CMG: A Class-Mixed Generation Approach to Out-of-Distribution Detection

Mengyu Wang[1]*, Yijia Shao[1]*, Haowei Lin[1], Wenpeng Hu[1], and Bing Liu[2](✉)

[1] Wangxuan Institute of Computer Technology, Peking University
{wangmengyu, shaoyj, linhaowei, wenpeng.hu}@pku.edu.cn
[2] University of Illinois at Chicago
liub@uic.edu

Abstract. Recently, contrastive learning with data and class augmentations has been shown to produce markedly better results for out-of-distribution (OOD) detection than previous approaches. However, a major shortcoming of this approach is that it is extremely slow due to the significant increase in data size and in the number of classes and the quadratic pairwise similarity computation. This paper shows that this heavy machinery is unnecessary. A novel approach, called CMG (***C**lass-**M**ixed **G**eneration*), is proposed, which generates pseudo-OOD data by mixing class embeddings as abnormal conditions to CVAE (conditional variational Auto-Encoder) and then uses the data to fine-tune a classifier built using the given in-distribution (IND) data. To our surprise, the obvious approach of using the IND data and the pseudo-OOD data to directly train an OOD model is a very poor choice. The fine-tuning based approach turns out to be markedly better. Empirical evaluation shows that CMG not only produces new state-of-the-art results but also is much more efficient than contrastive learning, at least 10 times faster.[3]

Keywords: Out-of-distribution detection · Data generation.

1 Introduction

Out-of-distribution (OOD) detection aims to detect novel data that are very different from the training distribution or *in-distribution* (IND). It has a wide range of applications, e.g., autonomous driving [40] and medical diagnosis [5]. So far, many approaches have been proposed to solve this problem, from distance-based methods [2, 3, 12, 20], to generative models [60, 38, 41, 36] and self-supervised learning methods [4, 11, 17, 19]. Recently, *contrastive learning* with *data augmentation* has produced state-of-the-art (SOTA) OOD detection results [53, 45].

However, data augmentation-based contrastive learning has a major drawback. It is extremely inefficient and resource-hungry due to a large amount of augmented data and quadratic pairwise similarity computation during training. For example, CSI [53] creates 8 augmented instances for each original image. Furthermore, every

* Equal contribution
[3] Code is available at: https://github.com/shaoyijia/CMG

2 samples in the augmented batch is treated as a pair to calculate contrastive loss. The performance is also poor if the batch size is small, but a large batch size needs a huge amount of memory and a very long time to train. It is thus unsuitable for edge devices that do not have the required resources. In Sec. 4.4, we will see that even for a moderately large dataset, CSI has difficulty to run.

In this paper, we propose a novel and yet simple approach, called CMG (***C**lass-**M**ixed **G**eneration*), that is both highly effective and efficient, to solve the problem. CMG consists of two stages. The first stage trains a pseudo-OOD data generator. The second stage uses the generated pseudo-OOD data and the IND training data to fine-tune (using an *energy function*) a classifier already trained with the IND data. We discuss the first stage first.

OOD detection is basically a classification problem but there is no OOD data to use in training. This paper proposes to generate pseudo-OOD data by working in the latent space of a Conditional Variational Auto-Encoder (CVAE). The key novelty is that the pseudo-OOD data generation is done by manipulating the CVAE's conditional information using *class-mixed embeddings*. CVAE generates instances from the training distribution on the basis of latent representations consisting of the conditional information and variables sampled from a prior distribution of CVAE, normally the Gaussian distribution. If the latent space features or representations are created with some *abnormal conditions*, the CVAE will generate "*bad*" instances but such instances can serve as pseudo-OOD samples. Our abnormal conditions are produced by mixing embeddings of class labels in the IND data, which ensures the generated pseudo-OOD data to be similar but also different from any existing IND data.

With the pseudo-OOD data generated, the conventional approach is to use the IND data and the pseudo-OOD data to build a classifier for OOD detection. However, to our surprise, *this approach is a poor choice*. We will discuss the reason in Sec. 3.3 and confirm it with experimental results in Sec. 4.5. We discovered that if we build a classifier first using the IND data and then fine-tune only the final classification layer using both the IND and pseudo-OOD data, the results improve dramatically. This is another novelty of this work. The paper further proposes to use an *energy function* to fine-tune the final classification layer, which produces even better results.

Our contributions can be summarized as follows:

(1) We propose a novel method using CVAE to generate pseudo-OOD samples by providing *abnormal conditions*, which are mixed embeddings of different class labels. To our knowledge, this has not been done before.[4]

(2) We discovered that the obvious and conventional approach of using the IND and pseudo-OOD data to train a classifier in one stage performed very poorly. Our two-stage framework with fine-tuning performs dramatically better. Again, to our knowledge, this has not been reported before.

[4] By no means do we claim that this CVAE method is the best. Clearly, other generators may be combined with the proposed class-mixed embedding approach too. It is also known that CVAE does not generate high resolution images, but our experiments show that low resolution images already work well.

(3) Equally importantly, since the classifier in CMG is not specified, CMG can be applied to existing OOD detection models to improve them too.

Extensive experiments show that the proposed CMG approach produces new SOTA results and is also much more efficient than contrastive learning for OOD detection, requiring only one-tenth of the execution time.

2 Related Work

Early ideas for solving the OOD detection problem focused on modifying softmax scores to obtain calibrated confidences for OOD detection [3, 13]. Many other score functions have also been proposed, e.g., likelihood ratio [46], input complexity [50] and typicality [37]. A recent work utilizes Gram matrices to characterize activity patterns and identify OOD samples [48].

Methods that use anomalous data to improve detection [16, 35] are more closely related to our work. Generative models have been used to anticipate novel data distributions. In some of these methods, generated data are treated as OOD samples to optimize the decision boundary and calibrate the confidence [60, 54]. In some other methods, generative models such as auto-encoders [59, 43] and generative adversarial networks (GAN) are used to reconstruct the training data [8, 42]. During GAN training, low quality samples acquired by the generator are used as OOD data [44]. Their reconstruction loss can also help detect OOD samples. There are also works using given OOD data to train a model [33]. It has been shown recently that using pre-trained representations and few-shot outliers can improve the results [9]. Self-supervised techniques have been applied to OOD detection too. They focus on acquiring rich representations through training with some pre-defined tasks [10, 25]. Self-supervised models show outstanding performance [25, 4]. CSI [53] is a representative method (see more below), which uses contrastive learning and data augmentation to produce SOTA results. However, it is extremely slow and memory demanding. Our CMG method for generating pseudo-OOD data is much more efficient. Some researchers also tried to improve contrastive learning based methods [49] and proposed distance-based methods [34]. However, our experiments show that CSI outperforms them. Our CMG method is a generative approach. But unlike existing methods that use perturbations to anticipate OOD data, CMG uses synthetic conditions and CVAE to obtain effective and diverse pseudo-OOD data.

Auto-Encoder (AE) is a family of unsupervised neural networks [47, 1]. A basic AE consists of an encoder and a decoder. The encoder encodes the input data into a low-dimensional hidden representation and the decoder transforms the representation back to the reconstructed input data [55, 7, 18]. Variational auto-encoder is a special kind of AE [23]. It encodes the input as a given probability distribution (usually Gaussian) and the decoder reconstructs data instances according to variables sampled from that distribution. CVAE is an extension of VAE [24]. It encodes the label or conditional information into the latent representation so that a CVAE can generate new samples from specified class labels. CVAE makes it easy to control the generating process, i.e., to generate samples with features of specified classes. We make use of this property of CVAE to generate high quality pseudo-OOD data.

3 Proposed CMG Method

OOD detection is commonly formulated as a classification problem without OOD data/class available in training. To effectively train an OOD detection model, an intuitive idea is to generate pseudo-OOD data and use them together with the IND data to jointly build a OOD detection model. We take this approach. We propose a method using Conditional Variational Auto-Encoder (CVAE) to generate pseudo-OOD data and a new fine-tuning framework based on an energy function to leverage the generated pseudo-OOD data to produce a highly effective and efficient OOD detection model.

3.1 Conditional Variational Auto-encoder

Conditional Variational Auto-Encoder (CVAE) is derived from Variational auto-encoder (VAE). We first introduce VAE which is a conditional directed graphical model consisting of three main parts, an encoder $q_\phi(\cdot)$ with parameters ϕ, a decoder $p_\theta(\cdot)$ with parameters θ and a loss function $\mathcal{L}(\mathbf{x};\theta,\phi)$, where $\mathbf{x}$ represents an input sample. The loss function is as follows:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})] + KL(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) \tag{1}$$

where $q_\phi(\mathbf{z}|\mathbf{x})$ is a proposal distribution to approximate the prior distribution $p_\theta(\mathbf{z})$, $p_\theta(\mathbf{x}|\mathbf{z})$ is the likelihood of the input $\mathbf{x}$ with a given latent representation $\mathbf{z}$, and $KL(\cdot)$ is the function to calculate Kullback-Leibler divergence. $q_\phi(\mathbf{z}|\mathbf{x})$ is the encoder and $p_\theta(\mathbf{x}|\mathbf{z})$ is the decoder. In Eq. (1), the expected negative log-likelihood term encourages the decoder to learn to reconstruct the data with samples from the latent distribution. The KL-divergence term forces the latent distribution to conform to a specific prior distribution such as the Gaussian distribution, which we use. After training, a VAE can generate data using the decoder $p_\theta(\mathbf{x}|\mathbf{z})$ with a set of latent variables $\mathbf{z}$ sampled from the prior distribution $p_\theta(\mathbf{z})$. Commonly, the prior distribution is the centered isotropic multivariate Gaussian $p_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z};\mathbf{0},\mathbf{I})$.

However, VAE does not consider the class label information which is available in classification datasets and thus has difficulty generating data of a particular class. Conditional variational Auto-Encoder (CVAE) was introduced to extend VAE to address this problem. It improves the generative process by adding a conditional input information into latent variables so that a CVAE can generate samples with some specific characteristics or from certain classes. We use c to denote the prior class information. The loss function for CVAE is as follows:

$$\mathcal{L}(\mathbf{x};\theta,\phi) = -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z},c)] + KL(q_\phi(\mathbf{z}|\mathbf{x},c)||p_\theta(\mathbf{z}|c)) \tag{2}$$

One CVAE implementation uses a one-hot vector to represent a class label y_c, and a weight matrix is multiplied to it to turn the one-hot vector to a class embedding $\mathbf{y}_c$. Then a variable $\mathbf{z}$, generated from the prior distribution $p_\theta(\mathbf{z})$, is concatenated with $\mathbf{y}_c$ to construct the whole latent variable. Finally, the generated instance $p_\theta(\mathbf{x}|\mathbf{z}, c)$ of class c is produced. We can formulate the process as:

$$p_\theta(\mathbf{x}|\mathbf{z}, c) = p_\theta(\mathbf{x}|[\mathbf{y}_c, \mathbf{z}]) \tag{3}$$

3.2 Generating Pseudo-OOD Data

CVAE's ability to control the generating process using the conditional information (e.g. class label in our case) inspired us to design a method to generate pseudo-OOD samples. This is done by the conditional decoder using atypical prior information c in $p_\theta(\mathbf{x}|\mathbf{z}, c)$. As introduced before, OOD data need to be different from in-distribution (IND) data but also resemble them. The continuity property of CVAE, which means that two close points in the latent space should not give two completely different contents once decoded [6], ensures that we can manipulate CVAE's latent space features to generate high quality pseudo-OOD data. Since we have no information of the future OOD data, we have to make use of the existing training data (i.e., IND data) to construct pseudo-OOD data. We can provide it with pseudo label information to generate pseudo-OOD data.

Specifically, we propose to construct pseudo class embedding by combining the embeddings of two existing classes in the IND training data. The formulation is as follows:

$$p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{k}, c_i, c_j) = p_\theta(\mathbf{x}|[\mathbf{k} * \mathbf{y}_{c_i} + (1 - \mathbf{k}) * \mathbf{y}_{c_j}, \mathbf{z}]) \tag{4}$$

where $\mathbf{k}$ is a vector generated from Bernoulli distribution $\mathcal{B}(0.5)$ with the same length as the class label embedding. $\mathbf{k}$ is basically for the system to randomly select the vector components of the two class embeddings with equal probability. Such a generated sample $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{k}, c_i, c_j)$ will not likely to be an instance of either class c_i or c_j but still keep some of their characteristics, which meets the need of the pseudo-OOD data. Furthermore, the pseudo class embedding has a great variety, owing to the diverse choices of classes and the vector $\mathbf{k}$. To generate pseudo-OOD samples, we also need to sample $\mathbf{z}$ from the encoder. In CVAE training, we ensure that $\mathbf{z}$ fits the Gaussian $\mathcal{N}(\mathbf{0}, \mathbf{I})$. To sample $\mathbf{z}$ for generating, we use another flatter Gaussian distribution $\mathcal{N}(\mathbf{0}, \sigma^2 * \mathbf{I})$, where $\sigma > 1 \in \mathbb{Z}$, to make the generated samples highly diverse.

3.3 Fine-tuning for OOD Detection

As discussed earlier, using the generated pseudo-OOD data and the original in-distribution (IND) training data to directly train a classifier for OOD detection is not a good approach. Here, we propose a fine-tuning method that uses the generated pseudo-OOD samples and the IND training data to learn an OOD detection model in two stages.

Stage 1 (IND classifier building and CVAE training): Only the original IND data is used to train a classification model $\mathcal{C}$. The classification model can be decomposed into two functions $f(\cdot)$ and $h(\cdot)$, where $f(\cdot)$ is the final linear *classifier* and $h(\cdot)$ is the *feature extractor*. $f(h(\mathbf{x}))$ is the classification output. A separate CVAE model is also trained for generating pseudo-OOD data.

Stage 2 (fine-tuning the classifier): We keep the trained feature extractor $h(\cdot)$ fixed (or frozen) and fine-tune only the classification/linear layer $f(\cdot)$ using both the IND and the pseudo-OOD data for OOD detection.

The CMG approach is in fact a framework, which is illustrated in Figure 1 with the 2-stage training process. The framework is flexible as the classifier in the first stage can use any model. Our pseudo-OOD data can help various classifiers improve the ability of OOD detection. Stage 2 is also flexible and can use different approaches. Here we introduce two specific approaches, which are both highly efficient. The second approach CMG-Energy produces new SOTA OOD detection results. As we will see in Sec. 4.4, fine-tuning an existing OOD detection model also enables it to improve.

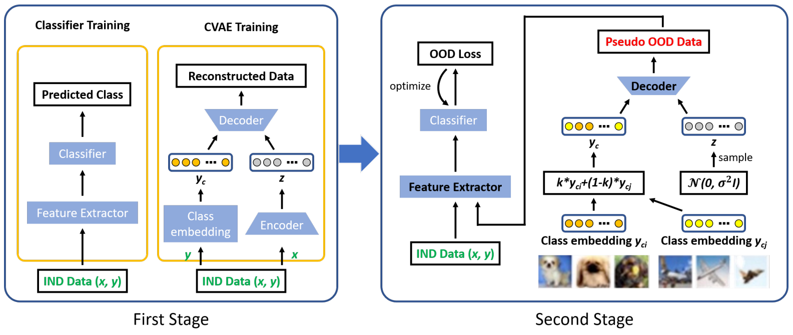


Fig. 1. CMG framework and its training process. The OOD loss can be cross entropy in CMG-softmax, cross-entropy+energy in CMG-energy, or other possible losses. Although we put Classifier Training and CVAE Training in First Stage, they are independent.

CMG-Softmax fine-tuning. In this approach to fine-tuning, we simply add an additional class (which we call the OOD class) in the classification layer to accept the pseudo-OOD data. If the IND data has N classes, we add parameters to the classifier to make it output $N+1$ logits. These added parameters related to the $(N+1)$th OOD class are randomly initialized. We then train the model by only fine-tuning the classification layer using the cross entropy loss with feature extractor trained in Stage 1 fixed. Finally, we use the softmax score of the $(N+1)$th class as the OOD score.

CMG-Energy fine-tuning. This approach adds an energy loss to the cross entropy loss ($\mathcal{L}_{ent} + \lambda\mathcal{L}_{energy}$) to fine-tune the classification layer using the IND and the pseudo-OOD data. No OOD class is added. The energy loss is,

$$\begin{aligned}\mathcal{L}_{energy} = \;&\mathbb{E}_{\mathbf{x}_{ind}\sim\mathcal{D}_{ind}}(\max(0, E(\mathbf{x}_{ind}) - m_{ind}))^2 \\ &+ \mathbb{E}_{\mathbf{x}_{ood}\sim\mathcal{D}_{ood}}(\max(0, m_{ood} - E(\mathbf{x}_{ood})))^2\end{aligned} \tag{5}$$

where $\mathcal{D}_{ind}$ denotes the IND training data, $\mathcal{D}_{ood}$ denotes generated pseudo-OOD data, and m_{ind} and m_{ood} are margin hyper-parameters. The idea of this loss is to make the OOD data get similar values for all N logits so that they will not be favored by any N IND data classes. Here N is the number of classes of the IND data. As the loss function shows, the OOD data are necessary. This loss was used in [33], which has to employ some *real* OOD data but such OOD data are often not available in practice. This loss cannot be used by other OOD methods since they have no OOD data available [3, 21, 53]. However, this is not an issue for us as we have pseudo data to replace real OOD training data.

Stage 2 produces an energy score calculated from a classification model for OOD detection for a test instance $\mathbf{x}$:

$$E(\mathbf{x}; f(h)) = -T \cdot \log \sum_{i=1}^{N} e^{f_i(h(\mathbf{x}))/T} \tag{6}$$

where $E(\mathbf{x}; f(h(\cdot)))$ denotes the energy of instance $\mathbf{x}$ with the classification model $f(h(\cdot))$, which maps $\mathbf{x}$ to N logits, where N is the number of classes in the IND data, $f_i(h(\mathbf{x}))$ is the i-th logit and T is the temperature parameter.

Reason for the 2-stage training. As discussed earlier, with the generated pseudo-OOD data, the obvious and intuitive approach to training an OOD detector is to use the IND data and pseudo-OOD data to build a classifier. However, as the results in Sec. 4.5 show, this is a very poor choice. The reason is that the pseudo-OOD data are not the real OOD data used in testing and their difference can be large because the real OOD data are completely unpredictable and can be anywhere in the space. This combined one-stage training fits only the pseudo-OOD data and may still perform poorly for the real OOD data. The proposed fine-tuning is different. Its Stage 1 training uses only the IND data to learn features for the IND data. In Stage 2, with the feature extractor $h(\cdot)$ fixed, we fine-tune only the final classification layer $f(\cdot)$ using the IND data and the pseudo-OOD data. Since the feature extractor $h(\cdot)$ is not updated by the pseudo-OOD data, the final model $f(\cdot)$ will not overfit the pseudo-OOD data and can give the model more generalization power for OOD detection. Experimental results will demonstrate the extreme importance of the proposed two-stage training.

4 Experiments

We construct OOD detection tasks using benchmark datasets and compare the proposed CMG with the state-of-the-art existing methods.

4.1 Experiment Settings and Data Preparation

We use two experimental settings for evaluation.

Setting 1 - Near-OOD Detection on the Same Dataset: In this setting, IND (in-distribution) and OOD instances are from different classes of the same dataset. This setting is often called *open-set detection.* We use the following 4 popular datasets for our experiments in this setting.

(1) **MNIST** [29]: A handwritten digit classification dataset of 10 classes. The dataset has 70,000 examples/instances, with the splitting of 60,000 for training and 10,000 for testing.

(2) **CIFAR-10** [26]: A 10-class classification dataset consisting of 60,000 32x32 color images with the splitting of 50,000 for training and 10,000 for testing.

(3) **SVHN** [39]: A colorful street view house number classification dataset of 10 classes. It contains 99289 instances with the splitting of 73257 for training and 26032 for testing.

(4) **TinyImageNet** [28]: A classification dataset of 200 classes. Each class contains 500 training samples and 50 testing samples of resolution 64x64.

We follow the data processing method in [51, 58] to split known and unknown classes. For each dataset, we conduct 5 experiments using different splits of known (IND) and unknown (OOD) classes. These same 5 splits are used by all baselines and our system. Following [51], for MNIST, CIFAR-10 and SVHN, 6 classes are chosen as IND classes, and the other 4 classes are regarded as OOD classes. The following 5 fixed sets of IND classes, 0-5, 1-6, 2-7, 3-8, and 4-9, are used and they are called **partitions 1, 2, 3, 4**, and **5**, respectively. The rest 4 classes in each case serve as the OOD classes. For TinyImageNet, each set of IND data contains 20 classes and the sets of IND classes in the 5 experiments are 0-19, 40-59, 80-99, 120-139, and 160-189 respectively. The rest 180 classes are regarded as the OOD classes. The reason for using different partitions is discussed in the supplementary material.

Setting 2 - Far-OOD Detection on Different Datasets: The IND data and OOD data come from different datasets. We use CIFAR-10 and CIFAR100 as the IND dataset respectively and each of the following datasets as the OOD dataset. When CIFAR-10 is used as the IND dataset, the following are used as the OOD datsets (when CIFAR100 is used as the IND dataset, CIFAR-10 is also one of the OOD datasets).

(1) **SVHN** [39]: See above. All 26032 testing samples are used as OOD data.

(2) **LSUN** [56]: This is a large-scale scene understanding dataset with a testing set of 10,000 images from 10 different scenes. Images are resized to 32x32 in our experiment.

(3) **LSUN-FIX** [53]: To avoid artificial noises brought by general resizing operation, this dataset is generated by using a fixed resizing operation on LSUN to change the images to 32x32.

(4) **TinyImageNet** [28]: See above. All 10,000 testing samples are used as OOD data.

(5) **ImageNet-FIX** [28]: 10,000 images are randomly selected from the training set of ImageNet-30, excluding "airliner", "ambulance", "parkingmeter", and "schooner" to avoid overlapping with CIFAR-10. A resizing operation is applied to transform the images to 32x32.

(6) **CIFAR100** [27]: An image classification dataset with 60,000 32x32 color images of 100 classes. Its 10,000 test samples are used as the OOD data.

4.2 Baselines

We compare with 10 state-of-the-art baselines, including 2 generative methods and 2 contrastive learning methods.

(1) **Softmax**: This is the popular classification score model. The highest softmax probability is used as the confidence score for OOD detection.

(2) **OpenMax** [3]: This method combines the softmax score with the distance between the test sample and IND class centers to detect OOD data.

(3) **ODIN** [32]: This method improves the OOD detection performance of a pre-trained neural network by using temperature scaling and adding small perturbations to the input.

(4) **Maha** [31]: This method uses Mahalanobis distance to evaluate the probability that an instance belongs to OOD.

(5) **CCC** [30]: This is a GAN-based method, jointly training the classification model and the pseudo-OOD generator for OOD detection.

(6) **OSRCI** [38]: This method also uses GAN to generate pseudo instances and further improves the model to predict novelty (OOD) examples.

(7) **CAC** [34]: This is a distance-based method, using the Class Anchor Clustering loss to cluster IND samples tightly around the anchored centers.

(8) **SupCLR** [21]: This is a contrastive learning based method. It extends contrastive learning to fully-supervised setting to improve the quality of features

(9) **CSI** [53]: This is also a supervised contrastive learning method. It uses extensive data augmentations to generate shifted data instances. It also has a score function that benefits from the augmented instances for OOD detection.

(10) **React** [52]: This method exploits the internal activations of neural networks to find distinctive signature patterns for OOD distributions.

For Softmax, OpenMax and OSRCI, we use OSRCI's implementation[5]. For SupCLR and CSI, we use CSI's code[6]. For ODIN, Maha, CCC, CAC and React, we use their original code[7][8][9][10][11]. We also use their default hyper-parameters.

4.3 Implementation Details

For MNIST, we use a 9-layer CNN as the encoder (feature extractor) and a 2-layer MLP as the projection head. CVAE includes a 2-layer CNN as the encoder and a 2-layer deconvolution network [57] as the decoder, as well as two 1-layer MLPs to turn features into means and variations. For CIFAR100, the encoder is a PreActResnet [15] and the projection head is a 1-layer MLP. Its CVAE is the same as for the other datasets below. For the other datasets, the encoder is a ResNet18 [14] and the projection head is a 2-layer MLP. CVAE also uses ResNet18

[5] https://github.com/lwneal/counterfactual-open-set
[6] https://github.com/alinlab/CSI
[7] https://github.com/facebookresearch/odin
[8] https://github.com/pokaxpoka/deep_Mahalanobis_detector
[9] https://github.com/alinlab/Confident_classifier
[10] https://github.com/dimitymiller/cac-openset
[11] https://github.com/deeplearning-wisc/react

as the encoder, and 2 residual blocks and a 3-layer deconvolution network as the decoder. The mean and variation projection are completed by two 1-layer MLPs. During the first stage of training, we use Adam optimizer [22] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and learning rate of 0.001. We train both the classification model and CVAE model for 200 epochs with batch size 512. In the second stage, the learning rate is set to 0.0001 and the fine-tuning process with the generated pseudo data are run for 10 epochs. The number of generated pseudo-OOD data is the same as the IND data (we will study this further shortly). Each batch has 128 IND samples and 128 generated OOD samples. There is no special hyper-parameter for CMG-softmax in stage 2. For CMG-energy, two special hyper-parameters of the energy loss m_{ind} and m_{ood} are decided at the beginning of stage 2 by IND and pseudo data. We calculate the energy of all training IND data and generated pseudo data. Then m_{ind} and m_{ood} are chosen to make 80% of IND data's energy larger than m_{ind} and 80% of pseudo data's energy smaller than m_{ood}. This ensures that 80% of data get non-zero loss. We use a NVIDIA-GeForce-RTX-2080Ti GPU for the experiments of evaluating the running speed of different methods.

4.4 Results and Discussions

Table 1 shows the results of the two OOD detection settings on different datasets. Due to the large image size, numerous IND classes and a large batch size requirement, we were unable to run SupCLR and CSI using TinyImageNet on our hardware and thus do not have their results in Setting 1. In Setting 2, CAC crashes owing to too many IND data classes. On average, our CMG achieves the best results in both Setting 1 and Setting 2. Specifically, CMG greatly outperforms the two GAN-based generation methods, CCC and OSRCI, which shows the superiority of CMG in generating and utilizing pseudo-OOD data. We also notice that our CMG-s (CMG-softmax) is slightly weaker than our CMG-e (CMG-energy), which shows the energy function is effective.

Table 2 demonstrates that CMG's fine-tuning (stage 2) can improve the 4 best performing baselines in Table 1, i.e., GAN-based OSRCI and contrastive learning based SupCLR and CSI, and a newest work React. Here after each baseline finishes its training, we apply fine-tuning of CMG's stage 2 to fine-tune the trained model using CMG-energy. We can see that the baselines OSRCI, SupCLR, CSI and React are all improved.

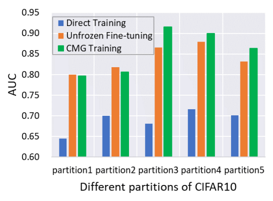
Table 3 shows that CMG is much more efficient than the contrastive learning methods. With the best overall performances on OOD detection, CMG spends about only 10% of contrastive learning training time.

4.5 Ablation Study

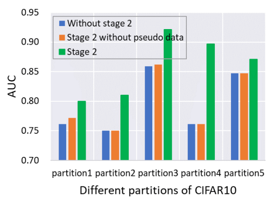
We now perform the ablation study with various options of CMG-e and report AUC scores on the 5 partitions of CIFAR-10 in Setting 1.[12]

[12] We also conducted some experiments using a pre-trained feature extractor. Using a pre-trained feature extractor can be controversial, which is discussed in the supplementary material.

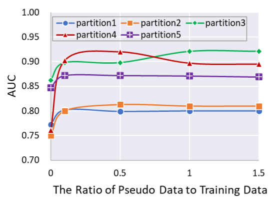
CMG's Two-stage Training vs. One-stage Direct Training. As we stated in the introduction, one-stage *direct training* using the IND training and the generated pseudo-OOD data produces very poor results compared with CMG's 2-stage training with only fine-tuning of the classification layer in the second stage. Here we show the comparison results. We compare three training strategies: (1) *Direct Training*, (2) *Unfrozen Fine-tuning*, i.e., keeping stage 1 but fine-tuning the whole model in stage 2 without freezing the feature extractor, and (3) *CMG training* (CMG-e). Figure 2(a) shows that Direct Training produces very poor results and Unfrozen Fine-Tuning is also weak. CMG Training (CMG-e) performs considerably better. We explained the reason in Sec. 3.3. In these experiments, the energy function in Eq. 6 is used to compute the OOD score. In the supplementary material, we also show that in experiment Setting 2, the same trend applies.



(a) Training Strategies.

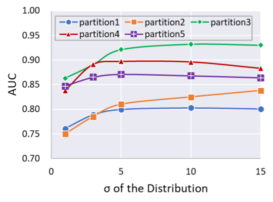


(b) CMG-e Stage 2.

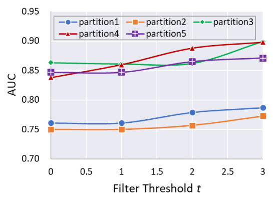


(c) Amount of Pseudo Data.

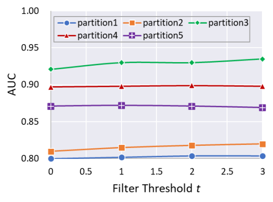
Fig. 2. Ablation studies: (a) different training strategies, (b) CMG-e stage 2, and (c) Amount of Pseudo-OOD Data.



(a) Influence of σ.



(b) Filter under $\sigma = 1$.



(c) Filter under $\sigma = 5$.

Fig. 3. Ablation study on different options in generating pseudo-OOD data. Figure 3(a) shows AUC results of different σ values of the sampling distribution. Figure 3(b) filters values near the center of the Gaussian distribution with $\sigma = 1$. Figure 3(c) filters values near the center of the Gaussian Distribution with $\sigma = 5$.

CMG Stage 2. To verify the effect of different options of stage 2, we compare the results of CMG-e model with **(1)** *without stage 2*, i.e., we directly compute the energy score using Eq. (6) on the classification model from stage 1, **(2)** *stage 2 without using pseudo-OOD data*, i.e., we use only IND data to fine-tune the classifier with loss using Eq. (5), and **(3)** *full stage 2*. Figure 2(b) shows that without stage 2, stage 1 produces poor results. Stage 2 without the generated pseudo-OOD data only improves the performance slightly. The full stage 2 with the generated pseudo-OOD data greatly improves the performance of OOD detection. These experiments prove the necessity of stage 2 and the effectiveness of the generated pseudo-OOD data.

Amount of Pseudo-OOD Data. We run experiments of stage 2 with different numbers of generated pseudo-OOD samples to analyze their effectiveness. Figure 2(c) shows that the model benefits significantly from only a few pseudo-OOD samples. With only 10% of that of the IND data, the pseudo-OOD data can already improve the results markedly, which indicates the importance of the pseudo-OOD data. The results are similar when pseudo-OOD samples are more than a half of the IND samples. We use the same number of pseudo-OOD samples as the IND samples in all our experiments.

Pseudo-OOD Data Distribution. The CVAE generator is trained to make the latent variables or features conform to the Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{I})$ (see Section 3.2). To make pseudo data diverse and different from the training data, we sample the latent variables $\mathbf{z}$ from a pseudo data sampling distribution $\mathcal{N}(\mathbf{0}, \sigma^2 * \mathbf{I})$. We conduct experiments to study the effect of the distribution. First, we study the influence of σ. Note that σ is 1 in training. With larger σ values, the sampled values will be more likely to be far from $\mathbf{0}$ (which is the mean) to make the latent features different from those seen in training.[13] Figure 3(a) shows the results, which indicate the necessity of using $\sigma > 1$ and results are similar for a large range of σ values. We use $\sigma = 5$ in all our experiments.

Intuitively, we may only keep latent features $\mathbf{z}$ that are far from the Gaussian distribution mean by filtering out values that are close to $\mathbf{0}$ (or the mean). We use a filtering threshold t to filter out the sampled $\mathbf{z}$ whose component values are within the range $[-t, t]$. Experimental results in Figure 3(b) allow us to make the following observations. When $\sigma = 1$, as t grows, the performance improves slightly. But comparing with Figure 3(a), we see that a larger σ improves the performance more. Figure 3(c) tells us that when $\sigma = 5$, the effect of filtering diminishes. For simplicity and efficiency, all our experiments employed $\sigma = 5$ without filtration. In the supplementary material, a visual analysis is done for our pseudo-OOD data to further illustrate their high quality.

[13] We include images generated with different choices of σ in the supplementary material. Images generated with larger σ's are more different from the IND data and show a more comprehensive coverage of the OOD area.

5 Conclusion

This paper proposed a novel and yet simple method for OOD detection based on OOD data generation and classifier fine-tuning, which not only produces SOTA results but is also much more efficient than existing highly effective contrastive learning based OOD detection methods. Also importantly, we discovered that using the IND data and the generated pseudo-OOD data to directly train a classifier performs very poorly. The proposed fine-tuning framework CMG works dramatically better. It is also worth noting that the proposed framework can improve the results of diverse state-of-the-art OOD methods too.

References

1. Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**(1), 53–58 (1989)
2. Bendale, A., Boult, T.: Towards open world recognition. In: CVPR. pp. 1893–1902 (2015)
3. Bendale, A., Boult, T.E.: Towards open set deep networks. In: CVPR. pp. 1563–1572 (2016)
4. Bergman, L., Hoshen, Y.: Classification-based anomaly detection for general data. In: International Conference on Learning Representations (2019)
5. Caruana, R., Lou, Y., Gehrke, J., Koch, P., Sturm, M., Elhadad, N.: Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In: 21th ACM SIGKDD. pp. 1721–1730 (2015)
6. Cemgil, T., Ghaisas, S., Dvijotham, K.D., Kohli, P.: Adversarially robust representations with smooth encoders. In: ICLR (2019)
7. Chen, M., Xu, Z.E., Weinberger, K.Q., Sha, F.: Marginalized denoising autoencoders for domain adaptation. In: ICML (2012), http://icml.cc/2012/papers/416.pdf
8. Deecke, L., Vandermeulen, R., Ruff, L., Mandt, S., Kloft, M.: Image anomaly detection with generative adversarial networks. In: Joint european conference on machine learning and knowledge discovery in databases. pp. 3–17. Springer (2018)
9. Fort, S., Ren, J., Lakshminarayanan, B.: Exploring the limits of out-of-distribution detection. Advances in NeurlPS **34** (2021)
10. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)
11. Golan, I., El-Yaniv, R.: Deep anomaly detection using geometric transformations. Advances in NeurlPS **31** (2018)
12. Gunther, M., Cruz, S., Rudd, E.M., Boult, T.E.: Toward open-set face recognition. In: CVPR Workshops. pp. 71–80 (2017)
13. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: ICML. pp. 1321–1330. PMLR (2017)
14. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778 (2016)
15. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV. pp. 630–645. Springer (2016)
16. Hendrycks, D., Mazeika, M., Dietterich, T.: Deep anomaly detection with outlier exposure. In: ICLR (2019), https://openreview.net/forum?id=HyxCxhRcY7
17. Hendrycks, D., Mazeika, M., Kadavath, S., Song, D.: Using self-supervised learning can improve model robustness and uncertainty. Advances in NeurlPS **32** (2019)

18. Hinton, G.E., Osindero, S., Teh, Y.W.: A fast learning algorithm for deep belief nets. Neural computation **18**(7), 1527–1554 (2006)
19. Hu, W., Wang, M., Qin, Q., Ma, J., Liu, B.: Hrn: A holistic approach to one class learning. Advances in Neural Information Processing Systems **33**, 19111–19124 (2020)
20. Júnior, P.R.M., De Souza, R.M., Werneck, R.d.O., Stein, B.V., Pazinato, D.V., de Almeida, W.R., Penatti, O.A., Torres, R.d.S., Rocha, A.: Nearest neighbors distance ratio open-set classifier. Machine Learning **106**(3), 359–386 (2017)
21. Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., Krishnan, D.: Supervised contrastive learning. Advances in NeurlPS **33**, 18661–18673 (2020)
22. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. In: ICLR (2015)
23. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
24. Kingma, D.P., Mohamed, S., Jimenez Rezende, D., Welling, M.: Semi-supervised learning with deep generative models. Advances in NeurlPS **27** (2014)
25. Kolesnikov, A., Zhai, X., Beyer, L.: Revisiting self-supervised visual representation learning. In: CVPR. pp. 1920–1929 (2019)
26. Krizhevsky, A., Hinton, G.: Convolutional deep belief networks on cifar-10. Unpublished manuscript **40**(7), 1–9 (2010)
27. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
28. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**, 7 (2015)
29. LeCun, Y., Cortes, C., Burges, C.: Mnist handwritten digit database (2010)
30. Lee, K., Lee, H., Lee, K., Shin, J.: Training confidence-calibrated classifiers for detecting out-of-distribution samples. In: ICLR (2018)
31. Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. Advances in NeurlPS **31** (2018)
32. Liang, S., Li, Y., Srikant, R.: Enhancing the reliability of out-of-distribution image detection in neural networks. arXiv preprint arXiv:1706.02690 (2017)
33. Liu, W., Wang, X., Owens, J., Li, Y.: Energy-based out-of-distribution detection. Advances in NeurlPS **33**, 21464–21475 (2020)
34. Miller, D., Sunderhauf, N., Milford, M., Dayoub, F.: Class anchor clustering: A loss for distance-based open set recognition. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 3570–3578 (2021)
35. Mohseni, S., Pitale, M., Yadawa, J., Wang, Z.: Self-supervised learning for generalizable out-of-distribution detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 5216–5223 (2020)
36. Nalisnick, E., Matsukawa, A., Teh, Y.W., Gorur, D., Lakshminarayanan, B.: Do deep generative models know what they don't know? In: ICLR (2019)
37. Nalisnick, E., Matsukawa, A., Teh, Y.W., Lakshminarayanan, B.: Detecting out-of-distribution inputs to deep generative models using typicality (2020)
38. Neal, L., Olson, M., Fern, X., Wong, W.K., Li, F.: Open set learning with counterfactual images. In: ECCV. pp. 613–628 (2018)
39. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning (2011)
40. Nitsch, J., Itkina, M., Senanayake, R., Nieto, J., Schmidt, M., Siegwart, R., Kochenderfer, M.J., Cadena, C.: Out-of-distribution detection for automotive perception. In: 2021 IEEE ITSC. pp. 2938–2943. IEEE (2021)
41. Oza, P., Patel, V.M.: C2ae: Class conditioned auto-encoder for open-set recognition. In: CVPR. pp. 2307–2316 (2019)

42. Perera, P., Nallapati, R., Xiang, B.: Ocgan: One-class novelty detection using gans with constrained latent representations. In: CVPR. pp. 2898–2906 (2019)
43. Pidhorskyi, S., Almohsen, R., Doretto, G.: Generative probabilistic novelty detection with adversarial autoencoders. Advances in NeurlPS **31** (2018)
44. Pourreza, M., Mohammadi, B., Khaki, M., Bouindour, S., Snoussi, H., Sabokrou, M.: G2d: Generate to detect anomaly. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2003–2012 (2021)
45. Qiu, C., Pfrommer, T., Kloft, M., Mandt, S., Rudolph, M.: Neural transformation learning for deep anomaly detection beyond images. In: International Conference on Machine Learning. pp. 8703–8714. PMLR (2021)
46. Ren, J., Liu, P.J., Fertig, E., Snoek, J., Poplin, R., Depristo, M., Dillon, J., Lakshminarayanan, B.: Likelihood ratios for out-of-distribution detection. Advances in NeurlPS **32** (2019)
47. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. nature **323**(6088), 533–536 (1986)
48. Sastry, C.S., Oore, S.: Detecting out-of-distribution examples with gram matrices. In: ICML. pp. 8491–8501. PMLR (2020)
49. Sehwag, V., Chiang, M., Mittal, P.: {SSD}: A unified framework for self-supervised outlier detection. In: ICLR (2021), https://openreview.net/forum?id=v5gjXpmR8J
50. Serrà, J., Álvarez, D., Gómez, V., Slizovskaia, O., Núñez, J.F., Luque, J.: Input complexity and out-of-distribution detection with likelihood-based generative models. In: ICLR (2020)
51. Sun, X., Yang, Z., Zhang, C., Ling, K.V., Peng, G.: Conditional gaussian distribution learning for open set recognition. In: CVPR. pp. 13480–13489 (2020)
52. Sun, Y., Guo, C., Li, Y.: React: Out-of-distribution detection with rectified activations. In: Advances in NeurlPS (2021)
53. Tack, J., Mo, S., Jeong, J., Shin, J.: Csi: Novelty detection via contrastive learning on distributionally shifted instances. Advances in NeurlPS **33**, 11839–11852 (2020)
54. Vernekar, S., Gaurav, A., Abdelzad, V., Denouden, T., Salay, R., Czarnecki, K.: Out-of-distribution detection in classifiers via generation. arXiv preprint arXiv:1910.04241 (2019)
55. Vincent, P., Larochelle, H., Bengio, Y., Manzagol, P.A.: Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th ICML. pp. 1096–1103 (2008)
56. Yu, F., Seff, A., Zhang, Y., Song, S., Funkhouser, T., Xiao, J.: Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. arXiv preprint arXiv:1506.03365 (2015)
57. Zeiler, M.D., Taylor, G.W., Fergus, R.: Adaptive deconvolutional networks for mid and high level feature learning. In: 2011 ICCV. pp. 2018–2025. IEEE (2011)
58. Zhou, D.W., Ye, H.J., Zhan, D.C.: Learning placeholders for open-set recognition. In: CVPR. pp. 4401–4410 (June 2021)
59. Zong, B., Song, Q., Min, M.R., Cheng, W., Lumezanu, C., Cho, D., Chen, H.: Deep autoencoding gaussian mixture model for unsupervised anomaly detection. In: ICLR (2018)
60. Zongyuan Ge, S.D., Garnavi, R.: Generative openmax for multi-class open set classification. In: Tae-Kyun Kim, Stefanos Zafeiriou, G.B., Mikolajczyk, K. (eds.) BMVC. pp. 42.1–42.12. BMVA Press (September 2017). https://doi.org/10.5244/C.31.42

Table 1. AUC (Area Under the ROC curve) (%) on detecting IND and OOD samples in 2 experimental settings. For Setting 1, the results are averaged over the 5 partitions. CMG-s uses CMG-softmax fine-tuning and CMG-e uses CMG-energy fine-tuning. Every experiment was run 5 times. Each result in brackets () in the *average* row for Setting 1 is the mean of the first three datasets as SupCLR and CSI cannot run on TinyImageNet. CAC crashed when using CIFAR100 as the IND data.

Datasets	Softmax	OpenMax	ODIN	Maha	CCC	OSRCI	CAC	SupCLR	CSI	React	CMG-s	CMG-e
Setting 1 - Near-OOD Detection on the Same Dataset												
MNIST	97.6	98.1	98.1	98.4	94.2	98.3	**99.2**	97.1	97.2	98.6	98.3	99.0
(std)	±0.7	±0.5	±1.1	±0.4	±0.8	±0.9	±**0.1**	±0.2	±0.3	±0.1	±0.2	±0.2
CIFAR-10	65.5	66.9	79.4	73.4	74.0	67.5	75.9	80.0	84.7	85.5	**86.3**	85.6
(std)	±0.5	±0.4	±1.6	±2.2	±1.4	±0.8	±0.7	±0.5	±0.3	±0.2	±**1.2**	±0.6
SVHN	90.3	90.7	89.4	91.5	64.6	91.7	93.8	93.8	**93.9**	92.8	91.8	92.1
(std)	±0.5	±0.4	±2.0	±0.6	±2.3	±0.2	±0.2	±0.2	±**0.1**	±0.1	±0.5	±0.4
TinyImageNet	57.5	57.9	70.9	56.3	51.0	58.1	71.9	\	\	51.9	72.2	**73.7**
(std)	±0.7	±0.2	±1.5	±1.9	±1.2	±0.4	±0.7	\	\	±0.0	±0.5	±**0.6**
Average	77.8	78.4	84.5	79.9	71.0	78.9	85.2	(90.3)	(91.9)	82.2	87.2	**87.6**
Setting 2 - Far-OOD Detection on Different Datasets												
CIFAR-10 as IND												
SVHN	80.2	82.7	83.2	97.5	83.3	80.2	87.3	97.3	**97.9**	92.2	95.8	96.2
(std)	±1.8	±1.9	±1.5	±1.6	±0.8	±1.8	±4.6	±0.1	±**0.1**	±1.1	±0.6	±2.5
LSUN	70.1	72.2	82.1	61.5	85.6	79.9	89.1	92.8	97.7	96.5	96.8	**97.7**
(std)	±2.5	±1.8	±1.9	±5.0	±2.3	±1.8	±3.4	±0.5	±0.4	±0.7	±1.4	±**0.9**
LSUN-FIX	76.7	75.6	84.1	77.8	86.6	78.2	85.5	91.6	93.5	90.6	**94.1**	93.7
(std)	±0.8	±1.2	±1.7	±2.1	±1.6	±0.5	±0.7	±1.5	±0.4	±1.9	±**0.9**	±0.4
TinyImageNet	62.5	65.2	68.7	56.8	83.2	70.0	86.4	91.4	**97.6**	94.3	94.8	95.2
(std)	±3.6	±3.1	±2.2	±2.1	±1.8	±1.7	±4.6	±1.2	±**0.3**	±0.5	±1.6	±2.7
ImageNet-FIX	75.9	75.6	74.8	79.0	83.7	78.1	85.6	90.5	**94.0**	92.0	89.7	92.9
(std)	±4.6	±0.7	±0.6	±3.1	±1.1	±0.3	±0.3	±0.5	±**0.1**	±2.2	±0.3	±1.2
CIFAR100	74.6	75.5	74.5	61.4	81.9	77.4	83.9	88.6	**92.2**	88.4	87.9	89.3
(std)	±0.5	±0.4	±0.8	±0.9	±0.5	±0.4	±0.2	±0.2	±**0.1**	±0.7	±0.4	±0.4
CIFAR100 as IND												
SVHN	66.7	65.9	71.7	**93.1**	66.0	65.5	\	83.4	88.2	88.6	90.0	90.2
(std)	±4.0	±4.5	±1.7	±**0.6**	±1.0	±1.1	\	±0.5	±0.7	±1.3	±2.4	±2.5
LSUN	48.1	53.7	66.0	**95.6**	68.7	74.4	\	81.6	80.9	88.1	85.9	88.3
(std)	±4.6	±6.7	±1.0	±**0.1**	±1.0	±0.8	\	±0.5	±0.5	±2.8	±2.6	±3.6
LSUN-FIX	47.5	50.4	72.6	63.4	59.3	69.7	\	70.9	74.0	69.7	76.5	**77.4**
(std)	±4.1	±5.8	±7.5	±3.4	±1.7	±0.6	\	±0.1	±0.2	±0.5	±8.5	±**2.4**
TinyImageNet	62.5	62.5	73.5	**93.3**	69.7	63.9	\	78.5	79.4	87.0	84.3	88.2
(std)	±2.9	±3.0	±3.3	±**0.7**	±1.6	±1.2	\	±0.8	±0.2	±3.2	±4.9	±2.0
ImageNet-FIX	64.8	64.5	76.9	61.2	60.6	63.8	\	75.0	**79.2**	78.9	72.5	76.5
(std)	±0.5	±0.6	±9.1	±0.9	±0.0	±0.9	\	±0.5	±**0.2**	±0.3	±2.2	±1.3
CIFAR-10	63.0	62.7	67.9	56.3	63.7	58.8	\	72.2	**78.2**	74.4	68.8	71.5
(std)	±1.0	±1.0	±2.1	±0.5	±0.5	±0.8	\	±0.6	±**0.2**	±1.3	±3.1	±2.9
Average	66.1	67.2	74.7	74.7	74.4	71.7	\	84.5	87.7	86.7	86.4	**88.1**

Table 2. AUC (Area Under the ROC curve) (%) results of the original model (denoted by **original**) and the model plus fine-tuning using CMG-energy (denoted by **+CMG-e**). Almost every +CMG-e version of the baselines outperforms the original model. Every experiment was run 5 times.

Datasets	OSRCI		SupCLR		CSI		React	
	original	+CMG-e	original	+CMG-e	original	+CMG-e	original	+CMG-e
Setting 1 - Near-OOD Detection on the Same Dataset								
MNIST	98.3±0.9	99.1±0.4	97.1±0.2	98.6±0.2	97.2±0.3	99.3±0.1	98.6±0.1	98.9±0.1
CIFAR-10	67.5±0.8	72.3±0.6	80.0±0.5	88.9±0.5	84.7±0.3	89.8±0.6	85.5±0.2	85.8±0.1
SVHN	91.7±0.2	92.1±0.1	93.8±0.2	96.5±0.3	93.9±0.1	96.7±0.2	92.8±0.1	92.8±0.1
TinyImageNet	58.1±0.4	59.9±0.3	\	\	\	\	51.9±0.0	51.8±0.1
Average	78.9	**80.9**	90.3	**94.7**	91.9	**95.3**	82.2	**82.3**
Setting 2 - Far-OOD Detection on Different Datasets								
CIFAR-10 as IND								
SVHN	80.2±1.8	79.3±2.5	97.3±0.1	93.0±1.3	97.9±0.1	97.8±0.6	92.1±1.1	98.2±0.8
LSUN	79.9±1.8	92.1±0.6	92.8±0.5	97.7±0.6	97.7±0.4	99.2±0.1	96.5±0.7	96.4±0.4
LSUN-FIX	78.2±0.5	81.2±1.0	91.6±1.5	94.1±0.3	93.5±0.4	96.2±0.3	90.6±1.9	91.9±0.2
TinyImageNet	70.0±1.7	83.2±1.7	91.4±1.2	96.3±0.8	97.6±0.3	98.7±0.3	94.3±0.5	94.0±0.6
ImageNet-FIX	78.1±0.3	78.5±0.2	90.5±0.5	92.9±0.3	94.0±0.1	95.7±0.1	92.0±2.2	91.3±0.1
CIFAR100	77.4±0.4	77.4±0.6	88.6±0.2	90.3±0.2	92.2±0.1	92.0±0.2	88.4±0.7	89.2±0.2
CIFAR-100 as IND								
SVHN	65.5±1.1	87.2±1.3	83.4±0.5	85.3±1.1	88.2±0.7	85.9±1.2	88.6±1.3	97.1±1.3
LSUN	74.4±0.8	76.5±0.7	81.6±0.5	84.3±0.9	80.9±0.5	89.9±0.8	88.1±2.8	89.3±0.8
LSUN-FIX	69.7±0.6	71.7±0.9	70.9±0.1	69.8±0.8	74.0±0.2	74.0±1.3	69.7±0.5	70.6±2.5
TinyImageNet	63.9±1.2	67.3±0.4	78.5±0.8	84.2±1.1	79.4±0.2	89.4±0.8	87.0±3.2	87.9±0.5
ImageNet-FIX	63.8±0.9	66.1±0.9	75.0±0.5	72.4±0.8	79.2±0.2	79.6±1.1	78.9±0.3	79.8±0.2
CIFAR-10	58.8±0.8	61.9±0.4	72.2±0.6	75.9±0.7	78.2±0.2	72.2±0.4	74.4±1.3	72.9±0.5
Average	71.7	**76.9**	84.5	**86.4**	87.7	**89.2**	86.7	**88.2**

Table 3. Execution time (min) of each method spent in running the whole experiment on benchmark datasets for Setting 1.

Datasets	Softmax	OpenMax	ODIN	Maha	CCC	OSRCI	CAC	SupCLR	CSI	React	CMG-e
MNIST	6	6	71	54	133	49	13	1260	1728	65	24
CIFAR-10	20	20	61	56	111	70	49	1110	1428	61	144
SVHN	20	20	142	140	196	71	37	1770	2471	79	249
TinyImageNet	22	22	64	54	46	79	64	\	\	65	131