2021 IEEE 37th International Conference on Data Engineering (ICDE) | 978-1-7281-9184-3/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICDE51399.2021.00210

AutoOD: Neural Architecture Search for Outlier Detection

Yuening Li[1], Zhengzhang Chen[2,*], Daochen Zha[1], Kaixiong Zhou[1], Haifeng Jin[1], Haifeng Chen[2], and Xia Hu[1]
[1]*Department of Computer Science and Engineering, Texas A&M University*, USA
{yueningl,daochen.zha,zkxiong,jin,xiahu}@tamu.edu
[2]*NEC Laboratories America*, Princeton, USA
{zchen,haifeng}@nec-labs.com
[*]*Corresponding author.*

***Abstract*—Outlier 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 task with complex data, the process of building an effective deep learning based system for outlier detection still highly relies on human expertise and laboring trials. Moreover, 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 outlier detection due to the lack of intrinsic search space and low sample efficiency. To bridge the gap, in this paper, we propose AutoOD, an automated outlier detection framework, which aims to search for an optimal neural network model within a predefined search space. Specifically, we 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 AutoOD achieves the best performance, comparing with existing handcrafted models and traditional search methods.**

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]. Centered around this goal, outlier detection plays a very important role in various real-world applications [2]–[8], such as fraud detection, cyber security, medical diagnosis, and social network analysis.

Driven by the success of deep learning, there has been a surge of interests [6], [9]–[11] in adopting deep neural networks for outlier 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 outlier detection methods as the scale of data increases [12]. 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 sub-optimal performance.

Neural Architecture Search (NAS) [13]–[15] is one promising means for automating the design of neural networks, where reinforcement learning and evolution have been used to discover optimal model architectures from data [16]. 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 outperforms hand-crafted 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 [13], [16], the unsupervised setting and the naturally imbalanced data have introduced new challenges in designing an automated outlier detection framework. (1) **Lack of search space**. It is non-trivial to determine the search space for an outlier detection task. In particular, since there is no class label information in the training data of an outlier detection task, objective functions play an important role to differentiate between normal and anomalous behaviors. Thus, in contrast to the supervised learning tasks, we often need to find a suitable definition of the outlier 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 [17]. 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 [18]. However, these clustering-based techniques will be less effective if the anomalies form significant clusters among themselves [1]. 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 outlier detection needs to cover not only the architecture configurations, but also the outlier definitions with corresponding objective functions. (2) **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

978-1-7281-9184-3/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00210

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.

real-world outlier detection tasks, 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 paper, we propose **AutoOD**, an automated outlier 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 outlier detection. It covers architecture settings, outlier definitions, and corresponding loss functions. Given the predefined search space, we further propose an experience replay mechanism based on self-imitation learning to enhance sample efficiency. It can benefit the search process through exploiting good experience in the historical episodes. To evaluate the performance of **AutoOD**, we perform an extensive set of experiments on benchmark datasets comparing with existing handcrafted models and traditional search methods.

The contributions of this paper are summarized as follows:

- We identify a novel and challenging problem (*i.e.*, automated outlier detection) and propose a generic framework **AutoOD**. To the best of our knowledge, AutoOD describes the first attempt to incorporate AutoML with an outlier detection task, and one of the first to extend AutoML concepts into applications from data mining fields.
- We carefully design a **search space** specifically tailored to the automated outlier detection problem, covering architecture settings, outlier definitions, and the corresponding objective functions.
- We introduce an *experience replay* mechanism based on the self-imitation learning to improve the sample efficiency.

II. Preliminaries and Problem Formulation

A. Deep AutoEncoder Based Outlier Detection

Classical outlier detection methods, such as Local Outlier Factor and One-Class SVMs, suffer from bad computational scalability and the curse of dimensionality in high-dimensional and data-rich scenarios [11]. 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 outlier detection. The AutoEncoder learns a representation by minimizing the reconstruction error from normal samples [19], [20]. 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], [17], [18] demonstrate the effectiveness of collaborating Deep AutoEncoders with classical outlier detection techniques, by introducing regularizers through plugging learned representations into classical outlier definition hypotheses. Specifically, there are three typical outlier assumptions: density, cluster, and centroid. The density based approaches [17] 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 [18]. The centroid based approaches [11] 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 **AutoOD** by utilizing Deep AutoEncoder with a variety of regularizers as the basic outlier detection algorithm. The framework of **AutoOD** could be easily extended to other deep structured approaches.

B. Problem Statement

Different from the traditional Neural Architecture Search, which focuses on optimizing neural network architectures for supervised learning tasks, automated outlier 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, *etc.*; Second, the outlier detection requires the designs of definition-hypothesis and corresponding objective function. Formally, we define the outlier detection model and the unified optimization problem of automated outlier detection as follows.

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

Automated Outlier Detection: Let the triple $(\mathcal{A}, \mathcal{H}, \mathcal{L})$ denote the search space of outlier 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:

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

where ω denotes the weights well trained on architecture A. $\mathcal{J}$ 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.

III. Proposed Method

In this section, we propose an automated outlier 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 an experience replay mechanism based on self-imitation learning to better exploit the past good experience and enhance the sample efficiency. An overview of **AutoOD** is given in Fig. 1.

A. Search Space Design

Because there is a lack of intrinsic search space for outlier detection tasks, here we design the search space for the Deep AutoEncoder based algorithms, which is composed of global

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.

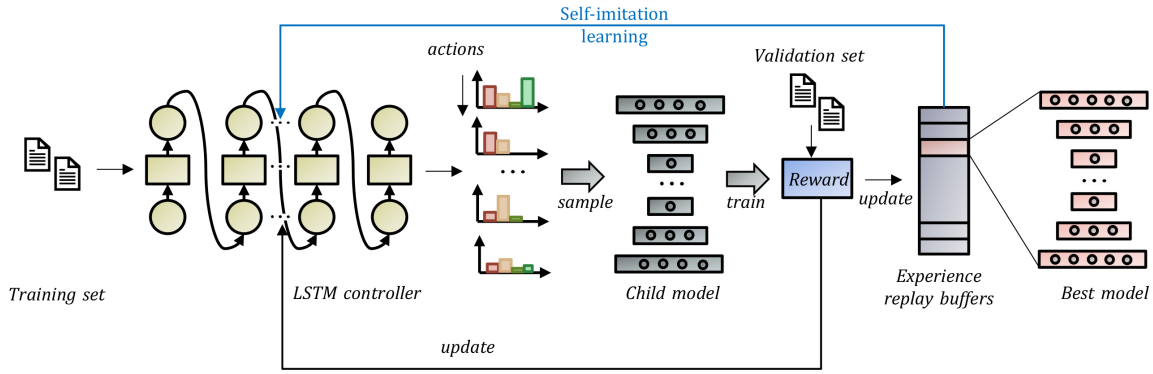

Fig. 1: An overview of **AutoOD**. With the pre-defined search space and the given dataset, we use an LSTM based controller to generate actions a. Child models are sampled from actions a and evaluated with the reward r. Once the search process of one iteration is done, the controller samples M child models as candidate architectures and then picks the top K from them. The top K architectures' controller outputs will fed as the input of the next iteration's controller. Parameters θ of the controller are updated with the reward r. Good past experiences evaluated by the reward function are stored in replay buffers for future self-imitations (shown as the blue line).

Definitions H	Regularizer Equations
Density [17]	$-\log\left(\sum_{k=1}^{K}\hat{\phi}_k \frac{\exp\left(-\frac{1}{2}(f(x_i;\omega)-\hat{\mu}_k)^T\hat{\mathbf{\Sigma}}_k^{-1}(f(x_i;\omega)-\hat{\mu}_k)\right)}{\sqrt{\lvert 2\pi\hat{\mathbf{\Sigma}}_k\rvert}}\right)$
Cluster [18]	$\sum_i\sum_j p_{ij}\log p_{ij}\left(\frac{\left(1+\lVert f(x_i;\omega)-\mu_j\rVert^2\right)^{-1}}{\sum_j\left(1+\lVert f(x_i;\omega)-\mu_j\rVert^2\right)^{-1}}\right)^{-1}$
Centroid [11]	$R^2+\sum_{i=1}^{n}\max\left\{0,\lVert f(x_i;\omega)-c\rVert^2-R^2\right\}$
Reconstruction [19]	$\frac{1}{n}\sum_{i=1}^{n}\lVert g(f(x_i;\omega))-x_i\rVert_2^2$

TABLE I: The set of four representative outlier detection hypotheses, where $f(\cdot)$ and $g(\cdot)$ denote encoder and decoder functions, respectively.

settings for the whole model, and local settings in each layer independently. Formally, we have:

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

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 the Section II (A), the encoder-decoder based outlier score *score* contains two terms: a reconstruction distance and an outlier 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 outlier detection hypotheses and their mathematical formulas from state-of-the-art approaches as shown in the Table I. We decompose the search space defined in Eq. 2 into the following 8 classes of actions:

Global Settings:

- `Definition-hypothesis` determines the way to define the "anomalies", which acts as a regularization term in the objective functions. We consider density-based, cluster-based, centroid-based, and reconstruction-based assumptions, as shown in Table I.
- `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:

- `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.
- `Convolution kernel` denotes the size of the kernel produced by the convolution operations in each layer, *i.e.*, $1\times1, 3\times3, 5\times5$, and 7×7.
- `Pooling type` denotes the type of pooling in each layer, including the max pooling and the average pooling.
- `Pooling kernel` denotes the kernel size of pooling operations in each layer, *i.e.*, $1\times1, 3\times3, 5\times5$, and 7×7.s
- `Normalization type` denotes the normalization type in each layer, including three options: batch normalization, instance normalization, and no normalization.
- `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 $4\times4\times(7\times4\times2\times4\times3\times8)^N$ possible settings. Suppose we have a $N=6$, the number of points in our search space is $3.9e{+}23$, which requires an efficient search strategy to find an

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.

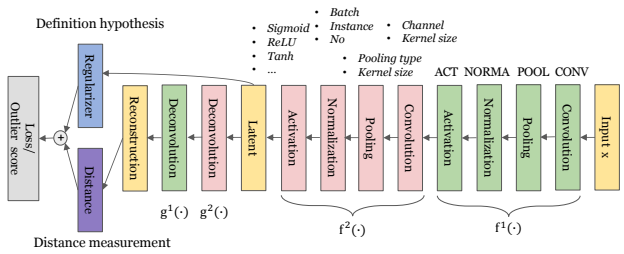

Fig. 2: An example of the search space in AutoOD with two layers, which is composed of global settings for the whole model (blue and purple parts), and local settings in each layer (red and green parts), respectively. All building blocks are wired together to form a direct acyclic graph.

optimal model out of the large search space. Fig. 2 illustrates an example of the proposed search space in AutoOD.

B. Search Process

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 [16]. 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 a reinforcement learning problem with an action $a_{1:T}$, and a reward function r. To find the optimal model, we ask our 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 of them 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 $\pi(m;\theta)$, $L(m,\omega)$ 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 non-differentiable, to maximize the expected reward r, we fix ω and apply the REINFORCE rule [21] 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 the experiment section. An empirical approximation of the Eq. 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.

C. Experience Replay via Self-Imitation Learning

The goal of this subsection 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 outlier detection tasks. In this paper, we propose to store rewards from historical episodes into experience replay buffers [22]: $\mathcal{B} = (a_{1:t}, r_{\boldsymbol{a}})$, where ($a_{1:t}$ and $r_{\boldsymbol{a}}$) are the actions and the corresponding reward. 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 the Eq. 5, the objective to update the controller's parameter θ through the replay buffer is:

$$\nabla_\theta \mathbb{E}_{a_{1:t}\sim\pi_\theta, b\sim\mathcal{B}}[-\log \pi_\theta\left(a_t|a_{1:t-1}\right)(r_{\boldsymbol{a}} - b)_+]. \tag{6}$$

Then, an empirical approximation of the Eq. 6 is:

$$L_{\text{replay}} = \frac{1}{n}\sum_{k=1}^{n}\sum_{t=1}^{T}\nabla_\theta - \log \pi_\theta\left(a_t|a_{1:t-1}\right)(r_{\boldsymbol{a}} - b)_+, \tag{7}$$

where n is the number of different child models that the controller samples in one batch and T is the number of tokens. The optimal model with the best performance on the validation set is utilized for the outlier detection tasks.

IV. Experiments

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

- **Q1**: How effective is AutoOD compared with **state-of-the-art handcrafted algorithms**?
- **Q2**: Whether or not the **experience replay** effective in the search process?
- **Q3**: Compared with **random search**, how effective is the proposed search strategy?

1) Datasets and Baselines: We evaluate **AutoOD** on four benchmark datasets, MNIST [25], Fashion-MNIST [26], CIFAR-10 [27], Tiny-ImageNet [28], and two synthetic noise datasets (*i.e.*, Gaussian and Uniform). The Gaussian dataset consists of $1,000$ random 2D images, where the value of each pixel is sampled from an i.i.d Gaussian distribution with mean 0.5 and unit variance. And the Uniform dataset consists of $1,000$ images, at which the value of each pixel is sampled from an i.i.d uniform distribution on $[0,1]$. We manually inject abnormal samples (a.k.a. out-of-distribution samples), which consists of images randomly sampled from other datasets. For all 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

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.

(a) In-distribution dataset: MNIST

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
CIFAR-10	**99.9**/**99.9**/99.7	91.3/90.3/**99.9**	99.2/**99.9**/97.6
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) In-distribution dataset: Fashion-MNIST

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
CIFAR-10	**99.9**/88.2/96.6	**99.5**/80.6/99.3	**99.9**/97.2/80.4
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) In-distribution dataset: CIFAR-10

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
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) In-distribution dataset: Tiny-ImageNet

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
CIFAR-10	**86.7**/82.9/58.0	**95.8**/75.3/85.7	**89.1**/75.3/28.2
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

TABLE II: Performance comparison on instance-level abnormal sample detection. The results from AutoOD and the two baselines MSP [23] and ODIN [24] are listed as AutoOD/MSP/ODIN. OOD: Out-of-distribution.

in the validation set and the test set are both 0.05. The train/validation/test split ratio is $6:2:2$. Two state-of-the-art methods MSP [23] and ODIN [24] are used as baselines.

2) Performance on Out-of-distribution Sample Detection: To answer the research question **Q1**, we compare AutoOD 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 AutoOD, we represent its performance by the best model found during the search process. In these experiments, we follow the setting in [23]: Each model is trained on individual dataset $\mathcal{D}_{in}$, which is taken from MNIST, Fashion-MNIST, CIFAR-10, and Tiny-ImageNet, respectively. At test time, the test images from $\mathcal{D}_{in}$ 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 down-sampling/up-sampling and reshaping their sizes as the same as $\mathcal{D}_{in}$.

As can be seen from Table II, in most of the test cases, the models discovered by AutoOD consistently outperform the handcrafted out-of-distribution detection methods with pre-trained models (ODIN [24]) and without pre-trained models (MSP [23]). It indicates that AutoOD could achieve higher performance in accuracy, precision, and recall simultaneously, with a more precise detection rate and fewer nuisance alarms.

3) 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 Eq. 7. Corresponding results are reported in Table III (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. Larger buffer size brings benefits to exploit past good experiences.

4) 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 [29]. We compare AutoOD 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, (3) the standard deviation of the model performance in every 20 epochs. From Fig. 3, we can observe that: Firstly, 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 AutoOD consistently outperform 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 AutoOD explores better models faster than the random search. Secondly, there is a clear increasing tendency in the mean performance of AutoOD, which can not 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. Thirdly, 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.

	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

TABLE III: Parameter analysis on Fashion-MNIST (in-distribution) and CIFAR-10 (out-of-distribution).

V. Conclusions

In this paper, we investigated a novel and challenging problem of automated deep model search for outlier detection. Different from the existing Neural Architecture Search methods that focus on discovering effective deep architectures for supervised learning tasks, we proposed **AutoOD**, an automated

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.

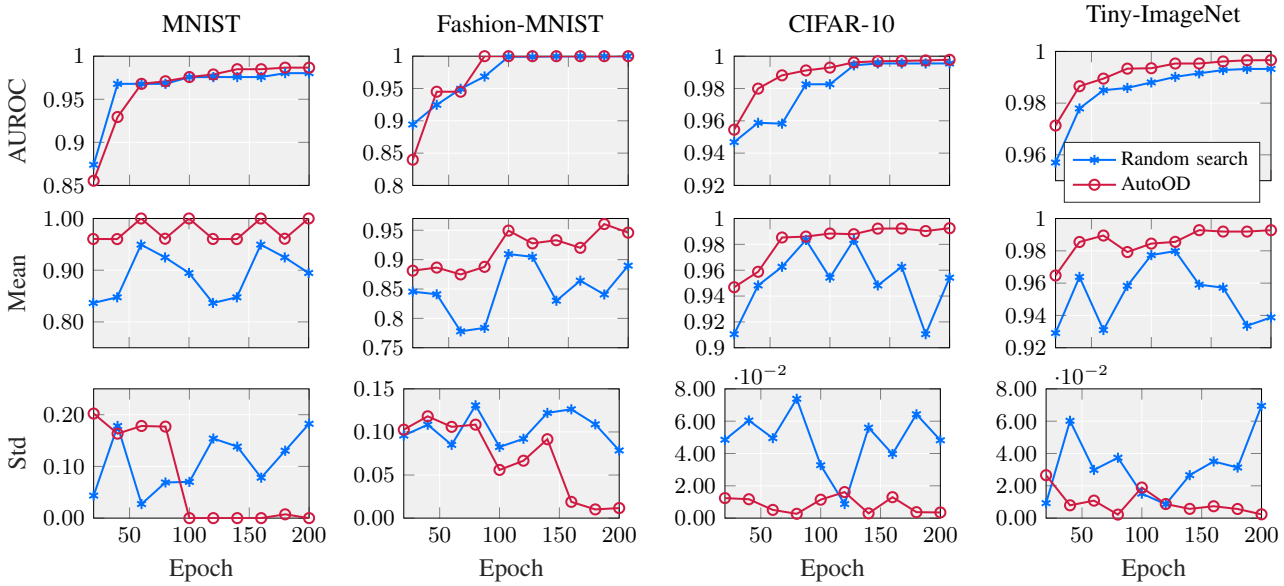

Fig. 3: Performance comparison with random search. (Top row) The progression of average performance in top-5 models for different search methods, *i.e.*, AutoOD (red lines with circles), and random search (blue lines with asterisks). (Middle row and bottom row) The mean and standard deviation of model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

unsupervised outlier detection framework, which aims to find an optimal neural network model within a predefined search space for a given dataset. **AutoOD** builds on the theory of 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 benchmark datasets for instance-level abnormal sample detection. The experimental results demonstrated the effectiveness of our approach.

Acknowledgement

The work is done during an internship at NEC, and is in part, supported by NSF (IIS-1750074, CNS-1816497, IIS-1718840). The views and conclusions in this paper are those of the authors and should not be interpreted as representing any funding agencies.

References

[1] V. Chandola and et al., "Anomaly detection: A survey," *CSUR*, 2009.
[2] B. Dong, Z. Chen, H. W. Wang, L.-A. Tang, K. Zhang, Y. Lin, Z. Li, and H. Chen, "Efficient discovery of abnormal event sequences in enterprise security systems," in *CIKM*, 2017.
[3] S. Wang, Z. Chen, D. Li, Z. Li, L.-A. Tang, J. Ni, J. Rhee, H. Chen, and S. P. Yu, "Attentional heterogeneous graph neural network: Application to program reidentification," in *SDM*, 2019.
[4] Z. Chen, W. Hendrix, and N. F. Samatova, "Community-based anomaly detection in evolutionary networks," *J. Intell. Inf. Syst.*, 2012.
[5] Y. Li, X. Huang, J. Li, M. Du, and N. Zou, "Specae: Spectral autoencoder for anomaly detection in attributed networks," in *CIKM*, 2019.
[6] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang, "Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection," *arXiv preprint arXiv:2008.13361*, 2020.
[7] 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 *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.
[8] S. Wang, Z. Chen, X. Yu, D. Li, J. Ni, L.-A. Tang, J. Gui, Z. Li, H. Chen, and S. P. Yu, "Heterogeneous graph matching networks for unknown malware detection," in *IJCAI*, 2019.
[9] Y. Li, N. Liu, J. Li, M. Du, and X. Hu, "Deep structured cross-modal anomaly detection," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.
[10] X. Huang, Q. Song, Y. Li, and X. Hu, "Graph recurrent networks with attributed random walks," in *KDD*, 2019.
[11] L. Ruff and et al., "Deep one-class classification," in *ICML*, 2018.
[12] D. Hendrycks, M. Mazeika, and T. Dietterich, "Deep anomaly detection with outlier exposure," *arXiv preprint arXiv:1812.04606*, 2018.
[13] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *ICLR*, 2016.
[14] Y. Li, D. Zha, N. Zou, and X. Hu, "Pyodds: An end-to-end outlier detection system," *arXiv preprint arXiv:1910.02575*, 2019.
[15] Y. Li, D. Zha, P. Venugopal, N. Zou, and X. Hu, "Pyodds: An end-to-end outlier detection system with automated machine learning," in *Companion Proceedings of the Web Conference 2020*, 2020.
[16] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *ICML*, 2018.
[17] B. Zong and et al., "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," *ICLR*, 2018.
[18] X. Guo, X. Liu, E. Zhu, and J. Yin, "Deep clustering with convolutional autoencoders," in *NIPS*, 2017.
[19] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *KDD*, pp. 665–674, 2017.
[20] M. Du, S. Pentyala, Y. Li, and X. Hu, "Towards generalizable forgery detection with locality-aware autoencoder," *arXiv preprint arXiv:1909.05999*, 2019.
[21] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, 1992.
[22] J. Oh and et al., "Self-imitation learning," *ICML*, 2018.
[23] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *ICLR*, 2017.
[24] S. Liang, Y. Li, and R. Srikant, "Enhancing the reliability of out-of-distribution image detection in neural networks," *ICLR*, 2017.
[25] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.
[26] H. Xiao and et al., "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv:1708.07747*, 2017.
[27] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," tech. rep., Citeseer, 2009.
[28] J. Deng and et al., "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009.
[29] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," *arXiv preprint arXiv:1902.07638*, 2019.

Authorized licensed use limited to: NEC Labs. Downloaded on September 15,2024 at 21:46:21 UTC from IEEE Xplore. Restrictions apply.