ACLS: Adaptive and Conditional Label Smoothing for Network Calibration Supplement

Hyekang Park[1] Jongyoun Noh[1] Youngmin Oh[1]
Donghyeon Baek[1] Bumsub Ham[1,2*]

[1]Yonsei University [2]Korea Institute of Science and Technology (KIST)

https://cvlab.yonsei.ac.kr/projects/ACLS

In this supplementary material, we first present detailed derivations of gradients for previous regularization-based calibration methods (Sec. S1). We then describe more details for calibration metrics and hyperparameters (Sec. S2). We also provide more results on our approach (Sec.S3).

S1. Gradient analysis

S1.1. FLSD

MbLS [14] has shown that a focal loss (FL) [13] can be regarded as a form of label smoothing (LS) [21]. Taking one step further, we observe that FLSD [18] can also be approximated as LS. Specifically, FLSD uses a sample-dependent focusing parameter γ to improve calibration of FL. γ is determined by a heuristic rule as follows:

$$\gamma = \begin{cases} 5, & p_{\hat{y}} \in [0, 0.2) \\ 3, & p_{\hat{y}} \in [0.2, 1) \end{cases}. \tag{1}$$

We empirically find that $\gamma = 3$ for 95% of samples on Tiny-ImageNet [12]. This suggests that we can approximate FLSD [18] as FL and thus it can be viewed in the form of LS. Based on the approximation in [14] and our observation, we formulate smoothing and indicator functions of FLSD as in Table 2 in the main paper.

S1.2. CPC

Gradients. CPC [1] calibrates $C(C-1)/2$ binary pairs of probabilities instead of regularizing original C-way softmax probabilities. The loss function of CPC is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{CPC}} = \mathcal{L}_{\text{CE}} + \lambda_1 \mathcal{L}_{\text{1v1}} + \lambda_2 \mathcal{L}_{\text{BE}} \tag{2}$$

where λ_1 and λ_2 are hyperparameters. $\mathcal{L}_{\text{1v1}}$ and $\mathcal{L}_{\text{BE}}$ are defined as follows:

$$\mathcal{L}_{\text{1v1}} = -\sum_{k \neq y}^{C} \log \frac{p_y}{p_y + p_k}, \tag{3}$$

and

$$\mathcal{L}_{\text{BE}} = -\sum_{k,l \neq y}^{C} \left(\log \frac{p_k}{p_k + p_l} + \log \frac{p_l}{p_k + p_l} \right). \tag{4}$$

We compute the gradients of $\mathcal{L}_{\text{1v1}}$ and $\mathcal{L}_{\text{BE}}$ w.r.t z_j as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{1v1}}}{\partial z_j} &= \frac{\partial}{\partial z_j} \left(-\sum_{k \neq y}^{C} \log \frac{p_y}{p_y + p_k} \right) \\ &= -\sum_{k \neq y} \frac{p_y + p_k}{p_y} \frac{\frac{\partial p_y}{\partial z_j}(p_y + p_k) - p_y \left(\frac{\partial p_y}{\partial z_j} + \frac{\partial p_k}{\partial z_j} \right)}{(p_y + p_k)^2} \\ &= \begin{cases} -\sum_{k \neq y} \frac{p_k}{p_j + p_k}, & j = y \\ \frac{p_j}{p_y + p_j}, & j \neq y \end{cases}, \end{aligned} \tag{5}$$

and

$$\begin{aligned} \frac{\partial \mathcal{L}_{\text{BE}}}{\partial z_j} &= \frac{\partial}{\partial z_j} \left(-\sum_{k,l \neq y}^{C} \left(\log \frac{p_k}{p_k + p_l} + \log \frac{p_l}{p_k + p_l} \right) \right) \\ &= -\sum_{k,l \neq y} \frac{p_l + p_k}{p_k} \frac{\frac{\partial p_k}{\partial z_j}(p_k + p_l) - p_k \left(\frac{\partial p_k}{\partial z_j} + \frac{\partial p_l}{\partial z_j} \right)}{(p_k + p_l)^2} \\ &\quad - \sum_{k,l \neq y} \frac{p_k + p_l}{p_l} \frac{\frac{\partial p_l}{\partial z_j}(p_l + p_k) - p_l \left(\frac{\partial p_l}{\partial z_j} + \frac{\partial p_k}{\partial z_j} \right)}{(p_l + p_k)^2} \\ &= -2 \sum_{k,l \neq y} \frac{\frac{\partial p_k}{\partial z_j}(p_k + p_l) - p_k \left(\frac{\partial p_k}{\partial z_j} + \frac{\partial p_l}{\partial z_j} \right)}{(p_k + p_l) p_k} \\ &= \begin{cases} 0, & j = y \\ 2 \sum_{k \neq y} \frac{p_j - p_k}{p_j + p_k}, & j \neq y \end{cases}. \end{aligned} \tag{6}$$

*Corresponding author

We compute the gradients of CPC w.r.t z_j using Eqs. (5) and (6) as follows:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_j} + \frac{\partial \mathcal{L}_{\text{CPC}}}{\partial z_j} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_j} + \lambda_1 \frac{\partial \mathcal{L}_{\text{1v1}}}{\partial z_j} + \lambda_2 \frac{\partial \mathcal{L}_{\text{BE}}}{\partial z_j}$$
$$= \begin{cases} p_j - \left(q_j + \lambda_1 \sum_{k \neq j} \frac{p_k}{p_j + p_k}\right) & j = y \\ p_j - \left(q_j + \lambda_1 \frac{p_j}{p_y + p_j} + \lambda_2 \sum_{k \neq y} \frac{p_k - p_j}{p_k + p_j}\right) & j \neq y \end{cases}. \tag{7}$$

where λ_1 and λ_2 are hyperparameters. We omit the constant (*e.g.*, 2) for brevity.

Smoothing and indicator functions. Reformulating Eq. (7), we define a smoothing function of CPC as follows:

$$f(z_j) = \begin{cases} -\lambda_1 \sum_{k \neq j}^{C} \frac{p_k}{p_k + p_j}, & j = y \\ \lambda_1 \frac{p_j}{p_y + p_j} + \lambda_2 \sum_{k \neq y} \frac{p_j - p_k}{p_j + p_k}, & j \neq y \end{cases}. \tag{8}$$

Eq. (8) shows that CPC adjusts the labels and the only difference between CPC and LS [21] is how they determine the degree of smoothing. For example, LS smoothes using fixed constants, while CPC determines the degree of smoothing based on the probabilities. Note that an indicator function of CPC is always 1.

S1.3. MDCA

Gradients. MDCA [9] presents a differentiable version of ECE that approximates ECE in a mini-batch. MDCA exploits it for a regularizer directly. The loss function is defined as follows:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MDCA}}$$
$$= \mathcal{L}_{\text{CE}} + \lambda \sum_{k=1}^{C} \frac{1}{N} \left| \sum_{n=1}^{N} p_i^{(n)} - \sum_{n=1}^{N} q_i^{(n)} \right|, \tag{9}$$

where λ is a hyperparameter and n is an index of an input sample. $\frac{1}{N} \sum_{n=1}^{N} q_i^{(n)}$ represents a ratio of input samples for class i in a mini-batch. Thus, $\mathcal{L}_{\text{MDCA}}$ in Eq. (9) encourages p_i to have the same value of the ratio for class i. We compute the gradients of Eq. (9) w.r.t z_j as follows:

$$\frac{\partial \mathcal{L}}{\partial z_j^{(n)}} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_j^{(n)}} + \frac{\partial \mathcal{L}_{\text{MDCA}}}{\partial z_j^{(n)}}$$
$$= p_j^{(n)} - q_j^{(n)} + \frac{\lambda}{N} \sum_{k}^{C} a_k \frac{\partial p_k^{(n)}}{\partial z_j^{(n)}}$$
$$= p_j^{(n)} - q_j^{(n)} + \frac{\lambda}{N} a_j p_j^{(n)} - \frac{\lambda}{N} \sum_{k=1}^{C} a_k p_k^{(n)} p_j^{(n)}$$
$$= p_j^{(n)} - \left(q_j^{(n)} - \frac{\lambda}{N} p_j^{(n)} \left(a_j - \sum_{k=1}^{C} a_k p_k^{(n)} \right) \right). \tag{10}$$

We define $a_k = \text{sgn}\left(\frac{1}{N} \left| \sum_{n=1}^{N} p_k^{(n)} - \sum_{n=1}^{N} q_k^{(n)} \right| \right)$, where we denote by $\text{sgn}(\cdot)$ a sign function whose value is 1 if the argument is positive and -1 otherwise. If $p_{\hat{y}}$ is overconfident, while p_j ($j \neq \hat{y}$) are underconfident (*i.e.*, a_j is 1 if $j = \hat{y}$ and -1 otherwise.), Eq. (10) reduces as follows:

$$\frac{\partial \mathcal{L}}{\partial z_j^{(n)}} = \begin{cases} p_j - (q_j - \lambda p_j (1 - \sum_k a_k p_k)), & j = \hat{y} \\ p_j - (q_j + \lambda p_j (1 + \sum_k a_k p_k)), & j \neq \hat{y} \end{cases}, \tag{11}$$

where we omit n and N for brevity.

Smoothing and indicator functions. We define a smoothing function of MDCA from Eq. (11) as follows:

$$f(z_j) = \begin{cases} \lambda p_j (1 - \sum_k a_k p_k), & j = \hat{y} \\ \lambda p_j (1 + \sum_k a_k p_k), & j \neq \hat{y} \end{cases}. \tag{12}$$

We observe that $|\sum_{k \neq \hat{y}} a_k p_k| \ll |a_{\hat{y}} p_{\hat{y}}|$ and thus we can formulate Eq. (12) as the fourth row in Table 2 in the main paper as follows:

$$f(z_j) = \begin{cases} \lambda p_j (1 - p_j), & j = \hat{y} \\ \lambda p_j (1 + p_{\hat{y}}), & j \neq \hat{y} \end{cases}. \tag{13}$$

Eq. (13) suggests that MDCA reduces the labels if the probability for $j = \hat{y}$ is overconfident, and raises the labels if the probabilities for $j \neq \hat{y}$ are underconfident. For cases other than specified in Eq. (11) (*i.e.*, a_j is 1 if $j = \hat{y}$ and -1 otherwise.), Eq. (10) still reduces the label if p_j is higher than the ratio for class j, and vice versa if p_j is lower than the ratio. Note that an indicator function of MDCA is always 1.

S1.4. MbLS

Gradients. While LS [21] adjusts q_j for all j, MbLS [14] regularizes each label selectively by exploiting a margin M. Specifically, its loss function is as follows:

$$\mathcal{L} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{MbLS}}$$
$$= \mathcal{L}_{\text{CE}} + \lambda \sum_{k=1}^{C} \max(0, z_{\hat{y}} - z_k - M), \tag{14}$$

When $z_{\hat{y}} - z_j - M \geq 0$, we compute its gradients w.r.t z_j as follow:

$$\frac{\partial \mathcal{L}}{\partial z_j} = \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_j} + \frac{\partial \mathcal{L}_{\text{MbLS}}}{\partial z_j}$$
$$= p_j - q_j + \lambda \frac{\partial}{\partial z_j} \left(\sum_{k=1}^{C} z_{\hat{y}} - z_k - M \right) \tag{15}$$
$$= \begin{cases} p_j - q_j, & j = \hat{y} \\ p_j - (q_j + \lambda), & j \neq \hat{y} \end{cases}.$$

Smoothing and indicator functions. Note that MbLS penalizes the label using a constant (*e.g.*, λ). Thus, we can define a smoothing function as follows:

$$f(z_j) = \lambda. \tag{16}$$

Considering that Eq. (15) reduces to the gradients of CE if $z_{\hat{y}} - z_j - M < 0$, we define an indicator function of MbLS as follows:

$$\mathbb{C}(z_j) = \begin{cases} 0, & j = \hat{y} \\ \mathbb{1}[z_{\hat{y}} - z_j \geq M], & j \neq \hat{y} \end{cases}. \tag{17}$$

S1.5. CRL

Gradients. CRL [17] formulates network calibration as an ordinal ranking problem [2]. It determines whether to apply regularization by using a ranking condition. Its loss function is as follows:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{CRL}} \\ &= \mathcal{L}_{\text{CE}} + \lambda \sum_{n,m} \max\left(0, -\text{sgn}(H(n,m))(p_{\hat{y}}^{(n)} - p_{\hat{y}}^{(m)})\right). \end{aligned} \tag{18}$$

where n and m are indices of input samples in the training dataset. We define $H(n,m) = h^{(n)} - h^{(m)}$, where $h^{(n)}$ stores a ranking history of the n-th sample in a training dataset. In practice, n and m are selected only in a mini-batch and $\mathcal{L}_{\text{CRL}}$ is calculated using a ranking loss [22]. When the ranking condition is satisfied (*e.g.*, $h^{(n)} < h^{(m)}$ but $p_{\hat{y}}^{(n)} \geq p_{\hat{y}}^{(m)}$), $\mathcal{L}_{\text{CRL}}$ has a non-zero value. In this case, we can compute the gradients of Eq. (18) w.r.t $z_j^{(n)}$ for the specific n as follows:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial z_j^{(n)}} &= \frac{\partial \mathcal{L}_{\text{CE}}}{\partial z_j^{(n)}} + \frac{\partial \mathcal{L}_{\text{CRL}}}{\partial z_j^{(n)}} \\ &= p_j^{(n)} - q_j^{(n)} + \lambda \frac{\partial p_{\hat{y}}^{(n)}}{\partial z_j^{(n)}} \\ &= \begin{cases} p_j^{(n)} - \left(q_j^{(n)} - \lambda p_j^{(n)}(1 - p_j^{(n)})\right), & j = \hat{y} \\ p_j^{(n)} - \left(q_j^{(n)} + \lambda p_{\hat{y}}^{(n)} p_j^{(n)}\right), & j \neq \hat{y} \end{cases}, \end{aligned} \tag{19}$$

since $-\text{sgn}(H(n,m)) = 1$ and $\frac{\partial p_{\hat{y}}^{(m)}}{\partial z_j^{(n)}} = 0$.

Smoothing and indicator functions. Based on Eq. (19), we first define a smoothing function of CRL as follows:

$$f(z_j) = \begin{cases} \lambda p_j(1 - p_j), & j = \hat{y} \\ \lambda p_{\hat{y}} p_j, & j \neq \hat{y} \end{cases}, \tag{20}$$

where we omit n for brevity. Considering that the gradients of Eq. (19) are the same for those of CE if the ranking condition is not satisfied, we can define its indicator function as follows:

$$\mathbb{C}(z_j) = \mathbb{1}\left[H(n,m)(p_{\hat{y}}^{(n)} - p_{\hat{y}}^{(m)}) < 0\right]. \tag{21}$$

CRL adjusts the labels based on the values of probability (Eq. (20)) only when the condition is satisfied (Eq. (21)). Thus, we can view CRL as a combination of AR and CR (ACR). Combining Eqs (20) and (21), we can obtain the gradients of CRL w.r.t z_j for the specific n in the form of Eq. (8) in the main paper.

S2. More details

S2.1. Metrics

Following [1, 4, 6, 7, 9, 14, 16, 17, 18, 19], we report expected calibration error (ECE) and adaptive ECE (AECE) as calibration metrics. ECE is defined as follows:

$$\text{ECE} = \mathbb{E}_{\mathbf{p}}\left[P\left(\hat{y} = y | p_{\hat{y}}\right) - p_{\hat{y}}\right]. \tag{22}$$

In practice, we approximate Eq. (22) by partitioning predictions into bin B. Specifically, we first divide a range $(0, 1]$ into $\mathcal{M}$ equidistance bins (*e.g.*, if $\mathcal{M} = 5$, the bins are $(0, 0.2]$, $(0, 0.4]$, $\cdots$, $(0.8, 1.0]$). Given a sample, we then assign the sample to the specific bin according to the value of prediction for each sample. For example, if the value of prediction is 0.9 and $\mathcal{M} = 5$, the sample is assigned to the fifth bin. Following this protocol, we reformulate Eq. (22) as follows:

$$\text{ECE} = \sum_{i=1}^{\mathcal{M}} \frac{|B_i|}{N} \left|\text{acc}\left(B_i\right) - \text{conf}\left(B_i\right)\right|, \tag{23}$$

where B_i is the i-th bin and N is the number of samples. To visualize reliability diagrams, we compute the average accuracies and the calibration errors in each bin. For computing AECE, we partition the range $(0, 1]$ into $\mathcal{M}$ bins, each of which includes the same number of samples, instead of dividing the range into the equidistance bins.

S2.2. Hyperparameters

To set the hyperparameters λ_1 and λ_2 of Eq. (10) in the main paper, we perform a grid search on the cross-validation split of Tiny-ImageNet [12]: $\lambda_1 \in \{0, 0.05, 0.1, 0.15\}$ and $\lambda_2 \in \{0, 0.005, 0.01, 0.015\}$. For ImageNet [3], we use the hyperparameters obtained from Tiny-ImageNet since the search on ImageNet is computationally demanding. We summarize in Table A the results of grid search on the cross-validation split of Tiny-ImageNet. We set λ_1 to 0.1 and λ_2 to 0.01, respectively. Following [14], we set the margin M to 10. We also provide the ablation study for M in Sec. S3.

Table A: Comparisons of ACC (%) and ECE (%) on the cross-validation split of Tiny-ImageNet [12] (ACC/ECE). We train ResNet-50 accordring to the values of λ_1 and λ_2. ECE is computed with 15 bins. -: Fail to converge.

λ_1 \ λ_2	0	0.005	0.01	0.015
0	64.62 / 4.39	64.43 / 3.42	55.81 / 2.94	-/-
0.05	64.83 / 2.36	64.83 / 1.95	64.76 / 1.73	64.48 / 1.41
0.1	65.43 / 2.67	64.74 / 1.94	65.40 / **1.31**	65.17 / 1.52
0.15	65.04 / 2.71	65.18 / 1.44	65.23 / 1.42	64.11 / 1.47

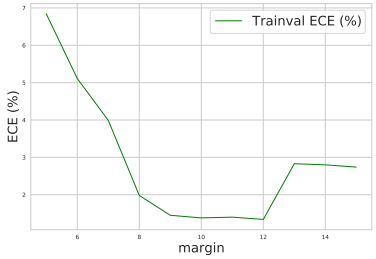

Figure A: Comparisons of ECE accordring to the value of M on the cross-validation (trainval) split of Tiny-ImageNet [12]. We use ResNet-50 [8] and compute ECE with 15 bins.

S3. More results and discussions

Effects of margins. We show in Fig. A ECE accordring to the value of M on the cross-validation split of Tiny-ImageNet [12]. From the figure, we have two findings: (1) If the margin is too small, the network is poorly calibrated. A plausible reason is that the condition is easily satisfied, and thus the regularization term penalizes well-calibrated probabilities also. (2) When the margin is too large, ECE is degraded also. This is because the margin condition is not satisfied even for the miscalibrated probabilities, disturbing calibration.

Additional results. We show in Table B quantitative comparisons with our method and other calibration approaches, including temperature scaling (TS) [7], MMCE [11], CutMix [10], CaPE-bin [15], and MCdrop [5], on image classification. For a fair comparison, we have reproduced all results with the same experimental configuration, including a network architecture and a dataset, except for the numbers of Patra et al. [20] which are taken from the paper. From Table B, we can clearly see that ACLS outperforms all methods by a significant margin in terms of ECE and AECE.

Comparisons with other non-label smoothing-based methods. We show quantitative comparisons with our method, CutMix [10], CaPE-bin [15], and MCdrop [5] in Table B. CutMix is a data augmentation strategy that replaces a region of an input image with a patch from another one. While CutMix is originally introduced to improve the generalization ability of neural networks, it has proven to be effective for network calibration. However, it often removes salient regions of images and produces inappropriate examples, which might degrade the discriminative ability of networks. CaPE is a regularization-based calibration method that exploits an accuracy of each sample directly. It assigns samples to particular bins and exploits accuracies for the bins (empirical accuracy) as target labels for network calibration, optimizing networks directly in terms of ECE. Computing the empirical accuracies of samples is however computationally demanding. MCdrop approximates the Bayesian inference with multiple predictions using dropout layers. This reduces the computational cost for uncertainty estimation, but still requires several forward passes for predictions. On the contrary, our method provides well-calibrated predictions in a single forward pass.

Table B: Quantitative results on the validation split of Tiny-ImageNet [12] in terms of ACC, ECE, and AECE. We compute the calibration metrics with both 10 and 15 bins using ResNet-50 [8], since Patra et al. [20] provides the numbers for 10 bins only. The number of CE+TS [7] in parentheses is an optimal temperature tuned on a held-out set of Tiny-ImageNet.

Method \ # of bins	10 bins			15 bins		
	ACC ↑	ECE ↓	AECE↓	ACC ↑	ECE ↓	AECE↓
CE	65.02	3.74	3.68	65.02	3.73	3.69
CE+TS (1.1) [7]	65.02	1.55	1.43	65.02	1.63	1.52
MMCE [11]	64.75	5.18	5.12	64.75	5.15	5.12
CutMix [10]	65.11	2.60	2.60	65.11	2.71	2.66
CaPE-bin [15]	59.78	2.91	2.61	59.78	2.99	2.96
MCdrop [5]	**65.33**	2.56	2.49	**65.33**	2.58	2.54
Patra et al. [20]	48.74	1.44	-	48.74	-	-
ACLS (Ours)	64.84	**0.91**	**0.92**	64.84	**1.05**	**1.03**

References

[1] Jiacheng Cheng and Nuno Vasconcelos. Calibrating deep neural networks by pairwise constraints. In *CVPR*, 2022. 1, 3

[2] Charles Corbière, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *NeurIPS*, 2019. 3

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 3

[4] Zhipeng Ding, Xu Han, Peirong Liu, and Marc Niethammer. Local temperature scaling for probability calibration. In *ICCV*, 2021. 3

[5] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 4

[6] Arindam Ghosh, Thomas Schaaf, and Matt Gormley. Adafocal: Calibration-aware adaptive focal loss. In *NeurIPS*, 2022. 3

[7] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger.

On calibration of modern neural networks. In *ICML*, 2017. 3, 4

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[9] Ramya Hebbalaguppe, Jatin Prakash, Neelabh Madan, and Chetan Arora. A stitch in time saves nine: A train-time regularizing loss for improved neural network calibration. In *CVPR*, 2022. 2, 3

[10] Xiaoqian Jiang, Melanie Osl, Jihoon Kim, and Lucila Ohno-Machado. Calibrating predictive model estimates to support personalized medicine. *Journal of the American Medical Informatics Association*, 19:263–274, 2012. 4

[11] Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *ICML*, 2018. 4

[12] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 2015. 1, 3, 4

[13] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1

[14] Bingyuan Liu, Ismail Ben Ayed, Adrian Galdran, and Jose Dolz. The devil is in the margin: Margin-based label smoothing for network calibration. In *CVPR*, 2022. 1, 2, 3

[15] Sheng Liu, Aakash Kaku, Weicheng Zhu, Matan Leibovich, Sreyas Mohan, Boyang Yu, Haoxiang Huang, Laure Zanna, Narges Razavian, Jonathan Niles-Weed, et al. Deep probability estimation. In *ICML*, 2022. 4

[16] Xingchen Ma and Matthew B Blaschko. Meta-cal: Well-controlled post-hoc calibration by ranking. In *ICML*, 2021. 3

[17] Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. In *ICML*, 2020. 3

[18] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In *NeurIPS*, 2020. 1, 3

[19] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *NeurIPS*, 2019. 3

[20] Rishabh Patra, Ramya Hebbalaguppe, Tirtharaj Dash, Gautam Shroff, and Lovekesh Vig. Calibrating deep neural networks using explicit regularisation and dynamic data pruning. In *WACV*, 2023. 4

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 1, 2

[22] Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *CVPR*, 2014. 3