Automated Anomaly Detection via Curiosity-Guided Search and Self-Imitation Learning

Yuening Li, Zhengzhang Chen, *Member, IEEE*, Daochen Zha, Kaixiong Zhou, Haifeng Jin, Haifeng Chen, *Member, IEEE*, and Xia Hu

***Abstract*— Anomaly detection is an important data mining task with numerous applications, such as intrusion detection, credit card fraud detection, and video surveillance. However, given a specific complicated task with complicated data, the process of building an effective deep learning-based system for anomaly detection still highly relies on human expertise and laboring trials. Also, while neural architecture search (NAS) has shown its promise in discovering effective deep architectures in various domains, such as image classification, object detection, and semantic segmentation, contemporary NAS methods are not suitable for anomaly detection due to the lack of intrinsic search space, unstable search process, and low sample efficiency. To bridge the gap, in this article, we propose AutoAD, an automated anomaly detection framework, which aims to search for an optimal neural network model within a predefined search space. Specifically, we first design a curiosity-guided search strategy to overcome the curse of local optimality. A controller, which acts as a search agent, is encouraged to take actions to maximize the information gain about the controller's internal belief. We further introduce an experience replay mechanism based on self-imitation learning to improve the sample efficiency. Experimental results on various real-world benchmark datasets demonstrate that the deep model identified by AutoAD achieves the best performance, comparing with existing handcrafted models and traditional search methods.**

***Index Terms*— Anomaly detection, curiosity-guided search, experience replay, neural architecture search (NAS), self-imitation learning.**

I. Introduction

WITH the increasing amount of surveillance data collected from large-scale information systems such as the Web, social networks, and cyber-physical systems, it becomes more and more important for people to understand the underlying regularity of the vast amount of data and to identify the unusual or abnormal instances [1]–[3]. Centered around this goal, anomaly detection plays a very important role in various real-world applications, such as fraud detection [4], [5], cybersecurity [6], medical diagnosis [7], and social network analysis [8]–[10].

Driven by the success of deep learning, there has been a surge of interests [11]–[14] in adopting deep neural networks for anomaly detection. Deep neural networks can learn to represent the data as a nested hierarchy of concepts to capture the complex structure in the data and thus significantly surpass traditional anomaly detection methods as the scale of data increases [15]. However, building a powerful deep neural network system for a real-world complex application usually still heavily relies on human expertise to fine-tune the hyperparameters and design the neural architectures. These efforts are usually time-consuming and the resulting solutions may still have suboptimal performance.

Neural architecture search (NAS) [16]–[18] is one promising means for automating the design of neural networks, where reinforcement learning (RL) and evolution have been used to discover optimal model architectures from data [19], [20]. Designing an effective NAS algorithm requires two key components: the search space and the search strategy, which define what architectures can be represented in principles and how to explore the search space, respectively. The discovered neural architectures by NAS have been demonstrated to be on par or outperform handcrafted neural architectures.

Although the recent years have witnessed significant progress of NAS techniques in some supervised learning tasks such as image classification and text classification [16], [20], the unsupervised setting and the naturally imbalanced data have introduced new challenges in designing an automated anomaly detection framework.

1) *Lack of Search Space:* It is nontrivial to determine the search space for an anomaly detection task. In particular, since there is no class label information in the training data of an anomaly detection task, objective functions play an important role to differentiate between normal and anomalous behaviors. Thus, in contrast to

Manuscript received September 30, 2020; revised March 5, 2021 and June 2, 2021; accepted August 11, 2021. Date of publication September 1, 2021; date of current version June 2, 2022. *(Corresponding author: Yuening Li.)*

Yuening Li, Daochen Zha, Kaixiong Zhou, and Haifeng Jin are with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 USA (e-mail: yueningl@tamu.edu; daochen.zha@tamu.edu; zkxiong@tamu.edu; jin@tamu.edu).

Zhengzhang Chen and Haifeng Chen are with NEC Laboratories America, Princeton, NJ 08540 USA (e-mail: zchen@nec-labs.com; haifeng@nec-labs.com).

Xia Hu is with the Department of Computer Science, Rice University, Houston, TX 77005 USA (e-mail: xia.hu@rice.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2021.3105636.

Digital Object Identifier 10.1109/TNNLS.2021.3105636

the supervised learning tasks, we often need to find a suitable definition of the anomaly and its corresponding objective function for a given real-world data. One typical way to define the anomalies is to estimate the relative density of each sample and declare instances that lie in a neighborhood with low density as anomalies [13]. Yet, these density-based techniques perform poorly if the data have regions of varying densities. Another way to define anomalies is through clustering. An instance will be classified as normal data if it is close to the existing clusters, while the anomalies are assumed to be far away from any existing clusters [11]. However, these clustering-based techniques will be less effective if the anomalies form significant clusters among themselves [4]. The proper definition of anomalies not only requires domain knowledge from researchers and experience from data scientists but also needs thorough and detailed raw data analysis efforts. Thus, different from the search spaces defined by the existing NAS, the search space of automated anomaly detection needs to cover not only the architecture configurations but also the anomaly definitions with the corresponding objective functions.

2) *Unstable Search Process:* The search process may easily become unstable and fragile when anomaly detection compounds with architecture search. On one hand, the imbalanced data distributions make the search process easily fall into the local optima [21]. On the other hand, the internal mechanisms of the traditional NAS may introduce bias in the search process. For instance, the weight sharing mechanism makes the architectures that have better initial performance with similar structures more likely to be sampled [22], [23], which leads to misjudgments of the child model's performance.
3) *Low Sample Efficiency:* Existing NAS algorithms usually require training a large number of child models to achieve good performance, which is computationally expensive, while in real-world anomaly detection tasks, anomalies or abnormal samples are very rare. Thus, it requires the search strategy to exploit samples and historical search experiences more effectively.

To tackle the aforementioned challenges, in this article, we propose AutoAD, an automated anomaly detection algorithm to find an optimal deep neural network model for a given dataset. In particular, we first design a comprehensive search space specifically tailored for anomaly detection. It covers architecture settings, anomaly definitions, and corresponding loss functions. Given the predefined search space, we further propose a curiosity-guided search strategy to overcome the curse of local optimality. The search agent is encouraged to seek out regions in the search space that are relatively unexplored. The uncertainty about the dynamics of the search agent is interpreted as the information gain between the agent's new belief and the old one. Moreover, we introduce an experience replay mechanism based on self-imitation learning to enhance sample efficiency. It can benefit the search process by exploiting good experience in the historical episodes. To evaluate the performance of AutoAD, we perform an extensive set of experiments on eight benchmark datasets. When tested on the two important anomaly detection tasks—instance-level abnormal sample detection and pixel-level defect region segmentation—our algorithm demonstrated the superior performance, comparing with existing handcrafted models and traditional search methods. The experimental results also show that AutoAD has the potential to be applied in more complicated real-world applications.

The contributions of this article are summarized as follows.

1) We identify a novel and challenging problem (i.e., automated anomaly detection) and propose a generic framework AutoAD. To the best of our knowledge, AutoAD describes the first attempt to incorporate AutoML with an anomaly detection task, and one of the first to extend AutoML concepts into applications from data mining fields.
2) We carefully design a search space specifically tailored to the automated anomaly detection problem, covering architecture settings, anomaly definitions, and the corresponding objective functions.
3) We propose a curiosity-guided search strategy to overcome the curse of local optimality and stabilize the search process.
4) We introduce an experience replay mechanism based on the self-imitation learning to improve the sample efficiency.
5) We conduct extensive experiments on eight benchmark datasets to demonstrate the effectiveness of AutoAD and provide insights on how to incorporate AutoAD to the real-world scenarios.

II. Related Work

In this section, we review the related work on NAS. Recently, NAS has attracted increasing research interests. Its goal is to find the optimal neural architecture in a predefined search space to maximize the model performance on a given task. Designing an NAS algorithm requires two key components: the search space and the search strategy (optimization algorithm) [17].

The search space defines which architectures can be represented in principles. The existing work of search space follows two trends: the macro and micro search [18], [20]. The macro search provides an exhaustive-architecture search space to encourage the controller to explore the space and discover novel architectures, whereas the micro search inductively limits the search space to accelerate the search process. The choice and the size of the search space determine the difficulty of the optimization problem. Yet, even for the case of the search space based on a single cell, it is still a challenging problem due to the discrete search space and the curse of high dimensionality (since more complex models tend to perform better, resulting in more design choices) [17]. Thereby, incorporating prior knowledge about the typical properties of architectures well-suited for a task can significantly reduce the size of the search space and simplify the search process. Recent research [24] has validated the importance

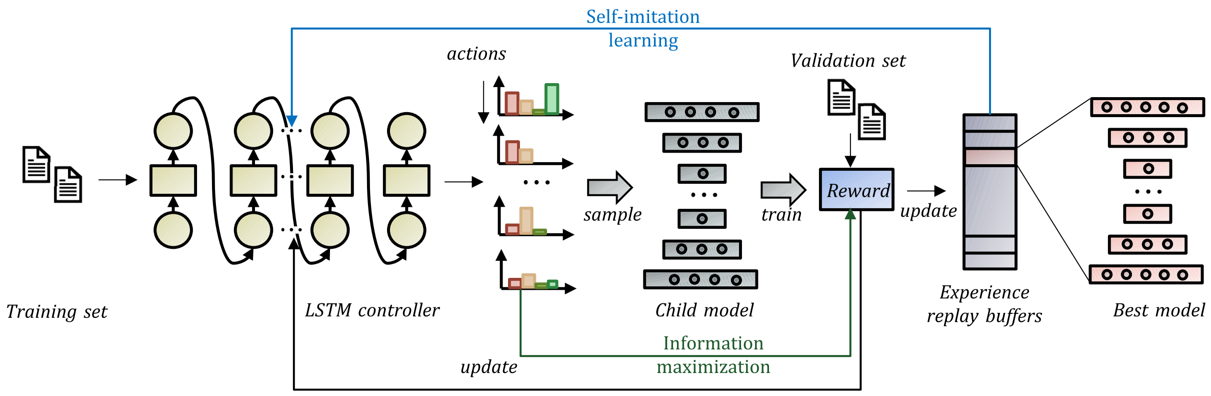

Fig. 1. Overview of AutoAD. With the predefined search space and the given dataset, we use an LSTM-based controller to generate actions a. Child models are sampled from actions a and evaluated with the reward r. Once the search process of one iteration is done, the controller samples M child models as candidate architectures and then picks the top K from them. The top K architectures' controller outputs will be fed as the input of the next iteration's controller. Parameters θ of the controller are updated with the reward r. r is also shaped by information maximization about the controller's internal belief of the model, which is designed to guide the search process (shown as the green line). Good past experiences evaluated by the reward function are stored in replay buffers for future self-imitations (shown as the blue line).

of the search space in the search process. With extensive experimental reproducibility studies, a task-tailored, carefully designed search space plays a more important role than the other search strategies. Recent works have proposed tailored search spaces with their applications, including image segmentation [25], adversarial training [26], and augmentation strategies [19]. To the best of our knowledge, our proposed AutoAD describes the first attempt to design the search space specifically customized to the anomaly detection task. AutoAD uses the micro search space to keep consistent with previous works. Yet, the contribution of AutoAD in search space is to design a hierarchical, general-purpose search space, including global settings for the whole model and local settings in each layer independently. Moreover, our proposed search space not only covers the hyperparameters as architecture configurations, such as the size of convolutional kernels and filters in each layer, but also incorporates the definition hypothesis and its corresponding objective function.

The search strategy focuses on how to explore the search space. Recent approaches include RL [16], [27], Bayesian optimization [28], and gradient-based methods [29]–[31]. Although these methods have improved upon human-designed architectures, directly borrowing existing NAS ideas from image classification to anomaly detection will not work. Due to the imbalanced data, the search process becomes more unstable in anomaly detection tasks [21]. As an internal mechanism in the traditional NAS, weight sharing also introduces the inductive bias in the search process, which intensifies the tendency [22]. Weight sharing [20] is proposed to transfer the well-trained weight before to a sampled architecture, to avoid training the offspring architecture from scratch. Recent research has validated that the weight sharing mechanism makes the architectures that have better initial performance with similar structures more likely to be sampled [22], which leads to misjudgments of the child model's performance. Our work builds upon RL-based method, which uses a recurrent neural network controller to choose blocks from its search space. Beyond that, we propose a curiosity-guided search strategy to stabilize the search process via encouraging the controller to seek out unexplored regions in the search space. Our search strategy formulates the search process as a classical exploration–exploitation tradeoff. On one hand, it is encouraged to find the optimal child model more efficiently; on the other hand, it avoids the premature convergence to a suboptimal region due to the inductive bias or insufficient search.

Previous work has explored RL in the context of anomaly detection. Oh and Iyengar [32] formulated sequential anomaly detection as an inverse RL problem, where the reward function is inferred from the behavior data. Pang *et al.* [33] proposed a deep RL approach to actively explore novel anomaly classes in semisupervised settings. Lai *et al.* [34] used deep RL to meta-learn an active learning strategy. However, these studies mainly focus on identifying anomalies with RL but do not consider neural architectures. In this work, we use RL to search the optimal neural architectures, which complements previous studies.

III. Preliminaries and Problem Formulation

In this section, we present the preliminaries and problem definition of our work.

A. Deep AutoEncoder-Based Anomaly Detection

Classical anomaly detection methods, such as local outlier factor [35] and one-class SVMs [36], suffer from bad computational scalability and the curse of dimensionality in high-dimensional and data-rich scenarios [14]. To tackle these problems, deep structured models have been proposed to process the features in a more efficient way. Among recent deep structured studies, Deep AutoEncoder is one of the most promising approaches for anomaly detection. The AutoEncoder learns a representation by minimizing the reconstruction error from normal samples [37]. Therefore, it can be used to extract the common factors of variation from normal samples and reconstruct them easily and vice versa. Besides directly employing the reconstruction error as the denoter, recent studies [11], [13], [14] demonstrate the effectiveness of collaborating Deep AutoEncoders with classical anomaly detection techniques, by introducing regularizers through plugging learned representations into classical anomaly definition hypotheses. Specifically, there are three typical anomaly assumptions: density, cluster, and centroid. The density-based approaches [13] estimate the relative density of each sample and declare instances that lie in a neighborhood with low density as anomalies. Under the clustering-based assumption, normal instances belong to an existing cluster in the dataset, while anomalies are not contained in any existing cluster [11]. The centroid-based approaches [14] rely on the assumption that normal data instances lie close to their closest cluster centroid, while anomalies are far away from them. In this work, we illustrate the proposed AutoAD by utilizing Deep AutoEncoder with a variety of regularizers as the basic anomaly detection algorithm. The framework of AutoAD could be easily extended to other deep-structured anomaly detection approaches.

B. Problem Statement

Different from the traditional NAS, which focuses on optimizing neural network architectures for supervised learning tasks, automated anomaly detection has the following two unique characteristics. First, the neural architecture in the autoencoder needs to be adaptive in the given dataset to achieve competitive performance. The hyperparameter configurations of neural architecture include the number of layers, the size of convolutional kernels and filters, and so on. Second, anomaly detection requires the designs of definition-hypothesis and corresponding objective function. Formally, we define the anomaly detection model and the unified optimization problem of automated anomaly detection as follows.

1) Anomaly Detection Model: The model of anomaly detection consists of three key components: the neural network architecture A of AutoEncoder, the definition-hypothesis H of anomaly assumption, and the loss function L. We represent the model as a triple (A, H, L).

2) Automated Anomaly Detection: Let the triple $(\mathcal{A}, \mathcal{H}, \mathcal{L})$ denote the search space of anomaly detection models, where $\mathcal{A}$ denotes the architecture subspace, $\mathcal{H}$ denotes the definition-hypothesis subspace, and $\mathcal{L}$ denotes the loss functions subspace. Given training set $\mathcal{D}_{\text{train}}$ and validation set $\mathcal{D}_{\text{valid}}$, we aim to find the optimal model $(A^\star, H^\star, L^\star)$ to minimize the objective function $\mathcal{J}$ as follows:

$$(A^\star, H^\star, L^\star) = \underset{A\in\mathcal{A}, H\in\mathcal{H}, L\in\mathcal{L}}{\arg\min} \mathcal{J}(A(\omega), H, L, \mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}) \quad (1)$$

where ω denotes the weights well trained on architecture A and $\mathcal{J}$ denotes the loss on $\mathcal{D}_{\text{valid}}$ using the model trained on the $\mathcal{D}_{\text{train}}$ with definition hypothesis H, and loss function L.

IV. Proposed Method

In this section, we propose an automated anomaly detection framework to find the optimal neural network model for a given dataset. A general search space is designed to include the neural architecture hyperparameters, definition hypothesis, and objective functions. To overcome the curse of local optimality under certain unstable search circumstances, we propose a curiosity-guided search strategy to improve search effectiveness. Moreover, we introduce an experience replay mechanism based on self-imitation learning to better exploit the past good experience and enhance the sample efficiency. An overview of AutoAD is given in Fig. 1.

TABLE I
Set of Four Representative Anomaly Detection Hypotheses (Density, Cluster, Central, and Reconstruction), Where $f(\cdot)$ and $g(\cdot)$ Denote Encoder and Decoder Functions, Respectively

H	Regularizer Equations
Den [13]	$-\log\left(\sum_{k=1}^{K}\hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(f(x_i;\omega)-\hat{\mu}_k)^T\hat{\mathbf{\Sigma}}_k^{-1}(f(x_i;\omega)-\hat{\mu}_k)\right)}{\sqrt{\left\|2\pi\hat{\mathbf{\Sigma}}_k\right\|}}\right)$
Clu [11]	$\sum_i\sum_j p_{ij}\log p_{ij}\left(\frac{\left(1+\left\|f(x_i;\omega)-\mu_j\right\|^2\right)^{-1}}{\sum_j\left(1+\left\|f(x_i;\omega)-\mu_j\right\|^2\right)^{-1}}\right)^{-1}$
Cen [14]	$R^2+\sum_{i=1}^{n}\max\left\{0,\left\|f\left(x_i;\omega\right)-c\right\|^2-R^2\right\}$
Rec [37]	$\frac{1}{n}\sum_{i=1}^{n}\left\|g\left(f\left(x_i;\omega\right)\right)-x_i\right\|_2^2$

A. Search Space Design

Because there is a lack of intrinsic search space for anomaly detection tasks, here, we design the search space for the Deep AutoEncoder-based algorithms, which is composed of global settings for the whole model and local settings in each layer independently. Formally, we have

$$\begin{aligned} A &= \left\{f^1(\cdot), \ldots, f^N(\cdot), g^1(\cdot), \ldots, g^N(\cdot)\right\} \\ f^i(x;\omega_i) &= \text{ACT}(\text{NORMA}(\text{POOL}(\text{CONV}(x)))) \\ g^i(x;\omega_i) &= \text{ACT}(\text{NORMA}(\text{UPPOOL}(\text{DECONV}(f(x))))) \\ \text{score} &= \text{DIST}(g(f(x;\omega)), x) + \text{DEFINEREG}(f(x;\omega)) \end{aligned} \quad (2)$$

where x denotes the set of instances as input data and ω denotes the trainable weight matrix. The architecture space A contains N encoder–decoder layers. $f(\cdot)$ and $g(\cdot)$ denote encoder and decoder functions, respectively. ACT$(\cdot)$ is the activation function set. NORMA denotes the normalization functions. POOL$(\cdot)$ and UPPOOL$(\cdot)$ are pooling methods. CONV$(\cdot)$ and DECONV$(\cdot)$ are convolution functions. As we discussed in Section II-A, the encoder–decoder-based anomaly score *score* contains two terms: a reconstruction distance and an anomaly regularizer. DIST$(\cdot)$ is the metric to measure the distance between the original inputs and the reconstruction results. DEFINEREG$(\cdot)$ acts as an regularizer to introduce the definition hypothesis from H. We revisit and extract the anomaly detection hypotheses and their mathematical formulas from state-of-the-art approaches, as shown in Table I. We decompose the search space defined in (2) into the following eight classes of actions.

Global Settings:

1) `Definition-hypothesis` determines the way to define the "anomalies," which acts as a regularization term in the objective functions. We consider density-,

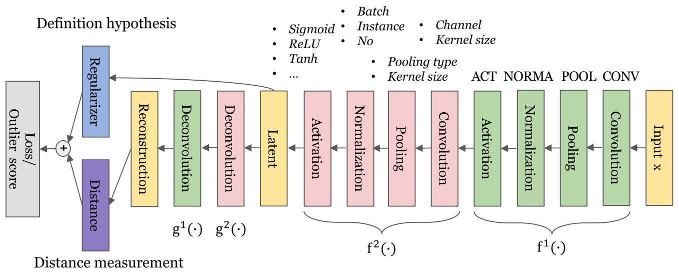

Fig. 2. Example of the search space in AutoAD with two layers, which is composed of global settings for the whole model (blue and purple parts) and local settings in each layer (red and green parts). All building blocks are wired together to form a direct acyclic graph.

cluster-, centroid-, and reconstruction-based assumptions, as shown in Table I.

2) `Distance measurement` stands for the matrix measuring the distance for the reconstruction purpose, including l_1, l_2, $l_{2,1}$ norms, and the structural similarity (SSIM).

Local Settings in Each Layer:

3) `Output channel` is the number of channels produced by the convolution operations in each layer, i.e., 3, 8, 16, 32, 64, 128, and 256.
4) `Convolution kernel` denotes the size of the kernel produced by the convolution operations in each layer, i.e., 1×1, 3×3, 5×5, and 7×7.
5) `Pooling type` denotes the type of pooling in each layer, including the max pooling and the average pooling.
6) `Pooling kernel` denotes the kernel size of pooling operations in each layer, i.e., 1×1, 3×3, 5×5, and 7×7.
7) `Normalization type` denotes the normalization type in each layer, including three options: batch normalization, instance normalization, and no normalization.
8) `Activation function` is a set of activation functions in each layer, including Sigmoid, Tanh, ReLU, Linear, Softplus, LeakyReLU, ReLU6, and ELU.

Thus, we use a $(6N + 2)$ element tuple to represent the model, where N is the number of layers in the encoder–decoder-wise structure. Our search space includes an exponential number of settings. Specifically, if the encoder–decoder cell has N layers and we allow action classes as above, it provides $4 \times 4 \times (7 \times 4 \times 2 \times 4 \times 3 \times 8)^N$ possible settings. Suppose that we have $N = 6$, the number of points in our search space is 3.9e + 23, which requires an efficient search strategy to find an optimal model out of the large search space. Fig. 2 shows an example of the proposed search space in AutoAD.

B. Curiosity-Guided Search

We now describe how to search the optimal model within the given search space. Inspired by the recent NAS work, the search strategy is considered as a meta-learning process. A controller is introduced to explore a given search space by training a child model to get an evaluation for guiding exploration [20]. The controller is implemented as a recurrent neural network. We use the controller to generate a sequence of actions for the child model. The whole process can be treated as an RL problem with an action $a_{1:T}$ and a reward function r. To find the optimal model, we ask the controller to maximize its expected reward r, which is the expected performance in the validation set of the child models.

There are two sets of learnable parameters: one set is the shared parameters of the child models, denoted by ω, and the other one is from the LSTM controller, denoted by θ. ω is optimized using stochastic gradient descent (SGD) with the gradient ∇_ω as

$$\nabla_\omega \mathbb{E}_{m\sim\pi(m;\theta)}[L(m;\omega)] \approx \nabla_\omega L(m,\omega) \tag{3}$$

where child model m is sampled from the controller's actions $\pi(m;\theta)$ and $L(m,\omega)$ is the loss function composed from the search space above, computed on a minibatch of training data. The gradient is estimated using the Monte Carlo method.

Since the reward signal r is nondifferentiable, to maximize the expected reward r, we fix ω and apply the REINFORCE rule [38] to update the controller's parameters θ as

$$\nabla_\theta \mathbb{E}_{P(a_{1:t};\theta)}[r\nabla_\theta \log P(a_t|a_{1:t-1};\theta)] \tag{4}$$

where r is computed as the performance on the validation set, rather than on the label-free training set. We define the reward r as the detection accuracy of the sampled child model. We also adopt different evaluation metrics, including AUROC, AUPR, and RPRO in Section V. An empirical approximation of (4) is

$$L = \frac{1}{n}\sum_{k=1}^{n}\sum_{t=1}^{T}(r_k - b)\nabla_\theta \log P(a_t|a_{1:t-1};\theta) \tag{5}$$

where n is the number of different child models that the controller samples in one batch and T is the number of tokens. b acts as a baseline function to reduce the estimate variance.

1) Curiosity-Driven Exploration: Despite being widely utilized due to search efficiency, weight sharing approaches are roughly built on empirical experiments instead of solid theoretical ground [22]. The unfair bias will make the controller misjudge the child-model performance: those who have better initial performance with similar child models are more likely to be sampled. In the meanwhile, due to the imbalanced label distribution in anomaly detection tasks, it is easy to make the controller fall into local optima.

To address these problems, AutoAD builds on the theory of curiosity-driven exploration [39], aiming to encourage the controller to seek out regions in the searching spaces that are relatively unexplored. It brings us a typical exploration–exploitation dilemma to guide the controller.

Bayesian RL [40], [41] offers us a formal guarantee as coherent probabilistic model for RL. It provides a principled framework to express the classic exploration–exploitation dilemma, by keeping an explicit representation of uncertainty and selecting actions that are optimal with respect to a version of the problem that incorporates this uncertainty [41]. Here, instead of a vanilla RNN, we use a Bayesian LSTM as the structure of the controller to guide the search. The controller's understanding of the search space is represented dynamically over the uncertainty of the parameters of the controller. Assuming a prior $p(\theta)$, it maintains a distribution prior over the controller's parameters through a distribution over θ. The controller models the actions via $p(a_t|a_{1:t};\theta)$, parameterized by θ. According to curiosity-driven exploration [39], the uncertainty about the controller's dynamics can be formalized as maximizing the information

$$I(a_t;\theta|a_{1:t-1}) = \mathbb{E}_{a_t\sim P(\cdot|a_{1:t-1})}\big[D_{\mathrm{KL}}[p(\theta|a_{1:t-1})\,||\,p(\theta)]\big] \quad (6)$$

where the Kullback–Leibler (KL) divergence can be interpreted as information gain, which denotes the mutual information between the controller's new belief over the model to the old one.

Thus, the information gain of the posterior dynamics distribution of the controller can be approximated as an intrinsic reward, which captures the controller's surprise in the form of a reward function. We can also use the REINFORCE rule to approximate planning for maximal mutual information by adding the intrinsic reward along with the external reward (accuracy on the validation set) as a new reward function. It can also be interpreted as a tradeoff between exploitation and exploration as

$$r_{\mathrm{new}}(a_t) = r(a_t) + \eta D_{\mathrm{KL}}[p(\theta|a_{1:t-1})\,||\,p(\theta)] \quad (7)$$

where $\eta \in \mathbb{R}_+$ is a hyperparameter controlling the urge to explore. However, it is generally intractable to calculate the posterior $p(\theta|a_{1:t-1})$ in (7).

2) Variational Bayes-by-Backprop: In this section, we propose a tractable solution to maximize the information gain objective presented in the previous subsection. To learn a probability distribution over network parameters θ, we propose a practical solution through a backpropagation compatible algorithm, bayes-by-backprop [40], [42].

In Bayesian models, latent variables are drawn from a prior density $p(\theta)$. During inference, the posterior distribution $p(\theta|x)$ is computed given a new action through Bayes' rule

$$p(a_t|a_{1:t-1}) = \frac{p(\theta)p(a_t|a_{1:t-1};\theta)}{p(a_t|a_{1:t-1})}. \quad (8)$$

The denominator can be computed through the integral

$$p(a_t|a_{1:t-1}) = \int_{\Theta} p(a_t|a_{1:t-1};\theta)p(\theta)d\theta. \quad (9)$$

As controllers are highly expressive parametrized LSTM networks, which are usually intractable as high dimensionality. Instead of calculating the posterior $p(\theta|\mathcal{D}_{\mathrm{train}})$ for a training dataset $\mathcal{D}_{\mathrm{train}}$, we approximate the posterior through an alternative probability densities over the latent variables θ as $q(\theta)$, by minimizing the KL divergence $D_{\mathrm{KL}}[q(\theta)\,||\,p(\theta)]$. We use $\mathcal{D}$ instead of $\mathcal{D}_{\mathrm{train}}$ in the following for brevity:

$$q(\theta) = \prod_{i=1}^{|\Phi|}\mathcal{N}(\theta_i|\mu_i;\sigma_i^2). \quad (10)$$

$q(\theta)$ is given by a Gaussian distribution, with μ Gaussian's mean vector and σ the covariance matrix.

Once minimized the KL divergence, $q(\cdot)$ would be the closest approximation to the true posterior. Let $\log p(\mathcal{D}|\theta)$ be the log likelihood of the model. Then, the network can be trained by minimizing the variational free energy as the expected lower bound

$$L[q(\theta),\mathcal{D}] = -\mathbb{E}_{\theta\sim q(\cdot)}\big[\log p(\mathcal{D}|\theta)\big] + D_{\mathrm{KL}}\big[q(\theta)\,||\,p(\theta)\big] \quad (11)$$

which can be approximated using N Monte Carlo samples from the variational posterior with N samples drawn according to $\theta \sim q(\cdot)$

$$L[q(\theta),\mathcal{D}] \approx \sum_{i=1}^{N} -\log p(\mathcal{D}|\theta^{(i)}) + \log q(\theta^{(i)}) - \log p(\theta^{(i)}). \quad (12)$$

3) Posterior Sharpening: We discuss how to derive a distribution $q(\theta|\mathcal{D})$ to improve the gradient estimates of the intractable likelihood function $p(\mathcal{D})$, which is related to variational autoencoders (VAEs) [43]. Inspired from strong empirical evidence and extensive work on VAEs, the "sharpened" posterior yields more stable optimization. We now use posterior sharpening strategy to benefit our search process.

The challenging part of modeling the variational posterior $q(\theta|\mathcal{D})$ is the high dimensional $\theta \in \mathbb{R}^d$, which makes the modeling unfeasible. Given the first term of the loss $-\log p(\mathcal{D}|\theta)$ is differentiable with respect to θ, we propose to parameterize q as a linear combination of θ and $-\log p(\mathcal{D}|\theta)$. Thus, we can define the hierarchical posterior of the form in (10)

$$q(\theta|\mathcal{D}) = \int q(\theta|\phi,\mathcal{D})q(\phi)d\phi \quad (13)$$

$$q(\theta|\phi,\mathcal{D}) = \mathcal{N}\big(\theta|\phi - \eta * -\nabla_\phi \log p(\mathcal{D}|\phi)\sigma^2 I\big) \quad (14)$$

with $\mu,\sigma \in \mathbb{R}^d$, and $q(\phi) = \mathcal{N}(\phi|\mu,\sigma)$ as the same setting in the standard variational inference method. $\eta \in \mathbb{R}^d$ can be treated as a per-parameter learning rate.

In the training phrase, we have $\theta \sim q(\theta|\mathcal{D})$ via ancestral sampling to optimize the loss as

$$\begin{aligned} L_{\mathrm{explore}} &= L(\mu,\sigma,\eta) \\ &= \mathbb{E}_{\mathcal{D}}[\mathbb{E}_{q(\phi)q(\theta|\phi,\mathcal{D})}[L(\mathcal{D},\theta,\phi|\mu,\sigma,\eta)]] \end{aligned} \quad (15)$$

with $L(\mathcal{D},\theta,\phi|\mu,\sigma,\eta)$ given by

$$\begin{aligned} L(\mathcal{D},\theta,\phi|\mu,\sigma,\eta) = &-\log p(\mathcal{D}|\theta) + \mathrm{KL}[q(\theta|\phi,\mathcal{D})\,||\,p(\theta|\phi)] \\ &+ \frac{1}{C}\mathrm{KL}[q(\phi)\,||\,p(\phi)] \end{aligned} \quad (16)$$

where the constant C is the number of truncated sequences.

Thus, we turn to deriving the training loss function for posterior sharpening. With the discussion above, we assume a hierarchical prior for the parameters such that $p(\mathcal{D}) = \int p(\mathcal{D}|\theta)p(\theta|\phi)p(\phi)d\theta d\phi$. Then, the expected lower bound

Algorithm 1 Automated Anomaly Detection

```
1: Input: Input datasets $\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}$, and search space $\mathcal{S}$.
2: Output: Optimal model with the best performance.
3: Initialize parameter $\theta, \omega$;
4: Initialize replay buffer $\mathcal{B} \leftarrow \emptyset$;
5: for each iteration do
6:   Perform curiosity-guided search via a LSTM controller
7:   for each step $t$ do
8:     Sample an action $a_t \sim \pi(a_{1:t-1}; \theta)$;
9:     $\omega \leftarrow \omega - \eta \nabla_\omega \mathbb{E}_{a_t \sim \pi(a_{1:t-1};\theta)}[L(a_{1:t-1}; \omega)]$; {Eq. 3}
10:    $\theta \leftarrow \theta - \eta L_{\text{explore}}(\mathcal{D}_{\text{train}}, \theta)$; {Eq. 15}
11:    $r_{\text{new}}(a_t) \leftarrow r(a_t) + \eta D_{\text{KL}}[p(\theta|a_{1:t-1}) \,||\, p(\theta)]$; {Eq. 7}
12:    Update controller via the new reward $r_{\text{new}}(a_t)$; {Eq. 4}
13:    if the performance of $a_t$ on $\mathcal{D}_{\text{val}}$ outperforms the actions stored in $\mathcal{B}$ then
14:      $\mathcal{B} \leftarrow \{\boldsymbol{a}, r\} \cup \mathcal{B}$; Update replay buffer;
15:    end if
16:  end for
17:  Perform self-imitation learning
18:  for each step t do
19:    Sample a mini-batch $\{\boldsymbol{a}, r\}$ from $\mathcal{B}$;
20:    $\omega \leftarrow \omega - \eta \nabla_\omega \mathbb{E}_{a_t \sim \pi(a_{1:t-1};\theta)}[L(a_{1:t-1}; \omega)]$; {Eq. 3}
21:    $\theta \leftarrow \theta - \eta L_{\text{replay}}(\mathcal{D}_{\text{valid}}, \theta)$; {Eq. 19}
22:  end for
23: end for
```

on $p(\mathcal{D})$ is defined as follows:

$$\begin{aligned}
\log p(\mathcal{D}) &= \log\left(\int p(\mathcal{D}|\theta) p(\theta|\phi) p(\phi) d\theta d\phi\right) \\
&\geqslant \mathbb{E}_{q(\phi,\theta|\mathcal{D})}\left[\log \frac{p(\mathcal{D}|\theta) p(\theta|\phi) p(\phi)}{q(\phi,\theta|\mathcal{D})}\right] \\
&= \mathbb{E}_{q(\theta|\phi,\mathcal{D})q(\phi)}\left[\log \frac{p(\mathcal{D}|\theta) p(\theta|\phi) p(\phi)}{q(\theta|\phi,\mathcal{D})q(\phi)}\right] \\
&= \mathbb{E}_{q(\phi)}\left[\mathbb{E}_{q(\theta|\phi,\mathcal{D})}\left[\log p(\mathcal{D}|\theta) + \log \frac{p(\theta|\phi)}{q(\theta|\phi,\mathcal{D})}\right] + \log \frac{p(\phi)}{q(\phi)}\right] \\
&= \mathbb{E}_{q(\phi)}\Big[\mathbb{E}_{q(\theta|\phi,\mathcal{D})}\big[\log p(\mathcal{D}|\theta) - \text{KL}[q(\theta|\phi,\mathcal{D}) \,||\, p(\theta|\phi)]\big] \\
&\qquad \times \text{KL}[q(\phi) \,||\, p(\phi)]\Big]. \qquad (17)
\end{aligned}$$

C. Experience Replay Via Self-Imitation Learning

The goal of this section is to exploit the past good experiences for the controller to benefit the search process by enhancing the sample efficiency, especially considering there are only a limited number of negative samples in anomaly detection tasks. In this article, we propose to store rewards from historical episodes into experience replay buffers [44]: $\mathcal{B} = (a_{1:t}, r_{\boldsymbol{a}})$, where ($a_{1:t}$ and $r_{\boldsymbol{a}}$) are the actions and the corresponding reward, respectively. To exploit good past experiences, we update the experience replay buffer for child models with better rewards and amplify the contribution from them to the gradient of θ. More specifically, we sample child models from the replay buffer using the clipped advantage $(r - b)_+$, where the rewards r in the past experiences outperform the current baseline b. Comparing with (5), the objective to update the controller's parameter θ through the replay buffer is

$$\nabla_\theta \mathbb{E}_{a_{1:t} \sim \pi_\theta, b \sim \mathcal{B}}[-\log \pi_\theta(a_t|a_{1:t-1})(r_{\boldsymbol{a}} - b)_+]. \qquad (18)$$

Then, an empirical approximation of (18) is

$$L_{\text{replay}} = \frac{1}{n} \sum_{k=1}^{n} \sum_{t=1}^{T} \nabla_\theta - \log \pi_\theta(a_t|a_{1:t-1})(r_{\boldsymbol{a}} - b)_+ \qquad (19)$$

where n is the number of different child models that the controller samples in one batch and T is the number of tokens.

Overall, the joint optimization process is specified in Algorithm 1, which consists of two phrases: the curiosity-guided search process and the self-imitation learning process. The optimal model with the best performance on the validation set is utilized for the anomaly detection tasks.

V. Experiments

In this section, we conduct extensive experiments to answer the following four research questions.

1) *Q1*: How effective is AutoAD compared with state-of-the-art handcrafted algorithms?
2) *Q2:* Whether or not the two key components of AutoAD, i.e., curiosity-guided search and experience replay, are effective in the search process?
3) *Q3:* Compared with random search, how effective is the proposed search strategy?
4) *Q4:* Does AutoAD have the potential to be applied in more complicated real-world applications?

A. Datasets and Tasks

We evaluate AutoAD on seven benchmark datasets for instance-level abnormal sample detection and pixel-level defect region segmentation tasks. We also conduct a case study on the CAT [45] dataset.

1) *MNIST [46]:* An image dataset consists of handwritten digits. It has a training set of 60 000 examples and a test set of 10 000 examples.
2) *Fashion-MNIST [47]:* A MNIST-like dataset contains fashion product with a training set of 60 000 examples and a test set of 10 000 examples. Each example is a 28 × 28 grayscale image associated with a label from ten classes.
3) *CIFAR-10 [48]:* A image dataset consists of 50 000 training images and 10 000 test images in ten different classes. Each example is a 32 × 32 three-channel image.
4) *Tiny-ImageNet [49]:* An image dataset consists of a subset of ImageNet images. It contains 10 000 test images from 200 different classes. We downsample each image to the size of 64 × 64.
5) *MVTec-AD [50]:* A benchmark dataset relates to industrial inspection in the application of anomaly detection.

TABLE II

PERFORMANCE COMPARISON ON INSTANCE-LEVEL ABNORMAL SAMPLE DETECTION. THE RESULTS FROM AUTOAD AND THE TWO BASELINES MSP [51] AND ODIN [52] ARE LISTED AS AUTOAD/MSP/ODIN. OOD: OUT-OF-DISTRIBUTION

(a)

OOD Dataset	AUROC	AUPR In	AUPR Out
Fashion-MNIST	**99.9**/97.9/97.9	**99.9**/99.7/99.6	**100**/90.5/91.0
notMNIST	**99.8**/97.2/98.2	**99.8**/97.5/98.4	**100**/97.4/98.0
CIFAR-10	**99.9**/**99.9**/99.7	91.3/90.3/**99.9**	99.2/**99.9**/97.6
LSUN	**99.9**/**99.9**/99.8	99.9/96.8/**100**	**99.9**/99.2/99.0
Tiny-ImageNet	**99.9**/99.4/99.6	99.8/99.6/**99.9**	**99.8**/96.8/97.5
Gaussian	**99.9**/99.7/99.9	**100**/99.8/**100**	**100**/99.7/**100**
Uniform	**100**/99.9/**100**	**100**/99.9/**100**	**100**/99.9/**100**

(b)

OOD Dataset	AUROC	AUPR In	AUPR Out
MNIST	**99.9**/92.9/72.9	**99.9**/82.8/91.6	**99.9**/94.2/46.1
notMNIST	**99.8**/96.9/80.2	**99.1**/81.6/94.2	**100**/99.4/57.7
CIFAR-10	**99.9**/88.2/96.6	**99.5**/80.6/99.3	**99.9**/97.2/80.4
LSUN	**99.5**/89.7/96.0	97.9/81.9/**99.2**	**99.9**/97.7/79.9
Tiny-ImageNet	**98.2**/87.7/95.5	90.7/80.4/**99.0**	**95.3**/97.1/82.5
Gaussian	**99.9**/97.2/89.6	**99.9**/82.24/98.0	**100**/99.5/48.2
Uniform	**99.9**/95.8/63.6	**99.9**/82.9/91.4	**99.9**/99.0/19.8

(c)

OOD Dataset	AUROC	AUPR In	AUPR Out
MNIST	**100**/98.4/99.9	**100**/99.4/**100**	**100**/89.4/99.4
Fashion-MNIST	**99.6**/98.2/99.4	98.1/96.1/**99.9**	**99.9**/98.8/97.3
notMNIST	**99.9**/96.8/98.1	**99.4**/99.0/99.2	**100**/89.4/90.2
LSUN	80.0/75.2/**85.6**	81.2/73.1/**83.5**	**85.8**/73.3/85.1
Tiny-ImageNet	**84.0**/72.6/81.6	**87.5**/73.5/76.9	**87.8**/80.6/84.8
Gaussian	**99.9**/86.3/98.8	**99.9**/90.5/99.1	**99.3**/77.0/97.9
Uniform	**99.9**/86.4/99.0	**99.9**/90.2/99.2	**99.9**/78.6/98.6

(d)

OOD Dataset	AUROC	AUPR In	AUPR Out
MNIST	**100**/99.8/94.8	**100**/98.2/98.9	**100**/98.2/79.9
Fashion-MNIST	**99.7**/70.4/73.8	**98.6**/88.4/92.6	**100**/85.5/39.5
notMNIST	**99.9**/80.6/82.7	**99.7**/90.4/95.2	**100**/85.8/56.0
CIFAR-10	**86.7**/82.9/58.0	**95.8**/75.3/85.7	**89.1**/75.3/28.2
LSUN	**88.6**/74.2/55.6	92.7/**99.5**/86.7	93.9/**99.5**/18.7
Gaussian	**99.9**/97.0/95.4	**100**/98.0/99.0	**99.9**/94.8/80.5
Uniform	**100**/96.0/87.5	**100**/97.4/96.8	**100**/99.3/62.8

It contains over 5000 high-resolution images divided into 15 categories in terms of different objects and textures. Each category comprises two parts: a training set of defect-free images, as well as a test set composed of defect-free images and the ones with various defects. We downsample each image to size 224 × 224.

6) *CAT [45]:* A cat dataset includes 10 000 cat images. We downsample each image to size 224 × 224.
7) *Gaussian Noise:* A synthetic Gaussian noise dataset consists of 1000 random 2-D images, where the value of each pixel is sampled from an independent and identically distributed (i.i.d) Gaussian distribution with mean 0.5 and unit variance. We further clip each pixel into the range [0, 1].
8) *Uniform Noise:* A synthetic uniform noise dataset consists of 1000 images, at which the value of each pixel is sampled from an i.i.d uniform distribution on [0, 1].

For the instance-level abnormal sample detection task, we use four benchmark datasets (i.e., MNIST [46], Fashion-MNIST [47], CIFAR-10 [48], and Tiny-ImageNet [49]) and two synthetic noise datasets (i.e., Gaussian and Uniform). Synthetic noise datasets consist of 1000 random 2-D images, where the value of each pixel is sampled from an i.i.d Gaussian distribution with mean 0.5 and unit variance. We further clip each pixel into the range [0, 1] or an i.i.d uniform distribution on [0, 1]. Different datasets contain different classes of images. We manually injected abnormal samples (also known as out-of-distribution samples), which consists of images randomly sampled from other datasets. For all the six datasets, we train an anomaly detection model on the training set, which only contains in-distribution samples, and use a validation set with out-of-distribution samples to guide the search, and another test set with out-of-distribution samples to evaluate the performance. The contamination ratio in the validation set and the test set is both 0.05. The train/validation/test split ratio is 6:2:2. Two state-of-the-art methods, including MSP [51] and ODIN [52] are used as baselines.

For the pixel-level defect region segmentation task, we use a real-world dataset MVTec-AD [50]. MVTec-AD contains high-resolution images with different objects and texture categories. Each category comprises a set of defect-free training images and a test set of images with various kinds of defects and images without defects. We train the model on the defect-free training set and split the whole test set into two halves for validation and testing. Three state-of-the-art methods, including AutoEncoder [53], AnoGAN [54], and feature dictionary [55], are used as baselines.

B. Baselines

We compare AutoAD with five state-of-the-art handcrafted algorithms and the random search strategy.

1) *MSP [51]:* The softmax probability distribution is used to detect the anomalies in tasks of computer vision, natural language processing, and automatic speech. The anomaly detection is performed based on the following assumption: the correctly classified examples have greater maximum softmax probabilities than those of erroneously classified and out-of-distribution examples.
2) *ODIN [52]:* The pretrained neural network is reused to detect the out-of-distribution images. ODIN separates the softmax probability distributions between in- and out-of-distribution instance, by using temperature scaling and adding small perturbations on the image data.
3) *AutoEncoder [53]:* The structure of convolutional AutoEncoders is applied for unsupervised defect segmentation on image data. More specifically, it utilizes the loss function based on SSIM and successfully examines interdependencies between local image regions to reveal the defective regions.
4) *AnoGAN [54]:* It is a deep convolutional generative adversarial network used to identify the anomalous image data. It learns a manifold of normal anatomical variability and maps images to a latent space to estimate the anomaly scores.

TABLE III
PERFORMANCE COMPARISON ON PIXEL-LEVEL DEFECT REGION SEGMENTATION. THE RESULTS OF RPRO AND AUROC ARE LISTED AS RPRO/AUROC. BASELINE RESULTS ARE DIRECTLY COLLECTED FROM THE ORIGINAL PAPER [50]

	Category	AutoAD	AutoEncoder	AnoGAN	Feature Dictionary
Textures	Carpet	**0.69 / 0.92**	0.38 / 0.59	0.34 / 0.54	0.20 / 0.72
	Grid	**0.89 / 0.94**	0.83 / 0.90	0.04 / 0.58	0.02 / 0.59
	Leather	**0.81 / 0.92**	0.67 / 0.75	0.34 / 0.64	0.74 / 0.87
	Tile	**0.26 / 0.94**	0.23 / 0.51	0.08 / 0.50	0.14 / 0.73
	Wood	**0.47 / 0.97**	0.29 / 0.73	0.14 / 0.62	**0.47** / 0.91
Objects	Bottle	**0.33 / 0.93**	0.22 / 0.86	0.05 / 0.86	0.07 / 0.78
	Cable	**0.17 / 0.87**	0.05 / 0.86	0.01 / 0.78	0.13 / 0.79
	Capsule	**0.16 / 0.95**	0.11 / 0.88	0.04 / 0.84	0.00 / 0.84
	Hazelnut	**0.46 / 0.97**	0.41 / 0.95	0.02 / 0.87	0.00 / 0.72
	Metal Nut	**0.30 / 0.88**	0.26 / 0.86	0.00 / 0.76	0.13 / 0.82
	Pill	**0.30 / 0.92**	0.25 / 0.85	0.17 / 0.87	0.00 / 0.68
	Screw	**0.34 / 0.96**	**0.34 / 0.96**	0.01 / 0.80	0.00 / 0.87
	Toothbrush	**0.60 / 0.90**	0.51 / 0.83	0.07 / **0.90**	0.00 / 0.77
	Transistor	**0.23 / 0.96**	0.22 / 0.86	0.08 / 0.80	0.03 / 0.66
	Zipper	**0.20 / 0.88**	0.13 / 0.77	0.01 / 0.78	0.00 / 0.76

5) *Feature Dictionary [55]:* It applies the convolutional neural networks and self-similarity to detect and localize anomalies in image data. More specifically, the abnormality degree of each image region is obtained by estimating its similarity to a dictionary of anomaly-free subregions in a training set.
6) *Random Search [24], [56]:* Instead of learning a policy to optimize the search progress, random search generates a neural architecture randomly at each step. It has been widely demonstrated that random search is a strong baseline hard to be surpassed in NAS.

C. Experimental Setup

We train the child models on the training set under the anomaly-free settings and update the controller on the validation set via the reward signal. The controller RNN is a two-layer LSTM with 50 hidden units on each layer. It is trained with the ADAM optimizer with a learning rate of $3.5\mathrm{e}^{-4}$. Weights are initialized uniformly in $[-0.1, 0.1]$. The search process is conducted for a total of 500 epochs. The size of the self-imitation buffer is 10. We use a Tanh constant of 2.5 and a sample temperature of 5 to the hidden output of the RNN controller. We train the child models by utilizing a batch size of 64 and a momentum of 0.9 with the ADAM optimizer. The learning rate starts at 0.1 and is dropped by a factor of 10 at 50% and 75% of the training progress, respectively.

D. Evaluation Metrics

We adopt the following metrics to measure the effectiveness.

1) AUROC [57] is the area under the receiver operating characteristic curve, which is a threshold-independent metric [57]. The receiver operating characteristic (ROC) curve depicts the relationship between TPR and FPR. The AUROC can be interpreted as the probability that a positive example is assigned a higher detection score than a negative example [58].
2) AUPR [59] is the area under the precision–recall (PR) curve, which is another threshold-independent metric [59], [60]. The PR curve is a graph showing the precision = TP/(TP +FP) and recall = TP/(TP + FN) against each other. The metrics AUPR-In and AUPR-Out denote the area under the PR curve, where positive samples and negative samples are specified as positives, respectively.
3) RPRO [50] stands for the relative per-region overlap. It denotes the pixel-wise overlap rate of the segmentations with the ground truth.

E. Results

1) Performance on Out-of-Distribution Sample Detection: To answer the research question Q1, we compare AutoAD with the state-of-the-art handcrafted algorithms for the instance-level abnomral sample detection task using metrics AUROC, AUPR-In, and AUPR-Out. Considering the automated search framework of AutoAD, we represent its performance by the best model found during the search process. In these experiments, we follow the setting in [52] and [62]. Each model is trained on individual dataset $\mathcal{D}_{\text{in}}$, which is taken from MNIST, Fashion-MNIST, CIFAR-10, and Tiny-ImageNet. At test time, the test images from $\mathcal{D}_{\text{in}}$ dataset can be viewed as the in-distribution (positive) samples. We sample out-of-distribution (negative) images from another real-world or synthetic noise dataset, after downsampling/upsampling and reshaping their sizes as the same as $\mathcal{D}_{\text{in}}$.

As can be seen from Table II, in most of the test cases, the models discovered by AutoAD consistently outperform the handcrafted out-of-distribution detection methods with pretrained models (ODIN) [52] and without pretrained models (MSP) [51]. It indicates that AutoAD could achieve higher performance in accuracy, precision, and recall simultaneously, with a more precise detection rate and fewer nuisance alarms.

2) Performance on Defect Sample Detection: To further answer question Q1, we test AutoAD on the pixel-level defect region segmentation task. The results from Table III show that AutoAD consistently outperforms the baseline methods by a large margin in terms of AUROC and RPRO. The higher AUROC demonstrates that the model found by AutoAD precisely detects images with defect sections out of the positive samples. The results also show that AutoAD has a better performance in RPRO. This indicates that the search process of AutoAD helps the model to locate and represent the anomaly regions in negative images.

As a step-by-step concrete example, given input datasets from different benchmarks, we first define the global search space, including definition hypothesis and distance measurement, and then define a three-layer encoder–decoder in the local space, and each layer contains output channel, convolution kernel, pooling type, pooling kernel, normalization type, and activation function. Table IV shows the best architectures discovered by AutoAD on both instance-level abnormal sample detection and pixel-level defect region segmentation tasks.

TABLE IV

ARCHITECTURES DISCOVERED BY AUTOAD FOR MNIST, FASHION-MNIST, CIFAR-10, TINY-IMAGENET, AND MVTEC-AD

Actions	MNIST	Fashion-MNIST	CIFAR-10	Tiny-ImageNet	MVTec-AD
Definition hypothesis	reconstruction	cluster	centroid	reconstruction	reconstruction
Distance measurement	l_1	$l_{2,1}$	$l_{2,1}$	$l_{2,1}$	$l_{2,1}+$ SSIM
Layer-1					
Output-channel	32	16	16	16	32
Convolution kernel	5×5	1×1	1×1	5×5	1×1
Pooling type	mean	average	average	average	average
Pooling kernel	1×1	1×1	1×1	1×1	3×3
Normalization type	no	no	no	no	no
Activation function	ReLU	Sigmoid	Sigmoid	Linear	LeakyReLU
Layer-2					
Output-channel	8	32	32	32	32
Convolution kernel	3×3	5×5	5×5	3×3	5×5
Pooling type	average	mean	mean	mean	mean
Pooling kernel	1×1	7×7	7×7	7×7	5×5
Normalization type	no	no	no	no	batch
Activation function	ELU	ReLU6	Sigmoid	Softplus	Tanh
Layer-3					
Output-channel	8	16	16	16	16
Convolution kernel	7×7	1×1	1×1	7×7	1×1
Pooling type	average	average	average	average	mean
Pooling kernel	5×5	1×1	1×1	1×1	5×5
Normalization type	no	instance	instance	instance	instance
Activation function	ReLU6	LeakyReLU	Sigmoid	LeakyReLU	Tanh

TABLE V

ABLATIONS AND PARAMETER ANALYSIS ON FASHION-MNIST (IN-DISTRIBUTION: NORMAL DATA) AND CIFAR-10 (OUT-OF-DISTRIBUTION: ANOMALIES). WE REPORT THE PERFORMANCE OF AUTOAD UNDER DIFFERENT SEARCH STRATEGIES AND HYPERPARAMETER SETTINGS, WITH 20_{TH}, 100_{TH}, AND 200_{TH} ITERATIONS

(a)

	AUROC_{20}	AUROC_{100}	AUROC_{200}	mean_{200}	std_{200}
η=0	85.27	96.23	96.51	90.01	0.093
η=0.01	85.46	96.54	98.50	91.20	0.097
η=0.1	85.46	96.93	98.41	91.84	0.131

(b)

	AUROC_{20}	AUROC_{100}	AUROC_{200}
no buffer	85.43	96.57	98.00
buffer size=5	88.05	97.12	98.04
buffer size=10	87.44	97.70	98.12

3) Effectiveness of Curiosity-Guided Search: To qualitatively evaluate the effectiveness of the curiosity-guided search for research question Q2, we perform ablation and hyperparameter analysis on the Fashion-MNIST dataset with samples from CIFAR-10 as anomalies. Specifically, we control the hyperparameter η in (7) for illustration. Note that the mathematical expression of $\eta = 0$ represents that there is no exploration. From Table V(a), we can observe that the following conditions hold. First, the absence of exploration would negatively impact the final performance and the AUROC after 200 epochs could drop 1.9%. Second, the curiosity-guided explorations help the controller find the optimal model faster. The better performance could be achieved in the 20th, 100th epochs when the controller has a larger weight η on explorations. This indicates that curiosity-guided search is a promising way for exploring more unseen spaces. Third, comparing AUROC between $\eta = 0.01$ and $\eta = 0.1$, we observe that there is no significant increase in the final performance after 200 epochs. This indicates that a higher rate of explorations cannot always guarantee a higher performance. Finally, if we treat the performance of the searched result as Gaussian distributions, the standard deviations of the AutoAD's performance keep increasing when η increases. This validates that the curiosity-guided search strategy increases the opportunity for the controller to generate child models in a more diverse way.

4) Effectiveness of Experience Replay: To further answer the question Q2, we evaluate the effectiveness of the experience replay buffers, by altering the size of the replay buffers $\mathcal{B}$ in (19). The corresponding results are reported in Table V(b). The results indicate that the increase of the buffer size could enhance model performance after 200 epochs. We also observe that the size of the buffer is sensitive to the final performance, as better performance would be achieved in the 20th, 100th epoch with a larger buffer size. This indicates that self-imitation learning-based experience replay is useful in the search process. A larger buffer size brings benefits to exploit past good experiences.

5) Comparison Against Traditional NAS: Instead of applying the policy gradient-based search strategy, one can use random search to find the best model. Although this baseline seems simple, it is often hard to surpass [24], [56]. We compare AutoAD with random search to answer the research question Q3. The quality of the search strategy can be quantified by the following three metrics: 1) the average performance of the top-5 models found so far; 2) the mean performance of the searched models in every 20 epochs; and 3) the standard deviation of the model performance in every 20 epochs. From Fig. 3, we can observe that the following conditions hold. First, our proposed search strategy is more efficient to find the well-performed models during the search process. As shown in the first row of Fig. 3, the performance of the top-5 models found by AutoAD consistently outperforms the random search. The results also show that not only the best model of our search strategy is better than that of random search, but also the improvement of average top models is much more significant. This indicates that AutoAD explores better models faster than the random search. Second, there is a clear increasing tendency in the mean performance of AutoAD, which cannot be observed in random search. It indicates that our search controller can gradually find better strategies from the past search experiences along the learning process, while the random search's controller has a relatively low chance to find a good child model. Third, compared with the random search, there is a clear dropping of standard deviation along the search process. It verifies that our search strategy provides a more stable search process.

F. Case Study

To answer the research question Q4, we provide further analysis for the pixel-level defect region segmentation task, to get some insights about how to further improve the detection performance in more complicated real-world settings. To make the anomaly sections and the rest sections more distinguishable in the latent space, we use the reconstruction error to learn intrinsic representation for positive samples to extract common patterns. Yet, it is hard to directly apply AutoAD into more complicated, real-world settings. Due to the pure data-driven strategy, the reconstruction-based denoters might be misled by background noises, other objects, or irrelevance

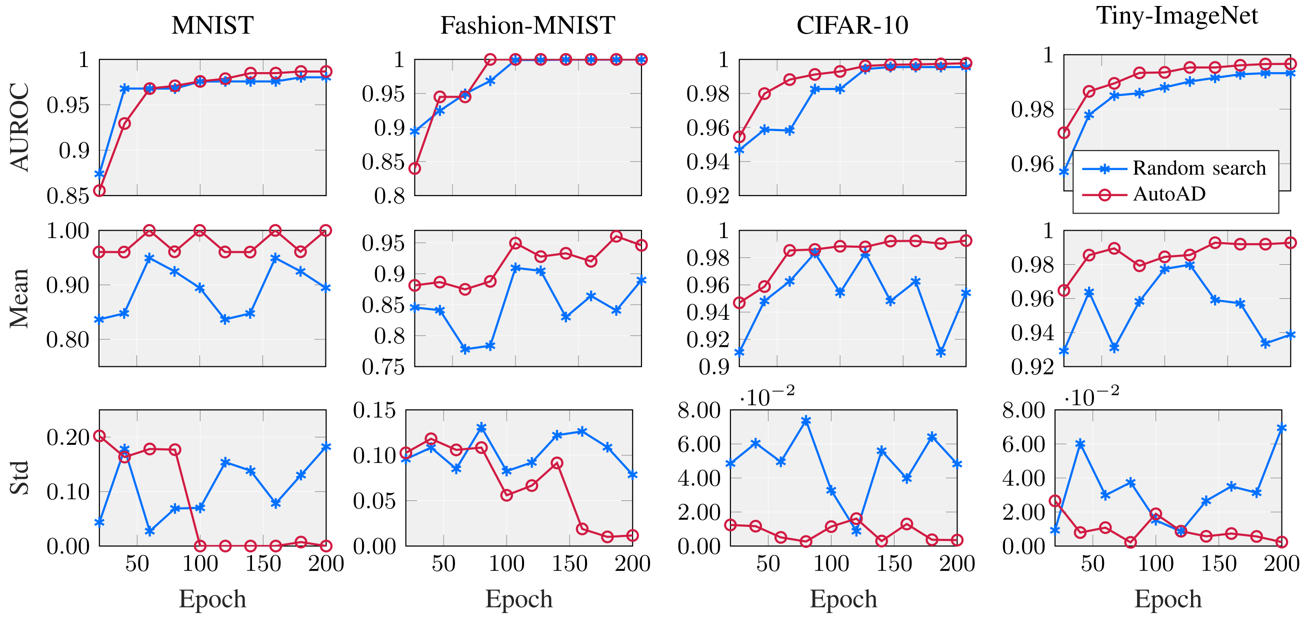

Fig. 3. Performance comparison with random search. Top row: progression of average performance in top-5 models for different search methods, i.e., AutoAD (red lines with circles) and random search (blue lines with asterisks). Middle and bottom rows: mean and standard deviation of model performances in every 20 epochs along the search progress.

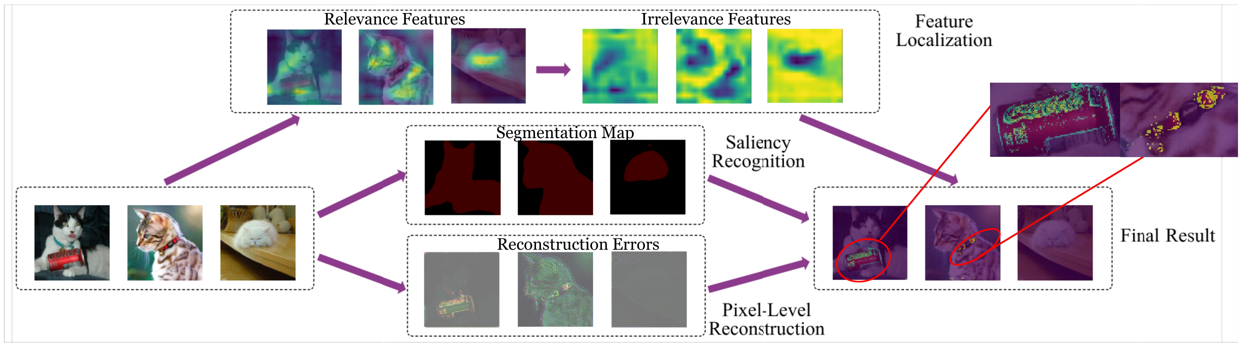

Fig. 4. Case study of anomaly segmentation. The pixel-level reconstruction is regularized by conducting pixel-level collaboratively with the results of feature localization and saliency recognition. AutoAD accurately identifies the anomaly regions (cola bin and collar) of the target object (cat) in the first two images and treats the third target object (cat) as abnormal-region-free.

features. We hereby introduce two strategies into AutoAD for regularization without increasing the model complexity.

1) Saliency Refinement Via Target Object Recognition: We introduce a mask map s into the reconstruction-based denoters from pretrained models. It is used for localizing and identifying target salient object, in order to eliminate the negative effect caused by background noises and other objects in the same image. To concisely localize and identify the salient object, the key idea is to extract dense features for semantic segmentation. In our experiment, we introduce s from DeepLabV3 [62], which is pretrained on PASCAL VOC 2012 [63].

2) Feature Augmentation via Gradient-Based Localization: In order to amplify the contribution of the irrelevance features from the salient object, we introduce the feature augmentation map to re-weight the reconstruction result. We also introduce a coarse localization map to highlight the irrelevance regions in the image from an interpretability perspective. Feature importance is reflected as gradients signal via backpropagation. The key idea is to use the gradient information flowing into the last convolutional layer of the CNN to assign importance values to each neuron for a particular decision of interest. Here, we follow the interpretation method from Grad-CAM [64]–[67], which is designed to highlight important features and pretrained on VGG-16 [68]. The feature augmentation map is defined as opposite to the Grad-CAM: $(1/mn)\sum_m \sum_n(1 - (\partial y_i/\partial A_{mn}))$.

After the two steps above, we reweight the reconstructions

$$\alpha_i = \|g(f(x_i;\mathcal{W})) - x_i\|_2^2 \odot \frac{1}{mn}\sum_m \sum_n \left(1 - \frac{\partial y_i}{\partial A_{mn}}\right) \odot s_i \tag{20}$$

where $x_i \in R^{m\times n}$ is a training sample, y_i is its target object class, $f(\cdot)$, $g(\cdot)$ denotes encoder–decoder structures produced by AutoAD, and A denotes the feature map activation of a latent layer. We use a real-world dataset CAT [45] for illustration. We find the optimal model via AutoAD and get the pixel-level reconstruction map. Then, we further refine the map via saliency recognition and feature localization strategies. As can be seen from Fig. 4, AutoAD achieves better visualization results after applying the reweighting tricks discussed above. We also observe that the model can successfully identify the anomaly regions (cola bins and collars) within the salient objects (kitties). Meanwhile, it reduces the effect caused by the background noises and irrelevant objects.

VI. Conclusion

In this article, we investigated a novel and challenging problem of automated deep model search for anomaly detection. Different from the existing NAS methods that focus on discovering effective deep architectures for supervised learning tasks, we proposed AutoAD, an automated unsupervised anomaly detection framework, which aims to find an optimal neural network model within a predefined search space for a given dataset. AutoAD builds on the theory of curiosity-driven exploration and self-imitation learning. It overcomes the curse of local optimality, the unfair bias, and inefficient sample exploitation problems in the traditional search methods. We evaluated the proposed framework using extensive experiments on eight benchmark datasets for instance-level abnormal sample detection and pixel-level defect region segmentation. The experimental results demonstrated the effectiveness of our approach.

References

[1] K. Padmanabhan, Z. Chen, S. Lakshminarasimhan, S. S. Ramaswamy, and B. T. Richardson, "Graph-based anomaly detection," in *Practical Graph Mining With R*. 2013.

[2] Y. Lin *et al.*, "Collaborative alert ranking for anomaly detection," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2018, pp. 1987–1995.

[3] Y. Li, N. Liu, J. Li, M. Du, and X. Hu, "Deep structured cross-modal anomaly detection," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2019, pp. 1–8.

[4] V. Chandola *et al.*, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, no. 3, pp. 1–58, 2009.

[5] W. Cheng, K. Zhang, H. Chen, G. Jiang, Z. Chen, and W. Wang, "Ranking causal anomalies via temporal and dynamical analysis on vanishing correlations," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 805–814.

[6] S. Wang *et al.*, "Attentional heterogeneous graph neural network: Application to program reidentification," in *Proc. SDM*, 2019, pp. 693–701.

[7] J. Lin, E. Keogh, A. Fu, and H. Van Herle, "Approximations to magic: Finding unusual medical time series," in *Proc. 18th IEEE Symp. Comput.-Based Med. Syst. (CBMS)*, Jun. 2005, pp. 329–334.

[8] Y. Li, X. Huang, J. Li, M. Du, and N. Zou, "SpecAE: Spectral autoencoder for anomaly detection in attributed networks," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, 2019, pp. 2233–2236.

[9] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 732–740.

[10] N. Liu, Q. Tan, Y. Li, H. Yang, J. Zhou, and X. Hu, "Is a single vector enough? Exploring node polysemy for network embedding," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2019, pp. 932–940.

[11] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *Proc. NIPS*, 2017, pp. 373–382.

[12] S. Wang *et al.*, "Heterogeneous graph matching networks for unknown malware detection," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019.

[13] B. Zong *et al.*, "Deep autoencoding Gaussian mixture model for unsupervised anomaly detection," in *Proc. ICLR*, 2018.

[14] L. Ruff *et al.*, "Deep one-class classification," in *Proc. ICML*, 2018, pp. 4393–4402.

[15] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," 2018, *arXiv:1812.04606*. [Online]. Available: http://arxiv.org/abs/1812.04606

[16] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. ICLR*, 2016.

[17] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 1997–2017, 2019.

[18] Y. Li, D. Zha, P. Venugopal, N. Zou, and X. Hu, "PyODDS: An end-to-end outlier detection system with automated machine learning," in *Proc. Companion Proc. Web Conf.*, Apr. 2020, pp. 153–157.

[19] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning augmentation strategies from data," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 113–123.

[20] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. ICML*, 2018, pp. 4095–4104.

[21] G. Swirszcz, W. Marian Czarnecki, and R. Pascanu, "Local minima in training of neural networks," 2016, *arXiv:1611.06310*. [Online]. Available: http://arxiv.org/abs/1611.06310

[22] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," 2019, *arXiv:1907.01845*. [Online]. Available: http://arxiv.org/abs/1907.01845

[23] G. Bender *et al.*, "Can weight sharing outperform random architecture search? An investigation with TuNAS," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 14323–14332.

[24] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," 2019, *arXiv:1902.07638*. [Online]. Available: http://arxiv.org/abs/1902.07638

[25] C. Liu *et al.*, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[26] X. Gong, S. Chang, Y. Jiang, and Z. Wang, "AutoGAN: Neural architecture search for generative adversarial networks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 3224–3234.

[27] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. ICLR*, 2018.

[28] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1946–1956.

[29] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. ICLR*, 2019.

[30] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *Proc. ICLR*, 2018.

[31] C. Liu *et al.*, "Progressive neural architecture search," in *Proc. ECCV*, 2018, pp. 19–34.

[32] M.-H. Oh and G. Iyengar, "Sequential anomaly detection using inverse reinforcement learning," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2019, pp. 1480–1490.

[33] G. Pang, A. van den Hengel, C. Shen, and L. Cao, "Toward deep supervised anomaly detection: Reinforcement learning from partially labeled anomaly data," 2020, *arXiv:2009.06847*. [Online]. Available: http://arxiv.org/abs/2009.06847

[34] K.-H. Lai, D. Zha, Y. Li, and X. Hu, "Dual policy distillation," 2020, *arXiv:2006.04061*. [Online]. Available: http://arxiv.org/abs/2006.04061

[35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[36] Y. Chen, X. Sean Zhou, and T. S. Huang, "One-class SVM for learning in image retrieval," in *Proc. Int. Conf. Image Process.*, 2001, pp. 34–37.

[37] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 665–674.

[38] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, May 1992.

[39] Y. Sun, F. Gomez, and J. Schmidhuber, "Planning to be surprised: Optimal Bayesian exploration in dynamic environments," in *Proc. AGI*, 2011, pp. 41–51.

[40] M. Fortunato, C. Blundell, and O. Vinyals, "Bayesian recurrent neural networks," in *Proc. ICLR*, 2017.

[41] M. Ghavamzadeh *et al.*, "Bayesian reinforcement learning: A survey," *Found. Trends Mach. Learn.*, to be published.

[42] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, “Weight uncertainty in neural network,” in *Proc. ICML*, 2015, pp. 1613–1622.

[43] D. P. Kingma and M. Welling, “Auto-encoding variational Bayes,” 2013, *arXiv:1312.6114*. [Online]. Available: http://arxiv.org/abs/1312.6114

[44] J. Oh *et al.*, “Self-imitation learning,” in *Proc. ICML*, 2018, pp. 3878–3887.

[45] W. Zhang, J. Sun, and X. Tang. (2009). *Cat Dataset*. [Online]. Available: https://archive.org/details/CAT_DATASET

[46] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[47] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms,” 2017, *arXiv:1708.07747*. [Online]. Available: http://arxiv.org/abs/1708.07747

[48] A. Krizhevsky *et al.*, “Learning multiple layers of features from tiny images,” Citeseer, Tech. Rep., 2009.

[49] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[50] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, “MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection,” in *Proc. CVPR*, Jun. 2019, pp. 9592–9600.

[51] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in *Proc. ICLR*, 2017.

[52] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” in *Proc. ICLR*, 2017.

[53] P. Bergmann, S. Löwe, M. Fauser, D. Sattlegger, and C. Steger, “Improving unsupervised defect segmentation by applying structural similarity to autoencoders,” 2018, *arXiv:1807.02011*. [Online]. Available: http://arxiv.org/abs/1807.02011

[54] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, “Unsupervised anomaly detection with generative adversarial networks to guide marker discovery,” in *Proc. IPMI*, 2017, pp. 146–157.

[55] P. Napoletano, F. Piccoli, and R. Schettini, “Anomaly detection in nanofibrous materials by CNN-based self-similarity,” *Sensors*, vol. 18, no. 2, p. 209, Jan. 2018.

[56] J. Bergstra and Y. Bengio, “Random search for hyper-parameter optimization,” *J. Mach. Learn. Res.*, vol. 13, no. 2, pp. 281–305, 2012.

[57] J. Davis and M. Goadrich, “The relationship between Precision-Recall and ROC curves,” in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*, 2006, pp. 233–240.

[58] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognit. Lett.*, vol. 27, no. 8, pp. 861–874, Jun. 2006.

[59] C. D. Manning, C. D. Manning, and H. Schütze, *Foundations of Statistical Natural Language Processing*. 1999.

[60] T. Saito and M. Rehmsmeier, “The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets,” *PLoS ONE*, vol. 10, no. 3, Mar. 2015, Art. no. e0118432.

[61] T. DeVries and G. W. Taylor, “Learning confidence for out-of-distribution detection in neural networks,” in *Proc. ICLR*, 2018.

[62] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation,” 2017, *arXiv:1706.05587*. [Online]. Available: http://arxiv.org/abs/1706.05587

[63] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL visual object classes challenge 2012 (VOC2012) results,” Tech. Rep., 2012.

[64] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual explanations from deep networks via gradient-based localization,” in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 618–626.

[65] M. Du, S. Pentyala, Y. Li, and X. Hu, “Towards generalizable deepfake detection with locality-aware autoencoder,” in *Proc. 29th ACM Int. Conf. Inf. Knowl. Manage.*, Oct. 2020, pp. 325–334.

[66] Y. Li *et al.*, “Learning disentangled representations for time series,” 2021, *arXiv:2105.08179*. [Online]. Available: http://arxiv.org/abs/2105.08179

[67] F. Yang, Z. Zhang, H. Wang, Y. Li, and X. Hu, “XDeep: An interpretation tool for deep neural networks,” 2019, *arXiv:1911.01005*. [Online]. Available: http://arxiv.org/abs/1911.01005

[68] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *Proc. ICLR*, 2015.

[69] D. Zha, K.-H. Lai, M. Wan, and X. Hu, “Meta-AAD: Active anomaly detection with deep reinforcement learning,” 2020, *arXiv:2009.07415*. [Online]. Available: http://arxiv.org/abs/2009.07415

[70] L. Li and A. Talwalkar, *What is Neural Architecture Search?*. Sebastopol, CA, USA: O’Reilly Media, 2018.

[71] L. Li and A. Talwalkar, “Random search and reproducibility for neural architecture search,” in *Proc. UAI*, 2020.