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 $\times$ 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 $f^\theta$ 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
```

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.

```
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 $u = \text{cdf}(\chi^2_{d'})(MD^2)$
```

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.

F. Proof of squared Mahalanobis Distance following a χ^2 distribution

In this section, we derive the proof that the squared Mahalanobis distance follow a χ^2 distribution with d' degrees of freedom, where d' is the dimension of the feature vectors used to calculate MD. A χ^2 distribution with d' degrees of freedom is defined as the distribution of a sum of the squares of d' independent standard normal random variables.

The squared Mahalanobis distance of $\boldsymbol{Z}$ and the mean vector $\vec{\mu}$ of a Multivariate Gaussian distribution is given as

$$D^2 = (\boldsymbol{Z} - \vec{\mu})^T \boldsymbol{\Sigma}^{-1} (\boldsymbol{Z} - \vec{\mu}) \tag{10}$$

$\boldsymbol{\Sigma}$ is the covariance matrix, which is symmetric. By property of matrices, the matrix inverse and it's square root are also symmetric. Thus,

$$\begin{aligned} D^2 &= (\boldsymbol{Z} - \vec{\mu})^T \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^{-\frac{1}{2}} (\boldsymbol{Z} - \vec{\mu}) \\ &= \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} (\boldsymbol{Z} - \vec{\mu})\right)^T \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} (\boldsymbol{Z} - \vec{\mu})\right) \end{aligned} \tag{11}$$

Let $\boldsymbol{W} = \boldsymbol{\Sigma}^{-\frac{1}{2}}$ and $\boldsymbol{X} = (\boldsymbol{Z} - \vec{\mu})$. The whitening transform is given as $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$ and $\boldsymbol{W}$ is also called the Mahalanobis whitening matrix. Eq. 11 can be written as

$$\begin{aligned} D^2 &= \boldsymbol{Y}^T \boldsymbol{Y} \\ &= ||\boldsymbol{Y}||^2 \\ &= \sum_{i=1}^{d'} Y_i^2 \end{aligned} \tag{12}$$

$(\boldsymbol{Z} - \vec{\mu}) \sim \mathcal{N}(0, \boldsymbol{\Sigma})$, and so $\boldsymbol{Y}$ has zero mean. The covariance of $\boldsymbol{Y}$ is given as

$$\begin{aligned} \boldsymbol{\Sigma_Y} &= \boldsymbol{W}\boldsymbol{\Sigma}\boldsymbol{W}^T \\ &= \boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma} \boldsymbol{\Sigma}^{-\frac{1}{2}T} \\ &= \boldsymbol{\Sigma}^{-\frac{1}{2}} \left(\boldsymbol{\Sigma}^{\frac{1}{2}} \boldsymbol{\Sigma}^{\frac{1}{2}}\right) (\boldsymbol{\Sigma}^{-\frac{1}{2}})^T \\ &= \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^{\frac{1}{2}}\right) \left(\boldsymbol{\Sigma}^{-\frac{1}{2}} \boldsymbol{\Sigma}^{\frac{1}{2}}\right) \\ &= (\boldsymbol{I})(\boldsymbol{I}) = \boldsymbol{I} \end{aligned} \tag{13}$$

The covariance of $\boldsymbol{Y}$ is an identity matrix, which means that the elements from $\boldsymbol{Y}$ are drawn from an independent standard Gaussian distribution *i.e.,* $Y_i \sim \mathcal{N}(0, 1)$.

From the definition of the χ^2 distribution, we can infer that D^2 follows a χ^2 distribution with d' degrees of freedom.