# Supplementary Material

# Appendix

## Table of Contents

# Organization of Appendix

In this appendix, we first present a table summarizing the major notations used by the main paper. Next, we provide detailed information about the training process and hyperaprameters setting. We provide the detailed proof of Lemma 1 and Theorem 2 in Section C. After that, we provide additional experimental details and results. Finally, we discuss the broader impacts, limitations, and future work of our DRE technique. The link to the source code can be found in the end of the Appendix.

# A Summary of Notations

Table 4 below shows the major notations used in the main paper. We further assign each notation into one of four major categories: dataset, DRO formulation, sparse training, and theoretical results.

Table 4: Symbols with Descriptions.

| Symbol Group | Notation | Description |
|---|---|---|
| Dataset | $\mathbf{X}$ | Set of training images |
| | $\mathbf{Y}$ | Set of training class labels |
| | $\mathbf{C}$ | Total classes |
| | $\hat{y}$ | Predicted class label |
| | $N$ | Total number of training samples |
| | $D$ | Dimensionality of each data sample |
| DRO | $D_f$ | $f$-divergence |
| | $\eta$ | Parameter controlling size of uncertainty set in DRO framework |
| | $z_n$ | Weight associated with $n^{th}$ data sample |
| Sparse Training | $M$ | Number of sparse sub-networks |
| | $\mathcal{K}$ | Density of the given network |
| | $\Theta$ | Parameter associated with given neural network |
| | $\hat{p}$ | Confidence associated with predicted class |
| | $l(\mathbf{x}_n, \Theta)$ | Loss associated with $n^{th}$ data sample |
| Theoretical Results | $\beta$ | Learning rate of the given network |
| | $P$ | Total number of patches in each data sample |
| | $d$ | Dimensionality of each patch |
| | $\mathbf{v}_{c,l}$ | Major $l^{th}$ feature associated with class c |
| | $L$ | Total number of features in each class class |
| | $D_N^S$ | Collection of single-view data samples |
| | $D_N^M$ | Collection of multi-view data samples |
| | $\cup$ | Collection of features |
| | $H$ | Number of convolution layers |
| | $F_c(\mathbf{x})$ | Logistic output for the $c^{th}$ class for the data sample $\mathbf{x}$ |
| | $\mathcal{P}_{\mathbf{v}_{c,l}}$ | Collection of patches containing feature $\mathbf{v}_{c,l}$ in sample $\mathbf{x}_j$ |
| | $\texttt{SOFT}_c$ | Softmax output for class $c$ |

# B Robust Loss Optimization in DRO

In this section, we first provide a detailed description on how we optimize the robust loss function in (1). We then explain how to set the uncertainty set by choosing a proper hyperparameter.

## B.1 Robust Loss Optimization

The optimization problem specified in (1) involves an inequality constraint so directly solving it may incur a higher computational overhead. Therefore, we consider a regularized version of the robust loss to train each base learner by using the following loss:

$$\mathcal{L}^{Robust} = \max_{\mathbf{z}\geq\mathbf{0},\mathbf{z}^\top\mathbb{1}=1} \sum_{n=1}^{N} z_n l_n(\Theta) - \lambda D_f\left(\mathbf{z}||\frac{\mathbb{1}}{N}\right) \tag{9}$$

where $l_n(\Theta) = l(\mathbf{x}_n, \Theta)$. Solving the above maximization problem leads to a closed-form solution for $\mathbf{z}^*$ as shown by the following lemma:

**Lemma 3.** *Assuming that $D_f$ is the KL divergence, then solving* (9) *leads to the following solution*

$$\mathcal{L}^{Robust} = \sum_{n=1}^{N} z_n^* l_n(\Theta) \tag{10}$$

*where $z_n^*$ is given by*

$$z_n^* = \frac{\exp\left(\frac{l_n(\Theta)}{\lambda}\right)}{\sum_{j=1}^{N} \exp\left(\frac{l_j(\Theta)}{\lambda}\right)} \tag{11}$$

It can be verified that there is a one-to-one correspondence between $\eta$ in (2) and $\lambda$ in (9). Given their roles in the corresponding equations, a large $\eta$ implies a small $\lambda$ and a small $\eta$ implies a large $\lambda$.

### B.2 Hyperparameter settings

The hyperparameter in the regularization term is chosen based on the difficulty of a dataset. Specifically, for DRE, we always consider the $\lambda \to \infty$ for the first sparse sub-network which is equivalent to Expected Risk Minimization (ERM). For the second and third sub-networks, we choose this hyperparameter based on the difficulty of data samples. It should be noted that we need to set higher $\lambda$ values for more difficult datasets as difficult samples are more common on those datasets. Using this notion, for Cifar10, we choose small $\lambda$ values so that the model can focus on the difficult samples that are few. For this, we choose $\lambda = 10$ for the second sparse sub-network and $\lambda = 500$ for the third sparse sub-network. Considering Cifar100 is more difficult, we would have more difficult samples and therefore higher $\lambda$ value is preferred. For this, we choose $\lambda = 50$ for the second sparse sub-network and $\lambda = 500$ for the third one. In the case of TinyImageNet, we have many difficult samples and therefore we choose relatively large $\lambda$ values. Specifically, we choose $\lambda = 100$ for the second sparse sub-network and $\lambda = 1,000,000$ for the third sparse sub-network.

## C Theoretical Proof

In this section, we provide detailed proofs of the theoretical results presented in the main paper.

### C.1 Proof of Lemma 1

*Proof.* For $y_n = c$, with respect to data sample $\{\mathbf{x}_n, y_n\}$, the gradient can be evaluated as

$$-\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{x}_n, y_n) = [1 - \texttt{SOFT}_c(F(\mathbf{x}_n))] \sum_{p\in[P]} \texttt{ReLU}[\langle \Theta_{c,h}, \mathbf{x}_n^p \rangle] \mathbf{x}_n^p \tag{12}$$

Assume that the given sample has a major feature $\mathbf{v}_{c,l}$, taking dot product with respect to $\mathbf{v}_{c,l}$ on both side of (12) leads

$$\langle -\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{x}_n, y_n), \mathbf{v}_{c,l} \rangle = [1 - \texttt{SOFT}_c(F(\mathbf{x}_n))] \sum_{p\in[P]} \langle \texttt{ReLU}[\langle \Theta_{c,h}, \mathbf{x}_n^p \rangle] \mathbf{x}_n^p, \mathbf{v}_{c,l} \rangle \tag{13}$$

Let's further assume that the feature set is orthonormal: $\forall c, c', \forall l \in [L], ||\mathbf{v}_{c,l}||_2 = 1$ and $\mathbf{v}_{c,l} \perp \mathbf{v}_{c',l'}$ when $(c,l) \neq (c',l')$. Using $\mathbf{x}^p = a^p \mathbf{v}_{c,l} + \sum_{\mathbf{v}' \in \cup \setminus \mathbf{v}_c} \alpha^{p,\mathbf{v}'} \mathbf{v}' + \epsilon^p$ given in (4), we have

$$\langle -\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{x}_n, y_n), \mathbf{v}_{c,l} \rangle = [1 - \texttt{SOFT}_c(F(\mathbf{x}_n))] \left( \sum_{p\in\mathcal{P}_{v,l}(\mathbf{x}_n)} \texttt{ReLU}[\langle \Theta_{c,h}, \mathbf{x}_n^p \rangle a^p] + \sum_{p\in[P]} \langle \epsilon^p, \mathbf{v}_{c,l} \rangle \right) \tag{14}$$

It should be noted that the term *i.e.,* $\sum_{v' \in \cup \setminus \mathbf{v}_c} \alpha^{p,v'} \langle \mathbf{v}', \mathbf{v}_{c,l} \rangle$ becomes zero due to the orthogonal properties of the feature set. Let us represent the second term by $\kappa$: $\sum_{p\in[P]} \langle \epsilon^p, \mathbf{v}_{c,l} \rangle = \kappa$. Then, we have

$$\langle -\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{x}_n, y_n), \mathbf{v}_{c,l} \rangle = (1 - \texttt{SOFT}_c(F(\mathbf{x}_n))) \left( \sum_{p\in\mathcal{P}_{v,l}(\mathbf{x}_n)} \texttt{ReLU}[\langle \Theta_{c,h}, \mathbf{x}_n^p \rangle a^p] + \kappa \right) \tag{15}$$

Furthermore, let us define $V_{c,h,l}(\mathbf{x}_j) = \sum_{p \in \mathcal{P}_{\mathbf{v}_{c,l}}(\mathbf{x}_j)} \texttt{ReLU}(\langle \Theta_{c,h}, \mathbf{x}_j^p \rangle a^p)$ then above equation further reduces to following

$$\langle -\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{x}_n, y_n), \mathbf{v}_{c,l} \rangle = (1 - \texttt{SOFT}_c(F(\mathbf{x}_n)))(V_{c,h,l}(\mathbf{x}_n) + \kappa) \tag{16}$$

Recall the above equation is the gradient with respect to the $n^{th}$ data sample. Considering the gradient with respect to all data samples with $y_n = c$, and let us consider the total loss, where the weight $z_n$ of each loss is assigned according to a distribution specified by the uncertainty set $\mathcal{U}$. Then, the total gradient is

$$\langle -\nabla_{\Theta_{c,h}} l(\Theta; \mathbf{X}, \mathbf{Y}), \mathbf{v}_{c,l} \rangle = \max_{\mathbf{z} \in \mathcal{U}} \sum_{n=1}^{N} z_n \left[ \mathbb{1}_{y_j = c}(V_{c,h,l}(\mathbf{x}_n) + \kappa)(1 - \texttt{SOFT}_c(F(\mathbf{x}_n))) \right] \tag{17}$$

Now using the standard gradient update rule with $\beta$ being the learning rate, we have

$$\langle \Theta_{c,h}^{t+1}, \mathbf{v}_{c,l} \rangle = \langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle + \beta \max_{\mathbf{z} \in \mathcal{U}} \sum_{n=1}^{N} z_n \left[ \mathbb{1}_{y_j = c}(V_{c,h,l}(\mathbf{x}_n) + \kappa)(1 - \texttt{SOFT}_c(F(\mathbf{x}_n))) \right] \tag{18}$$

Let $\mathbf{x}_k \in \mathcal{D}_N^S$ be the most difficult sample having $\mathbf{v}_{c,l}$ as the main feature. Also, consider $\mathbf{x}_n \in \mathcal{D}_N^M$ to be the easy sample with $y_n = c, y_k = c$. Then, we have

$$[1 - \texttt{SOFT}_c(F(\mathbf{x}_k))] \geq [(1 - \texttt{SOFT}_c(F(\mathbf{x}_n))],\ \forall n \in [1, N], n \neq k, y_n = c \tag{19}$$

Using above property, we can write the following using (18)

$$\begin{aligned} &\langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle + \beta \max_{\mathbf{z} \in \mathcal{U}} \sum_{n=1}^{N} z_n \left[ \mathbb{1}_{y_j = c}(V_{c,h,l}(\mathbf{x}_n) + \kappa)(1 - \texttt{SOFT}_c(F(\mathbf{x}_n))) \right] \\ &\leq \langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle + \beta N z_k (1 - \texttt{SOFT}_c(F(\mathbf{x}_k))) \end{aligned} \tag{20}$$

On the r.h.s., we have $z_n = \frac{1}{N}$ for ERM, which assigns equal weights to all samples. Under the assumption of $N_{\mathbf{v}_{c,l}} \ll N_{\cup \setminus \mathbf{v}_{c,l}}$, the contribution of the $N_{\mathbf{v}_{c,l}}$ on overall gradient will be negligible. In contrast, for the DRO framework, using (11), we have

$$z_k = \frac{1}{\sum_{j=1, j \neq k}^{N} \exp\left( \frac{l_j(\Theta) - l_k(\Theta)}{\lambda} \right) + 1} \tag{21}$$

Since $l_k(\Theta) > l_j(\Theta), \forall \lambda > 0, \lambda \neq \infty$, we have $z_k > \frac{1}{N}$. Using r.h.s. of (20) and incorporating $z_k = \frac{1}{N}$ for ERM and $z_k > \frac{1}{N}$, we have

$$\{\langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle + \beta(1 - \texttt{SOFT}_c(F(\mathbf{x}_k)))\}_{ERM} \leq \{\langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle + \beta(1 - \texttt{SOFT}_c(F(\mathbf{x}_k)))\}_{Robust} \tag{22}$$

This subsequently leads to the following:

$$\{\langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle\}_{Robust} > \{\langle \Theta_{c,h}^{t}, \mathbf{v}_{c,l} \rangle\}_{ERM}; \forall t > 0 \tag{23}$$

which completes the proof of Lemma 1. □

### C.2 Proof of Theorem 2

Let $\mathbf{x} \in \mathcal{D}_S^N$ from class c with $\mathbf{v}_{c,l}$ as the main feature and $\mathbf{v}'$ as the dominant feature learned through the memorization. Also consider $\mathbf{v}'$ to be the main feature characterizing class $k$. Then for any class $c'$, we can define the following

$$\texttt{SOFT}_{c'}(\mathbf{x}) = \frac{\exp(F_{c'}(\mathbf{x}))}{\sum_{j \in [C]} \exp(F_j(\mathbf{x}))} \tag{24}$$

In the above equation, $F_{c'}(\mathbf{x})$ can be written as

$$F_{c'}(\mathbf{x}) = \sum_{h \in [H]} \sum_{p \in [P]} \texttt{ReLU}[\langle \Theta_{c',h}, \mathbf{x}^p \rangle] \tag{25}$$

Substituting $\mathbf{x}^p$ from (4), we have

$$F_{c'}(\mathbf{x}) = \sum_{h\in[H]}\sum_{p\in[P]} \texttt{ReLU}\left[a^p\langle\Theta_{c\prime,h}, \mathbf{v}_{c,l}\rangle + \sum_{\mathbf{v}'\in\cup\setminus\mathbf{v}_c} \alpha^{p,\mathbf{v}'}\langle\Theta_{c\prime,h}, \mathbf{v}'\rangle + \langle\Theta_{c\prime,h}, \epsilon^p\rangle\right] \tag{26}$$

Substituting $c'$ by $k$, we have

$$F_k(\mathbf{x}) = \sum_{h\in[H]}\sum_{p\in[P]} \texttt{ReLU}\left[a^p\langle\Theta_{k,h}, \mathbf{v}_{c,l}\rangle + \sum_{\mathbf{v}' in\cup\setminus\mathbf{v}_c} \alpha^{p,\mathbf{v}'}\langle\Theta_{k,h}, \mathbf{v}'\rangle + \langle\Theta_{k,h}, \epsilon^p\rangle\right] \tag{27}$$

In case of ERM, the $\mathbf{v}_{c,l}$ signal is fairly weak during the training process due to $N_{\mathbf{v}_{c,l}} \ll N_{\cup\setminus\mathbf{v}_{c,l}}$. Therefore, the term $\langle\Theta_{k,h}, \mathbf{v}_{c,l}\rangle$ is negligible. Also, the last term $\langle\Theta_{k,h}, \epsilon^p\rangle$ is also small as this corresponds to the Gaussian noise. For the second term $\exists\mathbf{v}'$ for which $\langle\Theta_{k,h}, \mathbf{v}'\rangle$ is very high because of the spurious correlation. In contrast, for the robust loss, using Lemma 1, the model learns a stronger correlation with the true class parameter and therefore $\langle\Theta_{c,h}, \mathbf{v}_{c,l}\rangle$ is high. As such, both terms $\langle\Theta_{k,h}, \mathbf{v}_{c,l}\rangle$ as well as $\langle\Theta_{k,h}, \mathbf{v}'\rangle, \forall v\prime$ becomes low. As a result, we have

$$\{F_k(\mathbf{x})\}_{ERM} > \{F_k(\mathbf{x})\}_{Robust} \tag{28}$$

Substituting this inequality to (24), we have

$$\{\texttt{SOFT}_k(\mathbf{x})\}_{Robust} < \{\texttt{SOFT}_k(\mathbf{x})\}_{ERM} \tag{29}$$

This completes the proof of Theorem 2.

# D Experimental Details and Additional Results

In this section, we first provide a detailed description of datasets used in our experimentation followed by hardware description of our experimentation. Consequently, we provide examples of single-view and multi-view data samples. Next, we provide additional experimental results on Cifar10 and Cifar100 datasets with a $15\%$ density. After that, we provide additional baselines results on TinyImageNet. We also compare our model performance with different calibration techniques commonly used in dense networks. Then, we perform an in-depth ablation study. Parameter size and inference speed are discussed in the subsequent subsection. We also further investigate the diversity of the sparse subnetworks. Finally, we provide detailed qualitative analysis to support our proposed claim.

## D.1 Detailed Dataset Description

For general classification setting, we consider Cifar10, Cifar100 [12], and TinyImageNet [14] datasets. For the out of distribution setting, we consider corrupted version of Cifar10 and Cifar100, which are named as Cifar10-C and Cifar100-C [10], respectively. Finally, for open-set detection, we leverage SVHN [19] as the open-set dataset. The detailed description of each dataset is given below:

- *Cifar10.* This dataset consists of total 10 classes, each consisting of 5,000 training samples and 1,000 testing (evaluation) samples. Each image is a colored image with size $32 \times 32$.
- *Cifar100.* This dataset consists of 20 super classes where each super-class consists of 5 classes resulting into total 100 classes. Each class consists of 500 training samples and 100 testing samples. Each image is a colored image with size $32 \times 32$.
- *TinyImageNet.* The original dataset consists of 200 classes with 1,000,000 samples where each class has 500 training images, 50 validation images, and 50 test images. Each image is a colored image with size $64 \times 64$.
- *Cifar10-C.* Fifteen different types of corruptions are applied on the Cifar10 clean testing dataset where each corruption has 5 severity levels, ranging from 1 to 5 with 1 being least severe and 5 being most severe. The corruptions include Gaussian noise, shot noise, impulse noise, defocus blur, forsted glass blur, motion blur, zoom blur, snow, frost, fog, brightness, contrast, elastic, pixelate, and JPEG.

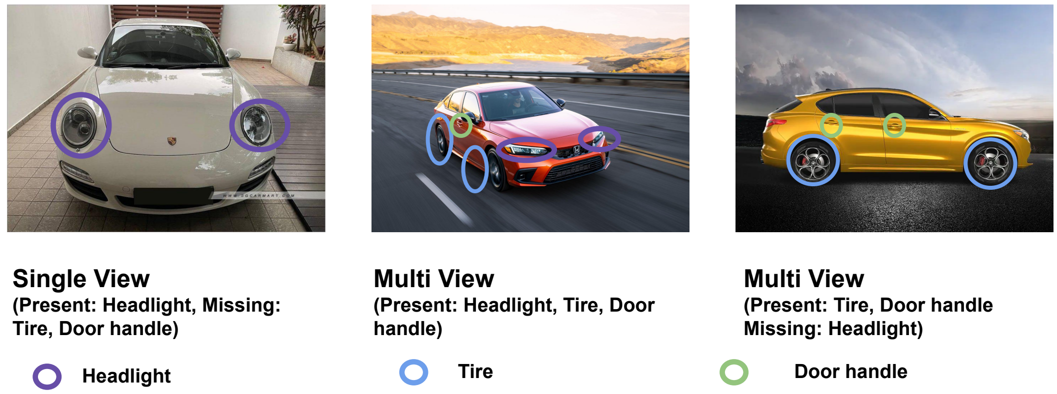



Figure 4: Examples of single-view and multi-view samples.

- *Cifar100-C*. Similar to Cifar10-C, fifteen different corruptions are applied on the Cifar100 clean testing dataset.
- *SVHN*. The Street View House Numbers (SVHN) dataset consists of 10 classes with digit 1 as class 1, digit 9 as class 9 and digit 0 as class 10. These are original, variable-resolution, colored house-number images with character level bounding boxes. We use this dataset as the open-set dataset in our experimentation.

### D.2 Hardware Details for Experimentation

All experimentations are conducted using NVIDIA RTX A6000 GPU with 48GB memory requiring 300 Watt power. For GPU, CUDA Version: 11.6, Driver Version: 510.108.03, and NVIDIA-SMI: 510.108.03 is used. In terms of CPU, our experimentation uses an Intel(R) Xeon(R) Gold 6326 CPU @ 2.90GHz with a 64-bit system and an x86_64 architecture.

### D.3 Single-view and Multi-view Examples

Figure 4 show the three example images, where the first image is a representative single-view data sample whereas the last two are multi-view samples. In this example, we consider three major features for cars: *i.e.,* `Tire`, `Headlight`, and `Door handle`. As only headlight feature is present in the first image, it belongs to the single-view category. For the second and third images, multiple features are presented and therefore we regard those images as multi-view data samples.

### D.4 Additional Result on Cifar10 and Cifar100

Table 5 shows the experimental result on Cifar10 and Cifar100 datasets with a $15\%$ density. As shown, the proposed technique has a far superior performance in terms of the ECE score compared to the competitive baselines. This is consistent with the results with a 9% density as presented in the main paper, which further justifies the effectiveness of our proposed technique.

### D.5 Additional Baseline Results on TinyImageNet

As mentioned in the main paper, the computational issue (*i.e.,* memory overflow) makes it impossible to run sparse learning techniques *i.e.,* CigL [15], DST Ensemble [17], and Sup-ticket [30] on the ResNet101 and WideResNet101 architectures to make a fair comparison. Therefore, in this section, we pick a lower capacity model (ResNet50) and compare the performance. Even for the ResNet50 architecture, CigL still runs into the memory overflow issue with a batch size of 128. Furthermore, lowering the batch size (*e.g.,* 16) makes the training process extremely

Table 6: Additional baseline results on TinyImageNet using ResNet50 with $\mathcal{K} = 15\%$.

| Training Type | Approach | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
|---|---|---|---|
| Sparse Training | *DST Ensemble* | 72.00 | 2.94 |
| | *Sup-ticket* | 68.68 | 10.96 |
| Mask Training | *DRE* | 71.57 | 1.51 |

Table 5: Accuracy and ECE performance with $15\%$ density for Cifar10 and Cifar100 Dataset.

| Training Type | Approach | Cifar10 | | | | Cifar100 | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ResNet50 | | ResNet101 | | ResNet101 | | ResNet152 | |
| | | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
| | *Dense*$^{\dagger}$ | 94.82 | 5.87 | 95.12 | 5.99 | 76.40 | 16.89 | 77.97 | 16.73 |
| Dense Training | *L1 Pruning* | 93.88 | 5.69 | 94.23 | 5.88 | 75.53 | 15.52 | 75.83 | 15.78 |
| | *LTH* | 92.97 | 4.03 | 93.15 | 5.69 | 74.36 | 15.13 | 74.77 | 15.22 |
| | *DLTH* | 95.15 | 6.21 | 95.65 | 6.96 | 77.98 | 16.24 | 78.23 | 16.54 |
| | *Mixup* | 93.22 | 4.02 | 93.38 | 5.68 | 74.48 | 15.10 | 74.68 | 15.16 |
| Sparse Training | *CigL* | 92.25 | 4.67 | 93.34 | 4.59 | 77.88 | 10.16 | 77.27 | 10.62 |
| | *DST Ensemble* | 89.57 | 2.10 | 88.64 | 1.34 | 64.57 | 9.76 | 64.75 | 9.27 |
| | *Sup-ticket* | 94.65 | 3.20 | 94.95 | 3.09 | 78.68 | 10.16 | 78.95 | 10.32 |
| Mask Training | *AdaBoost* | 94.07 | 5.65 | 94.76 | 5.14 | 75.98 | 23.55 | 76.28 | 24.27 |
| | *EP* | 94.41 | 3.90 | 94.42 | 4.07 | 75.66 | 14.79 | 76.05 | 14.79 |
| | *SNE* | 94.85 | 3.05 | 94.96 | 3.18 | 76.82 | 11.12 | 77.23 | 11.63 |
| | ***DRE*** | 94.87 | **1.71** | 94.74 | **1.34** | 75.86 | **4.90** | 76.46 | **5.81** |

slow even using a $48Gb$ GPU, where each training epoch takes more than half an hour, making model training extremely difficult. Therefore, we did not report the performance of CigL. It should be noted that CigL can be trained on Cifar10 and Cifar100 because of lower dimension of the input images and we have already reported its performance in the main paper. Table 6 shows the performance of DRE along with those from DST Ensemble and Sup-ticket on ResNet50. It is clear that DRE achieves better performance compared to these baselines.

### D.6 Performance from Ensemble Members

We investigate how performance varies in different sparse sub-networks. We use Cifar100 as an example and Table 7 report the individual sub-network performance on both accuracy and ECE. While each sparse sub-network is a relatively weaker learner (which is expected), they contribute to the final ensemble model in a complementary way, leading to a better ECE score as well as accuracy.

Table 7: Different subnetworks performance on Cifar100 Dataset.

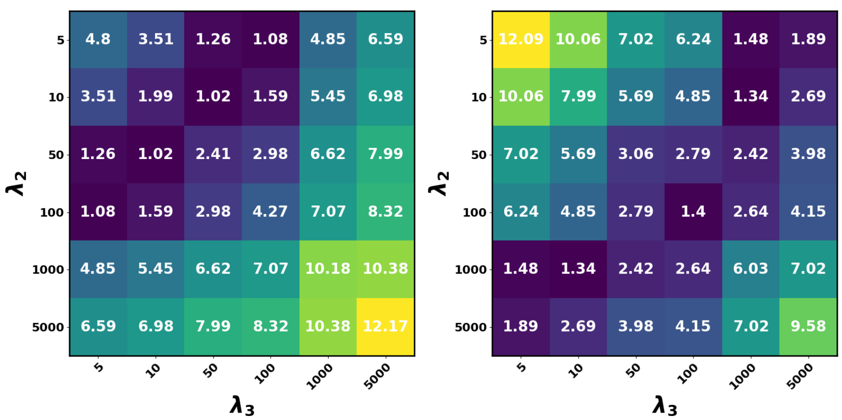
| Subnetworks | ResNet101 | | ResNet152 | |
|---|---|---|---|---|
| | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
| *Subetwork 1 (3%)* | 68.22 | 14.35 | 69.65 | 13.31 |
| *Subetwork 2 (3%)* | 69.03 | 1.39 | 70.00 | 3.39 |
| *Subetwork 3 (3%)* | 72.86 | 11.96 | 70.24 | 14.78 |
| *DRE* | 74.68 | **1.20** | 74.37 | 2.09 |

### D.7 Comparison with Common Calibration Techniques

In this section, we investigate whether existing calibration techniques designed for training dense networks can be leveraged to further improve the calibration performance of sparse networks. However, most of these techniques (*e.g.,* temperature scaling and mix-n-match) are post hoc techniques, which require a separate validation set to fine-tune the parameters. This means we need to further divide the training data into training and validation sets, which may negatively impact the generalization capability of the trained model (due to less training data). To make a comparison, we pick Temperature Scaling (TS) [9], Label Smoothing (LS) [27], and a few other techniques proposed in [31], including Ensemble Temperature Scaling (ETS) and Isotonic Regression One vs All combined with Temperature Scaling (IROvA-TS). We apply these calibration techniques on the top of the EP algorithm. Specifically, as LS does not require a separate validation set, we train it on the full training dataset using the LS loss (with $\epsilon = 0.1$). Other calibration techniques require a separate validation set and therefore we divide training data into training and validation with a 80:20 ratio. EP (No Validation) uses the full training dataset whereas EP (Validation) is trained using $80\%$ of the training data. Once the model is trained with $80\%$ of training data using EP, we further calibrate it using the aforementioned calibration techniques. Table 8 shows the results. There are two key observations: (i) the classification accuracy decreases for all calibration techniques at the expense of improving calibration performance as they require a separate validation set, and (ii) DRE achieves the best ECE in all cases, which further justifies its strong calibration performance.

Table 8: Different calibration techniques on the top of EP Algorithm with $\mathcal{K} = 9\%$.

| Approach | Cifar10 | | | | Cifar100 | | | |
|---|---|---|---|---|---|---|---|---|
| | ResNet50 | | ResNet101 | | ResNet101 | | ResNet152 | |
| | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
| TS | 93.42 | 0.96 | 93.42 | 1.37 | 73.06 | 1.72 | 73.40 | 2.45 |
| ETS | 93.42 | 0.97 | 93.42 | 1.37 | 73.06 | 1.76 | 73.40 | 2.40 |
| IROvA-TS | 89.90 | 1.45 | 88.69 | 0.89 | 60.87 | 1.56 | 60.77 | 2.86 |
| LS | 94.06 | 7.56 | 94.21 | 7.41 | 75.96 | 9.36 | 76.40 | 7.71 |
| EP (No Validation) | 94.20 | 3.97 | 94.35 | 4.03 | 75.05 | 14.62 | 75.68 | 14.41 |
| EP (Validation) | 93.42 | 4.46 | 93.42 | 4.83 | 73.06 | 15.56 | 73.40 | 15.88 |
| ***DRE*** | 94.60 | **0.7** | 94.28 | **0.7** | 74.68 | **1.20** | 74.37 | **2.09** |




(a) $\mathcal{K} = 5\%$ (b) $\mathcal{K} = 3\%$

Figure 5: (a-b) Impact of $\lambda$ on ECE using ResNet101 architecture on Cifar100 dataset.

Table 9: ACC and ECE with different: (a) backbones and (b) number of subnetworks.

| Approach | WideResNet28-10 | | ViT | |
|---|---|---|---|---|
| | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
| *EP* | 94.12 | 4.53 | 86.16 | 10.01 |
| *DRE* | 93.98 | **1.93** | 85.53 | **4.18** |

(a) Different backbones on Cifar10 Dataset.

| Approach | ResNet101 | | ResNet152 | |
|---|---|---|---|---|
| | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ |
| *DRE (M = 3)* | 94.87 | 1.71 | 94.74 | 1.34 |
| *DRE (M = 5)* | 94.79 | 0.84 | 94.69 | 0.62 |

(b) Different $M$ values on Cifar10 with $\mathcal{K} = 15\%$.

### D.8 Ablation Study

In this section, we first show the impact of $\lambda$ values on the prediction and calibration performance. We then investigate how the size of the ensemble affects it calibration performance. Finally, we show the effectiveness of the proposed technique as we vary the backbones. In addition to the backbones used in the main paper, we will further evaluate two other commonly used backbones, including WideResNet28 and Vision Transformer (ViT) [5] as backbones.

**Impact of the uncertainty set size.** For simplicity, we always keep one sparse sub-network in our framework to be with $\lambda_1 \to \infty$. The ECE performance with respect to different sets of $\lambda$ value for the remaining sub-networks is shown using the heatmap given in Figure 5 (a-b). As can be seen, it is important to choose $\lambda_2$ and $\lambda_3$ with very distinct values to achieve a low calibration error.

**Performance analysis of different backbones.** Table 9 (a) reports the performance of Cifar10 from both DRE and EP using different backbone architectures. In case of WideResNet28-10, the calibration error is low without sacrificing the accuracy. It also demonstrates that the superior performance of DRE is not limited to a specific backbone. In case of ViT, DRE still achieves a much lower calibration error than EP. However, using ViT as a backbone, the accuracy from both EP and DRE is lower and ECE is higher than other backbones. Existing studies show that without pretraining, the lack of useful inductive biases for ViT can cause performance drop [1]. Since no pretraining is conducted in both EP and DRE, it causes a lower accuracy (and a higher ECE).

**Impact of number of sparse-sub-networks.** In this analysis, we study the impact of number of sparse sub-networks. It should be noted that our work is not limited only for $M = 3$. We can instead increase the $M$ value. For example, Table 9 (b) shows the performance for ensemble model with $M = 5$, where each sub-network is trained with $\mathcal{K} = 3\%$ leading to a total $\mathcal{K} = 15\%$. We also show the performance with $M = 3$, where each sub-network is trained with $\mathcal{K} = 5\%$. As can be seen, if there is a sufficient learning capacity for each sub-network, the ECE score can further improve with the increase of $M$.

### D.9 Parameter Size and Inference Speed

Table 10: Parameter size and inference speed.

| Approach | ResNet50 | | ResNet101 | |
|---|---|---|---|---|
| | Params | Flops ($\times 10^9$) | Params | Flops ($\times 10^9$) |
| *Dense*$^\dagger$ | 23.6M | 4.14 | 42.5M | 7.88 |
| *SNE* | 3.5M | 1.31 | 6.3M | 2.53 |
| *DRE* | 3.5M | 1.31 | 6.3M | 2.53 |

We compare parameter size and inference speed of different types of sparse networks. Table 10 shows the FLOPS along with number of parameters associated with each technique. As can be seen, the proposed DRE has a comparable parameter size as that of the sparse network ensemble. In terms of computational times, our approach is comparable to the sparse network ensemble. Compared to a dense network, our technique has a much smaller parameter size with less FLOPS.

### D.10 Diversity on Sparse Sub-networks

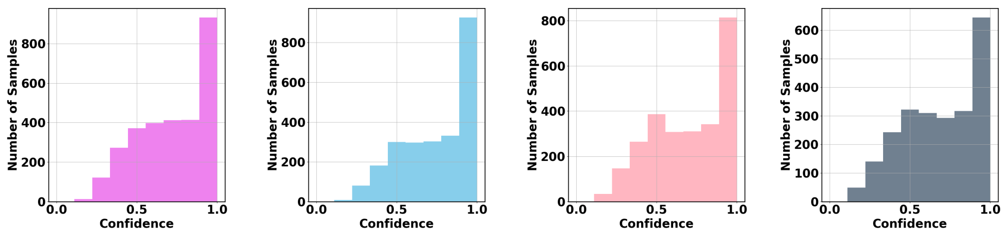
To justify our claim that our technique ensures the diverse sparse sub-networks, we adapt the disagreement metric ($d_{dist}$) from [17]. This metric measures the disagreement among sub-networks in terms of class label prediction. Table 11 below shows the results for Cifar10 and Cifar100 datasets. As shown, compared to Sparse Network Ensemble, DRE achieves higher disagreement which implies that the sparse sub-networks are more diverse.

Table 11: Accuracy, ECE, and prediction disagreement performance with a $\mathcal{K} = 15\%$ density.

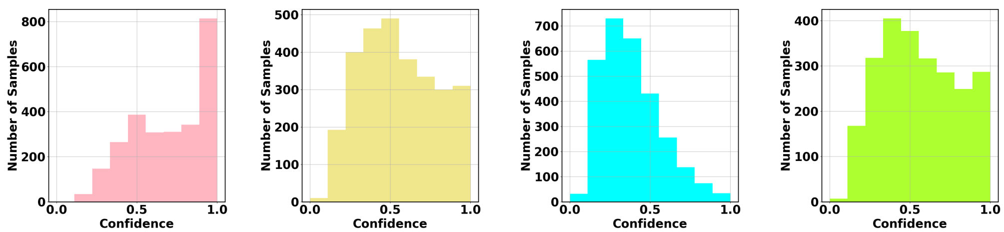
| Approach | Cifar10 | | | | | | Cifar100 | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | ResNet50 | | | ResNet101 | | | ResNet101 | | | ResNet152 | | |
| | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $d_{dist}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $d_{dist}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $d_{dist}$ | $\mathcal{ACC}$ | $\mathcal{ECE}$ | $d_{dist}$ |
| SNE | 94.85 | 3.05 | 0.048 | 94.96 | 3.18 | 0.049 | 76.82 | 11.12 | 0.20 | 77.23 | 11.63 | 0.20 |
| ***DRE (Ours)*** | 94.87 | **1.71** | 0.088 | 94.74 | **1.34** | 0.069 | 75.86 | **4.90** | 0.24 | 76.46 | **5.81** | 0.24 |

### D.11 Qualitative Analysis

In this section, we provide illustrative examples to further justify the proposed DRE is better calibrated compared to existing baselines. Figure 6 (a)-(d) show the confidence values for the wrongly classified samples using different baselines. As can be seen, all of the baselines suffer from the overfitting issue, resulting into the incorrect predictions with high confidence. In contrast, as shown in Figure 6 (e)-(f), the sparse sub-networks provide the confidence values in different ranges, where sub-network in (a) is learned from representative samples and (c) from the difficult ones. As these sub-networks are complementary with each other, the DRE has a much better confidence distribution for both the correct as well as incorrect samples. Figure 7 shows the confidence score of correctly classified data samples from the CIFAR100 dataset with different techniques. As shown, our DRE technique remains confident on the correct data samples while being not confident on the incorrect data samples. This result shows our approach is well calibrated and trustworthy compared with the competitive baselines. In summary, our proposed technique remains uncertain for incorrect samples while being confident on the correct samples resulting in a much improved calibration.

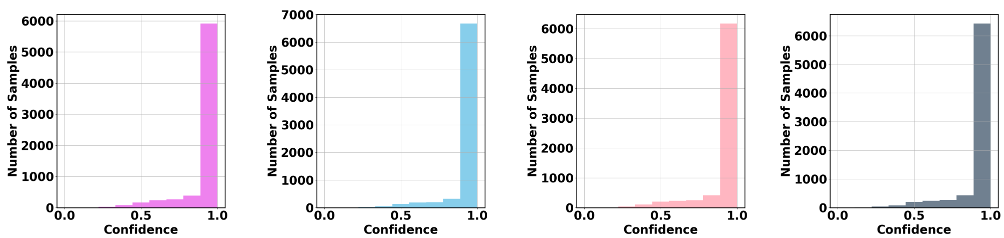



(a) Dense network (b) EP Sparse network 1 (c) EP Sparse network 2 (d) Sparse ensemble

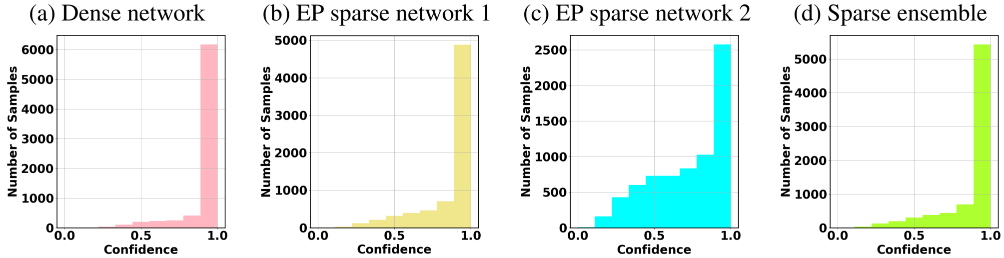



(e) DRO sparse Network 1 (f) DRO sparse network 2 (g) DRO sparse network 3 (h) DRE

Figure 6: Confidence scores of incorrectly classified samples in CIFAR100 with ResNet101




(a) Dense network (b) EP sparse network 1 (c) EP sparse network 2 (d) Sparse ensemble




(e) DRO sparse network 1 (f) DRO sparse network 2 (g) DRO sparse network 3 (h) DRE

Figure 7: Confidence scores of correctly classified samples in CIFAR100 with ResNet101

# E Broader Impact, Limitations, and Future Work

In this section, we first describe the potential broader impacts of our work. We then discuss the limitations and identify some possible future directions.

## E.1 Broader Impact

Sparse network training provides a highly promising way to significantly reduce the computational cost for training large-scale deep neural networks without sacrificing their predictive power. Besides energy savings, it also opens the gate for deploying deep neural networks to lightweight computing or edge devices that can further broaden the applications of AI in more diverse and resource constrained settings. The proposed robust ensemble framework provides a general solution to achieve calibrated training of deep learning models. As a result, the trained model is expected to provide more reliable uncertainty predictions, which could be an important step towards using AI in safety-critical domains.

### E.2 Limitations and Future Works

As an ensemble model, DRE involves multiple base learners (*i.e.,* sparse sub-networks). Consequently, it may lead to more computational overhead. This could create issues for real-time application as during the inference time, the input needs to be passed through all base learners to get the final output, which can slow down the prediction speed. A straightforward way to speed up the inference process is to execute all the base learners in parallel, which still incurs additional computational overhead. One interesting future direction is to investigate knowledge distillation and train a single sparse network from the ensemble model. Theoretical evidence [1] shows that knowledge distillation has the potential to largely maintain the ensemble performance while providing a promising way to train a single sparse network with an even higher sparsity level and improved inference speed.

## F Source Code

For the source code of this paper, please click here.

## G References

[1]. Dosovitskiy et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. ICLR2021.