Gaussian Latent Representations for Uncertainty Estimation using Mahalanobis Distance in Deep Classifiers

Aishwarya Venkataramanan, Assia Benbihi, Martin Laviale, Cedric Pradalier

Gaussian Latent Representations for Uncertainty Estimation using Mahalanobis Distance in Deep Classifiers

Aishwarya Venkataramanan[1,2,3] Assia Benbihi[4] Martin Laviale[1,3] Cédric Pradalier[2,3]

[1]Université de Lorraine, CNRS, LIEC, Metz, France
[2]Georgia Tech Europe, GT-CNRS IRL 2958, Metz, France
[3]LTSER-"Zone Atelier Moselle", Metz, France
[4]Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague

Abstract

Recent works show that the data distribution in a network's latent space is useful for estimating classification uncertainty and detecting Out-Of-Distribution (OOD) samples. To obtain a well-regularized latent space that is conducive for uncertainty estimation, existing methods bring in significant changes to model architectures and training procedures. In this paper, we present a lightweight and high-performance regularization method for Mahalanobis distance (MD)-based uncertainty prediction, and that requires minimal changes to the network's architecture. To derive Gaussian latent representation favourable for MD calculation, we introduce a self-supervised representation learning method that separates in-class representations into multiple Gaussians. Classes with non-Gaussian representations are automatically identified and dynamically clustered into multiple new classes that are approximately Gaussian. Evaluation on standard OOD benchmarks shows that our method achieves state-of-the-art results on OOD detection and is very competitive on predictive probability calibration. Finally, we show the applicability of our method to a real-life computer vision use case on microorganism classification.

1. Introduction

Current deep learning classification networks achieve superior performance and find widespread applications in various industrial domains such as biology and robotics [16, 30, 41]. While they achieve state-of-the-art accuracy, there remain two main challenges that hinder the deployment of deep classifiers in critical situations: the derivation of calibrated classification and a measure of the classification uncertainty. Without those, a network exposed to Out-of-Distribution (OOD) data makes incorrect predictions with high confidence [17] and no human-in-the-loop can catch such errors. It is thus necessary to obtain calibrated probabilities [17] *i.e.,* predict probabilities that represent true likelihood, and to estimate the uncertainty in the network's predictions to allow users to make informed decisions.

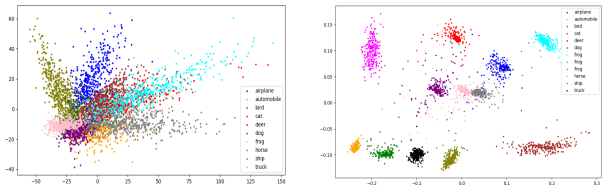

Figure 1. **Self-supervised latent space regularization with MAPLE** for uncertainty estimation and OOD detection. MAPLE improves class separation as illustrated by the PCA visualization of a CNN's latent space trained on CIFAR10 without regularization (left) and with MAPLE regularization (right). Our method constrains the latent representations to be approximately Gaussian to enable efficient distance-based uncertainty estimation.

Among deep uncertainty estimation approaches [1, 12, 14, 21] are Bayesian Neural Networks [2], MC-Dropout [13] and Deep Ensemble [26]. These stochastic methods require multiple forward-passes, so they are not scalable to large systems. Aware of the scalability requirements, current research focuses on estimating uncertainty from deterministic single-forward-pass networks [19,32,35, 38,42,45]. Distance-based methods belong to this category and are an attractive alternative for their excellent performance in OOD detection [29,46].

Distance-based methods rely on the distance between the test samples and the In-Distribution (ID) samples in a network's latent space to determine if the test samples

Corresponding author: aishwarya.venkataramanan@univ-lorraine.fr

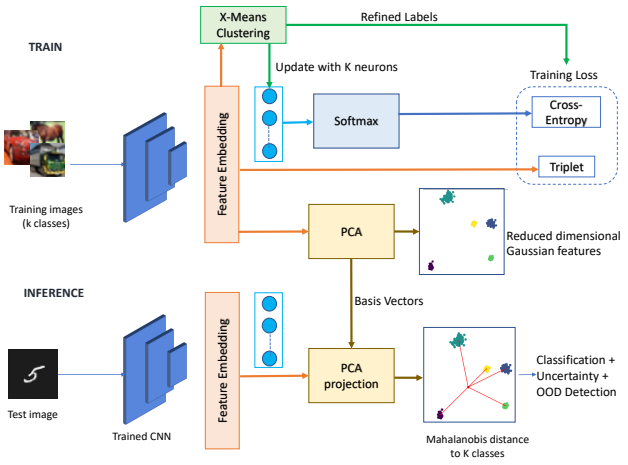

Figure 2. **Representation regularization with MAPLE for uncertainty estimation.** Our approach trains a classification network to learn representations that are approximately Gaussian for each class. During inference, the Mahalanobis distance between a test sample and the class centroids is used for classification, uncertainty estimation and OOD detection.

are OOD. A relevant distance is the Mahalanobis distance (MD) [34] for its superior performance over Euclidean Distance (ED) [24, 39, 48]. One key MD assumption though is that the in-distribution samples in the latent space should follow class-conditional Gaussian distributions. In practice, though, there is nothing in the classification training that constrains the latent space to fulfil such an assumption [7]. Instead, research on representation learning shows that each class is usually composed of several clusters of visually similar images [3, 9, 50]. This can be due to intra-class variance of images taken from different viewpoints, the presence of additional objects in the image, and variations in object shapes. In the network's latent space, these variations appear as distinct distributions or deviate from a Gaussian distribution. This breaks the MD assumption, which could lead to incorrect or imprecise uncertainty estimation. In this paper, we introduce MAPLE, a self-supervised representation learning method that regularizes a classification network's latent space to exhibit multivariate Gaussian distributions. MAPLE generates a latent space where class representations are Gaussian, making it compliant with the MD assumption and allows fast and high-performance MD-based OOD detection, uncertainty estimation, and calibrated classification. The effect of MAPLE is illustrated in Fig. 1 with the 2D projection of the latent space of a Convolutional Neural Network (CNN) trained on CIFAR10.

MAPLE stands for MAhalanobis distance based uncertainty Prediction for reLiablE classification, and is illustrated in Fig. 2. MAPLE relies on two components: i) a self-supervised intra-class label refinement through clustering in the latent space; ii) a deep metric learning loss that improves the class separation. During training, the representations associated to a class that deviate from a Gaussian distribution are divided into several clusters that are approximately Gaussian. The cluster assignments become the new labels of the representations, and the training goes on. Since each cluster gathers samples that exhibit similar intra-class variations, the clustering step is akin to automatic fine-grained annotation. The metric-learning then reinforces the fined-grained class separation by pushing apart the new classes. The combination of in-class clustering and metric learning results in classification representations that are well-clustered and approximately Gaussian, which makes them suitable for MD-based uncertainty estimation.

We evaluate MAPLE against existing uncertainty quantification methods on the following standard benchmarks: CIFAR10 [25] *vs.* SVHN [36]/CIFAR100 [25], CIFAR100 *vs.* CIFAR10/Tiny ImageNet [28], ImageNet [40] *vs.* ImageNet-O [20] for OOD detection and predictive probability calibration. Results show that MAPLE achieves the best compromise between performance and run time efficiency while being the most lightweight integration-wise. Also, it introduces minor architectural changes and does not require additional fine-tuning to OOD datasets.

We summarize the paper's contributions as follows. **i)** We develop a self-supervised representation learning method that constrains a classification network's latent space to be approximately Gaussian. **ii)** We show that such representations allow for reliable OOD detection and probability calibration using MD. **iii)** We design the method such that it has a minimal impact on the network's original architecture, and achieves results competitive with the state-of-the-art on OOD detection. The code is available in: https://github.com/vaishwarya96/MAPLE-uncertainty-estimation.git

2. Related Work

Multi-forward-pass Uncertainty Estimation. Traditional uncertainty quantification methods rely on Bayesian Neural Networks [15, 23] to learn a distribution over the network weights. To extract predictive probability variance, sampling [5] or variational methods [2] are used. The application of these methods is limited, as they increase the number of parameters by a factor of two and hinder convergence. As a lighter alternative, MC Dropout [13] enables dropout at test time and averages the network's output over several forward passes. While MC Dropout paves the way towards faster and lighter uncertainty estimation, it has been shown to produce over-confident predictions [26] and underestimate uncertainty [43]. To improve uncertainty estimation, Deep Ensembles [26] average the predictions from an ensemble of trained models and achieve state-of-the-art performance on several classification tasks. It remains com-

putationally expensive due to the training of multiple models and the several forward passes during inference. By deriving uncertainty from a single forward pass, MAPLE achieves significantly faster inference time without sacrificing performance.

Single-forward-pass Uncertainty Estimation. One line of work relies on the distribution of data samples in the network's latent space. A test sample is considered ID if it lies within the training data manifold, otherwise it is labelled as OOD. Methods differ in the way they regularize the representation space and the way they derive distances. DUQ [46] uses a Radial Basis Function (RBF) kernel in the representation space to measure distances between test samples and the centroids of various classes. Additionally, they use gradient penalty to obtain a regularized space, which improves the prediction's quality. SNGP [31] uses Spectral Normalization on the network's weights to satisfy the bi-Lipchitz condition, which is a more gradient-friendly regularization than DUQ. This condition preserves semantically meaningful distance changes in the representation space with respect to input changes. The prediction's uncertainty is then given by a Gaussian Process layer on the output. To improve the scalability of the Gaussian Process estimation, [45] proposes Deep Kernel Learning to process the input images with a distance-preserving network and fit a Gaussian on inducing points only. Contrary to these methods, MAPLE avoids the Gaussian Process estimation and gradient regularization during training and instead relies on simple metric learning. Similarly, [6] and VMDLS [7] simplifies the Gaussian enforcement by training the network with a Bregman divergence and KL-divergence loss respectively, so that each class representations follow an isotropic Gaussian distribution in the latent space for OOD detection. However, this ignores the possible intra-class variation within each class and requires the Gaussian variance to be tuned manually. Instead, MAPLE uses a simpler self-supervised clustering that automatically fits the data. Also, MAPLE makes the latent space not only suitable for OOD detection but also for calibrated probability prediction.

Mahalanobis-Distance for OOD detection. MD is a common distance in the OOD detection literature. Early work by Lee et al. [29] derives confidence values as a function of MD to predict the likelihood of a sample being ID. To obtain competitive performance, the method requires several tweaks such as adding noise to input samples, combining confidence values from multiple feature layers, and fine-tuning on OOD datasets. [24] proposes two light improvements: Partial MD and Marginal MD. In Partial MD, the MD is computed on lower dimensional representations with PCA. Marginal MD uses all training representations to fit a single Gaussian to calculate the MD. While both perform well on Far-OOD datasets *i.e.,* where ID and OOD samples are significantly distinct, their results are limited on Near-OOD [11], where the OOD samples are semantically similar to the ID ones. Relative MD (RMD) [39] improves the MD performance on Near-OOD by computing a global MD between the test sample and the samples of all classes combined, and then subtracting this value from the per-class MDs. All these methods exhibit satisfying performance, but their main limitation is their strong assumption that the image representations follow a Gaussian distribution, even though standard classification training does not enforce such a constraint. MAPLE addresses this limitation with a self-supervised regularization. By doing so, the features better fit the theoretical framework of MD-based OOD detection, thereby improving the performance.

3. Method

In this section, we describe MAPLE, a self-supervised regularization method for MD-based OOD detection, uncertainty estimation, and calibrated classification. It augments a standard CNN classifier with a self-supervised regularization to output both class probabilities and MD-based uncertainty. To enable MD for OOD detection, the representations of the training samples are dynamically clustered into multiple Gaussians using X-Means [37] during training. The samples are assigned new pseudo-class labels defined by their cluster assignment. The network is then optimized with the cross-entropy loss and the triplet loss. With periodic validation, the clusters are updated and the total number of classes change with every validation. At inference time, the MD between a test sample and each cluster's centroid is used to estimate the classification uncertainty and the probability of the point being OOD. Note that the only modification to the original network architecture is in the final layer, where the number of output neurons change according to the number of clusters identified. This makes MAPLE easy to integrate to any classification network. An algorithmic and computational description is provided in Appendix C.

3.1. Representation Regularization

This section describes the self-supervised automatic label refinement through clustering.

Notations. Consider a classification problem consisting of k classes with input samples $\boldsymbol{x}$ and labels $\boldsymbol{y} \in \{1, .., k\}$. The training dataset is denoted by $\boldsymbol{\mathcal{D}_{train}} = \{(x_n, y_n)\}_{n=1}^{N}$ and the validation dataset by $\boldsymbol{\mathcal{D}_{val}} = \{(x_m, y_m)\}_{m=N+1}^{M}$. Let the training input samples be represented as $\boldsymbol{x_{train}} = \{x_n\}_{n=1}^{N}$. The training is done with an off-the-shelf classification CNN. The penultimate layer of the CNN is used as a deep feature extractor $f^{\theta}(.)$, where $f^{\theta} : \mathbb{R}^D \rightarrow \mathbb{R}^d$ is a mapping from an input of dimension D to a representation (or feature) of dimension d, and θ is the model's parameters. The final layer consists of k neurons, followed by softmax activation to obtain the

Figure 3. **Visualizing intra-class label refinement and feature optimization.** The original data is not perfectly Gaussian due to intra-class variations. X-Means refines the labelling by dividing the samples into multiple clusters that are approximately Gaussian. The clusters are considered as separate classes during training. Triplet loss optimizes the representations by bringing the in-class samples together and separating them from other classes.

predictive probabilities. In addition to the standard cross-entropy loss used in CNNs, we use the triplet loss on the representations to train the network. See Appendix. B.6 for a recall of these standard losses.

Self-Supervised Dynamic Relabelling. During training, MAPLE updates the training labels to make them representative of the features' separation in the latent space. Every p epochs, the network is evaluated on $\boldsymbol{\mathcal{D}_{val}}$ and the classes with a false negative ratio higher than a threshold t are updated. This is representative of the scenarios where the samples of a given class are misclassified, which is typical of classes with high intra-class variations. For every class to update, the training representations belonging to such a class are extracted and clustered using X-Means [37]. The resulting clusters form well-separated groups and we use the cluster assigmnent as new pseudo-labels for the train samples. If k' additional clusters are introduced by X-Means, each of them are considered as independent classes. Thus, the number of classes becomes $K = k + k'$, and the final layer of the model is updated to have K neurons. Then, the network training continues with the new labels. During inference, the pseudo labels are remapped to the original set of k labels to identify their original class.

Fig. 3 illustrates the benefits of jointly using X-Means and the triplet loss on the representations: X-Means splits classes with high intra-class variations into separated classes that are semantically more representative of the data, and the triplet loss reinforces this separation.

The method introduces three hyperparameters: false negative ratio threshold t, frequency of validation epochs p and the maximum number of clusters (`max_num_cluster`), which is a parameter needed for X-Means. More details on the hyperparameters are provided in Appendix. B.5.

Clustering. The motivation for using X-Means over other commonly used clustering methods such as K-Means [33], DB-SCAN [10] and Gaussian Mixture Models (GMMs) are two-folds: (1) X-Means is scalable and automatically identifies the number of clusters based on the Bayesian Information Criterion (BIC); (2) BIC uses a maximum likelihood estimation of the variance under the spherical Gaussian assumption, which means that the samples are approximately spherical Gaussian in each cluster.

3.2. Representation Distance

This section describes the MD derivation over the latent representations. To avoid matrix singularities, the latent representations are first reduced using PCA.

Dimensionality reduction. Representations extracted from large neural networks usually have a high dimension and redundant dimensions. The MD requires calculating the inverse covariance matrix of these features, but the presence of redundancy causes the covariance matrix to be singular. Furthermore, [39] shows that the presence of non-informative dimensions could be detrimental to MD performance. This motivates the use of dimensionality reduction.

A common dimensionality reduction method is t-SNE [47], widely used for latent space's visualization. While t-SNE maintains the local distribution of points, it fails to represent global distributions accurately, which is undesirable in distance-based uncertainty predictions. Instead, we use Principal Component Analysis (PCA) for dimensionality reduction. The principal components are constructed from the covariance matrix of the standardized training representations. The eigen vectors of the covariance matrix are the principal components and the eigen values account for the amount of original information (variance) present in these components. We automatically estimate the number of principal components by the number of eigen values in decreasing order, required to explain 95% of the original data variance. This transformation is denoted by $g : \mathbb{R}^d \to \mathbb{R}^{d'}$, where d' is the dimension of the reduced features. With $\boldsymbol{x'_{train}} = f^{\theta}(\boldsymbol{x_{train}})$ the full dimensions training features, we denote $\boldsymbol{z_{train}} = g(\boldsymbol{x'_{train}})$ the reduced features.

Mahalanobis Distance. The MD is a generalized version of Euclidean distance that takes into account the data correlation to measure the distance. Hence, the MD is more accurate when predicting the distance between a point and a distribution of points. Here, MD is calculated on the PCA-reduced representations as follows. Let $\{z_i\}$ be the set of training representations after dimensionality reduction, μ_c be the class centroids with $c = 1, 2, ..., K$, and $\boldsymbol{\Sigma}$ be the shared covariance for all training samples, given by

$$\begin{aligned} \mu_c &= \frac{1}{N_c} \sum_{i:y_i=c} z_i \\ \boldsymbol{\Sigma} &= \frac{1}{N} \sum_{c} \sum_{i:y_i=K} (z_i - \mu_c)(z_i - \mu_c)^T \end{aligned} \tag{1}$$

The following Eq. 2 gives the Mahalanobis distance between the centroid μ_c of class c and a test sample $\tilde{x}$ with

reduced representation $\tilde{z} = g(f^{\theta}(\tilde{x}))$

$$MD_c(\tilde{x}) = \sqrt{(\tilde{z} - \mu_c)^T \mathbf{\Sigma}^{-1} (\tilde{z} - \mu_c)} \tag{2}$$

3.3. Classification and Uncertainty Estimation

We now show how to use the MD distance calculated in Eq. 2 for three purposes: classification, predictive probability, and uncertainty prediction.

MD-based Classification. The predicted class is the one whose centroid c^* is closest to the test sample $\tilde{x}$:

$$c^* = \underset{c}{\text{argmin}}(MD_c(\tilde{x})) \tag{3}$$

Note that this classification is inferred in addition to the usual classification done by the network by taking the maximum of the output logits.

Predictive Probability. We convert the MD into a calibrated classification probability using the following property: the squared MD on representations with dimension d' follows a chi-squared distribution $\chi^2_{d'}$ with d' degrees of freedom. The proof of this is provided in Appendix. **??**. The MD is converted as follows:

$$P^c_{MD} = 1 - \text{cdf}(\chi^2_{d'})(MD_c(\tilde{x})^2) \tag{4}$$

where cdf(.) is the cumulative distribution function. P^c_{MD} represents the probability that a test sample belongs to class c. When the test point belongs to a particular class, the MD to that class is low and the corresponding P^c_{MD} is high. The predictive probability is the one associated with the class c^* obtained in Eq. 3:

$$P^{c^*}_{MD} = \max_c(P^c_{MD}) \tag{5}$$

Note that contrary to a CNN softmax 'probabilities', this classification probability is calibrated and can be interpreted as a confidence in the classification output. This means P^c_{MD} represents the actual probability that a sample belongs the class c.

Uncertainty Prediction. We define the predictive uncertainty, which is the uncertainty in the network prediction as

$$u_{c^*} = 1 - P^{c^*}_{MD} \tag{6}$$

For small values of MD, u_{c^*} is around 0 and goes to 1 as the MD increases.

4. Experiments

We compare MAPLE with the following related works: two multi forward-pass methods MC-Dropout [13] (10 dropout samples) and Deep ensemble [26] (10 models), four single forward-pass methods: DUQ [46], SNGP [31], DUE [45] and VMDLS [7]. Following the standard evaluation on OOD detection, we evaluate the methods on classification, predictive probability calibration, and OOD detection on the following benchmark datasets: CIFAR10 [25] *vs.* SVHN [36]/CIFAR100 [25], CIFAR100 *vs.* CIFAR10/Tiny ImageNet [28] and ImageNet [40] *vs.* ImageNet-O [20].

Additionally, we compare the ID metrics for the corrupted version of CIFAR100 [20]. We also compare MAPLE with MD-based methods on OOD detection, namely, the approach by Lee et al. [29], Marginal MD [24] and RMD [39]. We used the near-OOD CIFAR10 *vs.* CIFAR100 for the comparison, which is notably challenging for OOD detection.

4.1. Evaluation Metrics

We report the standard evaluation metrics [31, 46] namely, the classification accuracy, the Expected Calibration Error (ECE), the Negative Log-Likelihood (NLL), the Area Under the Receiver Operating Characteristics (AUROC) and the Area Under the Precision-Recall curve (AUPR). For qualitative analysis, we use calibration plots. As mentioned previously, MAPLE produces two classification outputs, so we report the accuracies obtained from both the traditional softmax probability and the MD-based classification (Sec. 3.3). The ECE and the NLL are calculated from the predictive probability $P^{c^*}_{MD}$. AUROC and AUPR are calculated from the uncertainty u_{c^*}. The definition of these standard metrics are recalled in Appendix A.

4.2. Implementation Details

The CIFAR10 and CIFAR100 training follows [31, 45] and uses a Wide ResNet 28-10 [51] for the classification backbone. The hyperparameters for the trainings are $p = 10, t = 0.3$ and `max_num_cluster`$= 5$. The ImageNet training is performed on ResNet-50 [18]. The hyperparameters are $p = 20, t = 0.2$ and `max_num_cluster`$= 5$. Additional details on the dataset splits, hyperparameters, and the hardware used for training are provided in the Appendix B.

4.3. Results

We report the results on CIFAR10, CIFAR100 and ImageNet in Table (Tab.) 1, 2 and 3 respectively.

OOD Detection Results. MAPLE outperforms the baseline methods by upto 12% on OOD detection. Note that competitive approaches, such as SNGP and DUE, derive their performance from spectral normalization and Gaussian process layer, which are invasive training add-ons. In contrast, MAPLE relies only on the layers of a standard CNN architecture to achieve superior performance.

Classification Results. MAPLE achieves results competitive to state-of-the-art, only 1% below the top method Deep ensemble [26] whose score comes at the cost of

Method	ID metrics			OOD AUROC ↑		OOD AUPR ↑		Latency↓
	Accuracy ↑	ECE ↓	NLL ↓	SVHN	CIFAR100	SVHN	CIFAR100	(ms/sample)
Deterministic	95.0±0.01	0.094 ±0.002	0.138±0.01	0.801±0.01	0.765±0.01	0.794±0.01	0.762±0.01	**4.01**
MC Dropout [13]	96.0±0.01	0.048±0.001	0.293±0.01	0.932±0.01	0.835±0.01	0.965±0.01	0.829±0.01	27.10
Deep Ensemble [26]	**96.4±0.01**	0.014±0.001	**0.134±0.01**	0.934±0.01	0.864±0.01	0.935±0.01	0.885±0.01	38.10
DUQ [46]	94.5±0.02	0.023±0.001	0.222±0.01	0.927±0.01	0.872±0.01	0.973±0.01	0.833±0.01	8.68
SNGP [31]	95.7±0.01	0.016±0.001	0.153±0.01	0.991±0.01	0.911±0.01	0.994±0.01	0.907±0.01	6.25
DUE [45]	95.6±0.02	0.015±0.001	0.179±0.01	0.936±0.01	0.852±0.01	0.967±0.01	0.850±0.01	6.94
VMDLS [7]	95.1±0.01	-	-	0.932±0.01	0.868±0.01	0.953±0.01	0.864±0.01	5.61
MAPLE	95.6±0.01/95.4±0.01	**0.012±0.001**	0.142±0.01	**0.996±0.01**	**0.926±0.01**	**0.997±0.01**	**0.918±0.01**	4.96

Table 1. **CIFAR10 (ID) vs SVHN/CIFAR100 (OOD).** Results are averaged over 10 seeds. MAPLE outperforms all single and multi pass methods on OOD detection, and is significantly faster. Classification with MAPLE is very competitive with the state-of-the-art and the predicted probabilities are better calibrated. Blue: classification based on prediction from softmax probability. Orange: MD-based classification.

Method	Accuracy↑		ECE↓		NLL↓		OOD AUROC↑		OOD AUPR ↑	
	Clean	Corrupted	Clean	Corrupted	Clean	Corrupted	CIFAR10	TinyImageNet	CIFAR10	Tiny ImageNet
Deterministic	79.0±0.02	52.2±0.03	0.108±0.012	0.279±0.003	1.342±0.03	2.834±0.03	0.697±0.01	0.748±0.01	0.713±0.01	0.747±0.01
MC Dropout [13]	79.4±0.02	46.3±0.05	0.115±0.010	0.293±0.004	0.986±0.02	2.868±0.02	0.786±0.01	0.787±0.01	0.781±0.01	0.790±0.01
Deep ensemble [26]	**79.6±0.01**	54.0±0.06	**0.029±0.008**	0.254±0.005	**0.706±0.01**	2.893±0.02	**0.798±0.01**	0.811±0.01	0.792±0.01	0.801±0.01
DUQ [46]	77.6±0.02	50.5±0.04	0.112±0.015	0.277±0.006	1.303±0.03	2.811±0.02	0.740±0.01	0.759±0.01	0.747±0.01	0.761±0.01
SNGP [31]	78.7±0.01	50.5±0.03	0.129±0.012	0.286±0.003	1.080±0.01	**2.676±0.02**	0.743±0.01	0.783±0.01	0.749±0.01	0.765±0.01
DUE [45]	77.8±0.02	49.3±0.05	0.134±0.014	0.305±0.005	1.454±0.02	2.756±0.03	0.732±0.01	0.754±0.01	0.734±0.01	0.768±0.01
MAPLE	78.9±0.02/78.6±0.01	**54.2±0.04**/54.0±0.03	0.065±0.001	**0.245±0.004**	1.112±0.01	2.715±0.02	0.793±0.01	**0.828±0.01**	**0.799±0.01**	**0.817±0.01**

Table 2. **CIFAR100 (ID) vs CIFAR10/Tiny ImageNet (OOD).** Results are averaged over 10 seeds. We also evaluate the ID metrics for the corrupted version of CIFAR100 from [20]. MAPLE achieves the best performance on OOD detection. It is very competitive with other single-pass methods on the classification task. Blue: Classification based on prediction from softmax probability Orange: MD-based classification.

Method	ID metrics			OOD metrics		Latency ↓
	Accuracy ↑	ECE ↓	NLL ↓	AUROC ↑	AUPR ↑	(ms/sample)
Deterministic	75.7±0.03	0.058±0.004	0.925±0.03	0.553±0.01	0.546±0.01	**42.64**
MC Dropout [13]	75.3±0.04	0.032±0.003	**0.922±0.02**	0.614±0.08	0.609±0.05	93.74
Deep ensemble (3 models) [26]	**76.4±0.08**	0.024±0.004	0.930±0.03	0.625±0.05	0.616±0.04	130.78
MAPLE	75.6±0.05/75.2±0.07	**0.021±0.003**	0.928±0.03	**0.637±0.06**	**0.635±0.04**	55.26

Table 3. **ImageNet (ID) vs ImageNet-O (OOD).** Results are averaged over 10 seeds. MC-Dropout is performed for 10 forward passes. MAPLE achieves the best performance on OOD detection. It is very competitive with other single-pass methods on the classification task. Blue: Classification based on prediction from softmax probability Orange: MD-based classification.

training and inference on several models. Note that both MAPLE accuracies, the softmax probability and the MD-based one are close. A finer analysis of the accuracy shows that the slight difference in accuracy with the MD-based classification occurs on samples the network is uncertain about: MAPLE achieves top accuracy on high-confidence predictions (above 80% and 90% confidence) and the accuracy slightly decreases for lower-confidence predictions. See Appendix. D.1 for an extended analysis.

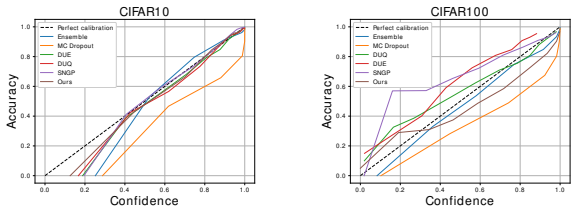

Figure 4. **Calibration plots.** A perfectly calibrated plot is when the predicted confidence equals the true likelihood *i.e.,* the accuracy. This is shown by the linear dotted line in the plots.

Calibration Results. MAPLE is competitive with state-of-the-art SNGP [31] and Deep Ensembles. The calibration plot for CIFAR10 and CIFAR100 and shown in Fig. 4. On CIFAR10, all methods are well-calibrated, except for MC-Dropout that is overconfident in its predictions, which explains its high ECE score. When the accuracy is below 0.4, baseline methods become overconfident, whereas MAPLE is closer to optimal calibration and achieves the best ECE score.

Additional results on Gaussian test, are provided in Appendix. D.

4.4. Comparison with other MD methods

Setup. MAPLE is compared against MD-based OOD detectors [24, 29, 39]. These methods are tailored for OOD detection, so we report the metric relevant to this task only for the sake of fairness. We report the AUROC score on the challenging near-OOD dataset CIFAR10 *vs.* CIFAR100. The experiments are done with a Wide ResNet 28-10 [51].

Method	Lee et al. [29]	Marginal MD [24]	RMD [39]	Ours
AUROC ↑	0.893	0.838	0.897	**0.926**

Table 4. **Comparison with MD-based OOD detection.** MAPLE performs better in OOD detection than existing MD-based methods on the CIFAR10 *vs.* CIFAR100 setup.

OOD Detection Results. MAPLE achieves top-performance on Near-OOD detection (Tab. 4), which supports MAPLE's representation regularization. Note that the primary difference between MAPLE and the baselines is their lack of constraints on the latent representation. In contrast, we force the samples of every class to be Gaussian be-

Method	ID metrics			OOD metrics - SVHN		OOD metrics—CIFAR100		#Eig
	Softmax Accuracy ↑	MD-based Accuracy↑	ECE ↓	AUROC ↑	AUPR ↑	AUROC ↑	AUPR ↑	
DNN+MD (1)	0.950	0.943	0.086	0.752	0.762	0.583	0.564	-
DNN+PCA+MD (2)	0.950	0.946	0.053	0.855	0.839	0.813	0.859	12
DNN+PCA+ED (3)	0.950	0.943	0.105	0.829	0.804	0.734	0.765	12
DNN+Triplet+PCA+MD (4)	0.954	0.953	0.013	0.945	0.948	0.912	0.894	11
DNN+Clustering+PCA+MD (5)	0.947	0.945	0.032	0.922	0.908	0.811	0.815	12
MAPLE (6)	**0.956**	**0.954**	**0.012**	**0.996**	**0.997**	**0.926**	**0.930**	12

Table 5. **Ablation study.** We evaluate the influence of several MAPLE components. **PCA** (1 vs 2) results in a significant improvement of the OOD detection by discarding non-informative dimensions. The distances derived on these reduced features are better representative of the similarity between the input samples. The **MD** (2 vs 3) is better suited than ED for calibrated classification and OOD detection, which reiterates conclusions already found in previous works. The **triplet loss** (2 vs 4) improves both the accuracy and the OOD metrics by increasing the class separation. **Clustering** alone (2 vs 5) also contributes to a better separation of the classes, but the results are not as significant. The joint use of **triplet loss and clustering**, as done in MAPLE (6) achieves the best results on both classification and OOD detection. Note: #Eig refers to the number of principal components, whenever applicable.

fore calculating MD. Non-Gaussian samples lead to incorrect mean and covariance calculations, resulting in incorrect distance values. The error is more pronounced when the samples deviate from the Gaussian distribution by a large factor. This explains why the MD-based approaches underperform compared to MAPLE on Near-OOD.

4.5. Ablation analysis

In this study, we assess how the different components of MAPLE impact its performance. We train a Wide ResNet 28-10 [51] network on CIFAR10 and use SVHN and CIFAR100 as OOD datasets.

Dimensionality Reduction. We consider two scenarios: **(1) DNN+MD** - A baseline where a standard Deep Neural Network (DNN) is trained with the cross-entropy loss and with no feature regularization. The MD is computed on the raw features, and we add a value of $1e^{-20}$ to the diagonal elements [39] to avoid a singular covariance matrix. **(2) DNN+PCA+MD** - It follows (1) except that the MD is derived on PCA-reduced features.

Results: Dimensionality reduction (2) drastically improves the network's performance, as shown in the first line of Tab. 5. The improvement amounts to 7-30% on the OOD metrics and 3% on the ID metrics. One possible explanation is that the reduced dimensions are the ones that contribute to distinguishing ID samples from OOD ones, as previously observed by [39]. When including all the feature dimensions in the MD, the dimensions that do not contribute to discriminating ID and OOD samples add up and dominate the final MD score.

Distance Definition. We compare Mahalanobis distance and Euclidean distance (ED) in the network's latent space. We compare **(2) DNN+PCA+MD** with the new experiment **(3) DNN+PCA+ED** - It follows (2) except that the MD is replaced with ED. As for MD, the $\chi^2_{d'}$ distribution is used to obtain the probability values from ED (Sec 3.3).

Results: The results show that MD boosts the performance in terms of ID and OOD metrics. The improvement is ECE score is by 5%, and the OOD metrics improved by 3-9% when using MD. This is because MD takes into account the data correlation, which gives a better estimate of the probability and uncertainty values.

Representation training. To study the influence of the training on the representations, we consider three experiments: **(4) DNN+Triplet+PCA+MD** - We train the DNN using both cross-entropy and triplet loss. **(5) DNN+Clustering+PCA+MD** - We train using the cross-entropy loss only and periodically cluster the feature points using X-Means. **(6) MAPLE** - This is our proposed method that fuses (4) and (5). For all experiments, the MD is derived on the reduced features.

Results: Using the triplet loss (4) improves the performance considerably compared to training with the cross-entropy loss only (2). An explanation is that the triplet loss pulls in-class feature embeddings together, and pushes the other class features apart. This encourages the representations to be well separated and makes it easier to distinguish OOD features. Choosing the triplet loss for metric learning is empirically motivated: experiments using contrastive loss showed that triplet loss has a slightly better performance.

Periodic clustering (5) improves the ECE score by 2%, and the AUROC and AUPR scores on SVHN by about 7% compared to (2). However, there is a slight drop in accuracy by 0.3% and OOD metric by 4% on CIFAR100. One explanation is that clustering increases the chances of new classes to overlap. This phenomenon is illustrated in the centre plot of Fig. 3. The class overlap is particularly hindering when the new domain is close to the training one: with clustering (5), the SHVN scores are better but the near-OOD CIFAR100 performs better without clustering (2).

MAPLE uses clustering together with triplet loss and achieves top-performance. The triplet loss reduces the overlap introduced with the clustering by pulling apart the newly created classes. With MAPLE, the latent representations are approximately Gaussian and well-clustered resulting in better MD estimates and superior performance in both ID and OOD metrics. Compared to experiment (2), the calibration error drops by 4%and the OOD scores improved by 4-11%.

False Negative Ratio (t)	#Classes	Accuracy↑	ECE↓	SVHN AUROC↑	CIFAR100 AUROC↑
0.0	23	0.9449	0.014	0.922	0.888
0.1	18	0.9534	0.013	0.964	0.918
0.2	14	**0.9544**	0.012	0.991	0.925
0.3	12	0.9541	**0.012**	**0.996**	**0.926**
0.4	10	0.9535	0.013	0.961	0.921
0.5	10	0.9535	0.012	0.955	0.915

Table 6. **Metrics for different values of False Negative Ratio evaluated on CIFAR10** #Classes refers to the total number of output classes obtained after clustering. A low value of t results in overclustering, whereas a high t fails to detect classes with high variance.

False Negative Ratio t. We evaluate the influence of the clustering trigger *i.e.,* the False Negatives Ratio. We train MAPLE with a range of t values on CIFAR10 (Tab. 6).

Results: A low value of t results in overclustering, where multiple clusters contain similar images. This further increases the chances of misclassifications, leading to decrease in the metric values. On the other hand, high t values result in underclustering. Note that for $t > 0.3$, there are no additional clusters generated. This is because, the classes have false negative ratios that are below this threshold and so, they are not clustered. For CIFAR10, a t value of 0.3 yields the best results.

An extended ablation analysis on the influence of classification backbones, clustering methods, and hyperparameters is provided in Appendix. E.

5. Discussion

With the periodical clustering and the dynamic relabeling, a natural question that arises is *'Is there a drop in performance when the ground truth labels change during training?'*. Experimentally, we observe a drop in training accuracy by 2-3% in the following epoch after every clustering phase. However, the network makes up for the drop within 4-5 epochs of training.

It can happen that the clusters contain very few samples, which introduces label imbalance when classifying. This is exacerbated when the samples are over-clustered. To mitigate this, we restrict X-Means to only cluster the classes that get misclassified. These are the classes with a false negative ratio higher than the threshold t. Automatic clustering regularization [4, 22, 27] is left for future work.

6. Use Case: Microorganism Classification

We consider the real-life computer vision use-case of image-based diatom identification [8]. Diatoms are microorganisms present in the water. The distribution of diatoms in the water is a useful indicator for predicting the water quality. Diatoms consist of several species or 'taxa', each corresponding to a different class with a different appearance. Typical in several biology applications, the image dataset includes a lot of intra-class variance (Fig. 5). In this study, we evaluate the performance of different approaches when encountering taxa that were not previously trained on.

Method	Accuracy ↑	ECE ↓	AUROC ↑	AUPR ↑	Latency (ms/sample)↓
MC-Dropout [13]	0.936	0.039	0.548	0.589	129.7
Deep Ensemble [26]	**0.969**	**0.025**	0.589	0.570	146.81
SNGP [31]	0.954	0.196	0.798	0.826	26.25
MAPLE	0.963	0.036	**0.864**	**0.865**	**17.38**

Table 7. **Real Case Application: microorganism classification.** With its top performance and state-of-the-art speed, MAPLE makes for a particularly applicable method for classification and OOD detection on real case datasets.

We train a Wide ResNet 28-10 on 130 taxa and use 36 taxa as OOD. The dataset is particularly challenging since it is fine-grained and Near-OOD. Additional details on the dataset and experimental setup are provided in Appendix B.4. As shown in Tab. 7, MAPLE outperforms all baselines on OOD detection. While Deep Ensemble has a slightly better classification accuracy and ECE score, MAPLE significantly outperforms it in OOD with a 30% score boost and a runtime 8 times faster.

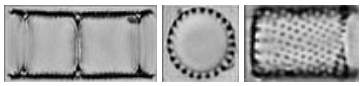

Figure 5. **Micro-organisms belonging to the same class.** These images of **one** diatom class show wide appearance changes due to different viewpoints during the acquisition. These translate into separate distributions in the latent space, deviating from Gaussian distribution. MAPLE's regularization makes the latent space Gaussian, hence suitable for MD calculation.

7. Conclusion

This paper presents MAPLE, a self-supervised regularization method for uncertainty estimation and out-of-distribution detection on CNN classifiers. The uncertainty is derived from the Mahalanobis Distance (MD) between an image representation and the class representations in the network's latent space. MAPLE derives meaningful MD distances by introducing a regularizer based on self-supervised label refinement and metric learning. Thus, MAPLE learns well-clustered representations that are approximately Gaussian for each class, which complies with the theoretical requirements of MD-based uncertainty estimation. Experimental results show that MAPLE achieves state-of-the-art results on out-of-distribution detection and is very competitive with existing methods on predictive probability calibration. MAPLE also has the significant advantage of introducing the least architectural changes. Finally, we demonstrate a real-life use-case of our method on microorganism classification for the automatic assessment of water quality in natural ecosystems.

References

[1] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76:243–297, 2021. 1

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR, 2015. 1, 2

[3] Simon Carbonnelle and Christophe De Vleeschouwer. Intra-class clustering: An implicit learning ability that regularizes dnns. In *International Conference on Learning Representations*, 2020. 2

[4] Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pages 132–149, 2018. 8

[5] Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691. PMLR, 2014. 2

[6] Jiacheng Cheng and Nuno Vasconcelos. Learning deep classifiers consistent with fine-grained novelty detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1664–1673, 2021. 3

[7] Or Dinari and Oren Freifeld. Variational-and metric-based deep latent space for out-of-distribution detection. In *The 38th Conference on Uncertainty in Artificial Intelligence*, 2022. 2, 3, 5, 6

[8] Hans Du Buf, Micha Bayer, Stephen Droop, Ritchie Head, Steve Juggins, Stefan Fischer, Horst Bunke, Michael Wilkinson, Jos Roerdink, José Pech-Pacheco, et al. Diatom identification: a double challenge called adiac. In *Proceedings 10th International Conference on Image Analysis and Processing*, pages 734–739. IEEE, 1999. 8

[9] Yan Em, Feng Gag, Yihang Lou, Shiqi Wang, Tiejun Huang, and Ling-Yu Duan. Incorporating intra-class variance to fine-grained visual recognition. In *2017 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1452–1457. IEEE, 2017. 2

[10] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996. 4

[11] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. Exploring the limits of out-of-distribution detection. *Advances in Neural Information Processing Systems*, 34:7068–7081, 2021. 3

[12] Yarin Gal et al. Uncertainty in deep learning. 2016. 1

[13] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016. 1, 2, 5, 6, 8, 13

[14] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021. 1

[15] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. In *Case Studies in Applied Bayesian Data Science*, pages 45–87. Springer, 2020. 2

[16] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020. 1

[17] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017. 1

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 11, 14

[19] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016. 1, 11

[20] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 2, 5, 6, 11

[21] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021. 1

[22] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 8

[23] Laurent Valentin Jospin, Hamid Laga, Farid Boussaid, Wray Buntine, and Mohammed Bennamoun. Hands-on bayesian neural networks—a tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2):29–48, 2022. 2

[24] Ryo Kamoi and Kei Kobayashi. Why is the mahalanobis distance effective for anomaly detection? *arXiv preprint arXiv:2003.00402*, 2020. 2, 3, 5, 6

[25] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 2, 5, 11, 13

[26] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. 1, 2, 5, 6, 8, 13

[27] Mans Larsson, Erik Stenborg, Carl Toft, Lars Hammarstrand, Torsten Sattler, and Fredrik Kahl. Fine-grained segmentation networks: Self-supervised segmentation for improved long-term visual localization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 31–41, 2019. 8

[28] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015. 2, 5

[29] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018. 1, 3, 5, 6

[30] Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen Awm Van Der Laak, Bram Van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017. 1

[31] Jeremiah Liu, Zi Lin, Shreyas Padhy, Dustin Tran, Tania Bedrax Weiss, and Balaji Lakshminarayanan. Simple and principled uncertainty estimation with deterministic deep learning via distance awareness. *Advances in Neural Information Processing Systems*, 33:7498–7512, 2020. 3, 5, 6, 8, 13

[32] Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020. 1

[33] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982. 4

[34] Prasanta Chandra Mahalanobis. On test and measures of group divergence. *Journal of Asiatic Society of Bengal*, 26:541–588, 1930. 2

[35] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. *Advances in neural information processing systems*, 31, 2018. 1

[36] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011. 2, 5

[37] Dan Pelleg, Andrew W Moore, et al. X-means: Extending k-means with efficient estimation of the number of clusters. In *Icml*, volume 1, pages 727–734, 2000. 3, 4, 14

[38] Janis Postels, Hermann Blum, Cesar Cadena, Roland Siegwart, Luc Van Gool, and Federico Tombari. Quantifying aleatoric and epistemic uncertainty using density estimation in latent space. *arXiv preprint arXiv:2012.03082*, 2020. 1

[39] Jie Ren, Stanislav Fort, Jeremiah Liu, Abhijit Guha Roy, Shreyas Padhy, and Balaji Lakshminarayanan. A simple fix to mahalanobis distance for improving near-ood detection. *arXiv preprint arXiv:2106.09022*, 2021. 2, 3, 4, 5, 6, 7

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 5, 11

[41] Andrew W Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander WR Nelson, Alex Bridgland, et al. Improved protein structure prediction using potentials from deep learning. *Nature*, 577(7792):706–710, 2020. 1

[42] Murat Sensoy, Lance Kaplan, and Melih Kandemir. Evidential deep learning to quantify classification uncertainty. *Advances in neural information processing systems*, 31, 2018. 1

[43] Lewis Smith and Yarin Gal. Understanding measures of uncertainty for adversarial example detection. *arXiv preprint arXiv:1803.08533*, 2018. 2

[44] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 14

[45] Joost van Amersfoort, Lewis Smith, Andrew Jesson, Oscar Key, and Yarin Gal. On feature collapse and deep kernel learning for single forward pass uncertainty. *arXiv preprint arXiv:2102.11409*, 2021. 1, 3, 5, 6, 13

[46] Joost Van Amersfoort, Lewis Smith, Yee Whye Teh, and Yarin Gal. Uncertainty estimation using a single deep deterministic neural network. In *International conference on machine learning*, pages 9690–9700. PMLR, 2020. 1, 3, 5, 6, 13

[47] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 4

[48] Garnik Vareldzhan, Kirill Yurkov, and Konstantin Ushenin. Anomaly detection in image datasets using convolutional neural networks, center loss, and mahalanobis distance. In *2021 Ural Symposium on Biomedical Engineering, Radioelectronics and Information Technology (USBEREIT)*, pages 0387–0390. IEEE, 2021. 2

[49] Aishwarya Venkataramanan, Pierre Faure-Giovagnoli, Cyril Regan, David Heudre, Cécile Figus, Philippe Usseglio-Polatera, Cedric Pradalier, and Martin Laviale. Usefulness of synthetic datasets for diatom automatic detection using a deep-learning approach. *Engineering Applications of Artificial Intelligence*, 117:105594, 2023. 12

[50] Aishwarya Venkataramanan, Martin Laviale, Cécile Figus, Philippe Usseglio-Polatera, and Cédric Pradalier. Tackling inter-class similarity and intra-class variance for microscopic image-based classification. In *International Conference on Computer Vision Systems*, pages 93–103. Springer, 2021. 2

[51] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5, 6, 7, 11, 12, 14

[52] Zhonghua Zhao, Shanqing Guo, Qiuliang Xu, and Tao Ban. G-means: a clustering algorithm for intrusion detection. In *Advances in Neuro-Information Processing: 15th International Conference, ICONIP 2008, Auckland, New Zealand, November 25-28, 2008, Revised Selected Papers, Part I 15*, pages 563–570. Springer, 2009. 14

A. Metrics Definitions

In this section, we provide the definitions and formulas of metrics used for evaluation in this paper. Let the samples be represented by $[(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)]$, where N is the total number of samples. x_i is the input and y_i is the corresponding label, having values between 1 and K.

Accuracy. This gives the fraction of samples that were correctly identified by the network.

$$acc = \frac{1}{N}\sum_{n=1}^{N} 1[\text{argmax}(p(y_n|x_n)) = y_n]$$

where, $p(y_n|x_n)$ is the predicted probability that the sample x_n belongs to the class y_n. A higher accuracy indicates better performance.

Expected Calibration Error. ECE is a measure of predictive probability calibration error. The output probability is divided into a histogram of B equally spaced bins. The expected calibration error gives the difference between the *observed relative frequency* (accuracy) and the *average predicted frequency* (confidence).

$$ECE = \sum_{b=1}^{B} \frac{n_b}{N} |acc(b) - conf(b)|$$

where n_b is the number of samples in bin b, N is the total number of samples, $acc(b)$ and $conf(b)$ are the accuracy and confidence of bin b. A lower ECE score means that the accuracy and confidence are aligned, indicating better calibration.

Negative Log Likelihood. NLL calculates the negative log-likelihood for the predicted class probability. While it is generally used for optimization using cross-entropy loss, it is also commonly used to evaluate the prediction uncertainty. A lower NLL score is preferred.

$$NLL = \frac{-1}{N}\sum_{n=1}^{N} log(p(y_n|x_n))$$

Area Under Receiver Operating Characteristic Curve. AUROC indicates the ability to separate ID and OOD samples. To calculate this metric, the predicted uncertainty is used to determine if a sample is ID or OOD. This can be considered as a binary classification problem. The area under the plot between the true positive rate and the false positive rate gives the AUROC value. Higher AUROC value means better separation between ID and OOD.

Area Under Precision-Recall Curve. AUPR, like AUROC measures the ability to separate ID and OOD samples. Considering ID and OOD separation as a binary classification problem, the area under the plot between precision and recall values give the AUPR score.

B. Experimental details

B.1. CIFAR10 *vs.* CIFAR100/SVHN

CIFAR10 [25] consists of 10 classes. We split the original training set consisting of 50000 samples into train and validation set, in the ratio of 80:20. The validation set was used for hyperparameter tuning. The test set consists of 10,000 samples, used for inference. For OOD analyses, we use the test set of SVHN and CIFAR100, which consists of 26,032 and 10,000 samples respectively. The OOD images are normalized the same way as train images during inference.

The network architecture is Wide ResNet 28-10 [51]. The feature embedding layer has a dimension of 640. After training MAPLE , the number of classes were 12, and hence, the final layer has a dimension of 12, followed by softmax. We trained the model for 200 epochs. We used an SGD optimizer with a learning rate of 0.05. The momentum was set to 0.9 and weight decay of $1e^{-4}$. The training was performed using PyTorch on a 12Gb NVIDIA GeForce GTX 1080Ti with a batch size of 64. The dimension of the reduced features from PCA is 12.

B.2. CIFAR100 *vs.* CIFAR10/Tiny ImageNet

CIFAR100 [25] consists of 100 classes. We split the original training set consisting of 50000 samples into train and validation set, in the ratio of 80:20. The validation set was used for hyperparameter tuning. The test set consists of 10,000 samples, used for inference. Additionally, inference and ID metrics were also calculated for the corrupted version (CIFAR100-C [20]). For OOD analyses, we use the test set of Tiny ImageNet and CIFAR100, which consists of 10,000 samples each. The OOD images are normalized the same way as train images during inference.

The network architecture is Wide ResNet 28-10 [51]. The feature embedding layer has a dimension of 640. After training MAPLE , the number of classes were 118, and hence, the final layer has a dimension of 118, followed by softmax. We trained the model for 200 epochs. We used an SGD optimizer with a learning rate of 0.05. The momentum was set to 0.9 and weight decay of $1e^{-4}$. The training was performed using PyTorch on a 12Gb NVIDIA GeForce GTX 1080Ti with a batch size of 64. The dimension of the reduced features from PCA is 34.

B.3. ImageNet *vs.* ImageNet-O

The ImageNet dataset [40] consists of 1,000 classes with 1,281,167 train, 50,000 validation and 10,000 test images. For OOD analysis, ImageNet-O [19] is used, which consists of 200 classes and 2000 images. The OOD images are normalized the same way as train images during inference.

The ResNet-50 [18] was used for training. The feature embedding layer has a dimension of 640. After training

MAPLE , the number of classes were 1223, and hence, the final layer has a dimension of 1223, followed by softmax. We trained the model for 300 epochs. We used an Adam optimizer with a learning rate of 0.01. The training was performed using PyTorch on a 2 24Gb NVIDIA GeForce RTX 3090 with a batch size of 64. The dimension of the reduced features from PCA is 66.

B.4. Diatoms

The diatom dataset consists of 9895 individual RGB images of size 256×256, belonging to 166 classes [49]. We divide it into ID dataset consisting of 130 classes (7874 images) and the remaining 36 classes as OOD (2021 images). 70% of the ID images were used for training, 10% for validation and 20% for testing. While training, horizontal and vertical flips were used for data augmentation.

The network architecture is Wide ResNet 28-10 [51]. The feature embedding layer has a dimension of 640. After training, there were a total of 158 classes, hence the output layer consists of 158 neuron with a softmax activation. We trained the model for 100 epochs with an Adam optimizer. The learning rate was $2e^{-4}$ and batch size 4. The training was performed using PyTorch on a 12Gb NVIDIA GeForce 1080Ti. The dimension of the features after PCA reduction was 31.

B.5. Hyperparameter Tuning

Our training depends on the following hyperparameters: (1) **Frequency of epochs** p - After every p epochs, validation is performed to obtain the new cluster assignments using X-Means. (2) **False negative ratio threshold** t - t is a threshold used to decide the class features to be clustered. From the normalized confusion matrix obtained during the validation step, the classes having false negative greater than t are clustered using X-Means. (3) **Maximum number of clusters** - This is a parameter of X-Means, that specifies the upper bound to the number of clusters that X-Means can generate for each class.

To find the optimal value of these parameters, a grid search was performed. For the grid search, the values of hyperparameters used were: False negative ratio threshold $t \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$, frequency of validation epochs $p \in \{5, 10, 15, 20\}$ and maximum number of clusters that X-Means can generate $\{3, 5, 7, 10\}$.

From the grid-search analysis, the best performance was obtained when $t = 0.3$, $p = 10$ and maximum number of clusters=5 for CIFAR10, CIFAR100 and the Diatom datasets. For ImageNet, $t = 0.2$, $p = 20$ and maximum number of clusters=5.

B.6. Loss Functions

For our training, we use the Cross-Entropy Loss and the Triplet Loss.

B.6.1 Cross-Entropy Loss

To estimate the cross-entropy loss, the final layer of the model is passed through a softmax layer to obtain probability values. Cross-entropy loss increases proportional to the difference between the predicted probability and the actual probability (typically 1) of the ground truth class. The cross-entropy loss is given by:

$$\mathcal{L}_{\text{cross-entropy}} = -\sum_{i=1}^{K} y_i \log(p_i) \tag{7}$$

where K is the total number of samples, y_i is the binary one-hot encoding value corresponding to ground truth class, which equals 1, and p_i is the probability predicted by the network.

B.6.2 Triplet Loss

To estimate the triplet loss, we use the feature embedding obtained from the penultimate layer of the classification network. Triplet loss tries to minimize the distance of intra-class data points, while maximizing the inter-class distance. Consider three input samples, which are feature embeddings extracted: anchor x'_a, positive x'_p and negative x'_n. x'_a and x'_p belong to the same class while x'_n belongs to a different class. The triplet loss is given as:

$$\mathcal{L}_{\text{triplet}} = \max\{||x'_a - x'_p|| - ||x'_a - x'_n|| + \alpha, 0\} \tag{8}$$

The final objective is

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cross-entropy}} + \mathcal{L}_{\text{triplet}} \tag{9}$$

C. Algorithm

The proposed method is summarized in Algorithm 1 and Algorithm 2. Algorithm 1 provides the steps using in training MAPLE . Algorithm 2 summarizes the procedure for estimating uncertainty from MD. At regular intervals of the training process, validation is performed, and the train feature representations are clustered using X-Means. The time complexity for X-Means is O(log K), where K is the number of clusters. The train features are reduced in dimension using PCA, which has a complexity of O(nd^2+d^3), where n is the number of train data and d is the feature dimension. Mahalanobis distance calculation requires calculating mean and the covariance matrix, which has a complexity of O(nd') and O(d'^3), where d' is the PCA reduced feature dimension.

Note that the operations such as the PCA covariance calculation and eigenvalue decomposition, and inverse covariance calculation for MD is to be performed only once, at the end of the training. During inference, the calculated mean and inverse covariance matrix can be used to calculate the Mahalanobis distance for all the test points.

Algorithm 1: MAPLE training

Data: Ground truth labels $\boldsymbol{y} \in \{1, 2, ...k\}$,
Input samples $\boldsymbol{x} \in \mathbb{R}^D$,
Train input samples $\boldsymbol{x_{train}} = \{x_n\}_{n=1}^N$,
Train dataset $\boldsymbol{\mathcal{D}_{train}} = \{(x_n, y_n)\}_{n=1}^N$,
Validation dataset $\boldsymbol{\mathcal{D}_{val}} = \{(x_v, y_v)\}_{m=1}^M$
Initialize: $n_c = k, p = 10, t = 0.3, max\_clusters = 5$
Model : $f^\theta : \mathbb{R}^D \to \mathbb{R}^d$
for *epoch = 1* **to** *max-epochs* **do**
  Train f^θ with $\boldsymbol{\mathcal{D}_{train}}$ and n_c classes and loss given by $L_{total} = L_{cross-entropy} + L_{triplet}$
  if *epoch%p==0* **then**
    $\boldsymbol{x'_{train}} = f^\theta(\boldsymbol{x_{train}})$
    Get softmax predictions on $\boldsymbol{\mathcal{D}_{val}}$
    if $n_c > k$, remap pseudo-labels to original class labels
    Compute confusion matrix
    for *i=1* **to** *k* **do**
      if *false_negative_ratio(i)* $> t$ **then**
        Cluster using X-Means.
        X-Means($\boldsymbol{x'_{train}(i)}$, max_clusters)
    $K \leftarrow$ total number of clusters obtained from all the classes
    $n_c = K$
    Update $\boldsymbol{\mathcal{D}_{train}}$ with pseudo-labels from clustering

Algorithm 2: MAPLE Prediction

Data: Train feature embeddings $\boldsymbol{x'_{train}}$
Input: Test sample $\tilde{\boldsymbol{x}}$
Compute the reduced dimensional train features: $\boldsymbol{z_{train}} = g(\boldsymbol{x'_{train}})$
Compute individual class means and shared covariance $\mu_c, \boldsymbol{\Sigma}$
$\mu_c = \frac{1}{N_c} \sum_{i:y_i=c} z_i$
$\boldsymbol{\Sigma} = \frac{1}{N} \sum_c \sum_{i:y_i=K} (z_i - \mu_c)(z_i - \mu_c)^T$
Get reduced dimensional feature for $\tilde{\boldsymbol{x}}$: $\tilde{\boldsymbol{z}} = g(f^\theta((\tilde{\boldsymbol{x}}))$
Compute Mahalanobis distance: $MD(\tilde{\boldsymbol{x}}) = \sqrt{(\tilde{\boldsymbol{z}} - \mu_c)^T \boldsymbol{\Sigma}^{-1} (\tilde{\boldsymbol{z}} - \mu_c)}$
Get the prediction probabilities: $P_{MD} = 1 - \text{cdf}(\chi^2_{d'})(MD^2)$
Predicted class = argmax(P_{MD})
Compute uncertainty $u = \text{cdf}(\chi^2_{d'})(MD^2)$

D. Additional Experiments

In this section, we provide results for additional evaluation of MAPLE .

D.1. Accuracy based on prediction confidence

We evaluate the accuracy of prediction when selecting samples with predictive confidence above a given threshold. In other words, classification is performed only when the network's confidence is above a threshold. This is representative of real-life applications where a network's prediction is considered only when the confidence is high. We consider three probability thresholds: 0.50, 0.80 and 0.90. For all samples with predictive probability above these values, we report the classification accuracy. Table 8 gives the results on the test set of CIFAR10 [25] dataset.

Method	acc@.50	acc@.80	acc@.90
MC Dropout [13]	0.962	0.976	0.988
Deep ensemble [26]	**0.967**	0.987	**0.995**
DUQ [46]	0.950	0.977	0.982
SNGP [31]	0.959	0.978	0.985
DUE [45]	0.962	0.974	0.979
MAPLE	0.958	**0.989**	**0.995**

Table 8. **Accuracy on CIFAR10 with different confidence levels.** MAPLE achieves top accuracy at confidence levels of 0.80 and 0.90.

Results. MAPLE achieves the best accuracy at confidence values of 0.80 and 0.90 on CIFAR10. Overall, on CIFAR10, MAPLE has competitive accuracy with the other approaches. This shows that even though MAPLE is computationally efficient, it can achieve the same level or better performance as the other methods.

D.2. Gaussian test

In Section 3.1, it was theoretically shown that X-Means creates clusters of feature points that are Gaussian. In this section, we empirically test this. A commonly adopted method to check for multivariate Gaussian is to use a quantile-quantile plot, where an observed quantile is compared with a theoretical one. If the samples are Gaussian, their squared MD follows a χ^2 distribution. Thus, we use $MD^2_{c^*}$ of the samples feature embeddings as our observed quantile and compare with theoretical χ^2 quantiles.

For our test, we use the reduced feature embeddings, $\boldsymbol{z_{train}}$, from a standard classifier network and MAPLE . The $MD^2_{c^*}$ of samples are calculated and plotted with χ^2 quantiles with d' degrees of freedom, where d' is the dimension of feature embeddings. We measure the error, which is the mean absolute difference between the two quantiles,

to test which method generates feature embeddings that are closer to a Gaussian. In the ideal situation, this value should be zero. The larger the error, the greater is the deviation from a Gaussian distribution.

Table 9 shows the errors computed on feature embeddings from CIFAR10 and CIFAR100 dataset. From the results, MAPLE's error is reduced by over 50%, which shows that the feature representations of MAPLE are more Gaussian than when using a standard DNN classifier.

Method	CIFAR10	CIFAR100
Standard CNN	3.540	4.479
MAPLE	**1.395**	**1.982**

Table 9. **Mean absolute error between squared MD and χ^2 distribution.** The lower the error, the more Gaussian are the samples. MAPLE's training generates sample distributions that are approximately Gaussian, fitting with the theoretical framework for MD calculation.

E. Extended Ablation Analyses

E.1. MAPLE evaluated on different backbones

MAPLE is tested on three networks: Wide ResNet 28-10 [51], ResNet-18 [18] and EfficientNet-B0 [44]. Table 10 gives the quantitative metrics for evaluation on CIFAR10 *vs.* SVHN and CIFAR100. While it is expected that the accuracy depends on the architecture used, the calibration and OOD detection are also influenced by the architecture. Wide ResNet, which has more number of parameters than the other two architectures, learns better feature representations for discriminating each class. As the model parameters decrease, there are overlapping feature points between different classes, which explains the lower accuracy and worse calibration and OOD metrics.

Architecture	Accuracy ↑	ECE ↓	SVHN AUROC ↑	CIFAR100 AUROC ↑
Wide ResNet 28-10 [51]	**0.954**	**0.012**	**0.996**	**0.926**
ResNet-18 [18]	0.945	0.029	0.979	0.886
EfficientNet-B0 [44]	0.902	0.035	0.942	0.893

Table 10. **MAPLE evaluated on different architectures.** The metrics improve as the model parameters increase, suggesting that the network learns better discriminative feature representations, thereby improving the performance.

E.2. Evaluation of different clustering methods

We analyse the performance of MAPLE on CIFAR10 when clustering is performed using K-Means, G-Means [52] and X-Means [37]. The value of K in K-Means is set to 3. Tab. 11 shows the results obtained. Based on the results, X-Means yields the best performance. K-Means and G-Means causes overclustering, which leads to worser performance on OOD detection. Using X-Means, we choose the optimal number of clusters, which performs superior to the others.

Clustering method	#Classes	Accuracy↑	ECE↓	SVHN AUROC↑	CIFAR100 AUROC↑
K-Means	30	0.952	0.154	0.871	0.850
G-Means	67	0.910	0.266	0.710	0.627
X-Means	12	**0.954**	**0.012**	**0.996**	**0.926**

Table 11. **Metrics for different frequency of validation epoch** #Classes refers to the total number of output classes obtained after clustering. K-Means and G-Means lead to overclustering, whereas using X-Means, the optimal number of clusters are generated leading to better performance.

E.3. Effect of maximum number of clusters

Tab. 12 shows the results when the maximum number of clusters that can be generated for every class by X-Means is varied, along with different values of false negative ratio t for CIFAR10. For $t > 0.5$, none of the classes are clustered, and hence we do not include them. From the results, when the maximum number of clusters are low, MAPLE fails to capture all the within-class variances, whereas higher values result in overclustering. With the maximum number of clusters as 5, MAPLE achieves the best performance.

E.4. Effect of frequency of validation epochs.

Tab. 13 summarizes the metrics for CIFAR10 when the number of epochs after which the validation and cluster refinements are performed is varied. A low value of validation epochs does not give the network enough time to learn representations for the new clusters generated. Whereas, with larger number of epochs, the number of cluster refinements are low. In both these situations, the network does not identify the optimal clusters. MAPLE gives the best results when the validation is performed every 10 epochs.

Max. number of clusters	t	#Classes	Accuracy↑	ECE↓	SVHN AUROC↑	CIFAR100 AUROC↑
3	0.1	14	0.9542	0.012	0.996	0.925
	0.3	10	0.9540	0.014	0.972	0.919
	0.5	10	0.9533	0.012	0.958	0.917
5	0.1	18	0.9534	0.013	0.964	0.918
	0.3	12	0.9541	0.012	**0.996**	**0.926**
	0.5	10	0.9535	0.012	0.955	0.915
7	0.1	18	0.9537	0.013	0.959	0.894
	0.3	13	**0.9545**	0.012	0.992	0.921
	0.5	10	0.9531	0.013	0.944	0.911
10	0.1	26	0.9519	0.014	0.909	0.863
	0.3	22	0.9521	0.013	0.918	0.886
	0.5	11	0.9534	0.012	0.952	0.908

Table 12. **Effect of maximum number of clusters per class on MAPLES's performance.** A high value of cluster numbers causes overclustering whereas a low value does not generate enough clusters. A value of 5 results in optimal number of clusters for MAPLE to learn meaningful representations.

Validation epochs	#Classes	Accuracy↑	ECE↓	SVHN AUROC↑	CIFAR100 AUROC↑
5	16	0.895	0.025	0.914	0.876
10	12	0.954	0.012	**0.996**	**0.926**
15	12	**0.955**	0.012	0.987	0.922
20	10	0.953	0.013	0.968	0.917

Table 13. **Metrics for different frequency of validation epoch** #Classes refers to the total number of output classes obtained after clustering. With lower validation epochs, the clustering is too frequent for the network to learn meaningful representations. At lower frequency, the number of cluster refinements are not sufficient.