APPENDIX

Supplementary Materials

A. Definition of label Noise

In this paper, we employed two distinct categories of datasets: those containing real label noise and those with synthetic label noise. In the case of the CIFAR-100 dataset, we created artificial noise by implementing the techniques detailed in the study conducted by [38]. Conversely, the mini Webvision dataset is a real noisy dataset. This strategy enables us to assess the efficacy of our approach on datasets featuring both synthetic and real label noise, thereby offering a comprehensive insight into its performance across diverse scenarios.

B. NCOD+

To use NCOD+ for the ensemble network architecture experiments, we have also employed two additional regularization terms as used in [38]. The definition of these two regularizations are taken directly from their work and are used to improve the consistency and class balance of the network's predictions. These regularization terms are the consistency regularizer L_C and class-balance regularizer L_B. Both of these terms are defined using the Kullback-Leibler divergence and help to improve the accuracy of the network's predictions by encouraging consistency and preventing the network from assigning all data points to the same class. The final loss function for NCOD+ is constructed by combining these regularization terms with our original loss function.

C. Network structures and hyperparameters

In our experiments, we utilized the Torch library version 1.12.1 and used specific hyperparameters and network types for NCOD and NCOD+ on different datasets. These details are provided in table IV. It is worth noting that for particular experiments, retraining the last saved model for a small number of additional epochs, e.g., from 5 to 10, improved the performance of the model. This highlights the effectiveness of our method even when applied to pre-trained models. Also, to avoid computational overhead, we calculated the mean representation of each class only once for each epoch, particularly at the beginning of each epoch.

D. Experimental Settings

In our experiments on CIFAR-100, we use simple data augmentations including random crop and horizontal flip, following the same approach used in previous works ([15];[6]). To improve the performance of our method, NCOD+, we also use unsupervised data augmentation techniques as described in ([50]) to create additional views of the data for consistency training. For the mini WebVision, we first resize the images to 256x256, and then perform a random crop to 227x227, along with a random horizontal flip. All images are standardized by means and variances to ensure consistency during the experiments. In our experiments, we use the Stochastic Gradient Descent (SGD) optimizer without weight decay for parameter U. We keep all the hyperparameters fixed for different levels of label noise to ensure fair comparison across experiments. To perform a fair comparison; we use the same settings of hyperparameters and architectures for both NCOD and NCOD+. Table IV provides a detailed description of the used hyperparameters.

	CIFAR-100			mini webvision	
Architecture	ResNet34	PreActResnet18	PreActResnet34	InceptionResNetV2	InceptionResNetV2, ResNet-50 both(pretrained)
batch size	128	128	128	32	256
learning rate	0.02	0.02	0.02	0.02	0.1
lr decay	[80,120] multistep gamma =0.1	Cosine Annealing	Cosine Annealing	[50] multistep gamma =0.1	init 0.1 LambdaLr warmup =5
weight decay	$5 * 10^{-4}$	$5 * 10^{-4}$	$5 * 10^{-4}$	$5 * 10^{-4}$	0.001
training epochs	150	300	300	100	25
training examples	50 k	50k	50k	$66k$	$66k$
lr for u	sym =0.1	sym = 0.1 Asym =0.3	sym = 0.1 Asym =0.3	0.3	0.3
wd for u	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$	$1e^{-8}$
init. std. for u	$1e^{-9}$	$1e^{-9}$	$1e^{-9}$	$1e^{-9}$	$1e^{-9}$
λ^c	0.0	0.9	0.9	0.0	0.0
λ^b	0.0	0.1	0.1	0.0	0.0

TABLE IV: The table shows the hyperparameters used for our experiments, which were kept consistent across all experiments to ensure a fair comparison with other methods, particularly SOP ([38]). This consistency in hyperparameters allows for an accurate comparison of the performance of our method with respect to other methods and helps to eliminate any bias that may be introduced by variations in the experimental setup. The table provides a detailed description of the settings that were used to conduct the experiments, which is useful for reproducing the results and for understanding the experimental conditions under which our method was evaluated. It's also worth noting that, in case we retrain from the previous best-saved model, the learning rate for 'u' is given as high as 3. This is done to fine-tune the model and improve its performance. Additionally, The values of λ^c and λ^b are the coefficients for consistency regularization and class balance regularization respectively, as used in the experiments done by [38]. These coefficients are used to balance the trade-off between consistency and class balance in the model, and their values are determined through experimentation.

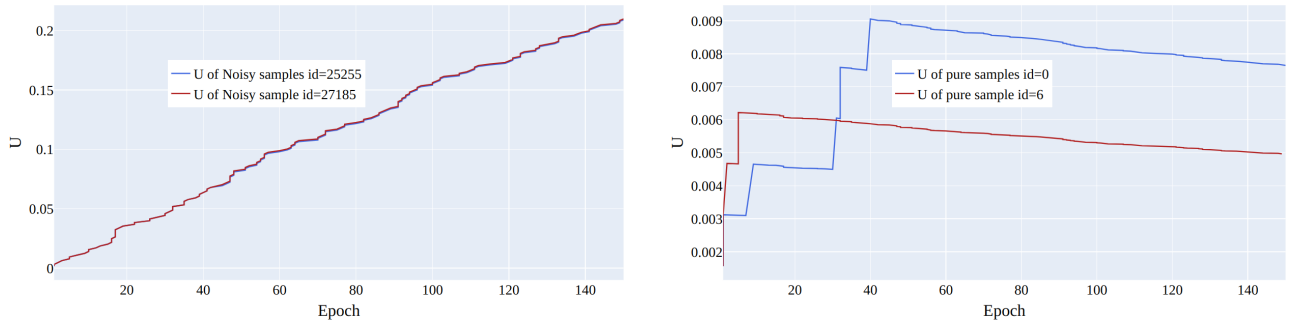
E. Additional details

It is important to note that in Figure 6, the average similarity of samples starts at around 0.20 instead of 0. This is because, in the first epoch, the class representative embeddings are initialized randomly, resulting in different similarity values and soft labels for each sample. This ensures that the noisy labeled data has no initial advantage and can begin learning. As training continues, the similarity of pure samples increases and predictions improve, but at some point, it starts overfitting, and accuracy decreases. However, the inclusion of u in our method prevents overfitting by removing the effect of noisy samples in the cross-entropy loss. As previously stated, in the first few epochs, even the similarity of pure samples is a bit low, causing u to learn for them as well. But eventually, as their similarity increases, u starts decreasing for pure samples. In the case of noisy samples, as their similarity drops, their predictions decrease, and u increases to compensate for the effect of this in $\mathcal{L}_1$ and to decrease the loss of $\mathcal{L}_2$. The change in u can be seen in Figure 10 for 2 noisy and 2 pure samples and their corresponding similarities can be seen in Figure 11.

FURTHER ANALYSIS OF OUR TRAINING STRATEGY

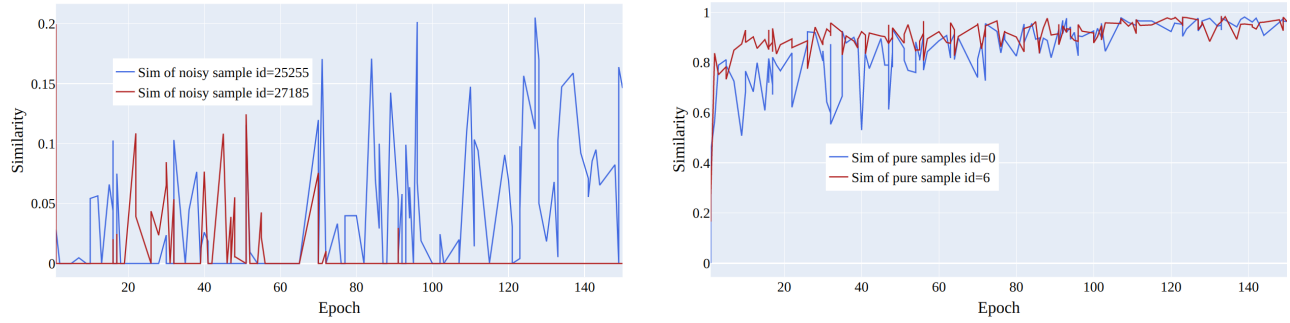
For each class, c, the set of samples in the training dataset that have class c as their label can be partitioned into C subsets, where each subset is comprised of examples belonging to the same clean label. To make learning from data possible, we must assume that each subset of samples with the corrupted label in c should be smaller than the subset having the clean label of c. If that is not the case, then the observed label is independent of the data, and no learning is possible. More formally, let n_c be the number of samples for class c in the training dataset. For each class c, we denote by $n_{c,c'}$ the number of samples labeled as c whose correct label is c' instead, and by $n_{c,c}$ the number of samples whose labels are correctly labeled as c. The above assumption can thus be expressed as $n_{c,c} \geq n_{c,c'} \; \forall \; c' \neq c, \; c' \in \{1, \ldots, C\}$.

Our training procedure can be explained by considering three consecutive stages.



(a) Noisy samples (b) Pure samples

Fig. 10: The figure depicts the learning of parameter u for two noisy and two pure samples in 50% symmetrical noise for CIFAR-100



(a) Noisy samples (b) Pure samples

Fig. 11: The figure shows how similarity changes during training for two noisy and two pure samples in 50% symmetrical noise for CIFAR-100

a) Initialization.: At the beginning of the learning phase, since the weights are initialized randomly, all the errors on the training samples are of comparable magnitude. More formally, let n be the number of samples of the training set; then, we can write the loss as

$$\begin{aligned}
&\sum_{i=1}^{n} \mathcal{L}_\theta(f(\theta, x_i) + u_i \cdot y_i, \tilde{y}_i) \\
&= \sum_{c=1}^{C} \sum_{j=1}^{n_c} \mathcal{L}_\theta(f(\theta, x_j) + u_j \cdot y_j, \tilde{y}_j) \\
&= \sum_{c=1}^{C} \sum_{c'=1}^{C} \sum_{j=1}^{n_{c,c'}} \mathcal{L}_\theta(f(\theta, x_j) + u_j \cdot y_j, \tilde{y}_j)
\end{aligned} \tag{4}$$

Given that initially, the representations for each sample are chosen randomly, the values of the loss $\mathcal{L}_\theta(f(\theta, x_j) + u_j \cdot y_j, \tilde{y}_j)$ for each sample belonging to the class c, are all similar. For this reason, the major contribution to the final value of the loss is given by the largest subset of samples, which in this case is the subset of samples correctly labeled. To minimize the loss, thus, the best strategy consists of minimizing the loss on that subset of samples, and the gradient will point to the direction mostly aligned with their gradients. Therefore, the updated representations will be influenced mostly by the samples with the correct labels.

b) Neighborhood Similarity and Avoiding Memorization.: The similarity between samples and their class embeddings is the primary factor contributing to the loss due to two phenomena. Firstly, correct samples are more likely to be close to the right class embedding, and secondly, their subset carries more weight in determining the average location in the embedding space. Higher similarity with the class embedding leads to a larger contribution to the loss. However, the model tends to overfit on noisy labels, which is addressed by introducing the outlier discounting parameter u in our loss, reducing the weight of samples with high u values.

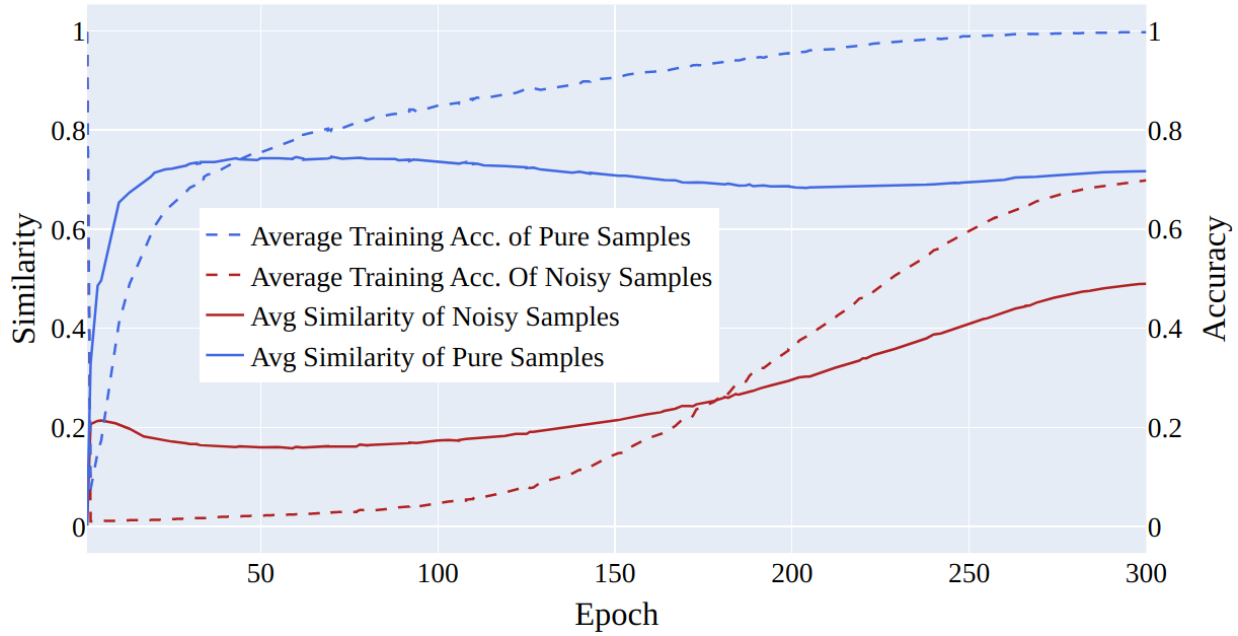


Fig. 12: Shows the model initially learns for pure samples which correspond to higher similarities and after approx. 100 epochs it significantly starts learning for smaller insignificant similarities corresponding to noisy samples. We obtained the plot using 50% symmetrical noise.

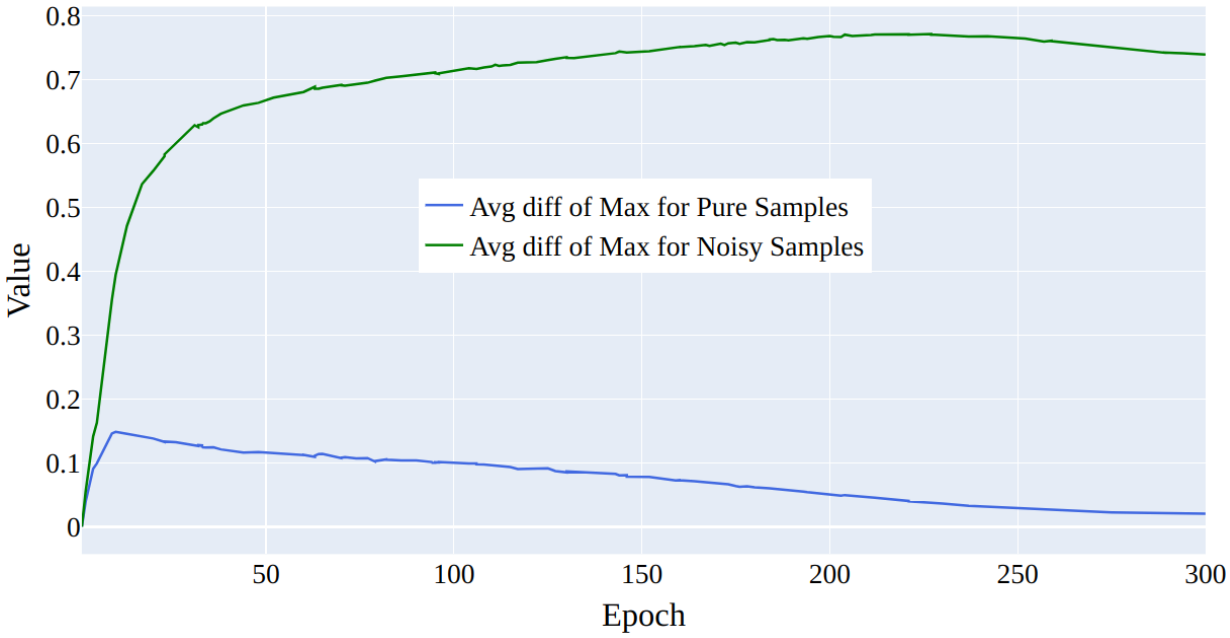


Fig. 13: Average difference between the probability of the predicted class and the probability assigned to the label in the dataset for both noisy and pure samples.

In fig. 13 we plot the computed average of the difference between the probability assigned to the predicted class and the probability assigned to the true label for both noisy and pure samples. More in detail, given a sample (x_i, y_i) we are considering a network output represented by the vector $f(\theta, x_i)$ of C components after applying softmax. Here, y_i represents the label the sample has in the dataset, for the sample, denoted by c. We can denote the individual components of the vector as $f(\theta, x_i)_j$ for $j = 1, ..., C$.

The quantity that we are plotting is given by the difference between the maximum value of $f(\theta, x_i)_j$ across all

j and the component $f(\theta, x_i)_c$ corresponding to the label y_i. In other words, we are computing the expression: $\max(j = 1, ..., Cf(\theta, x_i)_j) - f(\theta, x_i)_c$

Theorem (V.1)**.** *Let (x_i, y_i) be a sample, θ^t be the parameters of the network at epoch t, and u_i^t be the parameter for outlier discounting relative to sample i at epoch t. Let $\hat{c}_i^t$ be the prediction of the network at time t, and y_i be the class of sample i. Suppose $\hat{c}_i^t = y_i$, then $u_i^{t+1} < u_i^t$ when $u_i^t \neq 0$, and $u_i^{t+1} = 0$ otherwise. When $\hat{c}_i^t \neq y_i$, $u_i^{t+1} \geq u_i^t$. Moreover $u_i^t < 1$ for $t \in \mathbb{N}$.*

Proof. We write $\hat{c}_i(t)$, to emphasize that also the class predicted by the network depends on the epoch since the network parameters are updated accordingly eq. (2). The update rule for u will be

$$\begin{aligned} u_i^{t+1} &= u_i^t - \beta\, \partial_{u_i} \mathcal{L}_2 \\ &= u_i^t - 2\frac{\beta}{C}(\delta_{c_i, \hat{c}_i(t)} - 1 + u_i^t) \end{aligned} \tag{5}$$

$\delta_{c_i, \hat{c}_i}$ is 1 if the label predicted by the network coincides with the label of the sample in the dataset while it is 0 otherwise. So the update becomes:

$$u_i^{t+1} = \begin{cases} u_i^t(1 - 2\frac{\beta}{C}) \text{ if } \hat{c}_i(t) = c_i \\ u_i^t(1 - 2\frac{\beta}{C}) + 2\frac{\beta}{C} \text{ if } \hat{c}_i(t) \neq c_i \end{cases}$$

Notice that in our setting β is the learning rate used to learn the outlier discounting and C is the number of classes, so $2\frac{\beta}{C} < 1$. From the equation above it follows that if $u_i^0 = 0$,

$$u_i^{t+1} = 2\frac{\beta}{C} \sum_{k=0}^{t} \left(1 - 2\frac{\beta}{C}\right)^k \left(1 - \delta_{\hat{c}_i(t-k), c_i}\right)$$

From this writing, we can see that the maximum of the sum is reached when predictions are always wrong. In this case, the sum becomes a geometric sum with a ratio smaller than one and we obtain $\lim_{t\to\infty} u_i^t = 1$. We can also notice that if the classes coincide u^t is multiplied by a value less than one, so it either decreases in magnitude or remain at zero if its initial value was zero. If the prediction is incorrect, since $u_i^t < 1$, u^{t+1} increases. Indeed $u_i^t(1 - 2\frac{\beta}{C}) + 2\frac{\beta}{C} > u_i^t$ if and only in $u_i^t < 1$. □

F. Proof Remark V.2

Proof. Let c be a constant, $0 < c < 1$, we consider the function $f(x) = \log(x + c) - (1 - c)\log(x)$. For $x > 0$, f is continuous and derivable and $f'(x) = \frac{1}{x+c} - \frac{1-c}{x}$. Studying the sign of the derivative we can observe that $f'(x) \geq 0$ if $x \geq 1 - c$. It follows that f has a minimum in $x = 1 - c$ and since $1 - c < 1$, $f(1 - c) = -(1 - c)\log(1 - c) > 0$, namely the function f is always positive, and the desired inequality is valid. □

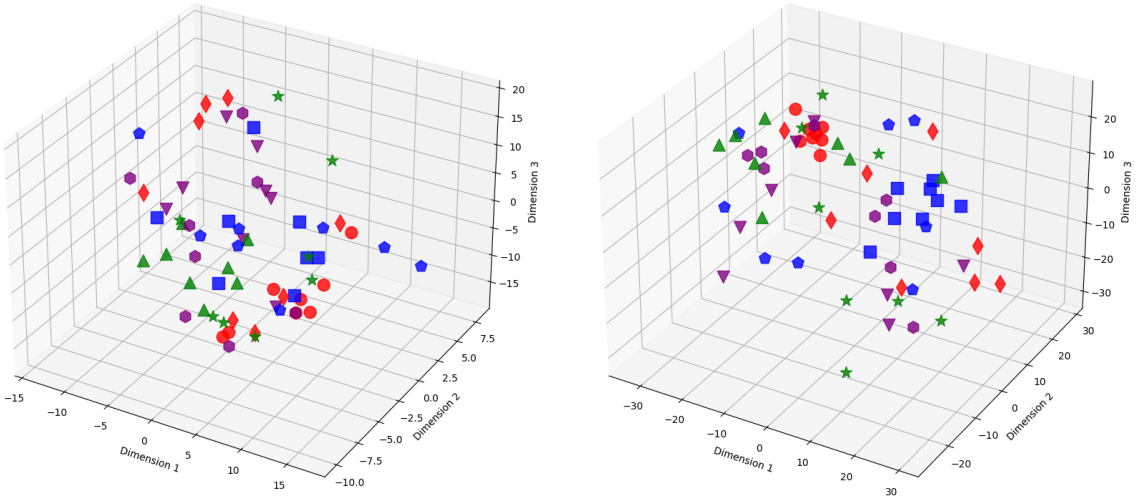
The 3D visualizations in appendix F show the evolution of latent representations in the penultimate layer of the model during the training process. The figures in this section show the shifts of these representations for training samples selected from the CIFAR-100 dataset with 20% symmetric noise for different epochs of training using our loss, NCOD. In order to make the plots easier to understand we choose to consider only samples belonging to four classes, in particular classes 0,1,2,3 from CIFAR 100 that correspond to "Apple", "Aquarium Fish", "Baby" and "Bear" respectively.

Different colors represent classes. For each color, there are two different shapes that distinguish noisy and pure samples: Blue (square: pure, pentagon: noisy), Red (circle: pure, diamond: noisy), Green (triangle-up: pure, star: noisy), and Purple (triangle-down: pure, hexagon: noisy).

In this section, we included the missing figure showing the latent space for epochs 1 and 14 when using our loss function appendix F. We also add an enlarged version fig. 1, that is fig. 15. As can be seen from the figures, in the first epoch, for both losses, the arrangement of samples in latent space is causal. Later, around epoch 14 we begin to see clusters related to the corrected samples (also for both losses) while the noisy samples are scattered in space. Later in training, we can see completely different behaviour between our loss and cross entropy, indeed at epoch 96 for cross entropy loss four clusters related to the four classes were formed, and the noisy examples are also part of these clusters, see subfigure (c) in fig. 15. Using our loss instead the clusters relative to the four classes are still visible but it can be observed that noisy samples stay at a distance from clusters related to classes, subfigure (d) in fig. 15.

We utilize t-SNE to reduce the 512 dimensional embeddings to a 3D dimensional embedding. Despite the reduction in dimensionality the formation of clusters and the different behavior of latent representations of noisy and pure samples is still visible.

The results in the previous section show a qualitative idea of what happens in training using the two losses. To get a quantitative measure Of the behavior of noisy and clean samples with respect to the formation of clusters in latent space we



(a) Epoch 1

(b) Epoch 14

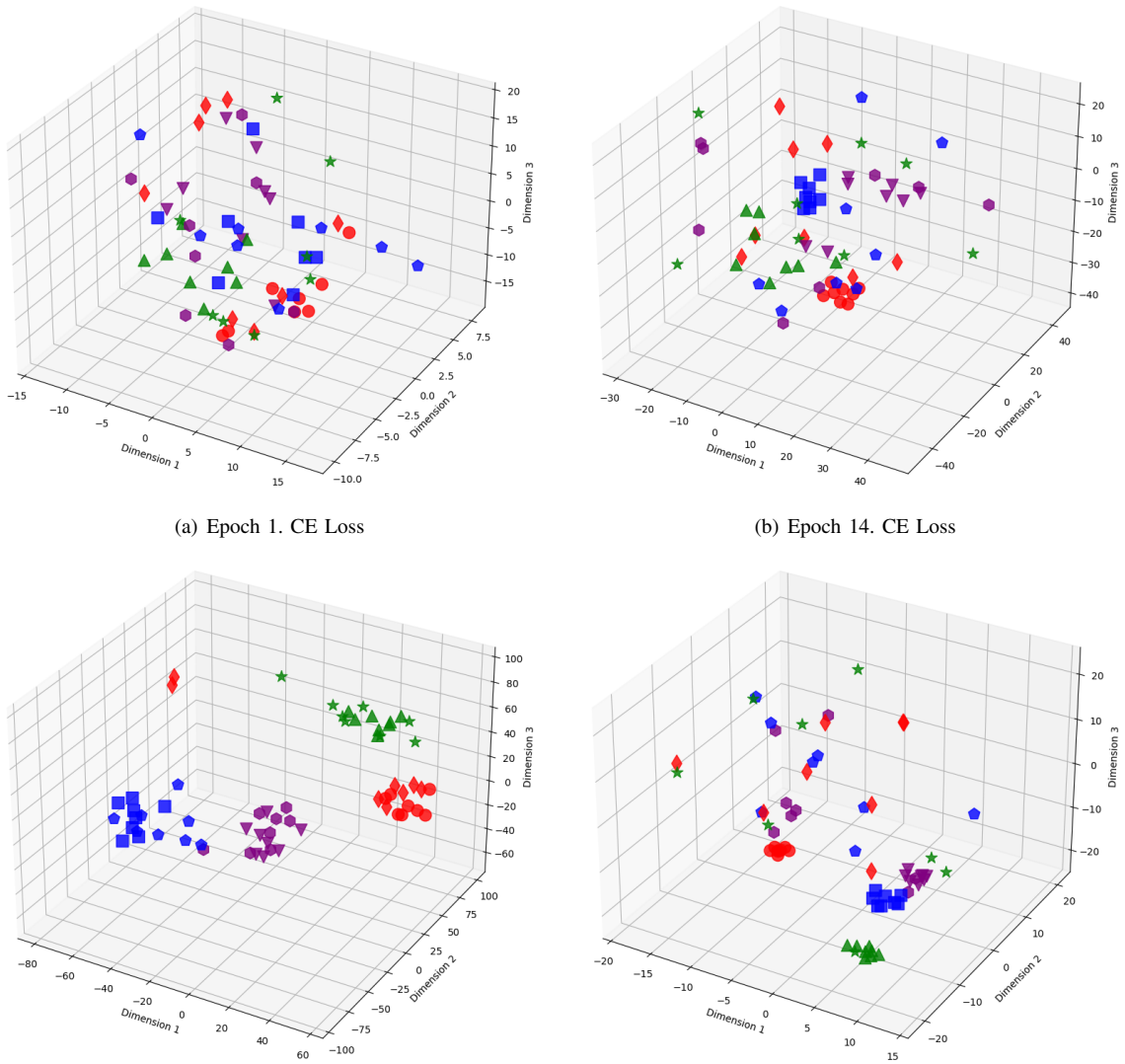
Fig. 14: The figure illustrates sample embeddings of four classes from CIFAR-100 with 20% symmetrical noise at different training epochs using our loss function (NCOD). Colors represent classes, and shapes distinguish noisy and pure samples: Blue (square: pure, pentagon: noisy), Red (circle: pure, diamond: noisy), Green (triangle-up: pure, star: noisy), and Purple (triangle-down: pure, hexagon: noisy).

chose to measure the distribution of samples around the centres of the clusters, the seeds, as explained in the main paper. The appendix F shows the distribution of the distance in latent space between the seed of each of the four classes chosen for fig. 1 and the other samples belonging to those classes for CIFAR 100 dataset with 20% symmetrical noise.

We can see that for both our loss and cross entropy loss at the beginning of training the distribution of distances at epoch 1 is unimodal, see fig. 1 subfigure (a) and (e), while at epoch 18, for both, we can see that some classes have a bimodal distribution, we have two peaks in the data, indicating the presence of two different groups, subfigure (b) and (f). For our loss, fig. 1 subfigure (f) the first peak is centred respectively around 15-18 depending on the classes, while the second peaks for all the classes are centred around 25-30, depending on the classes. Plotting the same plot only for noisy and clean sample separated shows exactly that for noisy samples is a unimodal the peak is exactly around 25-30 while the distribution for the pure samples is unimodal with a peak centred around 15. This shows that the two groups of the bimodal distribution are exactly noisy and clean samples. This can be observed in more detail in fig. 18 which shows the distributions of distances from seeds separately for noisy and clean samples.

Examining fig. 19 and delving into the insightful distribution analyses of sample embeddings conducted on the CIFAR-100 dataset, an intriguing parallel emerges with the results seen in the MNIST dataset. Notably, both the cross-entropy (CE) and our novel NCOD loss function initially exhibit analogous behavior, commencing with a unimodal distribution as captured in (a) and (e) of fig. 19, then transitioning through training to a bimodal distribution as depicted in (b) and (f) of fig. 19. However, as training progresses further the distribution of embeddings for the cross-entropy method begins to shift back towards a unimodal state, shown in (c), the embeddings under NCOD retain their bimodal nature, as demonstrated in (g). This captivating pattern persists, culminating in the eventual complete unimodality of distribution for CE, as seen in (d), juxtaposed against the unwavering bimodality of NCOD, magnificently depicted in (h). It's important to underscore the significance of this behavior. A bimodal distribution signifies a perceptive distinction between pure and noisy samples, while a unimodal distribution lacks this discriminatory power. This revelation adds an extra layer of brilliance to our findings.

To establish the robustness of our analysis across diverse datasets and architectures, we replicated the investigation we did on CIFAR 100 in fig. 7 on the MNIST dataset using a CNN. More precisely the network we used is made of 2 Convolution layers (kernel size equal to 3, number of channels for the first layer is 16 and for the second 32) followed by a MaxPool (kernel size equal to 2) layer and followed by 2 fully connected layers (the first fully connected layer has dimension 128). Our results reveal a noticeable contrast in training accuracy between pure and noisy samples. Initially, pure samples exhibit superior training accuracy, whereas noisy samples display markedly lower accuracy. Upon employing the cross-entropy approach, we observed a gradual reduction in the disparity of training accuracies as the training proceeded. This phenomenon contributed to a decrease in test accuracy. Conversely, when utilizing our loss function, the gap in training accuracy between pure and noisy samples either remained stable or expanded. This dynamic led to an improvement in test accuracy. For a comprehensive



(a) Epoch 1. CE Loss

(b) Epoch 14. CE Loss

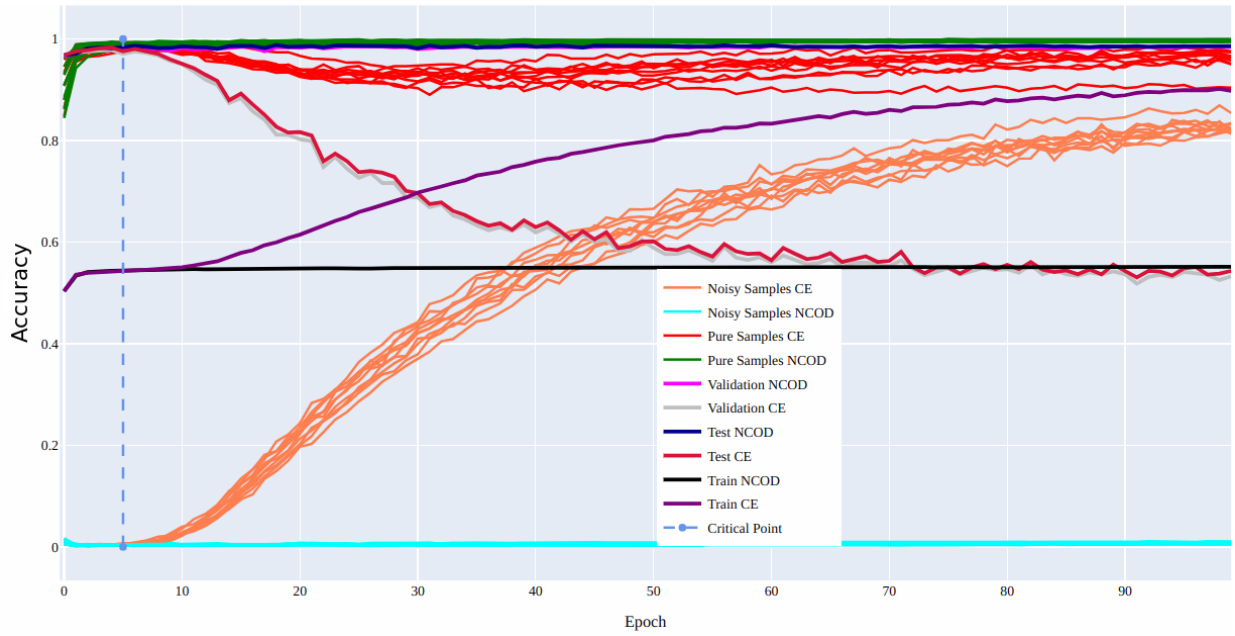
(c) Epoch 96. CE Loss

(d) Epoch 96. NCOD Loss

Fig. 15: Sample embeddings of four classes from CIFAR-100 with 20% symmetrical noise. Colors represent classes, and shapes distinguish noisy and pure samples: Blue (square: pure, pentagon: noisy), Red (circle: pure, diamond: noisy), Green (triangle-up: pure, star: noisy), and Purple (triangle-down: pure, hexagon: noisy).

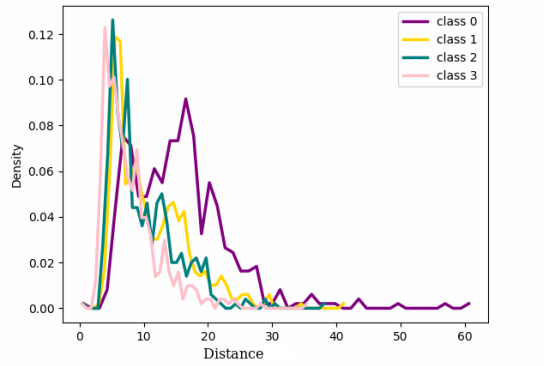
exploration of these outcomes, please consult fig. 16(a). The figure presents a comparison of training for CE loss and NCOD loss for the ten different classes in the MNIST dataset. We also plot the test accuracy for both loss functions.

In this figure, there's a specific point labeled as "critical". This is where things get interesting. When we use cross-entropy, the training accuracy for nearly all the noisy samples in every class starts to go up at this critical point. This increase in training accuracy, however, leads to a drop in the accuracy of the model's predictions during testing. On the other hand, with our unique loss function (NCOD), it's a different story. This loss function ensures that the training accuracy for noisy samples doesn't increase as it does with the cross-entropy method. Instead, the model primarily learns from clean, noise-free samples as evident from the figure. This special behavior contributes to an overall improvement in the accuracy of the model's predictions during testing.

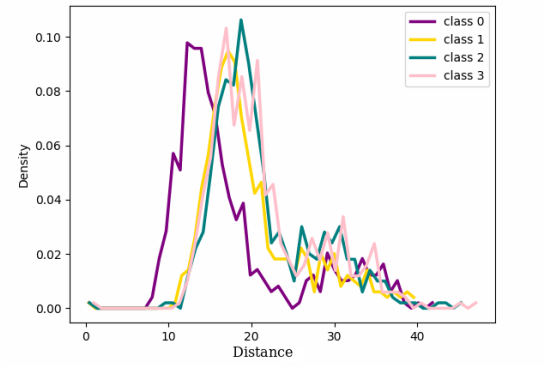


(a) MNIST with 50% symmetric noise

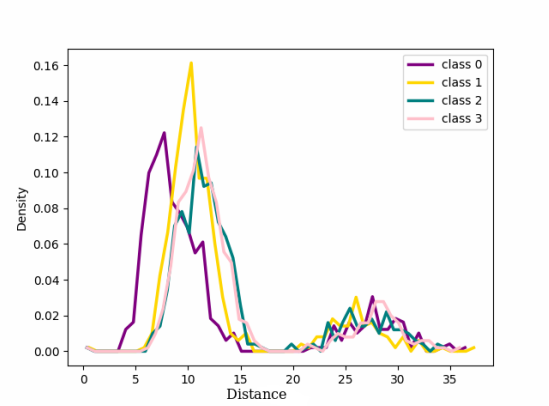
Fig. 16: The figure shows how our loss function (NCOD) performed compared to Cross entropy (CE).



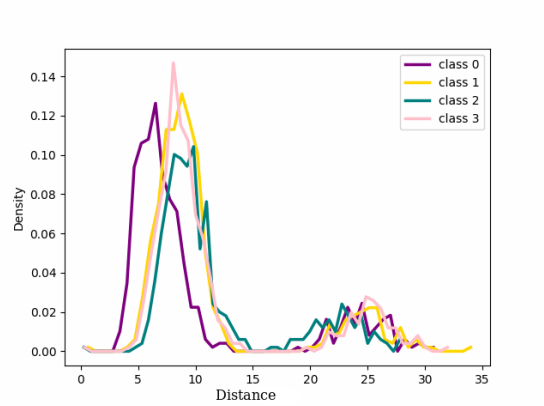
(a) Epoch 1. Cross Entropy Loss



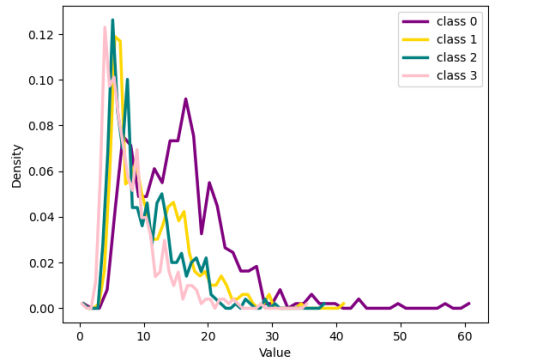
(b) Epoch 18. Cross Entropy Loss



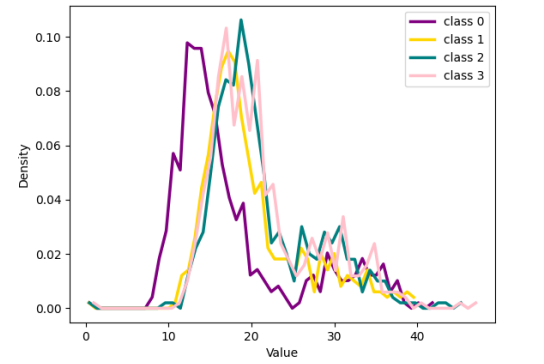
(c) Epoch 96. Cross Entropy Loss



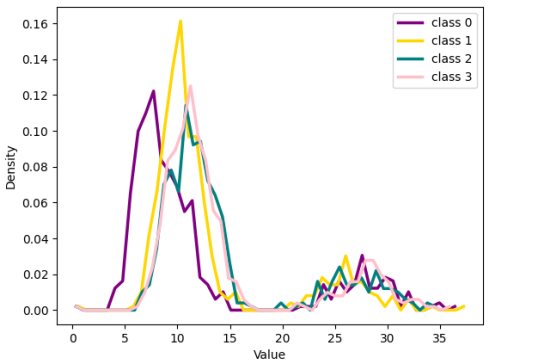
(d) Epoch 124. Cross Entropy Loss



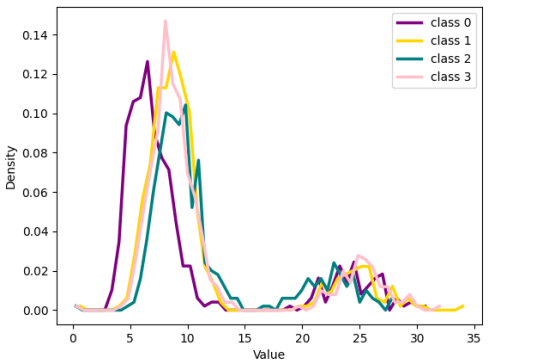
(e) Epoch 1. NCOD Loss



(f) Epoch 18. NCOD Loss

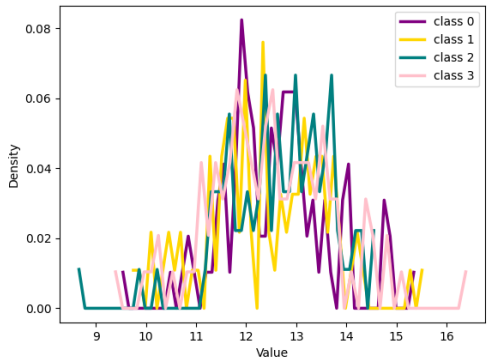


(g) Epoch 96. NCOD loss

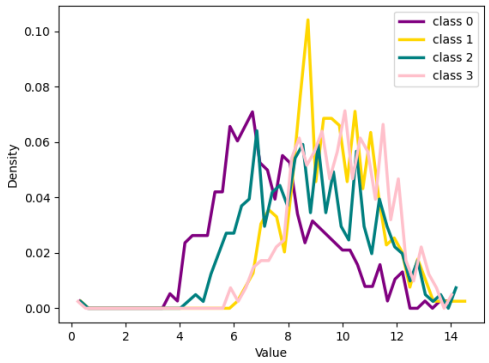


(h) Epoch 124. NCOD loss

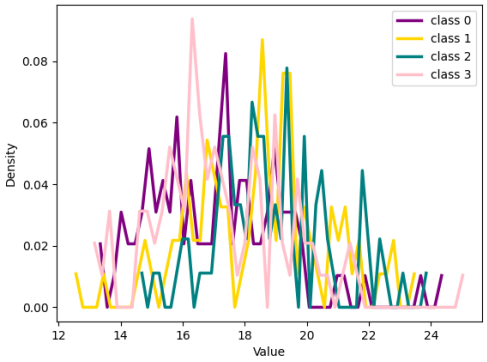
Fig. 17: The figure illustrates the distribution of four classes from CIFAR-100 with 20% symmetrical noise at different training epochs using our loss function and cross-entropy loss function.



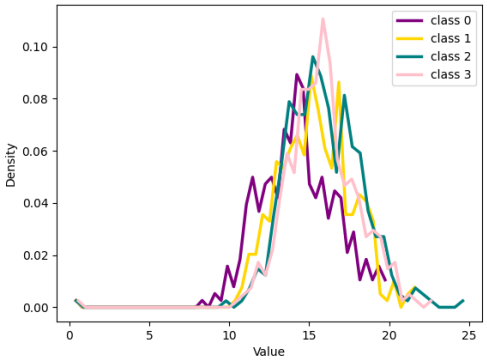
(a) Epoch 18. CE loss. Noisy Samples



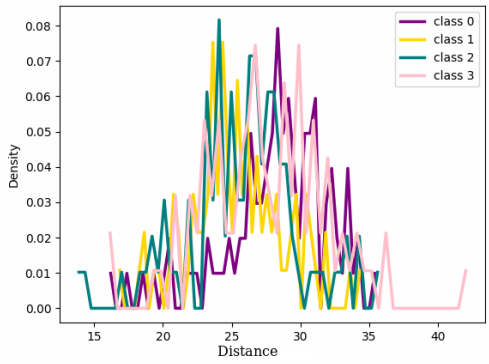
(b) Epoch 18. CE loss. Clean Samples



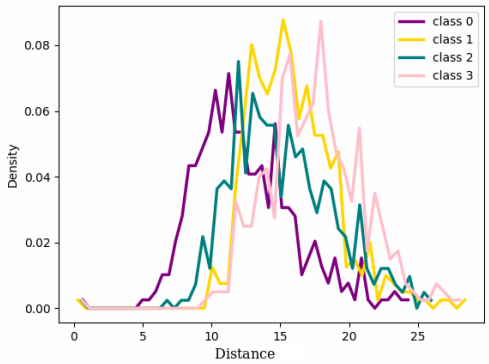
(c) Epoch 96. CE loss. Noisy Samples



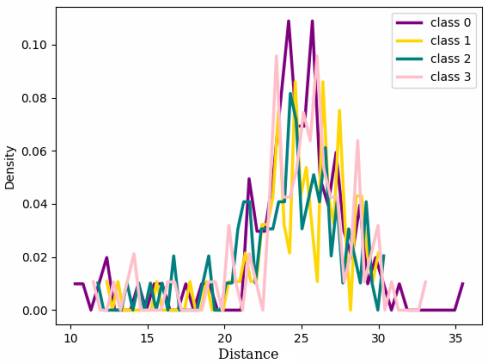
(d) Epoch 96. CE loss. Clean Samples



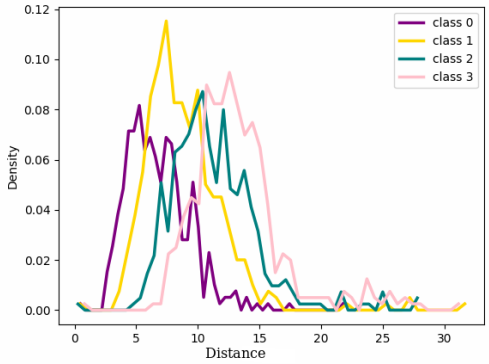
(e) Epoch 18. NCOD loss. Noisy Samples



(f) Epoch 18. NCOD loss. Clean Samples

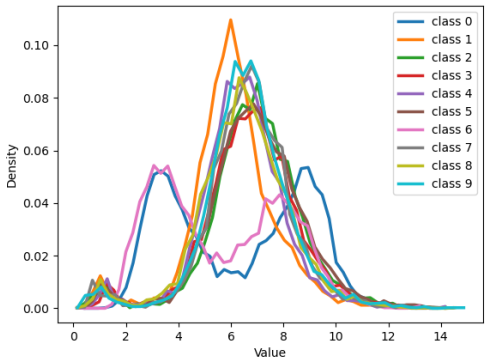


(g) Epoch 96. NCOD Loss. Noisy Samples

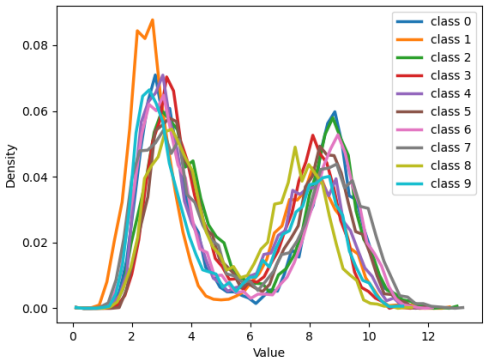


(h) Epoch 96. NCOD loss. Clean Samples

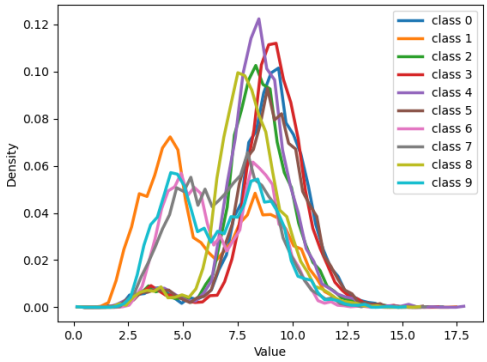
Fig. 18: The figure illustrates the distribution of the four classes from CIFAR 100 with 20% symmetrical noise for noisy and clean samples respectively at different training epochs using Cross Entropy loss or NCOD loss function.



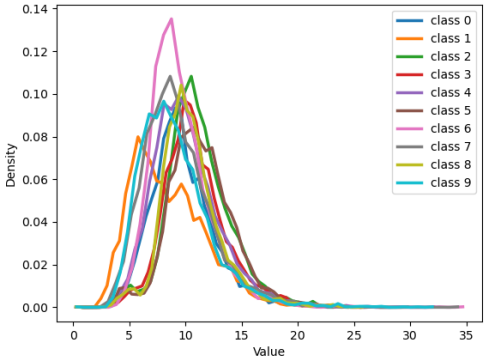
(a) Epoch 0. Cross entropy Loss



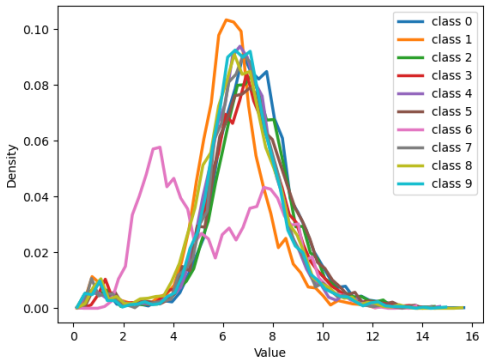
(b) Epoch 1. Cross entropy Loss



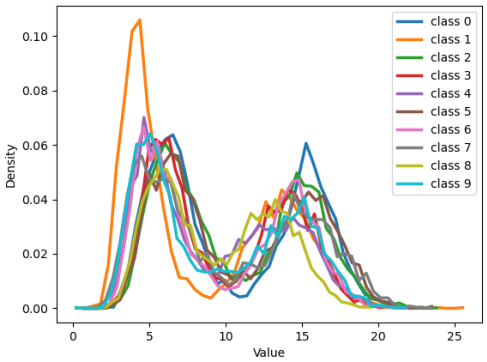
(c) Epoch 10. Cross entropy Loss



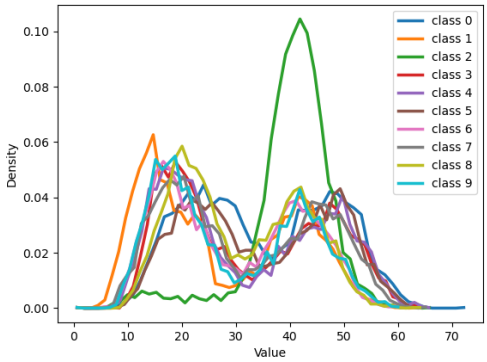
(d) Epoch 20. Cross entropy Loss



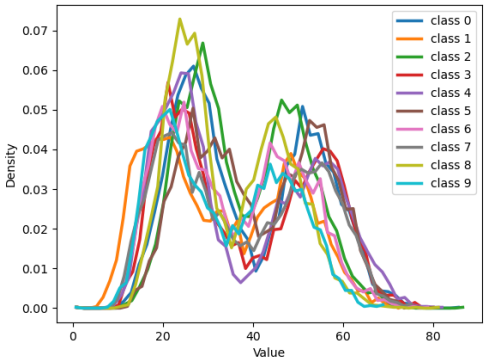
(e) Epoch 0. NCOD loss



(f) Epoch 1. NCOD loss



(g) Epoch 10. NCOD loss



(h) Epoch 20. NCOD loss

Fig. 19: The figure illustrates the distribution of all classes from MNIST with 50% symmetrical noise at different training epochs using Cross Entropy loss and NCOD loss.