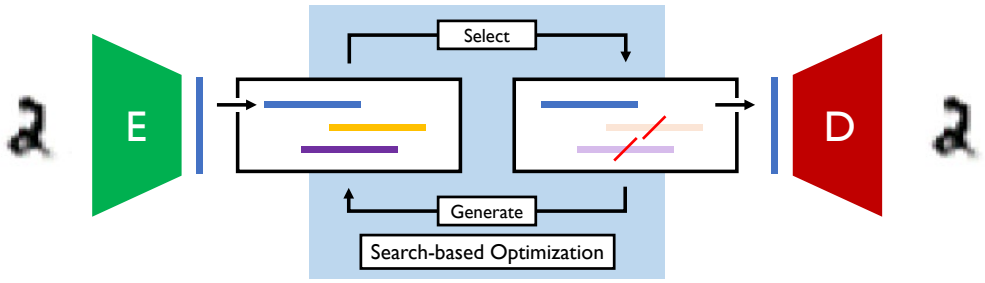


Fig. 1. Diagram of SINVAD. SINVAD uses the VAE encoder (E) to encode images into a *latent space* (bars). Search-based optimization is performed not on raw pixels, but on latent representations. A representation is transformed back to an image using the VAE decoder (D). The use of VAE networks before and after keeps search within the subset of realistic images.

3 SINVAD: SEARCH-BASED INPUT SPACE NAVIGATION USING VAES

This section presents the formal definition of the search-based input generation problem for DNN and subsequently introduces SINVAD.

3.1 Problem Statement

A common application of DNNs is as image classifiers. In such applications, a neural network N acts as a function that maps the space of images $\mathcal{I} = [0,1]^{c \times w \times h}$ to a space representing the probability of each class $\mathcal{P} = \{v \in [0,1]^n : \sum_{i=1}^{n} v_i = 1\}$, where there are n categories. Formally, the DNN is a mapping $N_p : \mathcal{I} \rightarrow \mathcal{P}$, and a categorical mapping $N_c(i) = \arg\max N_p(i)$ is derived from an image $i \in \mathcal{I}$.

In practice, the semantic meaning of most images is unclear, so the neural network is trained on a subset of images from the space of realistic images $\mathcal{D}$ and maps to $\mathcal{P}$. When we say 'realistic images', we mean that for any image in $\mathcal{D}$, a reasonable person could identify that the image would naturally be part of a certain dataset. While this definition is not the most rigorous, generally there is a boundary of $\mathcal{D}$ that people agree upon (see the human study results of Riccio and

Tonella [32]), albeit being difficult to pin down and somewhat subjective. In test generation, this becomes a problem; the space of all images of a certain resolution is usually much larger than the space of all such images that are within $\mathcal{D}$. If one looks at the MNIST dataset for example, the space of grayscale 28 by 28 images is clearly larger than the space of digit images, for most random grayscale images look more like static noise than recognizable numbers [41]. Then, due to the vast size difference between $\mathcal{I}$ and $\mathcal{D}$ and the complexity of the neural networks, it is often easy to obtain images that meet a certain criteria, but at the same time are not particularly interesting or meaningful. For example, it is relatively easy to construct an image that resembles noise, but is classified with high confidence as a cheetah [28]. This points to an important factor regarding neural network robustness: in general, we are less interested in robustness of the neural network over the entire image space, and are more curious of the robustness *over the realistic image space* $\mathcal{D}$. Test generation techniques for DNNs not appreciating this factor will likely generate pixel-level perturbations. To emphasize, our concern in this paper is on the implications of when neural networks are released in the wild, e.g. in what situations would there be a risk of failure? For such a goal, we believe that there is much benefit to identifying and generating realistic test cases.

3.2 SINVAD

We propose to use DNN based generative models, such as VAEs or GANs, to solve this problem. A schema of our approach SINVAD is provided in Figure 1. Generative models estimate the distribution of a provided dataset over the space of all images $\mathcal{I}$. In particular, VAEs effectively try to find a mapping between $\mathbb{R}^d$ and $\mathcal{D}$ (where d is the dimension of the latent space); the encoder ($E : \mathcal{D} \rightarrow \mathbb{R}^d$) maps from images to latent representations, and decoders ($D : \mathbb{R}^d \rightarrow \mathcal{D}$) operate vice versa. By executing traditional search-based optimization algorithms, but within the latent representation space, we can effectively restrict our search space to $\mathcal{D}$, instead of the full image space $\mathcal{I}$. Hence the result of search will mostly be images that closely resemble true images from $\mathcal{D}$. Note that since the VAE itself has only been trained on a small sub-sample of $\mathcal{D}$, i.e. the training set, it may or may not be a good mapping to/from the actual $\mathcal{D}$. We thus have to investigate the practical value of our approach empirically.

As for the search algorithm used, depending on the objective, we use either hill climbing or a genetic algorithm. In hill climbing, we attempt to obtain images that have a target Surprise Adequacy [20]. In our hill climbing algorithm, SINVAD adds a random value sampled from the standard normal distribution to a single element in the vector representation. The concrete algorithm is described in Figure 2. Given `S`, a random sampling function, `rep2img`, a function that transforms an image representation to an actual RGB image, and finally a fitness function, the algorithm performs hill climbing and returns an image. In essence, a random neighbor is visited by modifying one element of the vector image representation by adding noise $\epsilon \sim N(0, 1)$. The fitness of this neighbor is evaluated; if it has better fitness than the original representation, the neighbor is selected. The sampling function `S` is either one that samples $\mathcal{N}(0, 1)$ element-wise, or one that samples one of the test images, seen by neither the VAE nor the classifier.

Meanwhile, to construct images that look like one category, but are classified as a different category, as in adversarial examples, we used the genetic algorithm, as it experimentally yielded smaller changes. Using the fitness function in Equation 1 and starting from a seed image i_0,

$$f(i) = \begin{cases} \infty & N_c(i) = N_c(i_0) \\ |E(i) - E(i_0)| & else \end{cases} \tag{1}$$

Hill Climbing Algorithm

Input: sampling function S, representation to image function `rep2img`, fitness function `fit`
Output: optimized image

1: $R \leftarrow$ `S()`
2: $I \leftarrow$ `rep2img`(R)
3: $f \leftarrow$ `fit`(I)
4: count $\leftarrow 0$
5: **while** count < limit **do**
6: **for** $i = 1$ **to** $i = \dim(R)$ **do**
7: $\epsilon \sim N(0, 1)$
8: $R' \leftarrow R$
9: $R'[i] \leftarrow R[i] + \epsilon$
10: $I' \leftarrow$ `rep2img`(R')
11: $f' \leftarrow$ `fit`(I')
12: **if** $f < f'$ **then**
13: $R \leftarrow R'; I \leftarrow I'; f \leftarrow f'$
14: **end if**
15: **end for**
16: **end while**
17: **return** I

Fig. 2. Hill-climbing algorithm used to generate images for Figure 5.

the genetic algorithm could lead to images that are semantically ambiguous. The upper case denotes the case in which the new image's classification has not changed; we do not want to accept such images. The lower case is when the classification of the new image is different from the original image; in such cases we want the image to look as similar to the original image as possible. Using the fitness function defined above, we employ a genetic algorithm to find images that satisfy our criteria. The algorithm is described in Figure 3. The seed representation R is obtained by first sampling a random image from the test dataset, and passing it through the VAE encoder to get the latent representation of the image. The `rep2img` function is the VAE decoder, as was in the hill climbing case. In line 3, the initial population is constructed; in our experiment, we simply sampled random vectors from $N(\mathbf{0}, I_{\dim(\mathrm{R})})$ and added them to R. Lines 6-9 describe the population evaluation; lines 10-24 describe how the next generation's population is constructed. Single-point crossover is used (line 18); mutation is performed by adding a small noise vector on each genome (line 19). The dimensions that have the most impact are iteratively identified, and mutations are only applied to those dimensions (lines 19-22). Here elementCompare compares two vectors element-wise and returns 0 if they are different, 1 if they are equal (line 19), while we randomly choose which dimensions to reset to the original representation R (lines 21-22). We find this significantly speeds up convergence. Individuals with smaller fitness are selected for the next generation.

3.3 Feasibility Study

Using the hill climbing and genetic optimization algorithms that we described, we first qualitatively demonstrate that VAE-based optimizations yield more *realistic* images through qualitative analysis.

To contrast the results of using different image representations (i.e., whether to use the raw pixels or the VAE latent vector), we first show interpolation results in two representations. Specifically, given two images, we interpolate over both raw pixels and the latent space of SINVAD; specifically, we obtain raw pixel interpolations for a certain pixel $[i, j]$ for images i_0 and i_1 through $i_t[i, j] = (1 - t) \times i_0[i, j] + t \times i_1[i, j]$, where t is the degree of mixture between the two

Genetic Algorithm

Input: seed representation R, representation to image function `rep2img`, fitness function `fit`
Output: optimized image

$I \leftarrow$ `rep2img`(R)
prevBest $\leftarrow$ `fit`(I)
population $\leftarrow$ `InitializePop`(R)
count $\leftarrow 0$
while count < genNum **do**
  popPhenotypes $\leftarrow$ map(`rep2img`, population)
  popScores $\leftarrow$ map(`fit`, popPhenotypes)
  bestIndividuals $\leftarrow$ top10(population, popScores)
  currBest $\leftarrow$ best(popScores)
  if prevBest = currBest **then**
    mutSize $\leftarrow 0.7 \times$ mutSize
  else
    mutSize $\leftarrow 0.1$
  end if
  children $\leftarrow$ []
  for $i = 1$ **to** i = childNum **do**
    parents $\leftarrow$ randomChoose(bestIndividuals, 2)
    child $\leftarrow$ pointCrossover(parents)
    diff $\leftarrow$ elementCompare(child, R)
    child $\leftarrow$ diff $\cdot\, N(\vec{0}, I_{\dim(R)})$ + child
    mask $\leftarrow$ Binomial(dim(R))
    child $\leftarrow$ mask $\cdot\, R + (1 - mask) \cdot child$
    Add **child** to **children**
  end for
  population $\leftarrow$ bestIndividuals + children
  count $\leftarrow$ count + 1
end while
bestImage $\leftarrow$ best(popPhenotype, popScores)
return bestImage

Fig. 3. Genetic algorithm used to generate images for Figure 6.

images: for $t = 0$, the image is the same as i_0, while for $t = 1$ vice versa. Meanwhile, VAE interpolated images for image i_0 and i_1 are obtained with: $i_t = D((1-t) \times E(i_0) + t \times E(i_1))$. We compare these interpolated images side by side in Figure 4. Note that the raw representation has no understanding of semantics, hence it mixes the images unnaturally. On the other hand, SINVAD tries to naturally interpolate between the images, keeping intermediate images realistic. This suggests that SINVAD can generate realistic images more consistently.

This difference manifests in optimization results as well. Specifically, using hill climbing we optimized images to have a target surprise adequacy with different starting conditions, as in Figure 5. Images in Figure 5 (a, b) used 'raw' pixel representations, where the precursor representation r has the same dimension as normal images. Meanwhile, images from SINVAD-based optimization are presented in Figure 5 (c), and real test images are presented in (d). Notice the contrast between Figure 5(a, b) and (c, d): while images using pixel search look like some noise has been added to an image (or the image itself looks like noise), SINVAD-generated images and naturally occurring data images both look

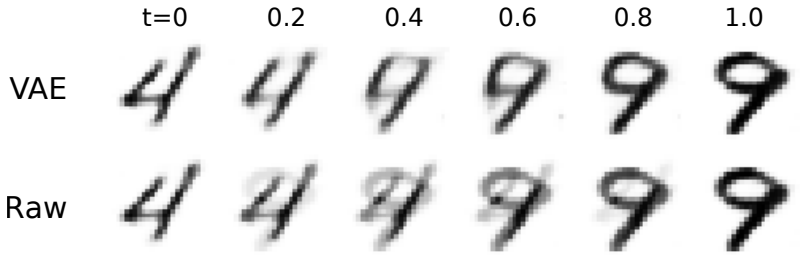
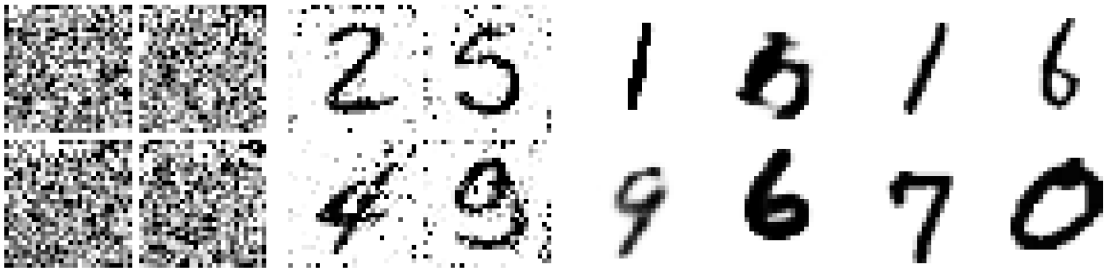


Fig. 4. Example of interpolation by SINVAD, done in the VAE latent space.

like natural digits. This qualitatively suggests that the representation used when optimizing images has a significant impact on the quality of the results of the search algorithm.



(a) Random seed, pixel search (b) test seed, pixel search (c) Random seed, SINVAD (d) Test dataset images

Fig. 5. Images produced by optimization either at the pixel level (a, b) or using SINVAD (c). For comparison, real images sampled from the MNIST test dataset are provided (d). Despite having the same fitness value, SINVAD-generated images resemble real dataset images to a greater extent.

Finally, using the genetic algorithm, we could generate semantically ambiguous images that represented the edge cases of classification, and which can thus play an important role when testing neural network classifiers, as shown in Figure 6. Observe that while the images resemble digits, precisely which digit they are is slightly ambiguous. On the other hand, raw pixel search does not yield realistic results as in Figure 5, reducing the utility in terms of analyzing and reasoning about the network.

4 RESEARCH QUESTIONS

We evaluate SINVAD via the five research questions below.

RQ1. Human Perception: How confused are humans when SINVAD generated images are presented, relative to other image generation techniques? To support our hypothesis that SINVAD restricts the search space so that it resembles the true data distribution $\mathcal{D}$, we perform a human study. In particular, we perform search using both SINVAD and pixel-based methods to find images close to a trained neural network's decision boundary. We then ask human evaluators how much the image resembles the class on each side of the decision boundary; the difference between these two scores indicates the level of evaluator confusion. The underlying assumption is that well-constructed

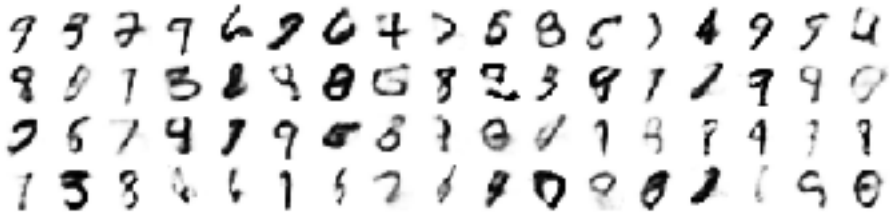


Fig. 6. Ambiguous images generated through GA optimization using Eq. 1 as the fitness function.

search spaces resembling $\mathcal{D}$ will generate images that are confusing to humans as well, while uninformed search spaces will not, because when search is performed on unrestricted search spaces, optimization techniques find a way to fool neural network classifiers without actually changing the semantics of the image; on the other hand, when search is restricted to a well-constructed space that is similar to $\mathcal{D}$, the only way to generate an image confusing to a neural network classifier is to generate a semantically confusing image. For example, pixel-based search techniques would mostly fool the neural network without changing the image semantics, while SINVAD, which search space resembles $\mathcal{D}$, would generate images that are recognizable but confusing.

RQ2. Boundary Testing: How close are SINVAD generated images to the decision boundaries of neural networks, as measured by class probability similarity? We employ the classifier probabilities of the two most likely classes of an input to confirm that SINVAD is capable of generating images close to the decision boundary of a neural network. If the probabilities of the top two classes have little difference, then an image can be rightfully said to be on or near the decision boundary, as the classifier is not 'confidently' declaring that an image is of a certain class; on the other hand, if the class probabilities are significantly different, the image is likely far from the decision boundary. We explore this property to check whether SINVAD can indeed generate images on the boundary of neural networks, enabling boundary testing.

RQ3. Problem Detection: How much more quickly does SINVAD identify problems in training data when compared to dataset-provided test data? We find pairs of categories that a given DNN image classifier is potentially vulnerable to using the semantic approximation that SINVAD provides. For example, given an image classifier, is it more prone to errors between categories 0 and 1, or 4 and 9? By identifying these 'weak spots' of a neural network, a developer could perform next steps, such as gathering more data or training the classifier further. To find these pairs, we use SINVAD and attempt to find images that look like class A but are classified as class B. The success rate of such an operation differs based on the (A, B) pair, and we suggest that this success rate is related to the risk of a classifier making this type of error. We experimentally verify this point by generating a 'polluted' training dataset in which some images are purposely mislabeled, and check if SINVAD finds these issues more quickly than the test error rate.

RQ4. VAE Architecture: How does the architecture of the VAE influence input realism? Meanwhile, the VAE architecture can have significant impact on the realism of an input. We investigate the impact that two architectural factors have on the output quality: the size of the latent space, d, and the choice of VAE layer (fully connected or convolutional). As a means of evaluation, we focus on how well a VAE can *reconstruct* an image. Reconstruction results are evaluated both qualitatively (via showing actual output images) as well as quantitatively, by measuring classifier accuracy on reconstructed images. While we could not retrain the VDVAE models used for the generation of ImageNet-32 images due to their high training cost (requiring up to 99GB of GPU memory [15]), we compare them

with linear VAEs to suggest and compare VAE architectures. We hope to provide guidelines on how to construct VAEs for SINVAD via this research question.

5 EMPIRICAL ANALYSIS

5.1 Experimental Setup

Four datasets are used throughout the paper. MNIST [26] is a grayscale digit image dataset, each 28×28 pixels large. There are 50,000 training, 10,000 validation, and 10,000 test images, but in this paper all training and validation images are used for training. Two closely related datasets are also used: Extended MNIST [9] (EMNIST) for alphabet letters, and Fashion-MNIST [40] (FMNIST) for various fashion items. Both have the same image size and a similar data size to MNIST. ImageNet-32 is a widely-used dataset containing RGB real-world images derived from the ImageNet [11] image classification benchmark, which has 1,000 categories, with images downscaled to a 32×32 pixel size from the original ImageNet data. There are 1,281,167 training images and 50,000 test images.

For MNIST, a VAE that has one hidden layer of size 1,600 for both encoder and decoder structures is used. The encoding vector size is 400. For our VAE architecture experiment in RQ4, we also employ a network with 8 hidden convolutional layers and encoding vector size 800 in both the encoder and decoder. VAEs for both datasets are trained for 50 epochs, which was sufficient for training to convergence. Finally, we use a pre-trained Very-Deep VAE (VDVAE) model [15] for the ImageNet-32 dataset that we use, as the ImageNet dataset is structurally complex and consequently requires significant more complexity to model using a VAE.

Regarding the neural networks to emulate the DNNs under test, a convolutional neural network with 6 hidden layers is used for MNIST, EMNIST, and FMNIST. Meanwhile, we use the AlexNet [23] architecture as a classifier for the ImageNet-32 dataset. The likelihood-based surprise adequacy (LSA) metric is calculated using the activation traces of the penultimate layer, i.e. the input vector to the final neural network layer which produces softmax logits.

SINVAD is implemented in Python version 3.6, using PyTorch version 1.5.1 to implement all neural networks. Code and model data are available from https://github.com/coinse/SINVAD. Any experiment requiring computational resources was done on machines with Intel Core i7 3.4GHz CPU, 32GB RAM, and a NVIDIA Titan X GPU.

5.2 Human Perception (RQ1)

It is important to thoroughly investigate whether SINVAD can actually induce a search space that resembles the realistic image distribution. To this end, we perform a quantitative human study to verify that SINVAD's search space resembles the realistic image space more than pixel-based methods. To do so, we evaluate how much each image generation technique is confusing to human evaluators, instead of directly asking users how realistic an image is.

The rationale behind this experiment setting is as follows. When one performs search over the unrestricted pixel space (or for that matter, a space that does not resemble $\mathcal{D}$), the search process simply exploits the inner workings of a neural network to generate an image that does not resemble any image in $\mathcal{D}$ and hence looks semantically unchanged (i.e., is clearly the same class identity with some noise added), while the subject neural network will classify it incorrectly. Such problems have been explored by Nguyen et al. [28], who demonstrate that unrestricted optimization can yield images meaningless to humans. Meanwhile, a search space that is closely tied to the underlying data distribution $\mathcal{D}$ will yield images that are confusing to humans as well, as the search process cannot simply find shortcuts to fool the neural network, assuming the neural network performs well enough on images from $\mathcal{D}$. Determining whether a technique can generate a recognizable image that is also confusing to humans also testifies to the robustness of the search space: the

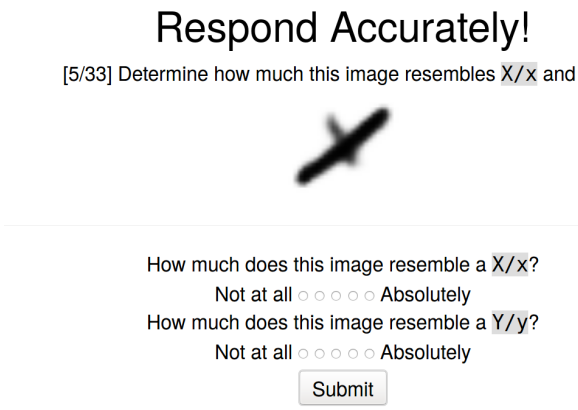


Fig. 7. Screenshot of a question posed to the human evaluators.

fact that a technique can generate such realistic confusing images that are clearly not in the original dataset suggests that the technique is a robust representation of the realistic data distribution $\mathcal{D}$ and not simply memorizing images. In turn, such a technique should be able to provide test cases that data-agnostic pixel-based techniques could not easily find.

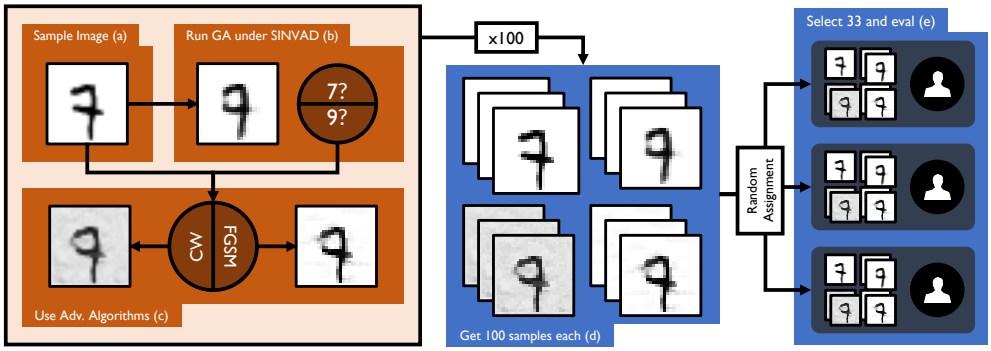


Fig. 8. Diagram of the human study performed.

A diagram explaining how the human study was performed is presented in Figure 8. The images used in the survey are constructed as follows. First, we sample a seed image (Figure 8 (a)). The original class of this image is called the *source class* (7 for the example in Figure 8 (a)). Next, we use SINVAD and the fitness function from Eq. 1 to generate an image that is on the decision boundary of a certain trained network. After doing so, the image will have a most probable class and a second most probable class based on the confidence of the classifier; we keep the record of these (the final optimized image and the two most probable categories (7 and 9) are depicted in Figure 8 (b)). Among these two, the one that was not the source class is called the *target class*. We now employ two adversarial attack techniques, FGSM [13] and C&W [6], that start from the same seed image as SINVAD and generate images that are on the decision

boundary of the same source and target classes, as shown in Figure 8 (c). We repeat this process 100 times to obtain 400 images overall (Figure 8 (c)). Each image is shown to a human evaluator, and the human will evaluate how much each image resembles the source and target class, respectively, answering on a Likert scale from one to five (Figure 8 (e)). An example screenshot of the question form, as shown to users, is provided in Figure 7. Each participant was asked to evaluate a total of 33 images, which were randomly assigned. Specifically, we took 10 random images from SINVAD, FGSM, and C&W, while 3 test images are used to verify that the workers understood the task.

We use two datasets, MNIST and EMNIST, for these experiments, as it is relatively easy from the human perspective to imagine confusing images for these datasets, while it is difficult to grasp what constitutes confusing images in some other datasets (if we think of FashionMNIST for example, it is difficult to think of an image that is plausibly both a sneaker and a t-shirt). 300 participants were recruited, and combined with the fact that each participant evaluates 3 real images and 30 potentially confusing images, each real image was evaluated about 10 times on average, while the confusing images were evaluated 30 times on average. The survey was performed via the Amazon MTurk platform[1]. Our human study was approved to be exempt from review by KAIST Institutional Review Board as it poses minimal risk for participants (IRB-21-208).

Images that are perceptually 'in between' the classes, and thus confusing, should have little absolute difference in their user-rated resemblance. We call the maximum score minus the absolute difference in the ratings of the categories the *confusion score*:

$$\text{Confusion score} = 5 - |\text{target score} - \text{source score}| \tag{2}$$

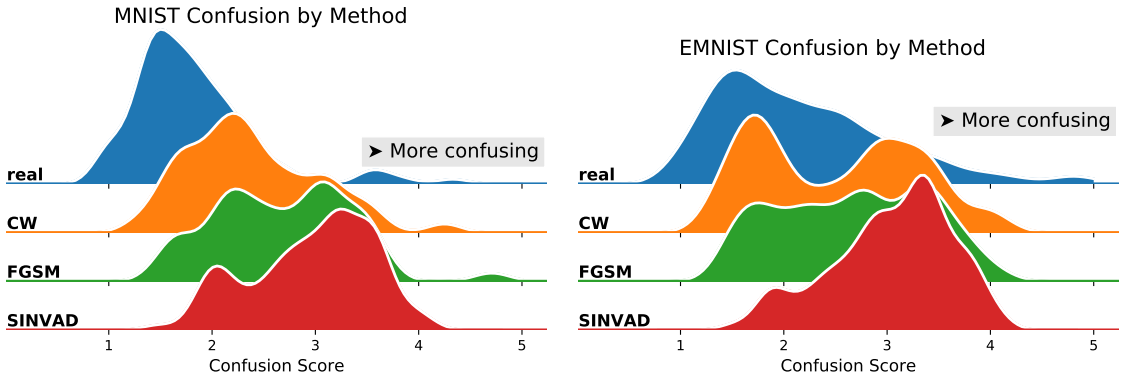
Using the images provided in Figure 8 (a-c) as examples, we show how the confusion score is calculated. A user presented with the sample image from Figure 8 (a) will be asked whether this image looks like a 7 or a 9. Suppose the user gave a score of five to class 7 and a score of one to class 9. The confusion score is then $5 - |5 - 1| = 1$, its lowest possible value - for the participant was sure that the image was a 7 and not a 9. On the other hand, suppose the participant saw the image in Figure 8 (b) and gave a score of three to class 7 and a score of three to class 9. The confusion score is then $5 - |3 - 3| = 5$, its highest value, as the participant was not sure of the image's identity.

Note that confusion score measures the *interclass* confusion, as opposed to *semantic confusion*, i.e. it measures how confusing an image is between two valid categories, and is not directly measuring whether an image is within the realm of semantic validity or not.

Figure 9 shows the distribution of the average confusion score for the 100 evaluated images of each technique. As these results show, SINVAD's confusion score distribution is shifted more toward the confusing side than any other image generation technique. This shows SINVAD's search space is more closely coupled with human perception of valid images when compared to other techniques. Statistical tests also back this point: by performing the Welch's test on the confusion score data, we could find that the difference with the second most confusing compared approach (FGSM) was statistically significant for both datasets ($p < 0.001$). Measuring the effect size of using SINVAD over FGSM using Cliff's delta, we found an effect size of 0.259 and 0.297 for MNIST and EMNIST, respectively, corresponding to a small to medium effect size [34]. Ultimately, these results demonstrate that SINVAD is indeed providing a better alternative to the pixel-based search space.

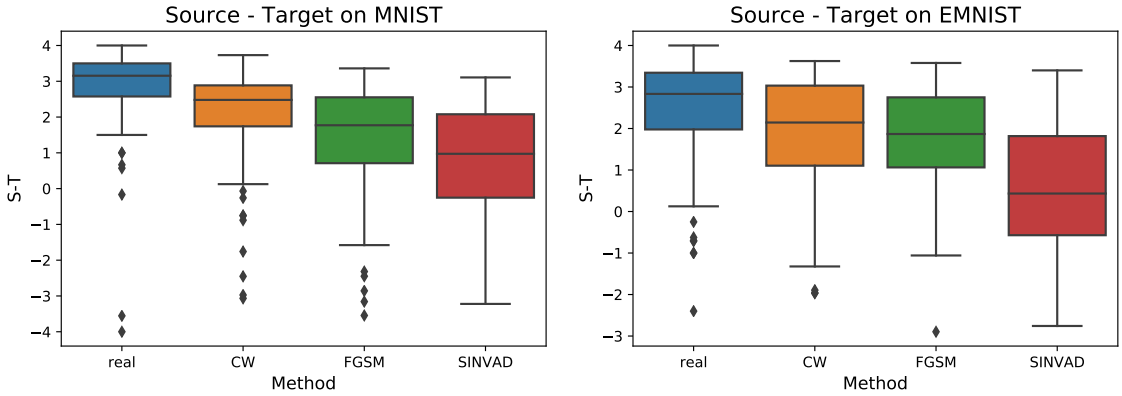
Meanwhile, we analyze how much influence the seed image has on human perception of the appropriate category. Recall that we started with the same seed image for SINVAD, FGSM, and C&W to generate images that are on the decision boundary of a given neural network. A method that indeed well models the perceptual space of the data, when

[1] https://www.mturk.com/



(a) (b)

Fig. 9. Confusion score distribution by image generation method.



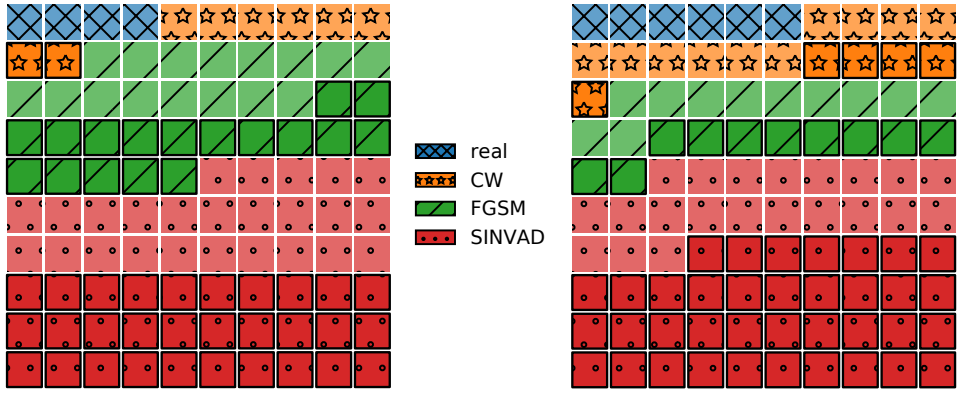
(a) (b)

Fig. 10. Boxplots for the target score subtracted from the source score. A high value means that the image significantly resembles the original label; a lower value means that the image more closely resembles the target label.

making confusing images, would not be biased toward either the seed image category or the target category. On the other hand, a method that has a similar average confusion score might still only be inching toward the semantics of a target class, mostly retaining the characteristics of the seed image. To investigate this issue, for each generated image, we subtract the target image score from the source image score. The closer the average value of this is to zero, the closer the image is to be truly perceptually confusing.

As the results depicted in Figure 10 show, SINVAD can generate images that are less likely to be skewed toward the seed image that it started with. Again, this shows that SINVAD indeed defines a search space that is closer to human perception, generating images that are more perceptually balanced than other methods.

It is also illuminating to compare how confusing images from different techniques are when they started with the same seed image. Using the confusion score we compare the evaluation results of images starting from the same seed



(a) MNIST (b) EMNIST

Fig. 11. For each quadruple of images starting from the same seed image (Figure 8(a-c)), a colored square is presented signifying which method's image was the most confusing within that quadruple. Boldly colored/outlined results signify that the difference in confusion was statistically significant.

(a) MNIST (b) EMNIST

Fig. 12. Example real images rated as 'most confusing'.

Dataset	Test	C&W	FGSM	SINVAD
MNIST	6.24	6.24	6.15	6.12
EMNIST	6.44	6.49	6.38	6.55

Table 1. The average acceptability of images for each method.

image for each of the 100 seed images. The results are presented in Figure 11. The dotted red squares represent seed images in which SINVAD was the most confusing method; as one can see, controlling for seed images, we find that SINVAD was again the most confusing method. While some real images were rated as most confusing, this is not due to a flaw in the experiment, as some of these real images had an actually dubious identity, as presented in Figure 12. To establish the significance of these results, we performed a two-sided Welch's unequal variances t-test on the confusion score results of each technique with a power of 0.05 and colored them darker in Figure 11. While the standard student's t-test assumes the sample means of each population would have equal variance, in our case (i) the number of evaluations for each image in the sample could differ, and (ii) we had no reason to think *a priori* the evaluation variance would be similar between methods: hence our use of Welch's test, which is suggested in these circumstances. Note that in over half of the cases, the difference in the confusion score between categories was statistically significant.

Dataset	Test	C&W	FGSM	SINVAD
MNIST	6.13	6.69	6.90	**7.20**
EMNIST	6.64	6.93	7.19	**7.35**

Table 2. The average time in seconds respondents took to evaluate an image from each method.

One may argue that using the confusion score alone would be a misleading measure. For example, simply showing static noise-like images would get a low rating for both classes, and thus the confusion score would be high. This can be mitigated by looking at the *sum* of the responses to the questions instead of just looking at the difference between the scores: we call this the *acceptability* of the image. The acceptability of all responses, averaged over each method, is presented in Table 1. The results show that SINVAD is comparable to other techniques and indeed real test images when it comes to image acceptability, demonstrating that SINVAD is not just making meaningless images, but rather is making meaningful images that are more confusing than baseline techniques while remaining semantically valid. Meanwhile, the difference between SINVAD and real images in terms of acceptability was not statistically significant under Welch's test in both datasets ($p > 0.05$).

Finally, we present indirect evidence that human evaluators found SINVAD's images confusing: the image evaluation time. One can assume that for confusing images, participants would generally take longer to answer, as they are presented with a difficult situation. The average time it took to evaluate images from each generation technique is presented in Table 2. SINVAD, on average, took more time than any other method to evaluate, providing indirect evidence that SINVAD could generate more confusing images. Statistical tests indicate that for MNIST, there is a statistically significant difference between SINVAD and FGSM processing time ($p < 0.05$), with an effect size of 0.045. On the other hand, for the EMNIST dataset, while SINVAD-generated images took longer to evaluate on average, the difference is no longer statistically significant ($p > 0.05$), and the effect size is lesser than that for MNIST (0.036). While it is difficult to clearly ascertain the reason behind this, we note that the EMNIST dataset is slightly more ambiguous than MNIST (note the number of real images labeled most confusing is greater in EMNIST), which causes all derivative images to be confusing and thus reduces their overall difference.

To conclude, the results of the human studies show both directly and indirectly that the images that SINVAD generates are perceptually the most confusing to humans. This in turn implies that the search space that SINVAD induces is more closely aligned to human perception than the pixel search space, supporting the conjecture that existing test generation techniques would benefit from searching in this space, rather than operating in the pixel space.

Answer to RQ1: A human study comparing human judgment on images reveals that SINVAD generates perceptually confusing images; SINVAD-generated images were evaluated as most confusing in 55% and 58% MNIST and EMNIST evaluations, respectively. This suggests that SINVAD is indeed searching using a search space that is aligned to human understanding of what constitutes a valid image.

5.3 Boundary Testing (RQ2)

The objective of the function provided in Eq. 1 was to obtain images close to the decision boundary of corresponding neural networks. To verify whether this is actually the case, we compare the prediction probabilities of the top two most likely classes. If an image is actually at the decision boundary, the final prediction probabilities should be similar; in

Dataset	SINVAD	Test set	p-value	Cliff's Delta
MNIST	**0.0638**	0.9957	$p < 10^{-4}$	-0.98
FashionMNIST	**0.1330**	0.9186	$p < 10^{-4}$	-0.92
ImageNet-32	**0.0423**	0.1953	$p < 10^{-4}$	-0.56

Table 3. A comparison of the difference in top-two category probabilities between provided test images and SINVAD-generated 'boundary' images. The SINVAD-generated images have smaller differences, suggesting that they are closer to the decision boundary.

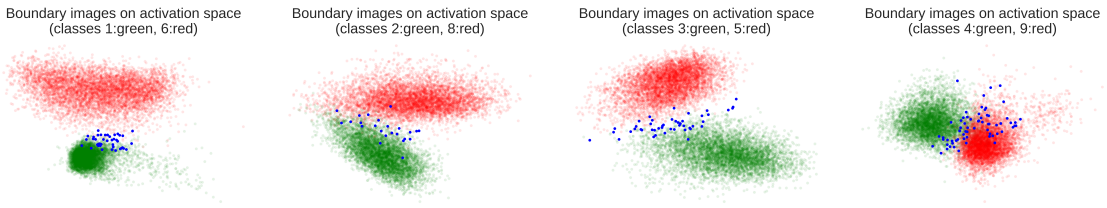
contrast, an image that the classifier deems very likely to be in one specific class is likely not at the decision boundary of the classifier. Using the objective function in Eq. 1, we first sample 10,000 images for MNIST and FashionMNIST datasets, and 200 for ImageNet, as inference takes a significantly longer time. Subsequently, we measure the prediction variability using the following metric:

$$U(x^*) = \max(\hat{y^*}) - \max(\hat{y^*} \setminus \{\max(\hat{y^*})\}) \tag{3}$$

where $\hat{y^*}$ is the set of the network's confidence values for each class on image x^*. We use this metric as it continuously measures how close an image is to the decision boundary of a neural network; when the difference is smaller, the image is closer to the decision boundary. Table 3 shows the comparison results for this experiment, conducted on MNIST, FMNIST, and the ImageNet-32 dataset.

As the table shows, the boundary images cause the probabilities of the top two classes to be more similar than when the same neural network is exposed to the default test images of each dataset. In turn, this shows that the SINVAD-generated images are close to the neural network's decision boundary itself. It is noteworthy that the images in the ImageNet-32 dataset are in general closer to the decision boundary. This is likely due to the greater complexity of the ImageNet dataset, as well as the greater number of categories (1,000). Consequently, the difference in boundary proximity between test set images and SINVAD-generated images is lesser than in the simpler MNIST-based datasets. Nonetheless, SINVAD can consistently generate images that are on the boundary of this more complex dataset as well. When using Welch's t-test to determine the significance of these differences, for all three datasets the difference was statistically significant ($p < 10^{-4}$). The effect size was large as well: for the ImageNet dataset, the difference was a large -0.56 (the minus indicating that SINVAD generated images that have a smaller probability difference between the top-two predictions); for MNIST and FashionMNIST, the effect size was even more substantial at -0.98 and -0.92, respectively.

Using the PCA dimensionality reduction technique, we can also qualitatively assess that the images that SINVAD makes are close to the decision boundary. We first gather the activation vectors of the test images of a chosen class pair. We use PCA to generate a projection that maximizes the projected variance of these vectors. Next, we obtain the activation vectors of the images that are between the selected class pair, and use the same PCA projection to see where the intermediate images are with respect to the test images. The projection results of a few example pairs are presented in Figure 13. Observe that the boundary images are indeed between the two clusters of each class in all cases, implying that the images that SINVAD generated are indeed nearby the decision boundary of the neural network.



(a) Images on 1 and 6. (b) Images on 2 and 8. (c) Images on 3 and 5. (d) Images on 4 and 9.

Fig. 13. PCA-projected location of SINVAD-generated boundary images in the activation vector space. Red and green dots are vectors of each image class, while blue dots are SINVAD-generated boundary images.

Answer to RQ2: The images generated by SINVAD were closer to the decision boundaries of DNNs than default test set images by a statistically significant degree ($p < 10^{-4}$). Thus, SINVAD can indeed generate images that are difficult for neural networks to classify by being on their decision boundary.

5.4 Problem Detection (RQ3)

To answer RQ3, we use SINVAD to identify pairs of classes that a given neural network finds particularly difficult to distinguish. We design a targeted fitness function shown in Equation 4, where i_0 is a seed image and t is a target class different from that of i_0.

$$f(i) = \begin{cases} \infty & \text{if } N_c(i) \neq t \\ |E(i) - E(i_0)| & \text{otherwise} \end{cases} \tag{4}$$

The results show that this optimization quite often fails to find an image of target class t starting from the seed image i_0, which is what we actually anticipate in this research question. Our focus is not on whether it succeeds or fails, but how often it succeeds, i.e., the proportions of successes. We posit that attacks on potentially confusing image pairs have a higher success rate, while image pairs that are clearly distinguished have a lower success rate. Concretely, we define a *GA class escape rate* as the number of times an attack successfully finds a solution that causes the DNN to classify an image as the target class divided by the number of all attempts. We can compare this to the targeted error rate which is the number of times an image in the dataset test set with the source class is classified as the target class. The change in GA class escape cases for MNIST is more pronounced than the targeted error cases of the test cases, as is showcased in Figure 14, which shows the number of successful "escape attempts" by the optimization technique out of 1000 attempts. It juxtaposes the confusion matrix using provided test cases and the GA class escape count for a simple neural network classifier.

To show the effectiveness of the GA class escape rate in detecting weaknesses of a given neural network, target labels are artificially mixed during the training phase according to a proportion α. For each α, a corresponding neural network is trained. For each such neural network, we measure the error rate between the mixed pair and the GA class escape rate. For the MNIST dataset, data between three pairs that are usually not confused per Figure 14(b) (Specifically, (0, 1), (2, 5), and (6, 7)) are mixed with varying αs; we use a similar procedure to identify pairs to mix in the ImageNet-32 dataset, namely (spider, Siamese cat), (jellyfish, maze), and (airliner, dog). The detection rate is defined as follows: for the SINVAD-led method, we calculate the success rate when there was no mixture s_0. For the success rate at mixture level α

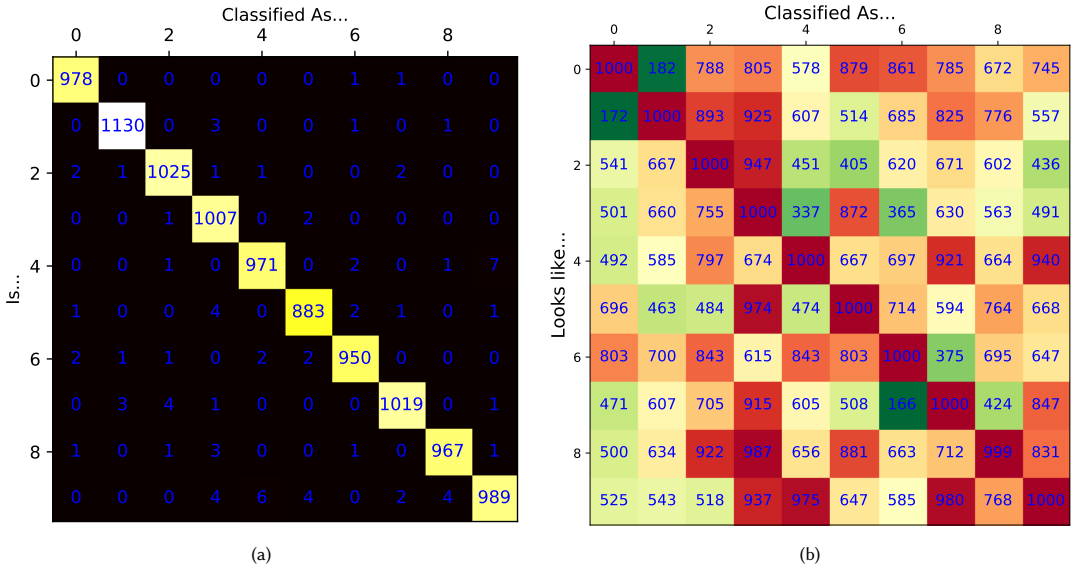


Fig. 14. Comparison between (a) the error matrix of the provided MNIST test set and (b) the number of successful optimizations of the GA algorithm out of 1000 attempts. Note that the differences are more pronounced in the GA algorithm's matrix.

s_α, the detection rate is defined as $s_\alpha - s_0$. Meanwhile, the detection rate for test accuracy is defined as $\frac{\text{errors}}{\text{number of images}}$. The ensuing detection rate results are presented in Figure 15.

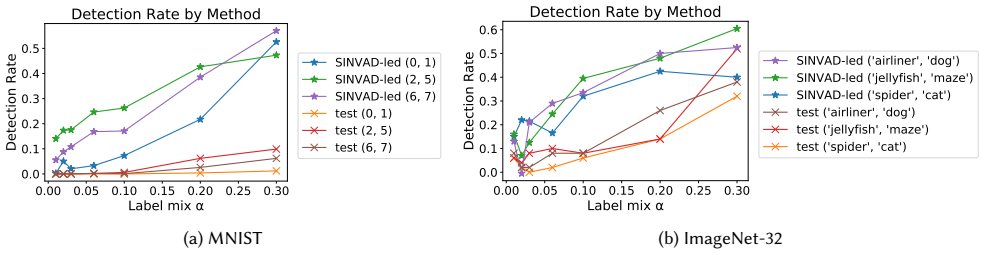


Fig. 15. Performance change when labels are mixed. GA class escape rate increases more rapidly than error rate when labels are mixed.

As evident from the figure, SINVAD-led detection rate has a tighter correlation relationship with the gradual mixing of labels in both the MNIST and ImageNet datasets; default test cases only pick up this change when labels have been polluted a great deal. This shows the effectiveness of SINVAD-generated test cases in finding potential flaws in neural networks. Further inspection reveals that while specific scale in escape rate may differ by class pair, the pattern in which escape rate is more sensitive to polluted data than usual test accuracy is clear. This is not just an issue of the detection rates being at a different scale; the average Pearson correlation coefficient for the MNIST dataset between SINVAD-led detection rate and α at the earlier stages of up to $\alpha \leq 0.1$ is 0.877, while the correlation between test error

rate is 0.525; for the ImageNet-32 dataset, SINVAD-led detection had a correlation of 0.829, while test error rate had a correlation of 0.454. Test error rate only catches up when the dataset is extremely polluted ($\alpha \geq 0.2$), after which the correlations between SINVAD-led detection rate and test error rate and α become similar (in MNIST, SINVAD's 0.977 to test error rate's 0.952; in ImageNet-32, SINVAD's 0.911 to test error rate's 0.930).

One can imagine this property being used in many contexts; for example, when there are many categories and it is difficult to think of all potentially confusing cases, one can use SINVAD to identify which category pairs are particularly at risk, and verify whether there may be any safety-critical confusions in each case. As such, we can conclude through this study that SINVAD can indeed be useful in testing and can reveal important properties of neural networks under test.

Answer to RQ3: Using images generated from SINVAD, problems in the network caused by inaccuracies of training data labels could be more consistently detected in weaker ($alpha \leq 0.1$) levels of pollution, as indicated by the correlation (0.877 for SINVAD versus 0.525 for test images in MNIST). This highlights the potential of SINVAD to identify problems in neural networks.

5.5 VAE Architecture (RQ4)

To answer RQ4, we evaluate the output quality of VAEs when the dimension d of the latent space or the layer type (fully connected or convolutional) is changed, we consider the reconstruction performance of different VAEs. We first present qualitative reconstruction results in Figure 16, with (a) and (b) being images from the MNIST dataset, while (c) and (d) are from the FashionMNIST dataset. For typical images, the VAEs will usually successfully reconstruct the image successfully even with very small latent spaces, as can be seen in Figure 16 (a) and (c). On the other hand, for atypical images, VAEs with smaller latent spaces struggle to capture the characteristics of the image. For example, in Figure 16 (b), the original image is a 7, but VAEs with smaller latent spaces reconstruct it as a 3 (Linear-1), 9 (Conv.-1), or 4 (Linear-2). However, we do not find a significant quality difference between linear and convolutional VAEs qualitatively.

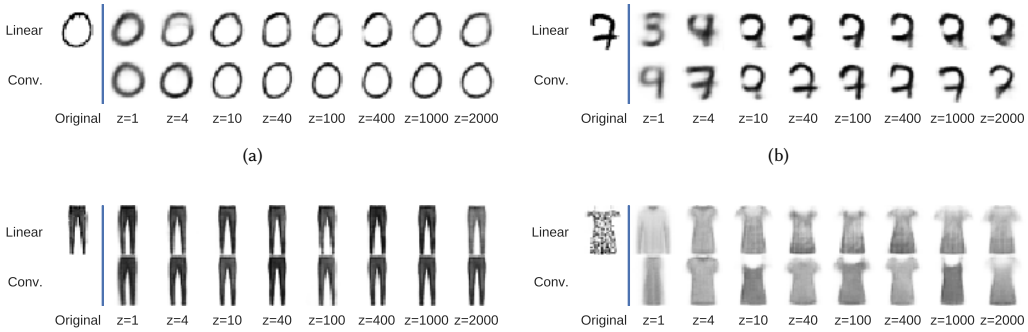
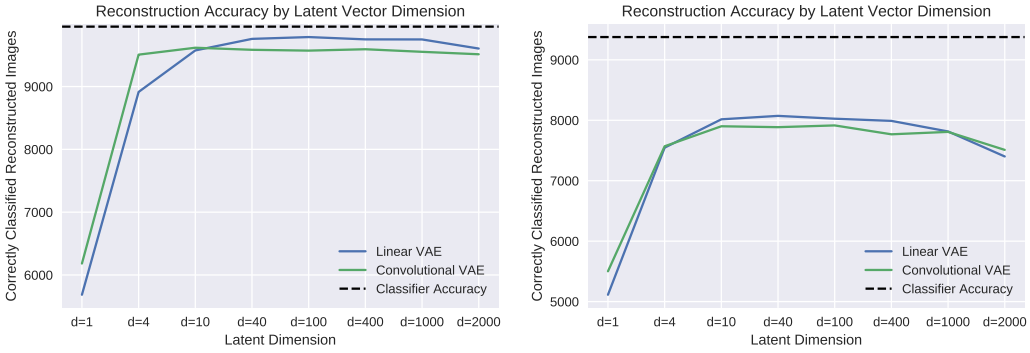


Fig. 16. Reconstruction examples by VAE architecture and latent dimension size.

These qualitative observations are further validated by our quantitative results, in which we evaluate how often a DNN classifier could recognize an image after the VAE reconstructs it from the latent space, in Figure 17. At smaller latent space sizes, VAEs fail to effectively reconstruct images, and consequently the classifier accuracy drops significantly.

As the latent space becomes larger, VAEs can capture more details of the image, and consequently reconstruction accuracy increases and plateaus, until the dimension of the VAE becomes larger than the dimension of the image itself ($d > 784$), at which point the reconstruction accuracy starts to drop. It is worth noting that smaller latent spaces may have benefits, as computation is quicker on smaller spaces and they are less likely to suffer from the curse of dimensionality [2]. Thus, provided reconstruction accuracy has plateaued, it may be advantageous to use VAEs with low-dimensional latent spaces when exploring the space with SINVAD. Meanwhile, we note that linear VAEs were slightly better at reconstructing images than convolutional VAEs for these datasets.



(a) MNIST (b) FashionMNIST

Fig. 17. Classifier accuracy on original test set and a 'reconstructed' test set that went through the VAE.

However, for more complex datasets, different VAE architectures can lead to substantially different results. While we could not perform similar retraining experiments with the VDVAE architecture due to its high training cost, on the ImageNet-32 dataset, we find that the VDVAE architecture that we used for our experiments leads to qualitatively superior reconstruction results (Figure 18). In addition, when evaluating the accuracy of our 10-class ImageNet classifier from RQ3, we find that its accuracy on the original test images was 67.4%, while the reconstructed image accuracy using VDVAE and linear VAEs was 59.5% and 32.6%, respectively. Thus, we can conclude that for more complex datasets, using more advanced and complex VAE architectures is recommended.

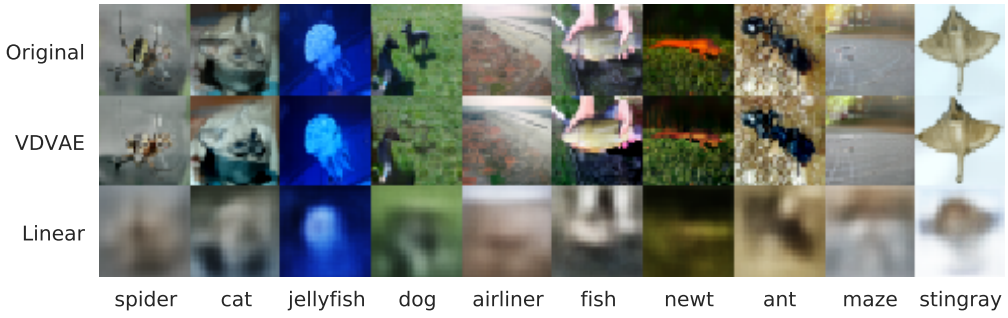


Fig. 18. Reconstruction of the ImageNet-32 dataset using linear VAE and VDVAE architectures.

Answer to RQ4: For simpler image datasets like MNIST, it is sufficient to use simple VAE architectures consisting of fully connected layers. As long as the latent vector size is not excessively small ($d < 10$), VAEs appear robust to different latent vector space sizes as well - reconstruction accuracy did not differ by more than 10%. For more complex datasets like ImageNet-32, the use of complex VAE architectures that can sufficiently model the inputs, such as the VDVAE architecture, is recommended.

6 THREATS TO VALIDITY

Internal validity is threatened when there are alternative factors that could explain our experimental results. One possible threat to our feasibility study is that optimization over raw pixels is perhaps not the best representation of all images. While this is possible, to construct better representations of images that incorporate global structures, one may need increasingly complex models, which may end up to being similar to search in the VAE latent space eventually. The human study results in RQ1 may be due to individual evaluator perception difference or mistakes. To mitigate such concerns, we performed the study on a large number of participants (ca. 300 for each dataset). The results presented in RQ2 and RQ3 are based on stochastic experiments; to ensure that these results are not from pure chance, we ran the corresponding experiments over a large number of images ($n \geq 100$).

External validity is threatened when the findings of a study may not generalize. In this work, we used a fixed number of neural network architectures for convenience; whether the results presented in this work will generalize to other neural networks is a question that requires further research. Meanwhile, the datasets used in this study were chosen as it was easier to discern the semantics of them, against datasets with a large number of categories that may be confusing to untrained humans. The results of SINVAD on other datasets must be verified with further experimentation.

7 DISCUSSION

7.1 Limitations

While SINVAD has demonstrated its capability to generate tests and help understand/diagnose the behavior of neural networks, it does not come without its own limitations. For one, there is the risk that SINVAD may be sensitive to the particular architectural details of the underlying VAE. As shown in RQ4, SINVAD is resilient to different values of hyperparameters such as the dimensionality of the latent layer. Nonetheless, as we recommend in RQ4, one may try changing the VAE architecture depending on the complexity of the underlying data. A related limitation is the capacity of the VAE; our experience indicates that as more complex datasets must be modeled, more complex VAE architectures must be used. In our paper, we used multi-layer perceptron-based simple VAEs for the simpler MNIST-related datasets, while we used more complex VDVAE architectures for the more complex datasets (ImageNet-32). As complex VAEs often are more expensive to train, it may take some time to find the appropriate architecture for a target dataset.

As VAEs must be trained and thus learn from a given dataset, they may reflect any bias or problem that the original dataset may have. For example, if a face recognition dataset has only a small proportion of faces from a certain ethnicity or gender, the VAE will reflect that distribution, and only rarely make such images. As such, it may be difficult to use SINVAD to identify such *fairness* issues. Nonetheless, there is a wide field of literature that deals with this issue [22], and we believe SINVAD can provide a complementary role with these techniques.

7.2 Future Directions

While our work focuses on DNNs that classify images, future work could potentially explore different modalities and different generative models. The generative model used in our work, Variational Autoencoders, are also capable of modeling non-image data such as text [33]. Indeed, work that uses VAEs in the context of multi-modal datasets [35] or test suites [31] have also been proposed. As such, a promising venue of future work would be to investigate whether performing search on generative models that induce a latent space is a general method for synthesizing meaningful software input and data, as other and more powerful generative models are introduced they can also be used. Extending further, one could investigate how SINVAD could be used to test non-DNN machine learning techniques. As long as the fitness function we use in our search is agnostic to the model under test (as was generally the case in our studies), i.e. the results do not rely on internal computations of the model, our method should be able to generalize without modification.

While we have demonstrated the ability to make confusing inputs (RQ1), the VAEs we experimented with are not producing labeled inputs. That is, we cannot directly ask the model to produce an image of a certain label. However, there are different VAE architectures that can tackle this problem, such as conditional variational autoencoders [37], and such architectures could be applied to SINVAD depending on the specification of the problem. Such an ability could lead to other interesting ways of probing the capabilities of neural networks.

8 CONCLUSION

This paper introduces SINVAD, a technique that uses generative models to focus the search for images so that more realistic tests can be generated. Our qualitative analysis and quantitative human study verify that the space induced by our method is indeed similar to human perception of acceptable input space for neural networks.

Along with the results of the human study, our quantitative and qualitative experiments suggest that SINVAD is capable of generating inputs that are more likely to provide guidance to DNN developers under a variety of testing scenarios. Specifically, we also show that SINVAD can find images close to decision boundaries, which can be used for boundary value testing and for diagnosing whether the boundary is actually where it is expected to be; experiments also show that SINVAD is capable of providing unique insights into the behavior of DNNs by providing interpretable results through targeted class escape rate analysis. Finally, we experiment with VAEs of different architectures, and provide a few guidelines on how to select the appropriate VAE architecture for a given domain.

In summary, we believe that the shifting of focus from the low-level, “syntactic”, here pixel-level, perturbations to semantically meaningful changes that search coupled with SINVAD allows can benefit many interesting software engineering and testing applications.

ACKNOWLEDGMENTS

Sungmin Kang and Shin Yoo were supported by the Engineering Research Center Program through the National Research Foundation of Korea (NRF) funded by the Korean Government MSIT (NRF-2018R1A5A1059921), as well as the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (no. 2018-0-00769, Neuromorphic Computing Software Platform for Artificial Intelligence Systems). Robert Feldt has been supported by the Swedish Scientific Council (No. 2020-05272, ‘Automated boundary testing for QUality of AI/ML modelS’) and partly also by the WASP project “BoundMiner”.

REFERENCES

[1] Andrea Agostinelli, Timo I. Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, Matt Sharifi, Neil Zeghidour, and Christian Frank. 2023. MusicLM: Generating Music From Text.

[2] Richard Bellman. 1966. Dynamic programming. *Science* 153, 3731 (1966), 34–37.

[3] Tom Brown, Dandelion Mane, Aurko Roy, Martin Abadi, and Justin Gilmer. 2017. Adversarial Patch. https://arxiv.org/pdf/1712.09665.pdf

[4] Taejoon Byun and Sanjai Rayadurgam. 2020. Manifold for machine learning assurance. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*. 97–100.

[5] Taejoon Byun, Abhishek Vijayakumar, Sanjai Rayadurgam, and Darren Cofer. 2020. Manifold-based Test Generation for Image Classifiers. In *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 15–22.

[6] Nicholas Carlini and David A. Wagner. 2016. Towards Evaluating the Robustness of Neural Networks. *CoRR* abs/1608.04644 (2016). http://arxiv.org/abs/1608.04644

[7] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. 2015. Deepdriving: Learning affordance for direct perception in autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*. 2722–2730.

[8] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. 2017. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1907–1915.

[9] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. 2017. EMNIST: an extension of MNIST to handwritten letters.

[10] Ekin Dogus Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning Augmentation Policies from Data. https://arxiv.org/pdf/1805.09501.pdf

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[12] David Foster. 2019. *Generative Deep Learning*. O'Reilly.

[13] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *International Conference on Learning Representations*. http://arxiv.org/abs/1412.6572

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'14)*. MIT Press, 2672–2680.

[15] Louay Hazami, Rayhane Mama, and Ragavan Thurairatnam. 2022. Efficient-VDVAE: Less is more. *arXiv preprint arXiv:2203.13751* (2022).

[16] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. 2017. Safety Verification of Deep Neural Networks. In *Computer Aided Verification*, Rupak Majumdar and Viktor Kunčak (Eds.). Springer, 3–29.

[17] Robin Jia and Percy Liang. 2017. Adversarial Examples for Evaluating Reading Comprehension Systems. *CoRR* abs/1707.07328 (2017).

[18] Sungmin Kang, Robert Feldt, and Shin Yoo. 2020. SINVAD: Search-Based Image Space Navigation for DNN Image Classifier Test Input Generation. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*. Association for Computing Machinery, New York, NY, USA, 521–528. https://doi.org/10.1145/3387940.3391456

[19] Guy Katz, Clark W. Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. 2017. Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *CoRR* abs/1702.01135 (2017). http://arxiv.org/abs/1702.01135

[20] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proceedings of the 41th International Conference on Software Engineering (ICSE 2019)*. IEEE Press, 1039–1049.

[21] Diederik P. Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. In *ICLR*.

[22] Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. 2018. Adaptive Sensitive Reweighting to Mitigate Bias in Fairness-Aware Classification. In *Proceedings of the 2018 World Wide Web Conference (WWW '18)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 853–862. https://doi.org/10.1145/3178876.3186133

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. 2012. ImageNet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems* 25 (01 2012). https://doi.org/10.1145/3065386

[24] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *CoRR* abs/1607.02533 (2016).

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature Cell Biology* 521, 7553 (5 2015), 436–444. https://doi.org/10.1038/nature14539

[26] Yann LeCun, Corinna Cortes, and CJ Burges. 2010. MNIST handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist* 2 (2010).

[27] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, Jianjun Zhao, and Yadong Wang. 2018. DeepGauge: Multi-granularity Testing Criteria for Deep Learning Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*. ACM, 120–131.

[28] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), 427–436.

[29] Nicolas Papernot, Patrick D. McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. 2015. The Limitations of Deep Learning in Adversarial Settings. *CoRR* abs/1511.07528 (2015).

[30] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems. In *Proceedings of the 26th Symposium on Operating Systems Principles (SOSP '17)*. ACM, 1–18.

[31] André Reichstaller and Alexander Knapp. 2017. Compressing Uniform Test Suites Using Variational Autoencoders. https://doi.org/10.1109/QRS-C.2017.128

[32] Vincenzo Riccio and Paolo Tonella. 2020. Model-based exploration of the frontier of behaviours for deep learning system testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 876–888.

[33] Fanny Roche, Thomas Hueber, Samuel Limier, and Laurent Girin. 2018. Autoencoders for music sound synthesis: a comparison of linear, shallow, deep and variational models.

[34] Jeanine Romano, Jeffrey D Kromrey, Jesse Coraggio, Jeff Skowronek, and Linda Devine. 2006. Exploring methods for evaluating group differences on the NSSE and other surveys: Are the t-test and Cohen's d indices the most appropriate choices. In *annual meeting of the Southern Association for Institutional Research*. Citeseer, 1–51.

[35] Mostafa Sadeghi, Simon Leglaive, Xavier Alameda-PIneda, Laurent Girin, and Radu Horaud. 2019. Audio-visual Speech Enhancement Using Conditional Variational Auto-Encoder.

[36] Lea Schönherr, Katharina Kohls, Steffen Zeiler, Thorsten Holz, and Dorothea Kolossa. 2018. Adversarial Attacks Against Automatic Speech Recognition Systems via Psychoacoustic Hiding. *CoRR* abs/1808.05665 (2018).

[37] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning Structured Output Representation using Deep Conditional Generative Models. In *Advances in Neural Information Processing Systems*, C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (Eds.), Vol. 28. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf

[38] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. 2018. Constructing Unrestricted Adversarial Examples with Generative Models. In *NeurIPS*.

[39] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-neural-network-driven Autonomous Cars. In *Proceedings of the 40th International Conference on Software Engineering (ICSE '18)*. ACM, 303–314.

[40] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms.

[41] Shin Yoo. 2019. SBST in the Age of Machine Learning Systems: Challenges Ahead. In *Proceedings of the 12th International Workshop on Search-Based Software Testing (SBST '19)*. IEEE Press, Piscataway, NJ, USA, 2–2.

[42] Jie M. Zhang, Mark Harman, Lei Ma, and Yang Liu. 2019. Machine Learning Testing: Survey, Landscapes and Horizons. *IEEE Transactions on Software Engineering (to appear)* (2019).

[43] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. 132–142.