A Algorithm

Algorithm 1 OOD detection

Input: Trained classifier that produces logits $L(x) = F_\theta(x)$. Individual data x. The number of classes K. K different trained $NF_{[1 \cdot K]}$ models. Threshold value θ.

procedure D(x)
  $L(x) \leftarrow F_\theta(x)$ ▷ Get the embeddings in the logit space
  $LLK_{Max} \leftarrow 0$ ▷ Get the embeddings in the logit space
  for each $c = 1, 2, \ldots K$ **do** ▷ Iterate each $NF_{[1 \cdot K]}$
    $\hat{LLK} \leftarrow NF_{[c]}(L(x))$ ▷ Get the likelihood for class c
    if $\hat{LLK} = LLK_{Max}$ **then** ▷ If the prediction matches the given label
      $LLK_{Max} = \hat{LLK}$ ▷ Train the NF over the ID cluster of class c
  if $LLK_{Max} \geq \theta$ **then** ▷ Check if x is an ID
    $x \rightarrow ID$
  else
    $x \rightarrow OOD$ ▷ If the data is not ID then it is OOD

Algorithm 2 Training of NF for each ID cluster

Input: Trained classifier that produces logits $L(x) = F_\theta(x)$. ID annotated training data $[c, x] \in X$. The number of classes K. K different $NF_{[1 \cdot K]}$ models.

procedure TRAIN INDIVIDAL NF
  for each round $t = 1, 2, \ldots$ **do**
    for each data $c, x \in X$ **do** ▷ Iterate the dataset in batches
      $L(x) \leftarrow F_\theta(x)$ ▷ Get the embeddings in the logit space
      $\hat{c} \leftarrow \arg\max L(x)$ ▷ Get the class prediction
      if $\hat{c} = c$ **then** ▷ If the prediction matches the given label
        $Train \rightarrow NF_{[c]}[L(x)]$ ▷ Train the c-th NF over the ID cluster of class c

B Proof of corollary 1

Corollary 3. *Given two (random) variables, the magnitude of their dot-product is a lower bound for the magnitude of their covariance.*

$$|\langle x, \omega \rangle| \leq |cov\{(x, \omega)\}|, \forall x, \omega \in \mathbb{R} \tag{5}$$

When a neural network is trained using a gradient-based method, the objective function is mainly driven by the dot-product between the weights and the training data $\sum_i x_i \omega_i$. Dot-product measures the (un-normalized cosine) similarity between two numerical entities (i.e., x, ω). This similarity, in this case, is just the scale of orthogonality

between two vectors x and ω. However, unlike covariance, dot-product is a geometric metric and does not explicitly convey any statistical similarity between the two entities. One can notice that the dot-product is a proxy for the covariance (eq. (6)). Eventually, it remains of primary interest to see the analytical relationship between x_i and ω_i w.r.t dot-product and covariance. An inequality relationship between dot-product and covariance offers analytical analysis for the statistical properties of parameters and data in the training process.

In the case of the absolute values, it is easy to prove that the following inequality holds for any value of x and ω (eq. (6)).

$$\big|\langle x, \omega \rangle\big| \leq \big|cov\{(x, \omega)\}\big| \tag{6}$$

Proof.

$$\big|\langle x, \omega \rangle\big| \leq \big|E\{(x - E[x])(\omega - E[\omega])\}\big| \tag{7}$$

using Monte-Carlo estimation for the left-hand side:

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i \{(x_i - E[x])(\omega_i - E[\omega])\}\right| \tag{8}$$

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i (x_i \omega_i - E[x]\omega_i + E[\omega]E[x] - E[\omega]x_i)\right| \tag{9}$$

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i x_i \omega_i - \sum_i E[x]\omega_i + \sum_i E[\omega]E[x] - \sum_i E[\omega]x_i\right| \tag{10}$$

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i x_i \omega_i - E[x]\sum_i \omega_i + E[\omega]E[x]\sum_i 1 - E[\omega]\sum_i x_i\right| \tag{11}$$

using the triangle inequality $|a + b| \leq |a| + |b|$ it leads to:

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i x_i \omega_i\right| + \left|-E[x]\sum_i \omega_i + E[\omega]E[x]\sum_i 1 - E[\omega]\sum_i x_i\right| \tag{12}$$

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i x_i \omega_i\right| + \left|E[\omega]E[x]\sum_i 1 - E[x]\sum_i \omega_i - E[\omega]\sum_i x_i\right| \tag{13}$$

Given the size of $[x_i]$ and $[\omega_i]$ is equal to N and $\sum_i \omega_i \approx NE[\omega]$ $\sum_i x_i \approx NE[x]$:

$$\left|\sum_i x_i \omega_i\right| \leq \left|\sum_i x_i \omega_i\right| + \big|E[\omega]E[x]N - NE[x]E[\omega] - NE[\omega]E[x]\big| \tag{14}$$

$$\left|\sum_i x_i\omega_i\right| \leq \left|\sum_i x_i\omega_i\right| + \left|-NE[\omega]E[x]\right| \tag{15}$$

In the case of either data or weights being centered ($E[\omega] \approx 0$ or $E[x] \approx 0$), the dot-product equals the covariance since (eq. (15)).

Utilizing this inequality $\left|\sum_i (x_i\omega_i)\right| \leq \left|cov(x,\omega)\right|$, one can conclude that maximizing the dot-product between the data and the weights directly translates into an increased statistical dependency between the two. Likewise, the dot-product is also minimal whenever the covariance between the data and the weights is minimal.

Eventually, $\left|\sum_i (x_i\omega_i)\right| \leq \left|cov(x,\omega)\right|$ can facilitate the imposition of statistical dependency between the transformation of the data and the network weights where the latter is trying to get statistically similar to the former.

C Proof of corollary 2

Corollary 4. ***(Co-variability)*** *Given three (random) variables (x,y,z) $\in \mathbb{R}$ where the first strongly co-varies with the second ($\left|cov(x,y)\right| \uparrow$) while the second does not co-vary with the third ($\left|cov(y,z)\right| \approx 0$), the first does not co-vary with the third either ($\left|cov(x,z)\right| \approx 0$).*

Proof. In order to prove this corollary, we make use of three of the following covariance property.

$$cov(const, X) = 0 \tag{16}$$

$$cov(X + Y, Z) = cov(X, Z) + cov(Y, Z) \tag{17}$$

$$cov(a + b * X, c + dY) = b * d * cov(X, Y) \tag{18}$$

Whenever the co-variability magnitude between two (random) variables increases, explaining one of the variables through the other becomes increasingly acceptable. The higher this co-variability magnitude ($\left|cov(x,y)\right|$), the more these two entities correlate, and the more it is possible to explain one while knowing the other.

Hence, given that $\left|cov(x,y)\right|$ is maximized, x and y can be expressed as $x \approx \sigma * y + constant$, such that $\sigma = \frac{cov(x,y)}{var(x)var(y)}$, therefore the covariance between x and z can be written as (eq. (19)):

$$cov(x, z) = cov(\sigma * y + constant, z) \tag{19}$$

Utilizing property eq. (17) we split the summation into:

$$cov(x, z) = cov(\sigma * y, z) + cov(constant, z) \tag{20}$$

Since a random variable does not co-vary with a constant (eq. (16)) we get:

$$cov(x, z) = cov(\sigma * y, z) + 0 \tag{21}$$

Getting the constant outside of the covariance (eq. (18))

$$cov(x, z) = \sigma * cov(y, z) \tag{22}$$

Given that $cov(y, z) \approx 0$ it enables this conclusion:

$$cov(x, z) \approx \sigma * 0 = 0 \tag{23}$$

D Proof for theorem theorem 1

Theorem 2 (Expectations of OOD embeddings). *Given two random variables ($x_{OOD}, \omega \in \mathbb{R}$) whose covariance magnitude is at the proximity of zero, the expectation of their dot-product is upper bounded by the property of their covariance:*

$$\Big|E\big\{\langle x_{OOD}, \omega\rangle\big\}\Big| \leq \Big|\sqrt{E[cov(x_{OOD}, \omega)]^2 + Var[cov(x_{OOD}, \omega)]}\Big| \tag{24}$$

Proof. Since covariance is an upper bound of the dot-product (eq. (6)), their corresponding expectations would not affect the inequality.

$$E\big\{\big|\langle x_{OOD_i}, \omega_i\rangle\big|\big\} \leq E\big\{\big|cov(x_{\text{OOD}}, \omega)\big|\big\} \tag{25}$$

Since the absolute value is a convex function it is possible to use Jensen's inequality $f\big\{E[x]\big\} \leq E\big\{f[x]\big\}$ again for both sides of the equation eq. (25).

$$\Big|E\big\{\langle x_{OOD_i}, \omega_i\rangle\big\}\Big| \leq E\big\{\big|\langle x_{OOD_i}, \omega_i\rangle\big|\big\} \tag{26}$$

Combining equation eq. (25) and eq. (26) the inequalities maintain the directions:

$$\Big|E\big\{\langle x_{OOD_i}, \omega_i\rangle\big\}\Big| \leq E\big\{\big|\langle x_{OOD_i}, \omega_i\rangle\big|\big\} \leq E\big\{\big|cov(x_{\text{OOD}}, \omega)\big|\big\} \tag{27}$$

Whereby removing the middle term would not affect the flow of inequalities.

$$\Big|E\big\{\langle x_{OOD_i}, \omega_i\rangle\big\}\Big| \leq E\big\{\big|cov(x_{\text{OOD}}, \omega)\big|\big\} \tag{28}$$

The right-hand side of equation eq. (28) can be re-written as:

$$E\big\{\big|cov(x_{\text{OOD}}, \omega)\big|\big\} = E\big\{\sqrt{cov(x_{\text{OOD}}, \omega)^2}\big\} \tag{29}$$

Since, $f(x) = \sqrt{x}$ is concave, using Jensen inequality for the concave functions $f\{E[x]\} \geq E\{f(x)\}$:

$$E\big\{\sqrt{cov(x_{\text{OOD}}, \omega)^2}\big\} \leq \sqrt{E\big\{cov(x_{\text{OOD}}, \omega)^2\big\}} \tag{30}$$

leads to

$$E\big\{\big|cov(x_{\text{OOD}}, \omega)\big|\big\} \leq \sqrt{E\big\{cov(x_{\text{OOD}}, \omega)^2\big\}} \tag{31}$$

The right hand side of equation eq. (31) is part of the $cov(x_{\text{OOD}}, \omega)$ variance.

$$var\{cov(x_{\text{OOD}}, \omega)\} = E\{cov(x_{\text{OOD}}, \omega)^2] - E\{cov(x_{\text{OOD}}, \omega)]^2 \tag{32}$$

where:

$$E\{cov(x_{\text{OOD}}, \omega)^2\} = E\{cov(x_{\text{OOD}}, \omega)\}^2 + var\{cov(x_{\text{OOD}}, \omega)\} \tag{33}$$

Eventually:

$$\left|\sqrt{E\{cov(x_{\text{OOD}}, \omega)^2\}}\right| = \left|\sqrt{E\{cov(x_{\text{OOD}}, \omega)\}^2 + var\{cov(x_{\text{OOD}}, \omega)\}}\right| \tag{34}$$

Combination of equation eq. (31) and eq. (34) yields:

$$E\{|cov(x_{\text{OOD}}, \omega)|\} \leq \sqrt{E\{cov(x_{\text{OOD}}, \omega)^2\}} \leq \left|\sqrt{E\{cov(x_{\text{OOD}}, \omega)\}^2 + var\{cov(x_{\text{OOD}}, \omega)\}}\right| \tag{35}$$

This concludes to:

$$\left|E\{\langle x_{OOD_i}, \omega_i\rangle\}\right| \leq \left|\sqrt{E\{cov(x_{\text{OOD}}, \omega)\}^2 + var\{cov(x_{\text{OOD}}, \omega)\}}\right| \tag{36}$$

Eventually, the lower the magnitude and the variance of ($cov(x_{\text{OOD}}, \omega)\}$) the lower the dot-product between x_{OOD_i} and ω_i

Since the co-variability of $cov(x_{\text{OOD}}, \omega)$ is minimal (corollary 2), the r.h.s of the eq. (36) is mostly affected by the variability of $cov(x_{\text{OOD}}, \omega)$. Hence, the output from $|E\{\langle x_{OOD_i}, \omega_i\rangle\}|$ will be lying around zero as far as x and ω are systematically statistically independent (i.e., $E\{cov(x_{\text{OOD}}, \omega)\}^2 \approx 0$ and $var\{cov(x_{\text{OOD}}, \omega)\} \approx 0$).

Empirical validation On a simple starting experiment with CIFAR-3 (only three classes from CIFAR-10) and SVHN where the classifier has been trained on the former dataset, one can notice that just before the training, both datasets behave as OOD (fig. 3). Moreover, once the training is over, SVHN persists on behaving like OOD while CIFAR-3 data are orthogonally clustered (fig. 3).

Eventually, the projection of OOD data from a well-trained layer maintains a tendency toward the origin of the embedding space. In contrast, the ID data maintain a tendency towards high positive values. This is empirically validated by comparing the density for each logit cell corresponding to OOD and ID data. To showcase the discrepancy between the ID and the OOD data in the predictive response, we overlay the plots of their estimated density from each logit cell. Density plots for ID data are segregated based on the annotated class; namely, for the first logit cell, we plot the density solely for the first-class data, for the second cell solely for the second class data, and so forth until the last designated class.

The model in table 7 is utilized to contrast fashion-MNIST and MNIST (figs. 5 and 6). The Resnet-35 [10] is then utilized to contrast CIFAR-10 versus SVHN (figs. 7 and 8) as well as Genome experiment fig. 9.

KDE plots for the ID data exhibit a positive trend. In contrast, the KDE plot for the OOD data shows a tendency toward the origin or, in some cases, slightly elevated

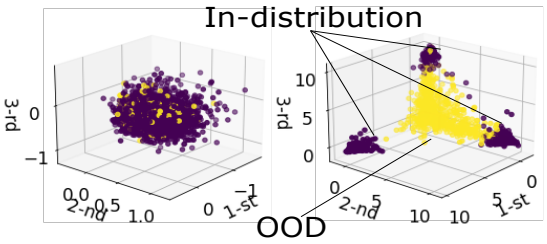
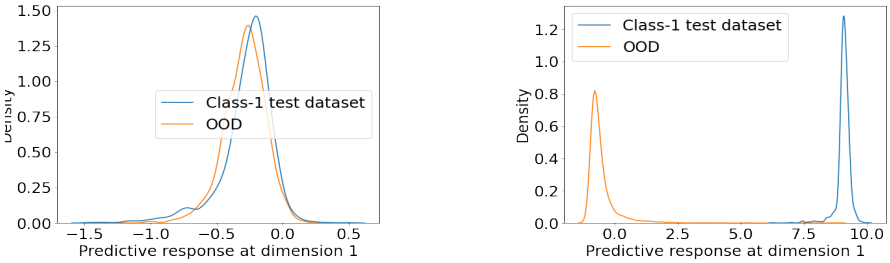


Fig. 3: CIFAR-10 as ID and SVHN as OOD. *Left*: Before the training, both ID and OOD maintain the tendency towards the center of the latent space. *Right*: After the training, the ID data are clustered, whereas the OOD persists towards the center.

towards the negative trend (figs. 5 to 9). This tendency toward negative values is most prevalent when the classifier performance in the ID cannot attain the best performance possible.

One can notice that for a dataset where the ID data classification is very high (i.e., MNIST, CIFAR-10, or SVHN), their corresponding density plot exhibit reduced spread (figs. 5, 7 and 8). Furthermore, the OOD data in this scenario maintain a more pronounced tendency towards the center with a reduced spread (figs. 5, 7 and 8).

On the contrary, in ID data whose classification accuracy is relatively low (i.e., Fashion-MNIST, Genome dataset), their corresponding density plot exhibits increased spread (figs. 6 and 9). Furthermore, the OOD data in this scenario overlaps with the ID data since the latter behaves a little bit like the former due to the low scale of co-variability with the model's weight.



(a) KDE over OODs and IDs before training.

(b) KDE over OODs and IDs after training.

Fig. 4: KDE over the logit response for both OODs and IDs shows that the DL classifier does not output a high-magnitude response before training. However, after training, solely IDs are displaced towards positive regions while the OODs remain in the center of the logit space.

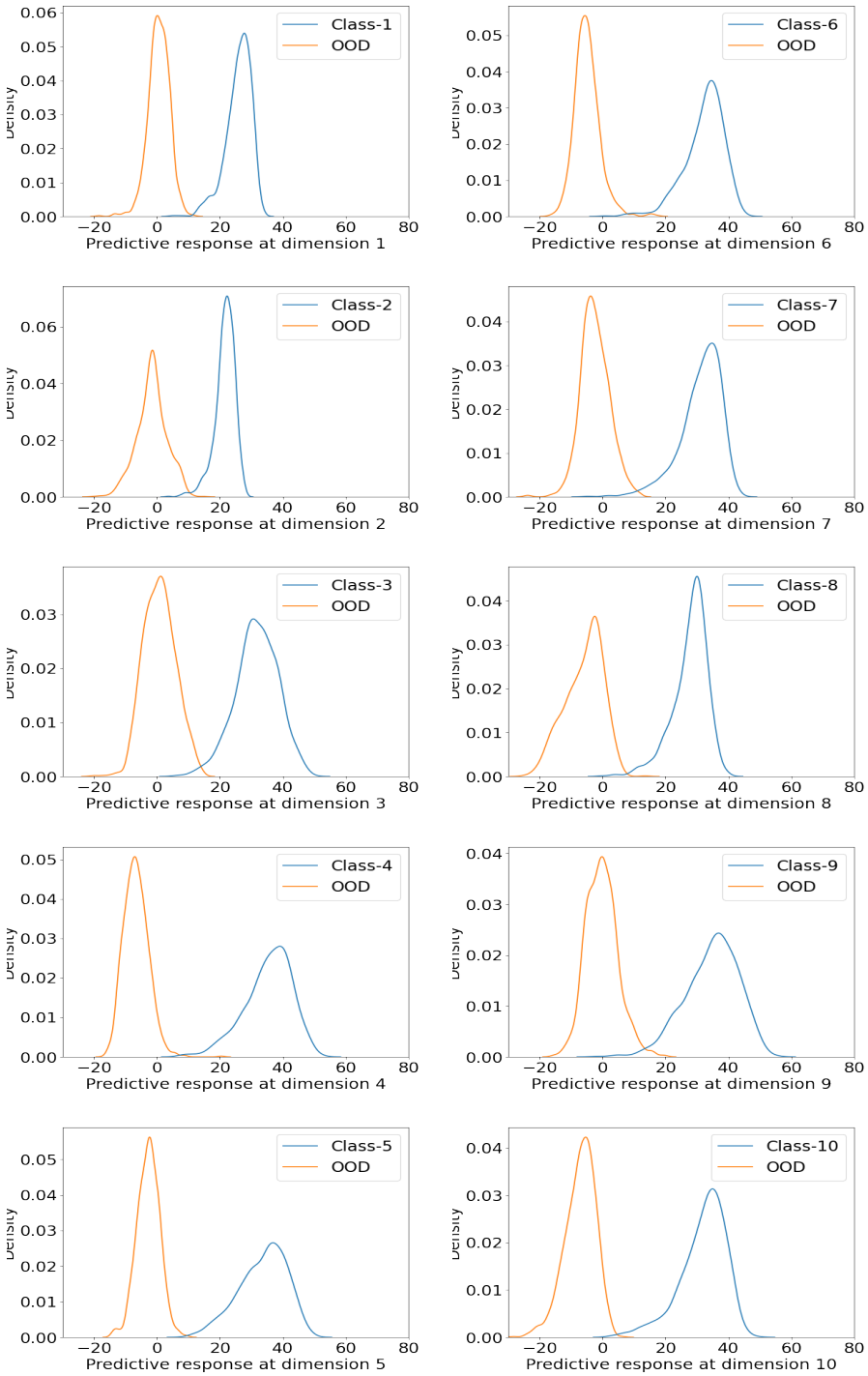


Fig. 5: MNIST (ID) vs. Fashion-MNIST (OOD) KDE plot for OOD and ID data.

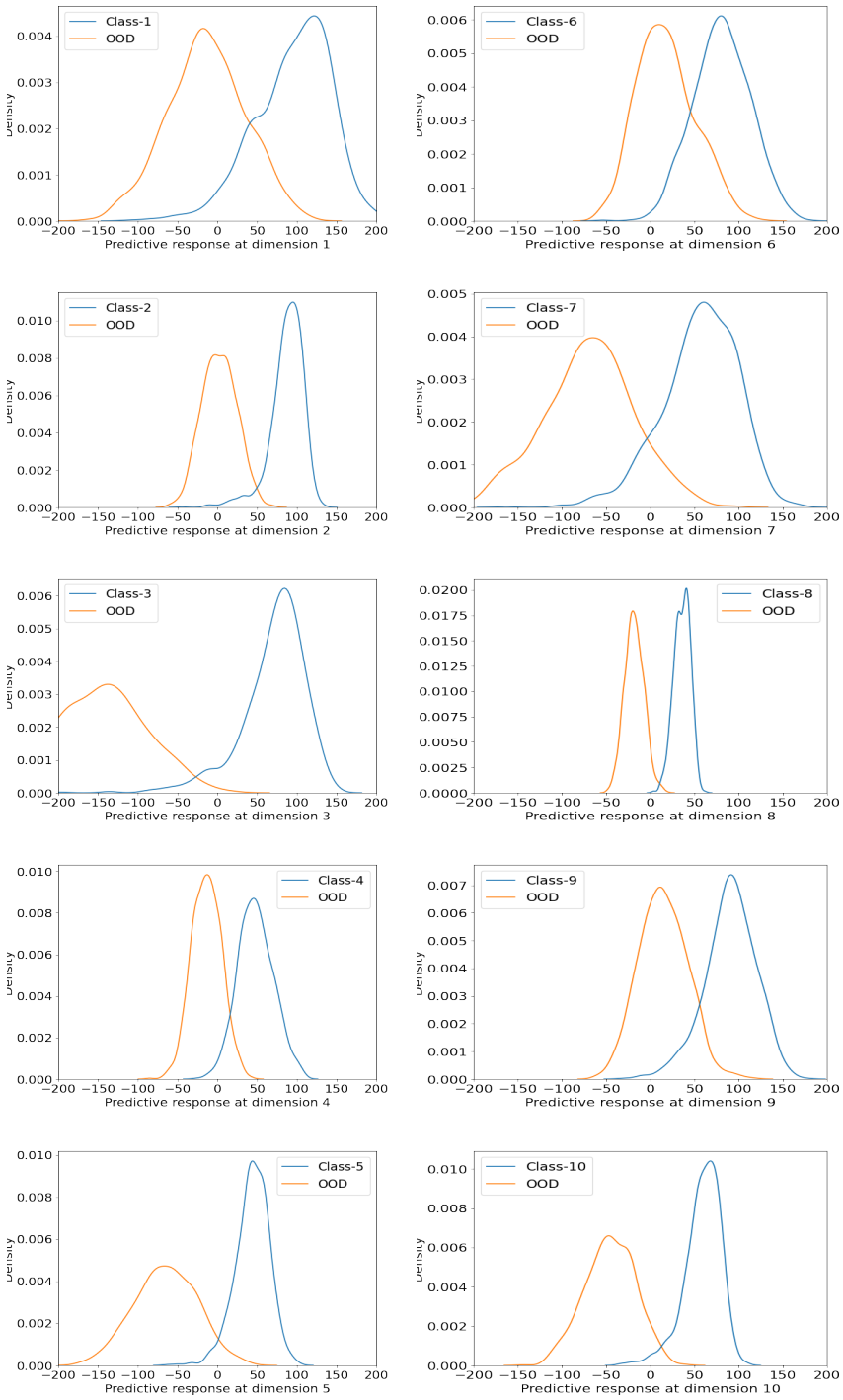


Fig. 6: Fashion-MNIST (ID) vs. MNIST (OOD) KDE plot for OOD and ID data.

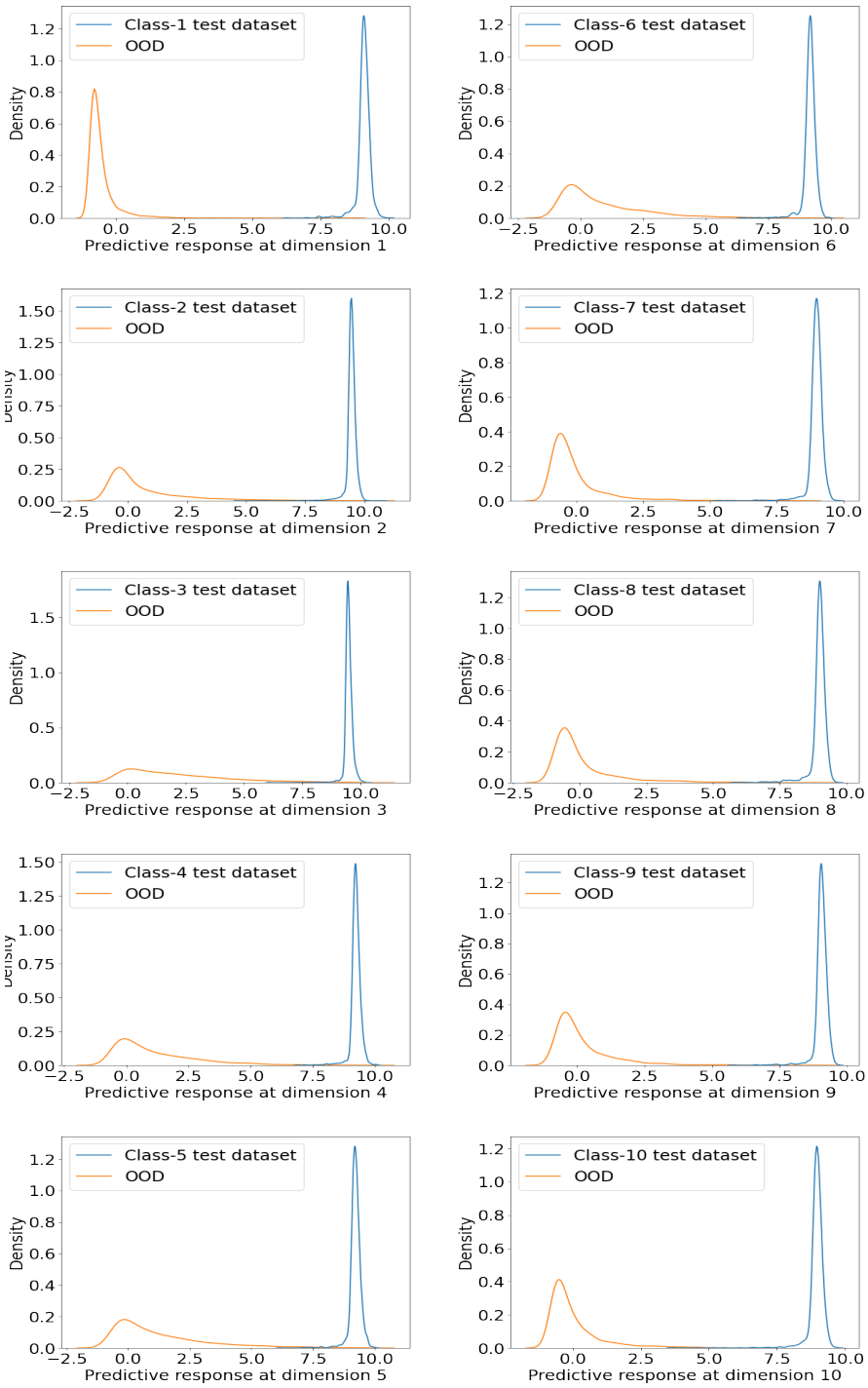


Fig. 7: SVHN (ID) vs CIFAR-10 (OOD) KDE plot for OOD and ID data.

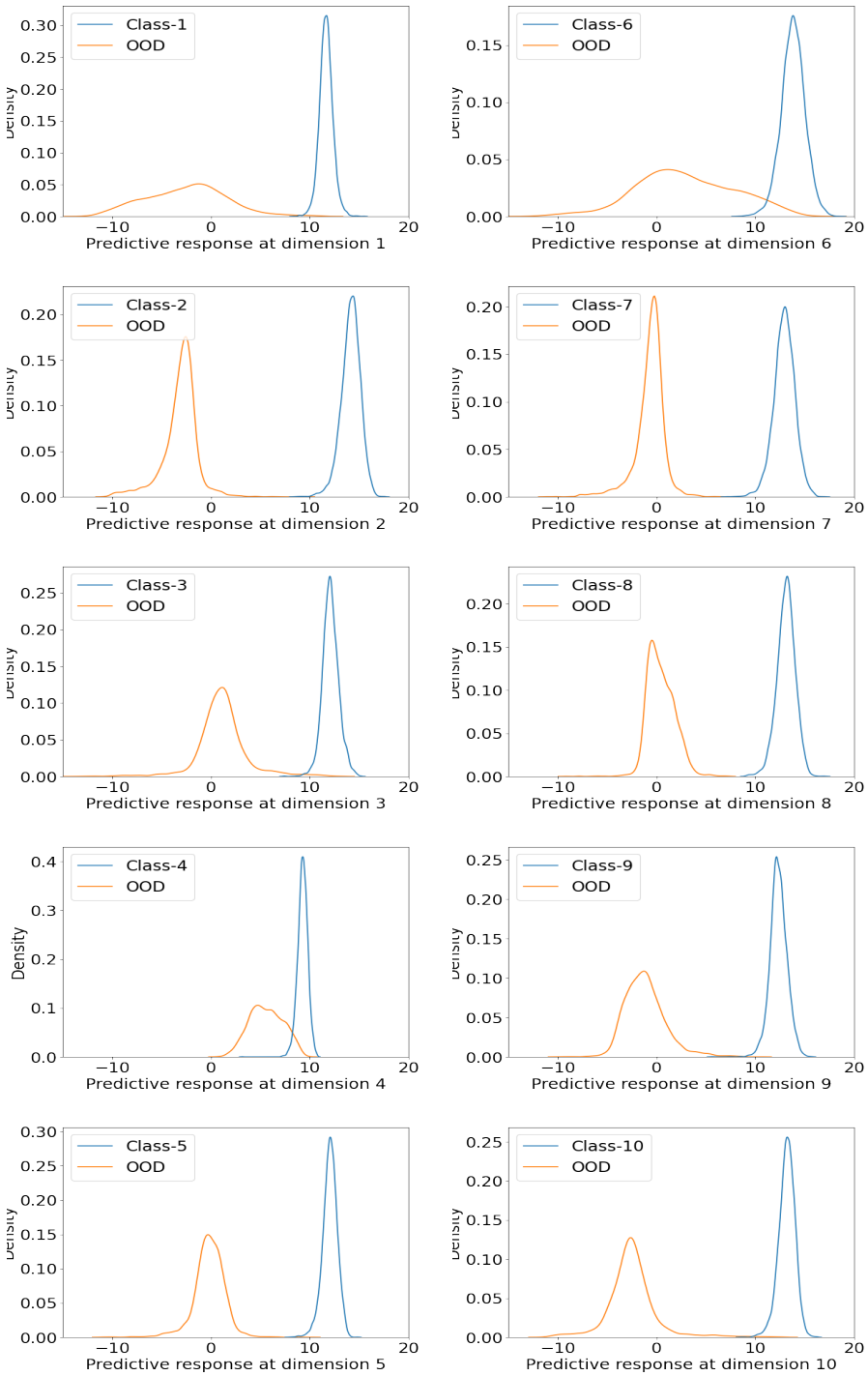


Fig. 8: CIFAR-10 (ID) vs SVHN (OOD) KDE plot for OOD and ID data.

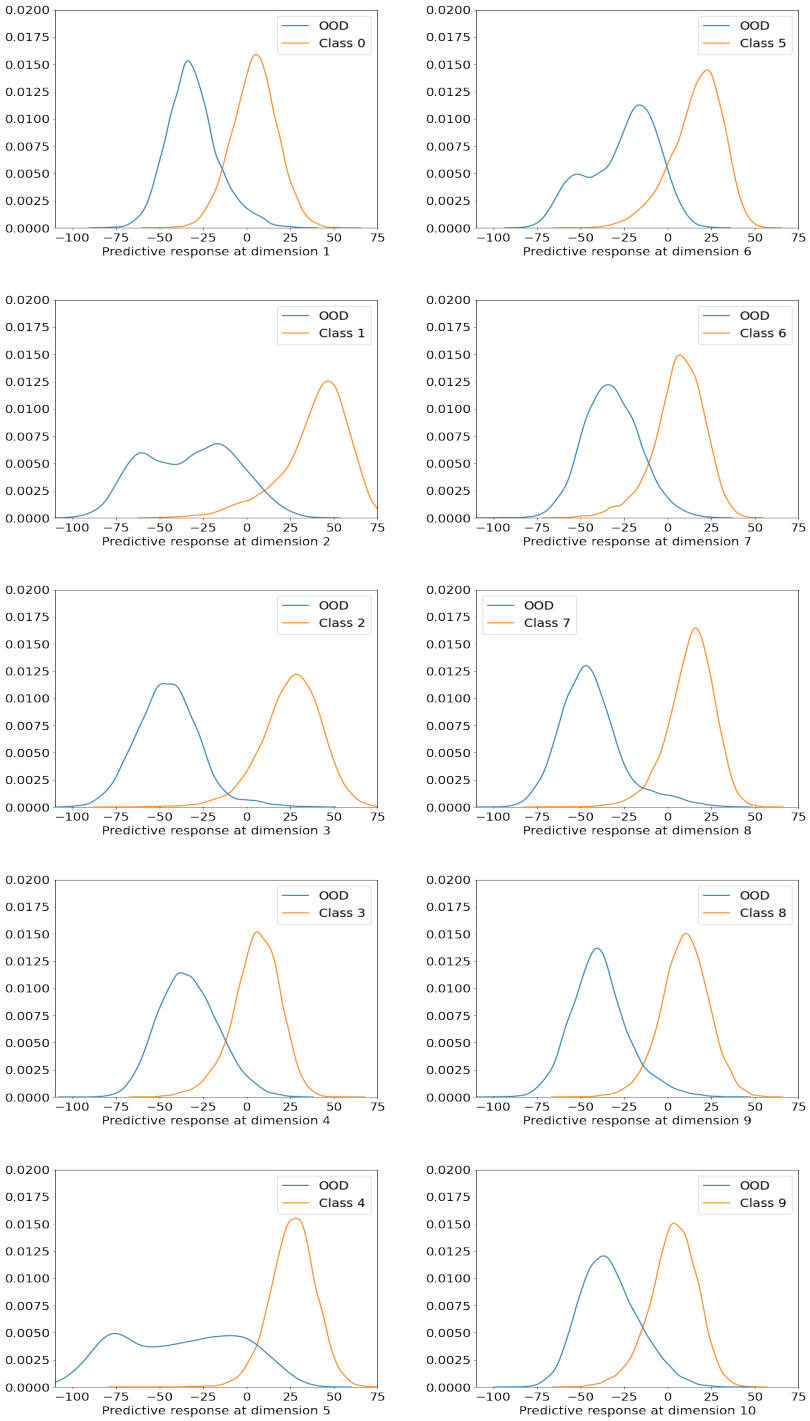


Fig. 9: Genome experiment where the first 10 classes are in distribution and the rest of the classes as OOD. KDE plot for OOD and ID data.

E Experiments on Fashion-MNIST image dataset

In order to test the capability of the proposed method, the ID and OOD datasets have to share a large scale of similarity while at the same time being contextually different. Fashion-MNIST (ID) and MNIST (OOD) [18] datasets are two popular datasets utilized for testing DL methods. These are datasets composed of black background and the content encoded in white pixels. These grayscale images share an immense scale of pixel correlation similarity. However, the Fashion-MNIST encodes ten different types of clothes, whereas MNIST ten different handwritten digits. In this experiment, MNIST is the OOD, whereas the fashion-MNIST is the training data, as vice versa is a far easier task [1].

To get the latent representation for the training data, a simple model (table 7) is trained until the validation accuracy converges at a 91.9%.(appendix E.1 for the training process). Once the softmax layer is removed, the rest of the classifier model serves as a mapping function for the training data into the latent space. Namely, the model's parameter gradients are nullified (appendix H). Ten identical NF (Real-NVP [4]) are trained on the embeddings of ten different classes of the same training data utilized for the classifier training.

The architecture of the NF consist of a simple MLP layer (table 14) where the base distribution is a multivariate standard Gaussian with the masking as in appendix H. Adam optimizer ($lr = 5 * 10^{-6}, wd = 10^{-6}$) was utilized in training both the classifier as well as the NF(s) (further details in (appendix E.1)). Since each NF is dedicated to one of the classes, each NF is trained using the embeddings of their respective classes, namely first NF is trained solely on the training embeddings from the first class only, the second NF solely on the second class only embeddings until the tenth NF.

To ensure the reproducibility of the method, a series of ten different experiments were conducted using ten different random seeds. To evaluate the method's performance at detecting OOD data, receive operating characteristic (ROC), and precision-recall curve (PRC) are reported.

The models have been randomly initialized at the beginning of the training process. The training data has been randomly shuffled for each of the ten experiments.

These metrics account for both errors (type I and II). Reporting the area under the curve (AUC) is more transparent as it shows the overall performance for the classification (ID vs. OOD) by marginalizing the threshold likelihood value that bounders this decision.

The proposed method outperforms the other methods while reaching an accuracy similar to the likelihood ratio. It is crucial to notice that the proposed method has a much smaller number of parameters than the one utilized in the likelihood ratio [28]. Furthermore, the likelihood ratio method [28] ensures this performance by simulating OOD exposure for tuning the hyper-parameters μ and λ. Eventually, leaving the method quite susceptible to the quality of the simulated OOD data.

E.1 Training details for the fashion-MNIST vs. MNIST

The model utilized as a classifier is in table 7 where the last layer (Linear-8) provides the embedding that is then fed through a softmax.

This model is then trained using the Adam optimizer using a learning rate of 10^{-4} with weight decay 10^{-6} and with $\beta_1 = 0.8$ and $\beta_2 = 0.999$. A batch size of 256 is applied for both test and train data. No augmentation or regularization is applied to the training process.

Table 7: A custom-designed model for the experiment on the fashion-MNIST vs. MNIST dataset.

Layer (Type)	Matrix	Nr Parameters
Conv2d-1	[64,28,28]	640
BatchNorm2d-2	[64,28,28]	128
Conv2d-3	[128,14,14]	73,856
BatchNorm2d-4	[128,14,14]	256
Conv2d-5	[256,5,5]	295,168
BatchNorm2d-6	[256,5,5]	512
Linear-7	[16]	16,400
Linear-8	[10]	170

Table 8: Fashion MNIST (ID) vs. MNIST (OOD) performance.

Experiment	1	2	3	4	5	6	7	8	9	10
AUC-ROC $^{(\%)\uparrow}$	99.2	99.3	99.3	99.1	99.3	99	99.3	99.1	99.3	99.4
AUC-PRC $^{(\%)\uparrow}$	99.4	99.5	99.5	99.3	99.5	99.2	99.5	99.3	99.4	99.4

Table 9: Difference masking for the NF.

Mask Type	1	2	3	4
Mask figure	fig. 10	fig. 11	fig. 12	fig. 13
AUC-ROC $^{(\%)\uparrow}$	96.2	98.8	99	99.4
AUC-PRC $^{(\%)\uparrow}$	97	98.5	99.3	99.4

F Experiments on CIFAR-10 image dataset

CIFAR-10 [15] and SVHN [9] are two popular datasets utilized in the deep learning community to test the proposed models' predictive output. In the first experiment, a Reset-35 model [10] is trained on the ID data as a classifier (appendix F.1) until validation accuracy converges at a 99.9%. Once the validation accuracy is saturated, the softmax layer is removed, and the rest of the model's parameters are frozen. Afterward, ten

different NFs (appendix H) with identical architecture are trained on the same CIFAR-10 training set until their corresponding log-likelihood converges. As in the previous experiments, each NFs is trained exclusively on the corresponding class-specific training data.

F.1 Training details for the CIFAR-10 vs. SVHN

The classifier model utilized in this case is a Resnet-34 [10]. This model is trained using stochastic gradient descent (SGD) with a cyclical learning rate starting at 10^{-1} with a cosine annealing operation with a periodicity of 200. Furthermore, the momentum is 0.9 while the weight decay $5 * 10^{-4}$. A batch size of 256 is applied for both test and train data.

No augmentation or regularization is applied to the training process.

Table 10: CIFAR-10 (ID) vs SVHN (OOD) performance.

Experiment	1	2	3	4	5	6	7	8	9	10
AUC-ROC $^{(\%)\uparrow}$	99.6	99.7	99.3	99.3	99.6	99.5	99.5	99.6	99.6	99.2
AUC-PRC $^{(\%)\uparrow}$	99.5	99.6	99.4	99.2	99.4	99.3	99.5	99.5	99.3	99

Table 11: SVHN (ID) vs CIFAR-10 (OOD) performance.

Experiment	1	2	3	4	5	6	7	8	9	10
AUC-ROC $^{(\%)\uparrow}$	99.6	99.8	99.8	99.6	99.7	99.7	99.7	99.6	99.7	99.6
AUC-PRC $^{(\%)\uparrow}$	99.4	99.6	99.6	99.2	99.5	99.5	99.4	99.3	99.5	99.2

G Experiments on Google genome dataset.

Another essential dataset where OOD detection is of primary importance is the genome dataset published by Google [28]. Bioinformatics has accelerated the understanding of new bacteria straight from the genome sequence. This understanding, however, requires the model prediction to have very high accuracy. To provide high accuracy in the ID genome, the model has been penalized during the training phase whenever it mistakes the types of bacteria already known from the annotations. Since the OOD genomes are not available during the training phase, it is impossible to penalize the model for this data type; thus, it can easily mistake them for ID.

Eventually, providing the proper diagnosis from just the genome dataset requires very high accuracy from the classifier model. However, it is impossible to ensure high accuracy on ID genome data without handling the appearance of OOD data at test time.

Initially, the classifier model was first to ensure high accuracy on the ID genome data. Therefore, a baseline Resnet [10] with 5.5 million (M) parameters was trained until it converges at a 91% accuracy on the ID validation data (appendix G.1). Afterward, the softmax layer is removed, and the rest of the model's weights are frozen.

Since the number of annotated classes on the ID training data is ten, an equal number of NF have been trained on the embeddings of the same training data. The architecture of each NF and the training setup is kept the same as in the previous two experiments. Each of the ten NF is trained exclusively in their respective class from training data until the log-likelihood converges.

Table 12: Performance comparison on the genome dataset where the mean performance is reported together with the variance in the brackets. Apart from our method result, the rest of the results are from the [28]

Method	AUC-ROC (%)↑	AUC-PRC (%)↑
Our Method (Ours)	**84.1 (0.9)**	**85.9 (0.8)**
Likelihood Ratio (μ)	73.2 (1.5)	71.9 (1.7)
Likelihood Ratio (μ, λ)	75.5 (0.1)	71.9 (0.6)
Mahalanobis distance	52.5 (1)	50.3 (0.7)
$p(\hat{y}\|x)$ with calibration	66.9 (0.5)	63.5 (0.4)
Adjusted ODIN	69.7 (1)	67.1 (1.2)
Ensemble, 5 classifiers	68.2 (0.2)	64.7 (0.2)
Ensemble, 10 classifiers	69 (0.1)	65.5 (0.2)
Ensemble, 20 classifiers	69.5 (0.1)	65.9 (0.1)
Pretrained ViT [6]	77.49 (0.04)	-

The OOD dataset was split into training validation and test in the previous work published by [28]. However, the validation set is rendered obsolete since the simple NFs architecture suffices, and no hyperparameter tuning is required. Therefore, the validation data is merged with the test set in this case.

The proposed architecture outperforms most previously proposed methods (table 12). Furthermore, it has a very high margin improvement relative to the SOTA method (table 12).

Despite the performance improvement, the reported performance on the genome dataset is still far from the performance provided in the image dataset examples (table 2, table 3). One can notice that the image training data have the distribution originating from a completely different domain relative to the OOD dataset. Thus the statistical similarity of the OOD image data relative to the weights of their respective classifier models is very different. As a result, their mapping in the latent space tends to be closer to the center and far from the embeddings of the ID data. Eventually, the image dataset is easier to tell apart from the ID relative to the OOD ones.

On the one hand, the difficulty scale in the genome dataset is more pronounced since both the ID and OOD data originate from the same domain while having different class content. Due to the domain similarity, the statistical properties between the model's weight and the OOD genomes have a lower discrepancy. As a result, part of the OOD

could be extended to the area where the training data are embeddings (fig. 9) and not collapsing entirely to the center of the latent space. Eventually, this could lead to a decrease in the OOD performance.

G.1 Training details for the Genome dataset

Resent baseline of 5.5 million parameters is the classifier for the Adam optimizer of with a learning 10^{-4} and a weight decay 10^{-6} and $\epsilon = 10^{-8}$ with a batch size of 256. No augmentation or regularization is applied to the training process.

Table 13: Genome dataset OOD.

Experiment	1	2	3	4	5	6	7	8	9	10
AUC-ROC $^{(\%)\uparrow}$	86	83.7	84.5	82.3	83.5	83.7	83.6	84.5	84.7	84.6
AUC-PRC $^{(\%)\uparrow}$	87.9	85.5	86.3	85.4	85.3	85.4	85.1	86	86.2	85.3

H Training details for the normalizing flow

H.1 Training normalizing flow

Modeling the density of the training data embeddings is equivalent to the uncertainty estimation. These density estimation models emulate a map of the data density over the continuity of the latent space. Namely, it extrapolates a high likelihood solely in regions where data are frequently encountered and low values if otherwise. NF is a suitable paradigm for density estimation since these models perform likelihood estimation for a given data while keeping dimensionality intact [26]. Although an NF requires more parameters relative to a traditional parametric model (i.e., multivariate Gaussian), these models can accommodate a large scale of nonlinearity in the shape of the target distribution.

Each of the NF is trained on data embeddings instead of raw data. However, traditional DL methods map individual data into the latent space without interacting with other data. As a result, the independence assumption also holds for the embedded data in the latent space. Furthermore, since each NF is trained from the embeddings of the designated clusters, one can assume that latent data originating from the same cluster originate from the same (identically) distribution. The preservation of i.i.d for the embeddings offers the possibility to train NF with embedded data in the same setting as traditional DL models.

In the current setup of the NF, the base distribution ($P_X(x)$) is accessible. In contrast, the target distribution (distribution of the training data in the latent space) is unknown ($P_U(u)$). However, it is possible to draw samples from the target distribution (i.e., embedding training data are simulated samples). Hence, the NF training objective maximizes the likelihood of the training data embeddings [26]. As a result, given the training data in the latent space (u), adjust the parameters (θ) of a bijective transformation

function (i.e., neural network $T_\theta^{-1}(x)$) that projects this data into regions of high volume inside the base distribution ($P_X(x)$) (eq. (37)). Whenever the data are projected into the tails of the base distribution, a high magnitude gradient would update the parameters (θ) of the neural network $T_\theta^{-1}(\bullet)$) (fig. 14). This would enable a volumetric transformation ($u = T_\theta^{-1}(x)$) of the base distribution ($P_X(x)$) to match the target density ($P_U(u)$) (eq. (37)).

$$P_X(x) = P_U(T_\theta^{-1}(x))\left|det\{J_{T_\theta^{-1}}(T_\theta^{-1}(x))\}\right| \tag{37}$$

In order to ensure a smooth volumetric transformation, the base distribution initially should possess a maximum entropy. A higher entropy of the base distribution would increase the possibility of a more considerable overlap with the target distribution. The transformation would require a small amount of rotation, scaling, and translation in a good overlap between the base and the target distribution. On the other hand, a low entropy base distribution could potentially have a low overlap with the target distribution and thus require more stretching and translation for the volumetric transformation.

The classifier tries to maximize the mean of the individual cluster in the training data embeddings while minimizing the variance of each of these clusters (section 2.1). Hence, we choose Gaussian as the base distribution since it conveys the maximum entropy for a dataset characterized by a specific mean and variance ([2]).

H.2 Reproducibility details

The NF in this approach is a real-valued non-volume preserving (real NVP) transformation where the affine coupling was the first introduced at [4] as in eq. (38).

$$y = m_i \odot x + (1 - m_i)\{x \odot exp(NN_t(m_i \odot x)) + NN_s(m_i \odot x)\} \tag{38}$$

where two different neural network NN_s and NN_t of the same architecture. The mask operation m_i is the row from the a given mask (fig. 10, fig. 11, fig. 12, fig. 13).

Different masks were utilized for different experiments in this work. The white cells symbolize one, whereas the black cells symbolize 0.

Table 14: The architecture for the both (NN_s and NN_t) of the neural network utilized in the affine coupling eq. (38)

Layer (Type)	Matrix shape	Parameters
MLP1	[10,256]	2560
ReLU1	[0]	0
MLP2	[256,256]	16384
ReLU2	[0]	0
MLP3	[256,10]	2560

Adam is the optimizer for the training of the NF with a learning rate of $5 * 10^{-6}$ and weight decay of 10^{-6} with a batch size of 256.

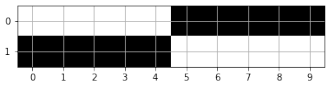


Fig. 10: Mask 1 dim=[2x10].

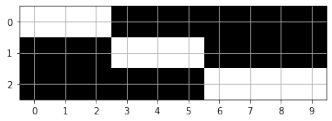


Fig. 11: Mask 2 dim=[3x10].

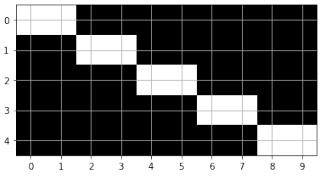


Fig. 12: Mask 3 dim=[4x10].

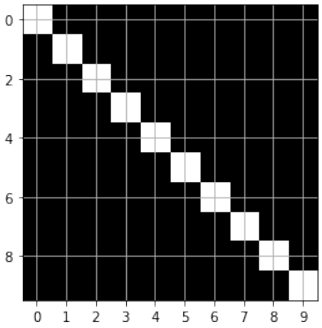


Fig. 13: Mask 4 dim=[10x10].

This framework is used in all three experiments (appendix E.1 appendix F.1 appendix G.1).

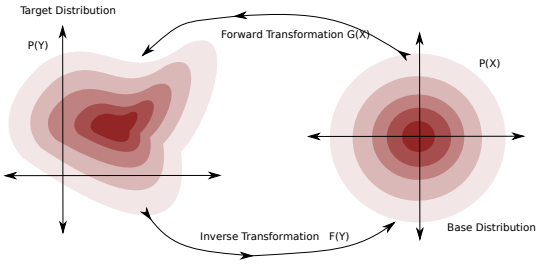


Fig. 14: Normalizing flow in a nutshell

References

[1] Joost van Amersfoort et al. *Uncertainty Estimation Using a Single Deep Deterministic Neural Network*. 2020.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[3] Stefan Depeweg et al. *Decomposition of Uncertainty in Bayesian Deep Learning for Efficient and Risk-sensitive Learning*. 2017.

[4] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using Real NVP". In: *CoRR* abs/1605.08803 (2016).

[5] Zhen Fang et al. "Is Out-of-Distribution Detection Learnable?" In: (2022).

[6] Stanislav Fort, Jie Ren, and Balaji Lakshminarayanan. *Exploring the Limits of Out-of-Distribution Detection*. 2021.

[7] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. *Analyzing and Improving Representations with the Soft Nearest Neighbor Loss*. 2019.

[8] Yarin Gal and Zoubin Ghahramani. "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. PMLR. New York, New York, USA: PMLR, June 2016, pp. 1050–1059.

[9] Ian J. Goodfellow et al. *Multi-digit Number Recognition from Street View Imagery using Deep Convolutional Neural Networks*. 2014.

[10] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015.

[11] Dan Hendrycks and Kevin Gimpel. "A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks". In: (2016).

[12] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. *Deep Anomaly Detection with Outlier Exposure*. 2018.

[13] Dan Hendrycks et al. *Using Self-Supervised Learning Can Improve Model Robustness and Uncertainty*. 2019.

[14] Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. *Why Normalizing Flows Fail to Detect Out-of-Distribution Data*. 2020.

[15] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: ().

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-100 (Canadian Institute for Advanced Research)". In: ().

[17] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. 2016.

[18] Yann LeCun and Corinna Cortes. "MNIST handwritten digit database". In: (2010).

[19] Kimin Lee et al. *A Simple Unified Framework for Detecting Out-of-Distribution Samples and Adversarial Attacks*. 2018.

[20] Shiyu Liang, Yixuan Li, and R. Srikant. *Enhancing The Reliability of Out-of-distribution Image Detection in Neural Networks*. 2020.

[21] Hao Liu and Pieter Abbeel. *Hybrid Discriminative-Generative Training via Contrastive Learning*. 2020.

[22] Jeremiah Liu et al. “Simple and Principled Uncertainty Estimation with Deterministic Deep Learning via Distance Awareness”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 7498–7512.

[23] Weitang Liu et al. *Energy-based Out-of-distribution Detection*. 2020.

[24] Andrey Malinin, Bruno Mlodozeniec, and Mark Gales. *Ensemble Distribution Distillation*. 2019.

[25] Eric Nalisnick et al. *Do Deep Generative Models Know What They Don't Know?* 2018.

[26] George Papamakarios et al. *Normalizing Flows for Probabilistic Modeling and Inference*. 2021.

[27] Tim Pearce, Alexandra Brintrup, and Jun Zhu. *Understanding Softmax Confidence and Uncertainty*. 2021.

[28] Jie Ren et al. *Likelihood Ratios for Out-of-Distribution Detection*. 2019.

[29] Chandramouli Shama Sastry and Sageev Oore. *Detecting Out-of-Distribution Examples with In-distribution Examples and Gram Matrices*. 2020.

[30] Jihoon Tack et al. *CSI: Novelty Detection via Contrastive Learning on Distributionally Shifted Instances*. 2020.

[31] Meet P. Vadera, Brian Jalaian, and Benjamin M. Marlin. *Generalized Bayesian Posterior Expectation Distillation for Deep Neural Networks*. 2020.

[32] Meet P. Vadera et al. *URSABench: Comprehensive Benchmarking of Approximate Bayesian Inference Methods for Deep Neural Networks*. 2020.

[33] Apoorv Vyas et al. *Out-of-Distribution Detection Using an Ensemble of Self Supervised Leave-out Classifiers*. 2018.

[34] Jim Winkens et al. *Contrastive Training for Improved Out-of-Distribution Detection*. 2020.

[35] Ev Zisselman and Aviv Tamar. *Deep Residual Flow for Out of Distribution Detection*. 2020.