Learning After Model Deployment

Derda Kaymak [a], Gyuhak Kim [b], Tomoya Kaichi [c], Tatsuya Konishi [c] and Bing Liu [a,*]

[a]University of Illinois Chicago, USA
[b]Accenture, USA
[c]KDDI Research, Japan

Abstract. In classic supervised learning, once a model is deployed in an application, it is fixed. No updates will be made to it during the application. This is inappropriate for many dynamic and open environments, where unexpected samples from unseen classes may appear. In such an environment, the model should be able to detect these novel samples from unseen classes and learn them after they are labeled. We call this paradigm ***Autonomous Learning after Model Deployment*** (ALMD). The learning here is ***continuous*** and involves ***no human engineers***. Labeling in this scenario is performed by human co-workers or other knowledgeable agents, which is similar to what humans do when they encounter an unfamiliar object and ask another person for its name. In ALMD, the detection of novel samples is dynamic and differs from traditional out-of-distribution (OOD) detection in that the set of in-distribution (ID) classes expands as new classes are learned during application, whereas ID classes is fixed in traditional OOD detection. Learning is also different from classic supervised learning because in ALMD, we learn the encountered new classes *immediately* and *incrementally*. It is difficult to retrain the model from scratch using all the past data from the ID classes and the novel samples from newly discovered classes, as this would be resource- and time-consuming. Apart from these two challenges, ALMD faces the data scarcity issue because instances of new classes often appear sporadically in real-life applications. To address these issues, we propose a novel method, PLDA, which performs dynamic OOD detection and incremental learning of new classes on the fly. Empirical evaluations will demonstrate the effectiveness of PLDA.

1 Introduction

A traditional machine learning application starts by training a model. Once the model accuracy is satisfactory, it is deployed to the application. During the application process, the model is fixed, i.e., no change to the model can be made. It assumes that the classes seen in the application must have been seen in training. This is commonly known as the *closed-world assumption* [10, 26, 5], meaning no samples from unseen classes can appear in applications. In contrast, the real world is an open environment, full of unknowns and novelties, also known as *out-of-distribution* (OOD) objects. To function effectively in this *open world*, an AI agent must continuously learn on the fly after deployment, rather than relying on periodic offline retraining initiated by human engineers. This means the *deployed model* should not be frozen, and more knowledge can be learned during application. We call this paradigm the *Autonomous Learning after Model Deployment* (**ALMD**). ALMD needs three key capabilities:

(1) detecting OOD samples continually based on the current set of classes (called *in-distribution* (ID) classes) that have been learned,

(2) obtaining the class labels of the detected OOD samples, and

(3) learning the OOD samples on the fly incrementally or continually. This is the *class-incremental learning* (CIL) setting of continual learning (CL) as the system learns more and more classes.

This paper focuses on (1) and (3). For (2), we do as humans do. When we humans encounter unknown objects, we usually ask others for their names (i.e., *class labels*). We assume the AI agent can ask human co-workers or other agents to provide labels for the detected OOD samples.[1] Furthermore, in ALMD, the data comes in a stream, and OOD detection and learning of the detected OOD samples are done online. ALMD is thus a **continual and autonomous learning** paradigm. ***Autonomous*** means that the AI agent **takes full control** of its learning process and learns from its **own experiences** [36]: (1) It discovers its tasks (OOD classes) to learn, (2) acquires class labels for the detected OOD samples through its interaction with human co-workers or other agents, and (3) learns the new classes incrementally. The whole process involves no human engineers.

ALMD Problem Setting. Since ALMD learns continually after model deployment, the initially deployed model M is assumed to be well-trained with a set of initial classes C of labeled data. After M is deployed in its application, it detects and learns more and more new classes. At the steady state, the set of all classes that the system has encountered is $C^A = C \cup C^L \cup C^E$, where C^L is the set of new classes that have appeared after deployment and are *well learned* after seeing a good number of training samples, and C^E is a set of emerging new classes that have been seen but are *not well learned yet*, i.e., not enough labeled training data have been seen to well-learn the classes. We denote $C^+ = C \cup C^L$ as the set of well-learned classes (ID classes) so far and $C^N = C^L \cup C^E$ as the set of all new classes seen after deploying M. With incremental learning of new classes, M becomes M^+, covering all classes in C^A. Each iteration of ALMD performs two main functions.

(1). *OOD Detection and Classification*. M^+ detects whether each incoming test sample x is OOD. If not, it is classified as one of the classes in C^+. OOD classes include those emerging classes in C^E as they still need some more data to be well-learned, but OOD detection can leverage the already-seen samples of these classes. This is different from existing OOD detection or continual OOD detection [51] (which detects OOD cases in continual learning).

(2). *Incremental Learning*. The system learns each detected OOD sample x after obtaining its class label by asking a human or a knowl-

* Corresponding Author. Email: liub@uic.edu.

[1] It is possible to use a vision-language model to help assign class labels. But it cannot guarantee correctness. We leave this to our future work.

edgeable agent. If x is assigned a class in C, do nothing. If x is assigned a class label in C^N, the current model M^+ is updated.

ALMD is thus related to three main areas of research, (1) **OOD detection** [60], (2) class incremental learning (CIL) [2] in continual learning [56, 9], particularly **online continual learning** or online CIL as online CIL also learns from the streaming data [3, 29, 39], and (3) **open world learning** [5, 11, 26]. These topics have been studied separately. However, OOD detection in ALMD has to be done continually (i.e., *continual OOD* (C-OOD) *detection*), unlike the traditional static OOD detection with a set of fixed *in-distribution* (ID) classes. The number of ID classes in C-OOD detection increases as the AI agent learns new classes of objects. ALMD is also very different from CIL or online CIL because CIL or online CIL does not do OOD detection. It also faces the major challenge of ***catastrophic forgetting*** (**CF**). CF refers to the phenomenon that the learner needs to modify the parameters learned for previous tasks in learning the new task, which may cause performance degradation for previous tasks. ALMD is also different from open world learning (OWL) [5, 11]. OWL still works in the pre-deployment stage, not in the post-deployment stage as we do. They are basically offline CIL that can also do OOD detection. The ID classes in OWL are only the set of classes learned in pre-deployment, which is fixed. In our case, the set of ID classes increases as more classes become well-learned classes post-deployment. [26] proposed a theoretical framework that is suitable for ALMD, but it does not present any algorithm. Its empirical work is only on CIL with no mechanisms for continual OOD detection. We will discuss more about the topic in Section 2.

This paper proposes a novel approach called PLDA (*P*ost-deployment Learning based on *L*inear *D*iscriminant *A*nalysis) to learn in the ALMD setting, i.e., performing the above two main functions. The method is based on *linear discriminant analysis* (LDA) [46], which obtains its features from a pre-trained model (PTM). LDA assumes that given the class, the data follows a normal distribution with a mean and a covariance matrix. It further assumes that the class covariances are identical, i.e., all classes share one covariance matrix but have different means. LDA uses the means and covariance for classification. However, LDA is not suitable for OOD detection because LDA is based on the likelihood ratio, which is only suitable for closed-world classification, as OOD detection needs a measure of absolute distance from a sample to a distribution. In this work, we use Mahalanobis distance (MD) and a related method for OOD detection with a novel idea.

After obtaining the label of a detected OOD sample, PLDA learns it immediately. Each new class still uses the same shared covariance matrix learned initially in M, but the mean of the class is updated. The pipeline of PLDA is given in Figure 1. It may **sound highly limiting** that LDA uses only the features from a PTM and assumes the same covariance matrix for all classes. However, as shown in Table 1 in Sec. 4.3, PLDA achieves a level of accuracy very close to the joint training upper bound accuracy using the pre-trained model ViT-B/14-DINOv2 [45], which has never been achieved before.

This paper thus makes the following contributions.

1. It proposes a realistic ALMD setting, which is important as AI agents working in the real open world need to continually learn new knowledge on the fly autonomously from its own experience after deployment to make it more and more knowledgeable.

2. It proposes a novel approach based on incremental updating of the model with a shared covariance and different means for different classes, which has **no CF** because PLDA does not do parameter updating after deployment and gives remarkably accurate results without many training samples from each new class. This is particularly important because it is hard to obtain many labeled samples after model deployment and it has not been done before.

3. In continual OOD (C-OOD) detection, we not only use the ID classes but also already detected OOD samples to help detect more OOD data more accurately. To our knowledge, this has not been done before either.

Experiments have been conducted to demonstrate the effectiveness of the proposed PLDA. The **code of PLDA** is available at [21].

2 Related work

OOD Detection. OOD detection has been studied under many names, e.g., novelty or outlier detection, anomaly detection, OOD detection, and open set recognition [10, 13, 5, 40], In recent years, deep learning approaches have produced state-of-the-art results [60]. One popular category of methods uses logits to compute OOD scores [18]. Some other works also use additional mechanisms [54, 37]. Many also improve the architecture and features [20, 54]. Yet, some others use ensembles [31]. Some approaches also expose the system to some OOD data during training [19, 47]. Some work also clusters the detected OOD samples into classes [15], which we don't do as we learn each OOD sample right after it is detected.

Our OOD detection method is most closely related to distance-based methods [33, 50]. However, our work does both OOD detection and continual learning. Unlike existing OOD detection methods, the number of ID classes in our case is not fixed but continues to increase. We also use newly identified OOD data (i.e., seen OOD classes that are not well-learned yet) to detect more OOD data.

Continual learning. The existing work mainly focuses on overcoming CF [22, 57]. Existing methods belong to several categories. *Regularization*-based methods deal with CF in learning a new task by using a regularizer to penalize changes to parameters that are important to previous tasks [28, 1]. *Replay*-based methods store some data from previous tasks. When learning a new task, the saved data and the new task data are used jointly to train the new task while also adjusting the previous task parameters so that their performance will not deteriorate significantly [2, 8]. *Pseudo-replay*-based methods build a data generator to generate previous task data to replace the replay data [59, 23]. *Parameter-isolation*-based methods use masks to protect the learned models for previous tasks so that they will not be updated in learning a new task, which avoids CF [41, 53, 58]. *Orthogonal projection*-based methods learn each task in an orthogonal space to the previous task spaces to reduce CF [61, 35]. Recently, the parameter-isolation approach and OOD detection are combined for class-incremental learning (CIL) [25]. However, this approach is not for open-world continual learning, but for traditional CIL.

Most of the above methods were proposed for *offline continual learning* (CL). Our work is more related to *online Cl*, which learns from a data stream [39]. There are many existing online CL methods [2, 48, 38, 4, 29, 43]. However, none of these methods does OOD detection, which makes it inapplicable to ALMD.

Our work is closely related to the work in [16, 44]. [16] uses *incremental linear discriminant analysis* (ILDA) [46] for online continual learning, but it does not detect OOD instances or work in ALMD after model deployment. Further, our work does not use ILDA. [16] already showed that even with a fixed covariance matrix, a certain

[2] *Class-incremental learning* (CIL) is a setting of continual learning that aims to learn a sequence of tasks incrementally, where each task consists of one or more classes to be learned. The classes in the tasks are disjoint. At test time, no task-related information, e.g., task-identifier, is given.

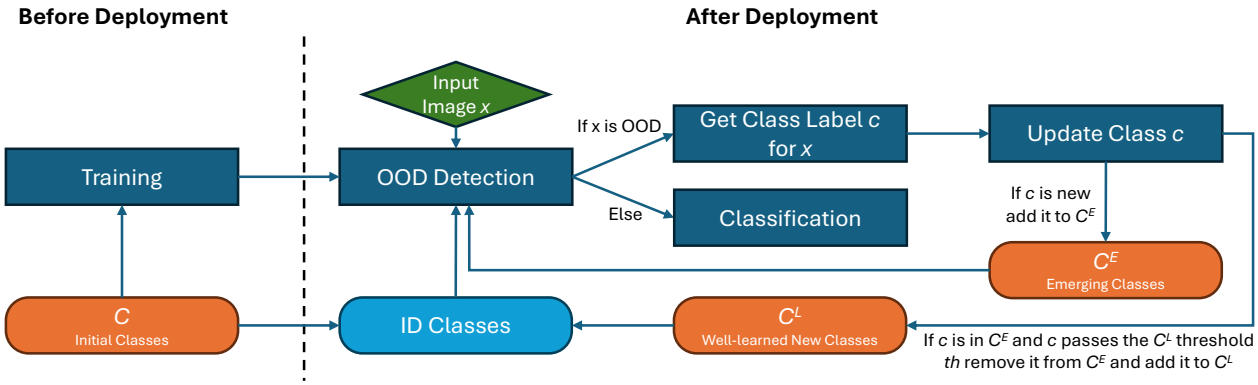

Figure 1. Pipeline of the proposed PLDA method. ID (in-distribution) classes, which are used in OOD detection, include both C and C^L.

amount of base classes is sufficient to create a continual learning model that is robust to increases in the number of classes and distribution changes. It is discussed in more detail in Section 4.3. [44] is based on a kerneled LDA for offline CL.

Open world learning. [5] can incrementally learn new classes, similar to CIL. It still works in the pre-deployment stage. In post-deployment or testing, the system can do classification (known classes) and OOD detection (unknown classes). We learn in the post-deployment stage, i.e., using OOD detection to identify OOD samples from unseen classes during model application and then learn the unseen classes into the model on the fly based on each detected OOD sample. [11] works in a similar setting as the method in [5]. [27] performed a theoretical study of open world continual learning, but it offers no method for OOD detection. [14] proposed SHELS for OOD detection and CL. Learning is still only in the pre-deployment stage. It does not integrate OOD detection and CL like ours. Their two functions can only be evaluated separately, and their CL is not on streaming data but on traditional offline CL. [51] proposed IncDFM that detects OOD using a pre-trained model. However, it does not continually learn. [52] investigated class-incremental novel class discovery (class-iNCD), focusing on discovering new classes. [17] addressed OOD detection in an unsupervised setting. In summary, none of these methods integrates OOD detection and continual learning, allowing the system to learn on the fly after model deployment.

3 Proposed approach: PLDA

We now present the proposed approach PLDA for solving ALMD. We start with the key ***challenges*** of ALMD and the main idea of the proposed techniques and their ***novelties***. Recall that ALMD has two main steps: (1) continual OOD (C-OOD) detection, and (2) class incremental learning (CIL). As discussed earlier, we assume that the class label for each detected OOD sample can be obtained by asking a human or another agent while working with them.

Both steps are highly challenging. For (1), C-OOD detection is dynamic. A traditional OOD detection model is built based on a set of fixed ID classes. The key novelty of our C-OOD detection is that we also use the identified OOD samples to detect more OOD samples. For (2), the key challenge is that the AI agent should not ask human users for labels of the detected OOD samples too many times, which means we must have a strong learning capability without using many labeled samples. For both (1) and (2), there is also the challenge of *catastrophic forgetting* (CF). PLDA deals with all these challenges with the help of *linear discriminant analysis* (LDA).

3.1 Linear discriminant analysis (LDA)

LDA is a statistical classification method that assumes each class is a normal distribution with parameters of covariance and mean for each class, i.e., (Σ_i, μ_i) [12]. Most LDA methods also make the simplifying homoscedasticity assumption that the class covariances are identical, i.e., $\Sigma_i = \Sigma_j = \Sigma$ for $i \neq j$. Thus, the differences between different classes are in only their means, μ_i's. This assumption is particularly useful for continual learning as the system does not have to save one covariance matrix for each class, which can consume a huge amount of memory as more classes are learned. LDA also makes it possible without using any replay data.

3.2 PLDA Method

The proposed method PLDA uses a *pre-trained model* (which is frozen throughout), *a continual OOD detection method*, and the *linear discriminant analysis* (LDA) method to solve the ALMD problem. The pre-trained model will be described in the experiment section. The proposed system PLDA consists of the following steps.

1. Building the Initial Model M. PLDA uses the pre-trained model f to provide the features for input samples, which are used by LDA to build a classifier M using the initial classes C. As mentioned earlier, LDA's classifier building is based on a mean μ_i for each class $i \in C$ and a single shared covariance matrix Σ across all classes. Thus, it produces the shared Σ and a separate mean μ_i for each class $i \in C$. The resulting model M is deployed in its application (Figure 1).

2. Post-deployment Continual Learning. After deployment, it continues to learn, which will update M after new classes are incrementally learned and M becomes M^+. In the continual learning process, Σ remains unchanged or frozen and it is also used by the newly detected classes, which, as discussed in Section 1, is by no means limiting. Each iteration has two sub-steps.

2.1. Continual OOD Detection and Classification. PLDA uses M^+ to detect whether each sample x in the online stream is an OOD sample. If not, it is classified to its class (see Figure 1). Note that M^+ is M initially. PLDA employs the covariance matrix Σ and the μ_i's for all classes encountered or seen so far to perform the tasks.

At a steady state, the set of all classes that the system has encountered is $C^A = C \cup C^L \cup C^E$, where C is the initial set of classes learned in M, C^L is a set of *well learned* new classes after seeing a good number of instances, and C^E is a set of emerging classes that have been seen but are *not yet well learned*. Note that, in the ALMD

setting, the classes are updated multiple times, with one sample at a time, throughout the AI agent's lifetime, as it encounters a sample from an OOD class. **Well learned** means that the mean of the class does not change much after more samples are added. A class becomes well-learned if its mean is updated at least th times, where th is the selected convergence threshold. We denote $C^+ = C \cup C^L$ as the set of well-learned classes so far and $C^N = C^L \cup C^E$ as the set of all new classes seen after the deployment of M. With incremental learning, M becomes M^+, covering all classes in C^A.

What is important here is that OOD detection not only uses the classes in C^+ but also leverages the covariance Σ and the current un-converged means of the classes in C^E to detect OOD samples belonging to C^E and other new classes. To our knowledge, no existing method does that. This is advantageous because a new sample may be similar to a class in C^E, which makes OOD detection more effective. In Sec. 3.3, we discuss the OOD detection methods used in our PLDA. If a test sample x is near a class in C^E, it is also considered an OOD sample.

2.2 Continual Learning - *class-incremental learning* (CIL). Here the continual learning setting is CIL, which incrementally learns more and more new classes. Specifically, PLDA learns each detected novel instance x after obtaining its class label by asking a human user or another knowledgeable agent. If x is assigned a class label in C, do nothing (i.e., no learning). If x is assigned a class label c_i in C^N, the current model M^+ is updated by updating the mean μ_i of the class c_i (covariance matrix Σ is not changed) as

$$\mu_i \leftarrow \frac{n_i \mu_i + z}{n_i + 1}, \tag{1}$$

where z is the feature $f(x)$ obtained from the pre-training model f and n_i is the number samples seen so far in class i.

This approach has two desirable properties.

(1) PLDA has no catastrophic forgetting (CF) during ALMD as we use a frozen pre-trained model (or feature extractor),[3] a fixed and shared covariance Σ, and a running mean for each class, which is independent of those of other classes. Thus, there is no interference across classes.

(2) Again, due to the sharing of covariance matrix Σ by all old and new classes, we achieve strong learning results with a small number of examples because, for each detected new class, PLDA only updates its mean based on the identified samples of the class.

3.3 OOD Detection Methods

Since PLDA produces a shared covariance Σ and one mean μ_i for each class i, we can naturally use Σ and μ_i related OOD detection methods, i.e., *Mahalanobis distance* (MD) and relative *Mahalanobis distance* (RMD). Each of these methods produces a confidence score using all classes $k \in C^A$ for the given feature vector $z = f(x)$, where x is the input. If the confidence score is below a *threshold* level, or it belongs to any class in C^E, that input is marked as OOD. Note that apart from these methods, there are numerous existing OOD detection methods (see Sec. 2). However, since our approach does not train a neural network, most existing methods are not suitable for use in PLDA. This is due to a few reasons. First, the number of our ID classes is not constant but keeps increasing, which means that the OOD detection model needs to be updated, causing CF. Second, since we cannot train all classes together in CL, those OOD detection methods that need to use logits are not applicable. Third, methods based on sample distances, e.g., KNN, are also inapplicable as we cannot save the past data in CL.

[3] By no means that using a pre-trained model without feature learning is a weakness. As we will see in the experiment section, the constantly advancing *pre-trained models* produce rich features for CL. The baselines that learn features produce poorer results even with replay data.

3.3.1 Mahalanobis distance (MD)

Mahalanobis distance [42] measures the distance between a data point (a feature vector in our case) and a normal distribution using the class mean vector μ and the covariance Σ, which is suitable for OOD detection [33]. Note that, each class mean μ_i and covariance Σ for the data used in building the initial model M are estimated as: $\mu_i = \frac{1}{N_i} \sum_{k:y_k=i} z_k$ and $\Sigma = \frac{1}{N} \sum_{i \in C} \sum_{k:y_k=i} (z_k - \mu_i)(z_k - \mu_i)^T$, where N denotes the number of samples, N_i denotes the number of samples of class i, and z_k is the feature of input sample x_k obtained from the pre-trained model, i.e., $z_k = f(x_k)$. Σ is the same for new classes, while μ_i for the new classes are incrementally computed using Eq. 1.

For $z = f(x)$ of a test sample x, we compute MD as,

$$MD_i(z; \mu_i, \Sigma) = \sqrt{(z - \mu_i)^T \Sigma^{-1} (z - \mu_i)} \tag{2}$$

where Σ^{-1} is the inverse of covariance matrix. The confidence score c is described as,

$$c(z) = \max_{i \in C^+} \{1/MD_i(z; \mu_i, \Sigma)\} \tag{3}$$

3.3.2 Relative Mahalanobis distance (RMD)

As noted in [50], MD has some limitations regarding the detection of OOD data and they proposed RMD by applying a simple addition to MD. It computes an additional mean $\mu_C = \frac{1}{N} \sum_{k=1}^{N} z_k$ and covariance $\Sigma_C = \frac{1}{N} \sum_{k=1}^{N} (z_k - \mu_C)(z_k - \mu_C)^T$, which, in our case, are only calculated based on the initial data with C classes used in building model M. RMD is computed as,

$$\begin{aligned} &RMD_i(z; \mu_i, \Sigma, \mu_C, \Sigma_C) = \\ &MD_i(z; \mu_i, \Sigma) - MD_A(z; \mu_C, \Sigma_C), \end{aligned} \tag{4}$$

where $MD_A(z; \mu_C, \Sigma_C) = \sqrt{(z - \mu_C)^T \Sigma_C^{-1} (z - \mu_C)}$. The confidence score is [50]

$$c(z) = \max_{i \in C^+} \{-RMD_i(z; \mu_i, \Sigma, \mu_C, \Sigma_C)\} \tag{5}$$

4 Experimental evaluation

We now evaluate the proposed method PLDA. We will see that PLDA can produce accuracy very close to those from joint fine-tuning, which learns all classes together as a single task in many epochs to reach the best classification accuracy. It is considered the upper bound of continual learning.

4.1 Datasets, compared methods, pre-trained model, and implementation

Datasets. We use three benchmark image classification datasets in our experiments.

(1) CIFAR-10 [30]: It contains 60,000 images, 50,000 training images and 10,000 testing images, distributed evenly across 10 classes.

(2) CIFAR-100 [30]: It contains 50,000 training images (500 per class) and 10,000 testing images (100 per class) of 100 classes.

(3) TinyImageNet [32]: It contains 200 classes, each with 500 training images. The validation set includes 50 images per class. Since the test data labels are unavailable, we use the validation set for testing.

– ID Class Set and **OOD Class Set**: For each experiment dataset, we divide the classes in the dataset into an equal number of *ID* (*in-distribution*) classes and *OOD* (*out-of-distribution*) classes. The **ID class set** is used to build the initial model M for deployment, while the **OOD class set** is used in incremental learning after model deployment.

– ID+OOD APP Set: We further divide the training set of each class in the ID class set into ***ID Train set*** and ***ID APP set***. We do the same for the OOD class set. ***ID+OOD APP set*** includes the data from both the ID and OOD APP sets. The ID+OOD APP set simulates the application (APP) data from a real-life data stream that needs to be classified.

For CIFAR-100 and TinyImageNet, each class has 500 samples in the original training set. After the split, the ID Train set has 450 samples and the ID+OOD APP set has 50 samples per class. CIFAR-10 has 5000 samples per class in its original training set, but to simulate the situation where the system does not ask the human user too many questions, we selected 4500 samples per class for the ID Train set and 50 samples per class for the ID+OOD APP set. We use a small number of samples in the ID+OOD APP set for each class to simulate the situation where in the application or after deployment, we don't see so many OOD samples. The rest of the OOD class data is not used in our experiment.

– Pre- and Post-Deployment: In pre-deployment, we perform joint training using LDA and a pre-trained model to build the model M using only the ID Train set. ID+OOD APP set is used only post-deployment.

Compared Methods. Although there are several related papers [5, 14, 51, 52], as discussed in Sec 2, no existing system can perform ALMD after model deployment as proposed in this paper.

Since our setting is closely related to online CL, we compare our method with 7 online CL baselines:

–**LwF** [34]: A regularization-based method that uses knowledge distillation to preserve the performance on previous tasks to deal with CF without storing old data.

–**iCaRL** [49]: A replay-based method that maintains a small exemplar memory and uses a nearest-mean-of-exemplars classifier.

–**AGEM** [7]: It mitigates forgetting by projecting gradients to avoid interference with stored memory samples.

–**ER** [8]: A replay-based method that replays a small buffer of past examples with random sampling alongside new data during training.

–**MIR** [2]: A replay-based method that prioritizes memory samples most vulnerable to forgetting for retrieval.

–**GDumb** [48]: A replay-based method that stores a balanced memory and retrains from scratch for evaluation.

–**GACL** [62]: A recent method that uses an analytic solution to avoid forgetting by achieving a weight-invariant property.

To ensure a fair comparison, we replaced their backbone model with ViT-B/14-DINOv2, and added adapters to prevent CF [25]. Note that these methods do not do OOD detection.

For each baseline, ID Train set is used to learn ID classes to build the initial model M as the first task, and OOD APP set is used to learn the OOD classes incrementally. However, since these baselines don't detect OOD samples, we assume that the baseline methods can do perfect OOD detection (thus ID APP set is not used). Even in this ideal case, these baseline methods perform poorer than PLDA.

Two upper-bound methods are also created, which are not continual learning methods.

– **Joint LDA:** This method applies LDA to learn a classifier using the data from all classes jointly, i.e., ID Train set and OOD APP set. ID APP set is not used as there is no post-deployment learning or OOD detection in this setting. This method gives the **upper bound results** of LDA.

– Joint Fine-tune: This method fine-tunes the pre-trained model using ID Train set and OOD APP set. We used the AdamW optimizer with a learning rate of 0.0001, CosineAnnealingLR scheduler, batch size of 128, and trained the model for 30 epochs, which is sufficient for convergence.

Ablations: We also create variations of the proposed method PLDA for ablation experiments.

– PLDA [X]: This creates three variations of PLDA, using different OOD detection methods (X), i.e., MD or RMD for OOD detection (see Section 3.3).

– PLDA [X](-C^E): This also creates two variations of PLDA ***without*** leveraging the un-converged means of the classes in C^E to help detect OOD samples. These will show that using C^E in OOD detection is helpful.

Pre-trained Models. For feature extraction, our primary pre-trained model is **ViT-B/14-DINOv2** [45], which is a self-supervised model with no class information leak. As another model we used, **DeiT-S/16-Kim** [24], based on DeiT-S/16 [55], was pre-trained using the labeled ImageNet-1k data. To prevent class information leaks between the pre-training and continual learning phases, 389 classes in ImageNet-1k similar to classes in CIFAR-10, CIFAR-100, and TinyImageNet, were excluded and pre-training was performed with the remaining 611 classes to produce DeiT-S/16-Kim. We also use another self-supervised model, **ViT-B/16-DINO** [6].

Implementation and Resource Usage. We used the LDA implementation in [16] and the **ViT-B/14-DINOv2** [45] pre-trained model. We run on a machine with an AMD EPYC 7502 32-Core Processor and NVIDIA RTX A6000 GPU. Each experiment requires approximately 6 GB of GPU memory and takes an average of 10 minutes.

OOD Detection and C^L Thresholds. In each setup, confidence scores for OOD detection are computed using two alternative methods, RMD and MD, and an image input is considered OOD if its confidence score is less than a certain threshold. These thresholds are set to 0.012 and 4.9 for RMD and MD, respectively. They are chosen empirically to ensure that the precision and recall for the OOD data are similar for each method. The threshold for C^L is empirically set to 30, which we will study later.

4.2 Two experiment setups and evaluations

Setup 1) Random ID+OOD APP Data Arrival. This is ***our main setup*** as it reflects real-life application scenarios. In this setup, after model deployment, the samples in the ID+OOD APP set arrive randomly in a data stream. The current model M^+ classifies each to its class or detects it as OOD. For each detected OOD sample (which may be correct or wrong), the proposed system PLDA asks the human user for its class label, and then PLDA incremental/continual learning is performed by updating its class mean. Clearly, in our experiment, ***no human user*** is involved. The system just uses the class label of the sample in the original data.

Setup 2) Class-Incremental OOD APP Data Arrival. This setup is for scenarios where an AI agent keeps going to new environments. This scenario may be rare. We use it to show the robustness of our approach. Each environment has a set of new OOD classes as a new task. This is similar to class incremental learning (CIL). In our case,

Table 1. Performance comparison of PLDA with baselines and ablation of PLDA using two different OOD detection methods (MD and RMD) on CIFAR-10, CIFAR-100, and TinyImageNet datasets in the *Random ID+OOD APP Data Arrival* setup (Setup 1). F-score gives the OOD detection performance. Note that baselines do not have F-score values as we assume their OOD detection is perfect (ideal case). Buffer size is the replay buffer size. C^L threshold is set to 30 for PLDA, higher values result in even higher accuracies and F-scores for the proposed method, with a cost of more number of asks.

		CIFAR-10		CIFAR-100		TinyImageNet		Average	
Methods	**Buffer Size**	**F-score**	**Accuracy**	**F-score**	**Accuracy**	**F-score**	**Accuracy**	**F-score**	**Accuracy**
Joint LDA			*97.03±0.00*		*85.73±0.00*		*81.89±0.00*		*88.22*
Joint Fine-tune			*93.22±0.16*		*89.74±0.13*		*85.90±0.10*		*89.62*
LwF	0		63.68±4.57		56.77±0.13		57.41±0.69		59.29
AGEM	5000		90.98±1.07		72.48±0.43		75.50±0.62		79.65
GACL	0		84.95±0.20		79.44±0.08		78.01±0.13		80.80
GDumb	1000		94.59±0.81		81.34±3.04		73.56±0.03		83.16
ER	5000		94.70±0.52		80.44±0.67		79.94±0.59		85.03
iCaRL	5000		95.48±0.73		81.06±0.09		79.23±0.59		85.26
MIR	5000		95.59±0.39		81.19±0.29		80.64±0.69		85.81
PLDA [MD]-C^E	0	75.20±0.81	96.61±0.19	73.88±0.15	83.96±0.06	76.86±0.07	81.68±0.11	75.31	87.42
PLDA [RMD]-C^E	0	75.50±0.18	96.53±0.10	74.48±0.14	84.21±0.09	78.17±0.22	81.49±0.08	76.05	87.41
PLDA [MD]	0	75.45±0.70	96.48±0.27	79.55±0.49	85.12±0.03	78.31±0.14	81.77±0.05	77.77	87.79
PLDA [RMD]	0	75.69±0.24	96.74±0.24	79.69±0.02	85.28±0.05	80.63±0.24	81.70±0.28	78.67	87.91

Table 2. Performance comparison of PLDA using different pre-trained models. All experiments use RMD as the OOD score method. **Joint** is the accuracy produced by joint fine-tuning the corresponding pre-trained model, considered the upper bound accuracy. Note that on CIFAR-10, due to highly imbalanced data, its Joint fine-tuning accuracy is low, but its Joint LDA accuracy is high (see Table 1).

	CIFAR-10			CIFAR-100			TinyImageNet			Average		
Pre-trained Models	**F-score**	**Accuracy**	**Joint**	**F-score**	**Accuracy**	**Joint**	**F-score**	**Accuracy**	**Joint**	**F-score**	**Accuracy**	**Joint**
DeiT-S/16-Kim	67.98±2.32	81.88±0.69	80.59±1.28	65.89±0.49	62.55±0.18	72.35±0.43	65.41±0.10	55.72±0.18	61.92±0.33	66.42	66.71	71.62
ViT-B/16-DINO	68.64±1.37	89.01±0.88	88.13±0.52	70.23±0.92	71.99±0.33	81.72±0.04	72.86±0.26	72.39±0.19	81.21±0.03	70.57	77.80	83.69
ViT-B/14-DINOv2	75.69±0.24	96.74±0.24	93.22±0.16	79.69±0.02	85.28±0.05	89.74±0.13	80.63±0.24	81.70±0.28	85.90±0.10	78.67	87.91	89.62

we divide the classes in the OOD class set evenly into 5 tasks (5 environments). Each task contains one-fifth of random samples from the ID APP set and all samples in the OOD APP set of the classes in the task. The OOD samples from one task finish before the data from the next task arrives. Acquiring labels of OOD samples and updating the class means for continual learning are the same as above.

4.3 Results: Comparison with baselines and ablations

Setup 1: Random ID+OOD APP Data Arrival:

Table 1 shows the OOD detection F-score, and accuracy of the combined ID and OOD classes after all data in the ID+OOD APP set are seen. PLDA variants (which use C^E) give significantly higher F-scores than their corresponding variants without using C^E. The final accuracy is also better. Note that the accuracy improvement is not large as the test results are obtained after the system sees all data, at which time the means of the OOD classes have mostly converged. We also observe that the baselines, which also do feature learning, are markedly poorer even in their ideal situation, i.e., OOD detection is perfect with no errors. The most recent online CL method GACL does poorly. Note that Joint fine-tuning is poor for CIFAR-10 due to highly imbalanced class distribution. 0 in the Buffer Size column means that the system does not store any replay data.

– ***Results of Different Pre-Trained Models.*** Table 2 shows that with the bigger model (DINOv2), the accuracy gets closer to the Joint fine-tuning upper bound. The OOD detection F-score also improves significantly. Our method performed even better than offline Joint fine-tuning for CIFAR-10, where the class imbalance is very high (4500 samples in each ID class in pre-deployment, but only 50 samples per class in each OOD class in post-deployment).

With bigger and more powerful pre-trained models appearing constantly, the results will improve further. There will be no need to learn new features or fine-tune the pre-trained model, which causes CF.

– ***Efficiency.*** Since PLDA uses the statistical methods LDA in learning and MD for OOD detection, which only needs to incrementally update the mean for each class and the shared covariance matrix, it is much more efficient than training in deep learning. In the post-deployment stage, PLDA performs only OOD detection and updating of the mean of each new class (no parameter training), which takes almost no time (less than 15 milliseconds), because the features are already learned in the pre-trained model. This makes PLDA especially suitable for ALMD in real time. Figure 3 shows the training and inference run time comparison between baselines and PLDA.

– C^L ***Threshold.*** The threshold *th* for C^L is set to 30 in the main experiments. We tested values between 10 and 50, and the results are shown in Figure 2. As the threshold (the number of mean updates or asks per class) increases, the total and the OOD class accuracy improves, but the ID class accuracy slightly drops (because more classes are considered in the classification). We chose 30 for all datasets as the threshold in our experiments because when the threshold reaches 30, the total accuracy stabilizes. It also achieves a good balance between accuracy and the number of asks (or mean updates). For example, by selecting 30 instead of 50 as the C^L threshold, we can reduce the number of asks by about 26.28%, losing only 0.36% in the total accuracy on average over the three datasets. The total accuracy covers both the ID classes and the OOD classes.

– **Shared Covariance Matrix.** As another ablation experiment, we explored the performance of using a separate covariance matrix for each OOD class, without sharing the covariance across classes. When tested on the CIFAR-100 dataset, we saw a drop in accuracy, from 85.28% to 82.90%. We hypothesize that this decrease in performance is due to the lack of sufficient samples for out-of-distribution (OOD) classes, making it challenging to construct accurate and robust covariance matrices for those classes. Our fixed shared covariance approach, on the other hand, works better because it combines information from a large number of training samples of multiple classes in the pre-deployment training, helping create a more reliable covariance estimate. This finding is supported by [16], which shows a robust performance with a fixed covariance computed from the ImageNet data. This further emphasizes the advantages of using fixed shared covariance, especially when dealing with class imbalance or limited data in the real world.

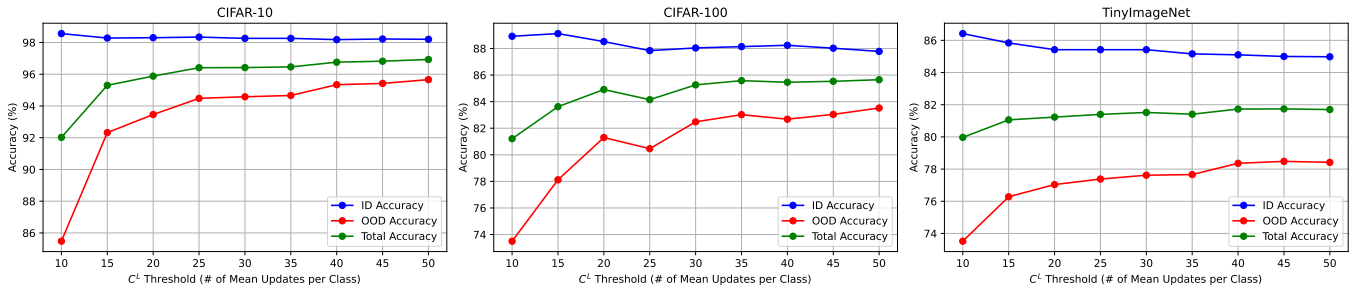

Figure 2. Accuracies of PLDA for different C^L threshold values for different datasets. While there is no CF, the ID accuracy drops as expected as more classes are learned, which makes the classification harder. The decrease in the ID accuracy is minimal relative to the significant gains in the OOD accuracy, demonstrating the model's resilience.

Table 3. Performance comparison of PLDA with baselines and ablation of PLDA using two different OOD detection methods (MD and RMD) on CIFAR-10, CIFAR-100, and TinyImageNet datasets in the *Class-Incremental OOD APP Data Arrival* setup (Setup 2). F-score gives the OOD detection performance. Baselines do not have F-score values as we assume their OOD detection is perfect. Buffer size is the replay buffer size.

		CIFAR-10		CIFAR-100		TinyImageNet		Average	
Method	**Buffer Size**	**F-score**	**Accuracy**	**F-score**	**Accuracy**	**F-score**	**Accuracy**	**F-score**	**Accuracy**
LwF	0		18.86±5.88		26.69±2.67		29.45±1.77		25.00
AGEM	5000		80.41±1.69		68.32±0.62		73.83±0.54		74.18
ER	5000		83.36±2.47		69.53±1.75		73.73±0.55		75.54
MIR	5000		87.91±2.96		71.16±0.62		74.56±1.27		77.88
GACL	0		84.82±0.13		79.27±0.11		77.96±0.15		80.68
iCaRL	5000		91.74±1.28		78.38±0.17		74.70±0.27		81.61
GDumb	1000		92.47±2.34		81.45±2.23		71.76±0.41		81.89
PLDA [MD]-C^E	0	74.76±0.47	96.40±0.14	72.79±0.10	83.72±0.11	75.88±0.07	81.47±0.22	74.47	87.20
PLDA [RMD]-C^E	0	76.31±0.92	96.71±0.12	73.77±0.17	84.33±0.18	76.70±0.09	80.91±0.08	75.59	87.32
PLDA [MD]	0	75.32±0.24	96.53±0.05	78.12±0.41	84.97±0.12	77.64±0.09	81.53±0.10	77.02	87.68
PLDA [RMD]	0	76.40±0.72	96.64±0.26	77.83±0.31	84.75±0.29	79.09±0.35	80.92±0.07	77.77	87.43

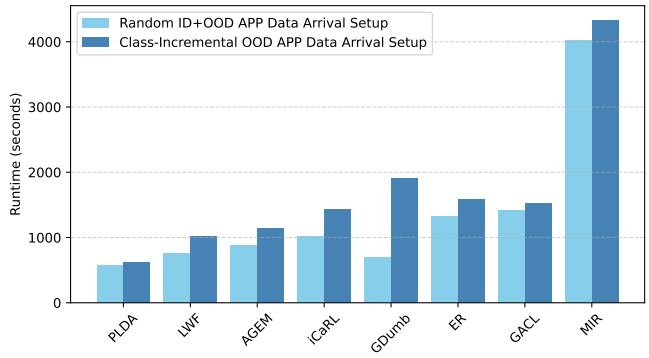

Figure 3. Run time comparison. PLDA achieves the best performance and the fastest run time among all methods across both setups.

Setup 2: Class-Incremental OOD APP Data Arrival: Table 3 shows the final classification accuracy and OOD detection F-score. PLDA shows a very similar performance to Setup 1, which shows the robustness of PLDA . The baselines are poorer because the tasks are incrementally arriving which causes more CF. In Setup 1, data from different classes arrive randomly, which is like arriving together with only one epoch of training.

5 Conclusion

This paper proposed the setting of *Learning After model Deployment* (ALMD). It enables the AI agent to learn based on its own experience in an autonomous manner. Although similar ideas like open world learning have been around for some time and some preliminary work has also been done, none of them have truly implemented learning after model deployment so that the AI agent can learn to classify more and more classes on the fly while working. We believe that ALMD is getting more and more important as more and more AI agents are deployed in real-life applications. It is highly desirable that these agents can learn continually after deployment by themselves to become more and more knowledgeable over time. This paper also proposed a method (called PLDA) based on LDA and a pre-trained model with several novel techniques to improve OOD detection in the ALMD process and to learn new classes easily by only updating class means, which has no catastrophic forgetting (CF) introduced in traditional continual learning due to network parameter updating in learning new classes or tasks. Experiment results have demonstrated the effectiveness of the proposed method PLDA.

One limitation of our work is that our OOD detection method may be effective, but it may not be state-of-the-art (SOTA). As explained in Sec. 3.3, it is difficult to use a current SOTA OOD detection method because they are unsuitable for continual OOD detection as they can cause serious catastrophic forgetting (CF) in our continual OOD detection setting. More work is needed to design more effective OOD detection methods for the continual learning context.

Ethics Statement

Our experiments used public domain benchmark datasets for image classification, which have been widely used in continual learning evaluations. They do not contain any unethical or offensive images. Our proposed system constructs a statistical model to categorize images into distinct classes. It does not generate any unethical content.

Acknowledgements

The work of Derda Kaymak and Bing Liu was supported in part by three NSF grants (IIS-2229876, IIS-1910424, and CNS-2225427), and an NVIDIA's Academia Grant, which provides cloud compute via its Saturn Cloud.

References

[1] H. Ahn, S. Cha, D. Lee, and T. Moon. Uncertainty-based continual learning with adaptive regularization. In *NeurIPS*, 2019.

[2] R. Aljundi, L. Caccia, E. Belilovsky, M. Caccia, M. Lin, L. Charlin, and T. Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019.

[3] R. Aljundi, M. Lin, B. Goujaud, and Y. Bengio. Gradient based sample selection for online continual learning. *NeurIPS*, 32, 2019.

[4] J. Bang, H. Koh, S. Park, H. Song, J.-W. Ha, and J. Choi. Online continual learning on a contaminated data stream with blurry task boundaries. In *CVPR*, pages 9275–9284, 2022.

[5] A. Bendale and T. Boult. Towards open world recognition. In *CVPR*, pages 1893–1902, 2015.

[6] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021.

[7] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019.

[8] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato. Continual learning with tiny episodic memories. *ICML*, 2019.

[9] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE TPAMI*, 44(7):3366–3385, 2021.

[10] G. Fei and B. Liu. Breaking the closed world assumption in text classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 506–514, 2016.

[11] G. Fei, S. Wang, and B. Liu. Learning cumulatively to become more knowledgeable. In *KDD-2026*, pages 1565–1574, 2016.

[12] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[13] N. Ghassemi and E. Fazl-Ersi. A comprehensive review of trends, applications and challenges in out-of-distribution detection. *arXiv preprint arXiv:2209.12935*, 2022.

[14] M. Gummadi, D. Kent, J. A. Mendez, and E. Eaton. Shels: Exclusive feature sets for novelty detection and continual learning without class boundaries. In *CoLLaS*, pages 1065–1085. PMLR, 2022.

[15] K. Han, S.-A. Rebuffi, S. Ehrhardt, A. Vedaldi, and A. Zisserman. Autonovel: Automatically discovering and learning novel visual categories. *IEEE TPAMI*, 44(10):6767–6781, 2021.

[16] T. L. Hayes and C. Kanan. Lifelong machine learning with deep streaming linear discriminant analysis. In *CVPR workshops*, 2020.

[17] J. He and F. Zhu. Out-of-distribution detection in unsupervised continual learning. In *CVPR*, pages 3850–3855, 2022.

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

[19] D. Hendrycks, M. Mazeika, and T. Dietterich. Deep anomaly detection with outlier exposure. *arXiv preprint arXiv:1812.04606*, 2018.

[20] R. Huang and Y. Li. Mos: Towards scaling out-of-distribution detection for large semantic space. In *CVPR*, pages 8710–8719, 2021.

[21] D. Kaymak, G. Kim, T. Kaichi, T. Konishi, and B. Liu. PLDA. https://github.com/drdkymk/PLDA, 2025. URL https://github.com/drdkymk/PLDA. GitHub repository.

[22] Z. Ke and B. Liu. Continual learning of natural language processing tasks: A survey. *arXiv preprint arXiv:2211.12701*, 2022.

[23] R. Kemker and C. Kanan. FearNet: Brain-Inspired Model for Incremental Learning. In *ICLR*, 2018.

[24] G. Kim, B. Liu, and Z. Ke. A multi-head model for continual learning via out-of-distribution replay. In *CoLLaS*, pages 548–563, 2022.

[25] G. Kim, C. Xiao, T. Konishi, Z. Ke, and B. Liu. A theoretical study on solving continual learning. *NeurIPS*, 35:5065–5079, 2022.

[26] G. Kim, C. Xiao, T. Konishi, Z. Ke, and B. Liu. Open-world continual learning: Unifying novelty detection and continual learning. *Artificial Intelligence*, 338:104237, 2025.

[27] J. Kim, J. Koo, and S. Hwang. A unified benchmark for the unknown detection capability of deep neural networks. *arXiv preprint arXiv:2112.00337*, 2021.

[28] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[29] H. Koh, D. Kim, J.-W. Ha, and J. Choi. Online continual learning on class incremental blurry task configuration with anytime inference. In *ICLR*, 2022.

[30] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[31] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *NeurIPS*, 2017.

[32] Y. Le and X. Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

[33] K. Lee, K. Lee, H. Lee, and J. Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *NeurIPS*, 2018.

[34] Z. Li and D. Hoiem. Learning Without Forgetting. In *ECCV*, 2016.

[35] S. Lin, L. Yang, D. Fan, and J. Zhang. Beyond not-forgetting: Continual learning with backward knowledge transfer. *NeurIPS*, 2022.

[36] B. Liu, S. Mazumder, E. Robertson, and S. Grigsby. Ai autonomy: Self-initiated open-world continual learning and adaptation. *AI Magazine*, 2023.

[37] X. Liu, Y. Lochman, and C. Zach. Gen: Pushing the limits of softmax-based out-of-distribution detection. In *CVPR*, 2023.

[38] Z. Mai, R. Li, H. Kim, and S. Sanner. Supervised contrastive replay: Revisiting the nearest class mean classifier in online class-incremental continual learning. In *CVPR Workshops*, pages 3589–3599, 2021.

[39] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51, 2022.

[40] A. Malinin, B. Mlodozeniec, and M. Gales. Ensemble distribution distillation. In *ICLR*, 2019.

[41] A. Mallya and S. Lazebnik. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. *arXiv preprint arXiv:1711.05769*, 2017.

[42] G. J. McLachlan. Mahalanobis distance. *Resonance*, 4(6):20–26, 1999.

[43] F. Mi, L. Kong, T. Lin, K. Yu, and B. Faltings. Generalized class incremental learning. In *CVPR workshops*, pages 240–241, 2020.

[44] S. Momeni, S. Mazumder, and B. Liu. Continual learning using a kernel-based method over foundation models. In *AAAI-2025*, volume 39, pages 19528–19536, 2025.

[45] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[46] S. Pang, S. Ozawa, and N. Kasabov. Incremental linear discriminant analysis for classification of data streams. *IEEE transactions on systems, man, and cybernetics*, 35:905–14, 11 2005.

[47] A.-A. Papadopoulos, M. R. Rajati, N. Shaikh, and J. Wang. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing*, 441:138–150, 2021.

[48] A. Prabhu, P. H. Torr, and P. K. Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *EECV*, 2020.

[49] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[50] J. Ren, S. Fort, J. Liu, A. G. Roy, S. Padhy, and B. Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021.

[51] A. Rios, N. Ahuja, I. Ndiour, U. Genc, L. Itti, and O. Tickoo. incdfm: Incremental deep feature modeling for continual novelty detection. In *ECCV 2022*, 2022.

[52] S. Roy, M. Liu, Z. Zhong, N. Sebe, and E. Ricci. Class-incremental novel class discovery. In *ECCV*, pages 317–333, 2022.

[53] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *ICML*, 2018.

[54] Y. Sun and Y. Li. Dice: Leveraging sparsification for out-of-distribution detection. In *ECCV*, 2022.

[55] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021.

[56] G. M. Van de Ven and A. S. Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[57] L. Wang, X. Zhang, H. Su, and J. Zhu. A comprehensive survey of continual learning: Theory, method and application, 2023.

[58] M. Wortsman, V. Ramanujan, R. Liu, A. Kembhavi, M. Rastegari, J. Yosinski, and A. Farhadi. Supermasks in superposition. *NeurIPS*, 2020.

[59] C. Wu, L. Herranz, X. Liu, J. van de Weijer, B. Raducanu, et al. Memory replay gans: Learning to generate new categories without forgetting. In *NIPS*, pages 5962–5972, 2018.

[60] J. Yang, K. Zhou, Y. Li, and Z. Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021.

[61] G. Zeng, Y. Chen, B. Cui, and S. Yu. Continuous learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 2019.

[62] H. Zhuang, Y. Chen, D. Fang, R. He, K. Tong, H. Wei, Z. Zeng, and C. Chen. GACL: Exemplar-free generalized analytic continual learning. In *NeurIPS*, 2024.