ADAFOCAL: CALIBRATION-AWARE ADAPTIVE FOCAL LOSS

Anonymous authors
Paper under double-blind review

ABSTRACT

Much recent work has been devoted to the problem of ensuring that a neural network's confidence scores match the true probability of being correct, i.e. the calibration problem. Of note, it was found that training with Focal loss leads to better calibrated deep networks than cross-entropy loss, while achieving the same level of accuracy Mukhoti et al. (2020). This success stems from Focal loss regularizing the entropy of the network's prediction (controlled by the hyper-parameter γ), thereby reining in the network's overconfidence. Further improvements in calibration can be achieved if γ is selected independently for each training sample. However, the proposed strategy (named FLSD-53) is based on simple heuristics which, when selecting the γ, does not take into account any knowledge of whether the network is under or over confident about such samples and by how much. As a result, in most cases, this strategy performs only slightly better. In this paper, we propose a calibration-aware sample-dependent Focal loss called AdaFocal that adaptively modifies γ from one training step to the next based on the information about the network's current calibration behaviour. At each training step t, AdaFocal adjusts the γ_t based on (1) γ_{t-1} of the previous training step (2) the magnitude of the network's under/over-confidence. We evaluate our proposed method on various image recognition and NLP tasks, covering a variety of network architectures, and confirm that AdaFocal consistently achieves significantly better calibration than the competing state-of-the-art methods without loss of accuracy.

1 INTRODUCTION

Neural networks have found tremendous success in almost every field including computer vision, natural language processing, and speech recognition. Over time, these networks have grown complex and larger in size to achieve state-of-the-art performance and they continue to evolve further in that direction. However, it has been well established that such high capacity networks suffer from poor calibration Guo et al. (2017), i.e. the confidence scores of the predictions do not reflect the real world probabilities of those predictions being true. For example, if the network assigns 0.8 confidence to a set of predictions, we should expect 80% of those predictions to be correct. However, this is far from reality since modern networks tend to be grossly over-confident. This is of great concern, particularly for mission-critical applications such as autonomous driving, medical diagnosis, wherein the downstream decision making not only rely on the predictions but also on their confidence.

In recent years, there has been a growing interest in developing methods for calibrating neural networks. These can be mainly divided into two categories (1) post-hoc approaches that perform calibration after training (2) methods that calibrate the model during training itself. The first includes methods such as Platt scaling Platt (1999), histogram binning Zadrozny & Elkan (2001), Isotonic regression Zadrozny & Elkan (2002), Bayesian binning and averaging Naeini et al. (2015); Naeini & Cooper (2016), and Spline fitting Gupta et al. (2021). Methods in the second category focus on training the model on an objective function that accounts for calibration as well, including Maximum Mean Calibration Error (MMCE) Kumar et al. (2018), Label smoothing Müller et al. (2019), and recently focal loss Mukhoti et al. (2020). These methods aim to produce inherently calibrated models which when combined with post training calibration methods lead to further improvements.

Contribution. Our work falls into the second category. We build upon the calibration properties of focal loss to propose a modification that further improves its performance. Firstly, we make the observation that while regular focal loss, with a fixed γ parameter, improves the overall calibration by preventing samples from being over-confident, it also leaves other samples under-confident. To alleviate this drawback, we propose a modification to the focal loss called AdaFocal that adjusts

the γ for each training sample (or rather a group of samples) separately by taking into account the model's under/over-confidence about a similar corresponding group from the validation set. We evaluate the performance of our method on three image classification tasks: CIFAR-10, CIFAR-100 and Tiny-ImageNet, and one text classification task: 20 Newsgroup using various model architectures, and show that AdaFocal substantially outperforms the regular focal loss and other state-of-the-art calibration techniques in the literature. We further study the performance of AdaFocal on an out-of-distribution detection task and find it to perform better than the competing methods. Overall, we find that the models trained using AdaFocal get innately calibrated to a level that most of the times do not significantly benefit from additional post-hoc calibration through temperature scaling.

2 Problem Setup and Definitions

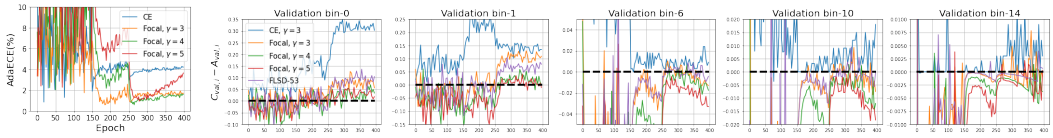
Consider a classification setting where we are given a set of training data $\{(\mathbf{x}_n, y_{\text{true},n})\}$, with $\mathbf{x}_n \in \mathcal{X}$ being the input and $y_{\text{true},i} \in \mathcal{Y} = \{1, 2, \ldots, K\}$ the associated ground-truth label. Using this data we wish to train a classifier $f_\theta(\mathbf{x})$ that outputs a vector $\hat{\mathbf{p}}$ over the K classes. We also assume access to a validation set for hyper-parameter tuning and a test set for evaluating its performance. For example, $f_\theta(\cdot)$ can be a neural network with learnable parameters θ, $\mathbf{x}$ is an image, and $\hat{\mathbf{p}}$ is the output of a *softmax layer* whose k^{th} element $\hat{p}_k$ is the probability score for class k. We refer to $\hat{y} = \arg\max_{k \in \mathcal{Y}} \hat{p}_k$ as the network's prediction and the associated probability score $\hat{p}_{\hat{y}}$ as the predicted confidence, and the same quantity for the jth example is $\hat{p}_{\hat{y},j}$.

In this setting, a network is said to be perfectly calibrated if the predicted confidence $\hat{p}_{\hat{y}}$ reflects the true probability of the network classifying $\mathbf{x}$ correctly i.e. $\mathbb{P}(\hat{y} = y_{\text{true}} \mid \hat{p}_{\hat{y}} = p) = p, \ \forall p \in [0, 1]$ Guo et al. (2017). Continuing our example, if the network assigns an average confidence score of 0.8 to a set of predictions then we should expect 80% of those to be correct. We define *Calibration Error* as $\mathcal{E} = \hat{p}_{\hat{y}} - \mathbb{P}(\hat{y} = y_{\text{true}} \mid \hat{p}_{\hat{y}})$ and the *Expected Calibration Error* as $\mathbb{E}_{\hat{p}_{\hat{y}}}[\mathcal{E}] = \mathbb{E}_{\hat{p}_{\hat{y}}}[\ |\hat{p}_{\hat{y}} - \mathbb{P}(\hat{y} = y_{\text{true}} \mid \hat{p}_{\hat{y}})|\ ]$ Guo et al. (2017). The true calibration error cannot be computed empirically with a finite sized dataset. Therefore, in the literature, the following three approximations are used instead for a dataset $\{(\mathbf{x}_n, y_{\text{true},n})\}_{n=1}^N$: (1) ECE Guo et al. (2017), uses M fixed size bins $\{B_i\}_{i=0}^{M-1}$ to compute a weighted average of the per-bin differences between confidence and accuracy. The bin B_i contains all examples j with $\hat{p}_{\hat{y},j}$ in the range $[\frac{i}{M}, \frac{i+1}{M})$. $\text{ECE} = \sum_{i=1}^{M} \frac{|B_i|}{N} |C_i - A_i|$, where $C_i = \frac{1}{|B_i|} \sum_{j \in B_i} \hat{p}_{\hat{y},j}$ is the average confidence and $A_i = \frac{1}{|B_i|} \sum_{j \in B_i} \mathbb{1}(\hat{y}_j = y_{\text{true},j})$ is the accuracy in bin B_i. Note that $E_i = C_i - A_i$ is the empirical approximation of the calibration error $\mathcal{E}$. (2) AdaECE Nguyen & O'Connor (2015) uses adaptively sized bins such that all bins contain an equal number of samples. It is computed as $\text{AdaECE} = \sum_{i=1}^{M} \frac{|B_i|}{N} \ |C_i - A_i|$, where $\forall i, j \ |B_i| = |B_j|$. (3) ClasswiseECE Kumar et al. (2018); Kull et al. (2019) estimates the calibration over all K classes: $\text{ClasswiseECE} = \frac{1}{K} \sum_{i=1}^{M} \sum_{k=1}^{K} \frac{|B_{i,k}|}{N} \ |C_{i,k} - A_{i,k}|$ where $C_{i,k} = \frac{1}{|B_{i,k}|} \sum_{j \in B_{i,k}} \hat{p}_{k,j}$ is the average confidence for the kth class in ith bin and $A_{i,k} = \frac{1}{|B_{i,k}|} \sum_{j \in B_{i,k}} \mathbb{1}(y_{\text{true},j} = k)$ is the accuracy of the kth class in ith bin. Lastly, as ECE has been shown to be a biased estimate of true calibration Vaicenavicius et al. (2019), we additionally use the following de-biased estimates of ECE namely $\text{ECE}_{\text{debiased}}$ proposed in Kumar et al. (2019) and $\text{ECE}_{\text{sweep}}$ as proposed in Roelofs et al. (2021) to further confirm the consistency of the results in this paper.

3 Calibration Properties of Focal loss

Focal loss Lin et al. (2017), given by $\mathcal{L}_{Focal}(p) = -(1-p)^\gamma \log p$, was originally proposed to improve the accuracy of classifiers by focusing on hard examples and down-weighting the well classified examples. Recently it was shown that training with focal loss results in significantly better calibration Mukhoti et al. (2020). This is because, while minimising the main KL divergence objective, it also increases the entropy of the prediction $\hat{\mathbf{p}}$ based on the relation: $\mathcal{L}_f \geq KL(q||\hat{\mathbf{p}}) - \gamma \mathbb{H}(\hat{\mathbf{p}})$, where q is the one-hot target vector. This prevents the network from being over confident and thus overall improving calibration.

However, as we show next, the regular focal loss with a fixed γ for all training samples does not achieve the optimal calibration. In Figure 1, we plot the per-bin calibration behaviour of ResNet50 trained on CIFAR-10 with different focal losses ($\gamma = 0, 3, 4, 5$). The calibration error subscripted by "val" $E_{val,i} = C_{val,i} - A_{val,i}$ is computed on the validation set with 15 equal-mass (adaptive) binning and shown here for two lower bins (indexed 0 and 1), two middle bins $(6, 10)$ and the highest bin (14). The rest of the bins and the bin boundaries are shown in Appendix A. From these figures,



(a) AdaECE (%) (b) $C_{val,top,i} - A_{val,i}$

Figure 1: Bin-specific calibration of ResNet-50 trained on CIFAR-10 with focal loss ($\gamma = 0, 3, 4, 5$) using 15 equal-mass (adaptive) binning on the validation set. (a) AdaECE, and (b) $C_{val,i} - A_{val,i}$ for bins $0, 1, 6, 10, 14$. The black horizontal line represents zero calibration error. These exemplify the downside of regular focal loss with a fixed common γ as the best performing γ differs between bins.

we see that although focal loss with $\gamma = 4$ achieves the overall lowest calibration error (AdaECE in Figure 1 (a)), there's no single γ that performs the best across all the bins. For example, in bin-$0, 1$ $\gamma = 4, 5$ seems to achieve better calibration whereas $\gamma = 0, 3$ are over-confident. For bin-$6, 10, 14$, $\gamma = 3$ seems to be better calibrated whereas $\gamma = 4, 5$ are under-confident and $\gamma = 0$ over-confident. This observation clearly points to the use of different γs for different bins to further improve the calibration. Such an attempt is presented in Mukhoti et al. (2020), called the sample-dependent focal loss(FLSD-53) which assigns $\gamma = 5$ if the training sample's true class posterior $\hat{p}_{y_{\text{true}}} \in [0, 0.2)$ and $\gamma = 3$ if $\hat{p}_{y_{\text{true}}} \in [0.2, 1]$. However, this strategy is fixed for every dataset-model pair and is based on simple heuristics of choosing higher γ for smaller values of $\hat{p}_{y_{\text{true}}}$ and relatively lower γ in the higher regions. However, from Figure 1(b), we see that FLSD-53 is also not the most optimal strategy across all the bins.

This motivates the design of a γ selection strategy that can appropriately assign a γ to a particular bin (or group of training samples) based on the magnitude and sign of the calibration error $E_{val,i}$ of a similar group of validation samples. However, at this point, one faces two challenges:

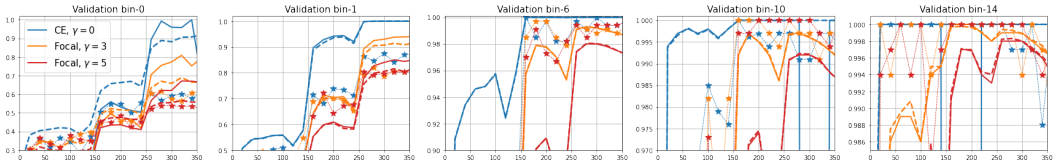
1. How do we find a correspondence between the confidence of training samples (which we can manipulate during training using the value of γ) and the confidence of the validation samples (which is the actual target)? In other words, in order to indirectly control the confidence of a particular group of validation samples, how do we know which particular group of training sample's confidence should be manipulated?

2. Given that we have established some kind of correspondence between a training group and a validation group, how do we arrive at the exact values of γ that will result in the perfect calibration?

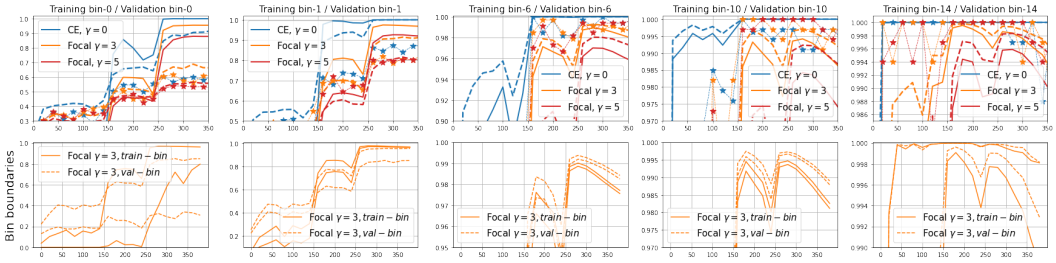
We try to answer the first question in the next section and the answer to the second question leads to the main contribution of the paper: AdaFocal.

4 Correspondence between confidence of train and val. samples

In order to find some correspondence, an intuitive thing to do would be to group the validation samples into M equal-mass validation-bins, use these validation-bin boundaries to group the training samples as well, and then compare the average confidence of the validation samples and the average confidence of the training samples in the same validation-bin to check for any correspondence. For binning the validation samples and determining the bin boundaries, we use the confidence score of the top predicted class $\hat{y}$ denoted by $\hat{p}_{val,top}$ (the average denoted by $C_{val,top}$). For the training samples, on the other hand, we either use the confidence of the top predicted class $\hat{y}$, denoted by $\hat{p}_{train,top}$ (the average denoted by $C_{train,top}$), or by the confidence of the true class y_{true} denoted by $\hat{p}_{train,true}$ (the average denoted by $C_{train,true}$). However, since during training we only care about manipulating $\hat{p}_{train,true}$, from hereon, we will consider only $\hat{p}_{train,true}$ (or $C_{train,true}$) when it comes to training samples. Nonetheless, for completeness we direct the reader to Appendix B which compares the two quantities $C_{train,true}$ and $C_{train,top}$), for the example case of focal loss $\gamma = 3$, to show that the two become almost the same as the training progresses. This is because as the model approaches towards 100% accuracy on the training set, the top predicted class and the true class become the same.



(a) Common validation-bin for both validation and training samples. **Solid line**: $C_{train,true,i}$ in validation-bin i, **Dashed line**: $C_{val,top,i}$ and **Star-dashed line**: $A_{val,i}$ in validation-bin i. For $\gamma = 3$, the validation-bin boundaries are the same as shown in sub-figure (b) below.



(b) Independent training and validation-bins. **Solid line**: $C_{train,true,i}$ in training-bin i, **Dashed line**: $C_{val,top,i}$ and **Star-dashed line**: $A_{val,i}$ in validation-bin i. Second row shows the bin boundaries for focal loss $\gamma = 3$.

Figure 2: ResNet-50 trained on CIFAR-10 with focal loss $\gamma = 0, 3, 5$. The figure compares the correspondence between average confidence of a group of training samples $C_{train,true,i}$ and a group of validation samples $C_{val,top,i}$.

With the terminologies now aside, in Figure 2 (a), for focal loss $\gamma = 0, 3, 5$, we plot and compare $C_{train,true,i}$ in validation-bin i [1] with $C_{val,top,i}$ and $A_{val,i}$ (accuracy of the validation samples) in the same validation-bin i. From these figures, we do observe a good correspondence between $C_{train,true,i}$ and $C_{val,top,i}$. For example in validation-bin-0 in Figure 2 (a), as γ increases from 0 to 3, to 5, the solid-line ($C_{train,true,i}$) gets lower and the same behaviour is reflected on the dashed-line ($C_{val,top,i}$). The same is observed in other bins as well. For reference, the bins not shown here are plotted in Appendix B.

This is very encouraging, as now we can expect (although loosely) that if we can increase or decrease the confidence of the training samples in some lower (or middle, or higher) probability region then the same will be reflected on the validation samples in a similar lower (or middle, or higher) probability region. Therefore, this now provides a way to indirectly control the value of $C_{val,top,i}$ by manipulating $C_{train,true,i}$. And from a calibration point of view, our strategy going forward would be to exploit this behaviour to keep $C_{train,true,i}$ (which we have control over during training) closer to $A_{val,i}$ so that, in turn, $C_{val,top,i}$ also stays closer to $A_{val,i}$ to overall achieve low calibration error $C_{val,top,i} - A_{val,i}$.

Before proceeding further, for completeness, we plot in Figure 2 (b) the case where $C_{train,true,i}$ is grouped independently into training-bins and $C_{val,top,i}$ into validation bins. The same figure also compares the boundaries of these bins. Since the binning is independent, the bin boundaries of training-bin i may not be the same as the bin boundary of validation-bin i, however as seen from the figure, they are quite close to each other. Overall, in this case as well, we observe a similar correspondence as mentioned above. For reference, the rest of the bins and their bin boundaries are shown in Appendix B. Going forward, however, for the ease of designing the algorithms in the paper, we will simply stick to the first case of using the validation-bin boundaries to group the training samples. This will allows us to maintain one-to-one correspondence between the probability boundaries of the ith training and validation group. Therefore, from hereon there is no separate training-bin involved and by "bin" we will always refer to validation-bin.

5 Our Proposed Method

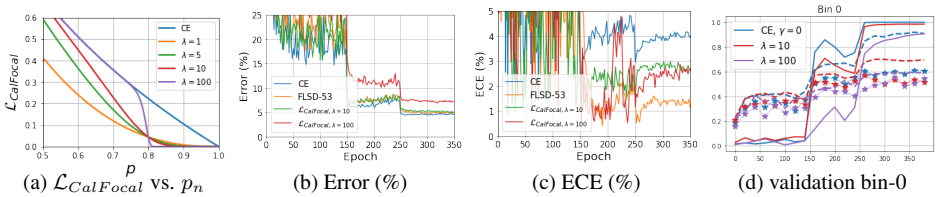
Let's denote the nth training sample's true class posterior $\hat{p}_{y_{\text{true}}}$ by p_n. Then our main goal is to keep p_n (or its averaged equivalent $C_{train,true,i}$) closer to $A_{val,i}$ so that the same is reflected on $C_{val,top,i}$. For this we can utilize the effect that the focal loss's γ parameter has on the confidence of training

[1] It may happen that no training samples falls into a particular validation-bin's boundaries. In that case, $C_{train,true,i}$ has been shown to drop to zero for example in bin-10 and bin-14 in Figure 2 (a).

sample i.e. increasing γ prevents $C_{train,true,i}$ from being high while decreasing γ pushes it in the other direction. Therefore, our overall approach would be to devise a strategy that adjusts the γ for each training sample based on how far either p_n is from $A_{val,i}$ or how far $C_{val,top,i}$ is from $A_{val,i}$. Such a γ-update-rule should ensure that: whenever the model is over-confident, i.e. $p_n > A_{val,i}$ (or $C_{val,top,i} > A_{val,i}$), γ is increased so that we get a smaller gradients which would prevent p_n from increasing further. On the other hand, when $p_n < A_{val,i}$ (or $C_{val,top,i} < A_{val,b}$) i.e. the model is under-confident, we decrease the γ so as to get larger gradients that in turn increases p_n. Note that, for focal loss however increasing γ does not always lead to smaller gradients. This mostly holds true in the region p_n approximately > 0.2 (see Figure 3(a) in Mukhoti et al. (2020)). However, in practice and as shown by the training-bin boundaries of bin-0 and bin-1 in Figure 2(b), we find majority of the training samples to lie above 0.2 during majority of the training, and therefore, for the algorithms in this paper, we simply stick to the above mentioned rule of increasing γ to decrease gradients.

In the the following sections, we first design a calibration-aware γ update strategy called CalFocal which, with some additional modifications, leads to the main algorithm AdaFocal.

5.1 Calibration aware focal loss (CalFocal)



(a) $\mathcal{L}_{CalFocal}$ vs. p_n (b) Error (%) (c) ECE (%) (d) validation bin-0

Figure 3: ResNet-50 trained on CIFAR-10 using $\mathcal{L}_{CalFocal}, \gamma = 0$. (d) $C_{train,true,i}$ (solid line), $C_{val,top,i}$ (dashed line), $A_{val,i}$ (star-dashed line).

Treating $A_{val,b}$ as the point that we want p_n to not deviate from, we make the focal loss parameter γ a function of $p_n - A_{val,i}$ to get

$$\mathcal{L}_{CalFocal}(p_n) = -(1-p_n)^{\gamma_n} \log p_n, \qquad \text{where, } \gamma_n = \exp(\lambda(p_n - A_{val,b})), \tag{1}$$

b is the validation-bin within the boundaries of which p_n falls. The hyper-parameterλ is the scaling factor which combined with the exponential function helps to quickly ramp up/down γ. The exponential function adheres to the γ-*update rule* mentioned earlier and ensures γ is always > 0.

Figure 3 (a) plots $\mathcal{L}_{CalFocal}$ vs. p_n for $A_{val,b} = 0.8$. We see that based on the strength of λ, the loss drastically drops down near $p_n = 0.8$ and then remains close to zero afterwards. This shows that main behaviour of $\mathcal{L}_{CalFocal}$ is to first push p towards 0.8 and then slow its growth towards overconfidence. Next, we find that CalFocal with $\lambda = 10, 100$ are able to reduce the calibration error, however, its still far from FLSD-53's performance. Also note that too high λ (=100) affects the accuracy of model as well. Figure 3 (d) compares $C_{train,true,i}$ with $C_{val,top,i}$ and $A_{val,i}$ for bin-0, where we find some evidence that the strategy of manipulating p_n (or equivalently $C_{train,true,i}$) to keep it closer to $A_{val,i}$ does get reflected on $C_{val,top,i}$ and it gets closer to $A_{val,i}$, thus reducing the calibration error.

Next, to reduce the computation and to avoid using a different γ_n for each training sample, one can use a common γ for all the training samples that fall into the boundaries of validation-bin b, by simply making it a function of $C_{val,b} - A_{val,b}$ instead of $p_n - A_{val,b}$.

$$\mathcal{L}_{CalFocal}(p_n) = -(1-p_n)^{\gamma_b} \log p_n, \qquad \text{where, } \gamma_b = \exp(\lambda(C_{val,b} - A_{val,b})) \tag{2}$$

As shown in appendix C, we find it's performance to be similar to (or slightly better than) Eq. 1.

Limitations. Lastly, note the following two limitations of CalFocal: (1) Let's say at some point, training with a high $\gamma = \exp(\lambda(C_{val,b} - A_{val,b}))$ starts reducing the calibration error $C_{val,b} - A_{val,b}$ over the next few epochs. Then, it would make more sense to continue training with the same high value of γ. However, due to CalFocal's update rule, γ will be reduced towards 1 as the $C_{val,b} - A_{val,b} \rightarrow 0$. (2) Let's say we find $C_{val,b} - A_{val,b}$ to be quite high at some point and that will set γ to a higher value as well. But let's say this γ is still not high enough to bring down the confidence, then we would want to increase it to an even higher value so that the gradient dies out further. But CalFocal is incapable of doing so as it will continue to hold at $\gamma = \exp(\lambda(C_{val,b} - A_{val,b}))$.

We address these two issues in the next sub-section to formulate the final algorithm for AdaFocal. For a detailed algorithmic description of CalFocal training, refer to Algorithm 1.

5.2 Calibration-aware Adaptive Focal loss (AdaFocal)

A straightforward way to address the above limitations is to make γ_t depend on γ_{t-1} as well. That is for the nth training example

$$\mathcal{L}(p_n, t) = -(1 - p_n)^{\gamma_{n,t}} \log p_n, \qquad \text{where, } \gamma_{n,t} = \gamma_{n,t-1} * \exp(p_n - A_{val,b}) \tag{3}$$

where, b is the validation-bin within the boundaries of which p_n lies. Note that this update rule do not involve λ and makes the loss function hyper-parameter free. However, note that, this function in Eq. 3 again requires to keep track of $\gamma_{n,t-1}$ for every training sample (e.g. for CIFAR-10, we would need to track 45000 $\gamma_{n,t-1}$) and for very large datasets this might be undesirable. To address this, as we also mentioned in CalFocal subsection, we can simply resort to using a common $\gamma_{b,t}$ for all the training samples that fall into a particular validation-bin b. Therefore, making γ a function of $C_{val,b} - A_{val,b}$ instead of $p_n - A_{val,b}$, we get

$$\mathcal{L}(p_n, t) = -(1 - p_n)^{\gamma_{b,t}} \log p_n, \qquad \text{where, } \gamma_{b,t} = \gamma_{b,t-1} * \exp(C_{val,b} - A_{val,b}) \tag{4}$$

This way we will need to store only M values of $\gamma_{b,t-1}$ which is usually very small ($= 15$ in this paper). Further, it makes more sense to update γ based on how far $C_{val,b}$ is from $A_{val,b}$ instead of how p_n is from $A_{val,b}$ because, as shown in Figure 3 bin-0, one may find $C_{val,b}$ quite closer to $A_{val,b}$ even when p_n is quite far away from $A_{val,b}$. At this point, we should stop updating γ further as we have already achieved very low $C_{val,b} - A_{val,b}$, even though $p_n - A_{val,b}$ is still a bit high. For these reasons, we use Eq. 4 as the basis for AdaFocal instead of Eq. 3.

Further, this update rule address the limitations of CalFocal in following way: let's say at some point we have over-confidence i.e. $E_{val,b} = C_{val,b} - A_{val,b} > 0$, therefore at the next step, γ will be increased to reign in the over-confidence, and will continue to increase at the same rate unless $E_{val,b}$ starts decreasing (this additional increase in γ was not possible with CalFocal). At this point, if we find $E_{val,b}$ starts reducing, then that would reduce the increase in γ over the next epochs, and will ultimately settle at the value that achieves $E_{val,b} = 0$ (CalFocal at $E_{val,b} = 0$ will cause γ to go down to 1). Next, if this current high value of γ now starts causing under-confidence i.e. $C_{val,b} - A_{val,b} < 0$, then the update rule will start reducing γ which will allow $C_{val,b}$ to be increased back to $A_{val,b}$.

However, note an undesirable property of Eq. 4 which is the unbounded exponential update that may cause γ to rise to an undesirably high value. This is because γ_t at any time step can be expanded as $\gamma_t = \gamma_{t-1} \exp(E_{val,t}) = \gamma_0 \exp(E_{val,0} + E_{val,1} + ... + E_{val,t-1} + E_{val,t})$. Thus if $E_{val,t} > 0$ for quite a few number of epochs γ_t may become so large that even if we get $E_{val,t} < 0$ in the subsequent epochs, it will still take too many steps to bring γ_t back down to a desired level. We remedy this by simply constraining γ to some upper bound $\gamma_{\max}$. Therefore, the final AdaFocal loss function is given by

$$\mathcal{L}_{AdaFocal}(p_n, t) = -(1 - p_n)^{\gamma_{b,t}} \log p_n, \text{ where, } \gamma_{b,t} = \min\{\gamma_{\max}, \gamma_{b,t-1} * e^{C_{val,b} - A_{val,b}}\} \tag{5}$$

with a detailed description of AdaFocal training given in Algorithm 1. For performance comparison of the constrained and unconstrained version of AdaFocal, refer to Appendix H.

Limitations. One may argue that $\gamma_{\max}$ is again a hyper-parameter, however, note that it does not require any special fine-tuning. Its sole purpose is to stop γ from exploding and thus any reasonable value in the range of 20 that leaves enough room for γ to move around works quite well in practice. In this paper, we use $\gamma_{\max} = 20$ and achieve consistent results.

6 Experiments

Experimental setup. We evaluate the performance of our proposed method on image and text classification tasks. For image classification, we use CIFAR-10, CIFAR-100 Krizhevsky (2009) and Tiny-ImageNet Deng et al. (2009) to analyze the calibration of ResNet50, ResNet-100 He et al. (2016), Wide-ResNet-26-10Zagoruyko & Komodakis (2016), and DenseNet-121 Huang et al. (2017) models. For text classification, we use the 20 Newsgroup dataset Lang (1995) and train the Global Pooling CNN model Lin et al. (2014). Further details about the datasets, models and experimental configurations are given in Appendix D.

Algorithm 1: CalFocal, AdaFocal

Input: $D_{train} = \{(\mathbf{x}_n, y_{true,n})\}_{n=1}^{N_{train}}$ and $D_{val} = \{(\mathbf{x}_n, y_{true,n})\}_{n=1}^{N_{val}}$;
Initialization at $t = 0$: **for** $i = 1$ **to** M **do**
  $B_{val,t,i} = \left(\frac{i-1}{M}, \frac{i}{M}\right]$ // equally spaced validation-bins;
  $C_{val,t+1,i} = A_{val,t+1,i} = \frac{2i-1}{2M}$ // mid point of the bin;
  $\gamma_{t,i} = 1$;
Training: **for** $t = 0$ **to** T **do**
  $L_t = 0$;
  for $n = 1$ **to** N_{train} **do**
    $p_n = f_{w_t}(\mathbf{x}_n)$ // denoting $p_n = p_{y_{true,n}}$;
    b = get_bin_index$(p_n, \{B_{t,i}\})$ // validation-bin inside which p lies;
    $L_t \mathrel{+}= -(1-p_n)^{\gamma_{t,b}} \log p$ // use $\gamma_{t,b}$ of bth bin to compute loss;
  w_{t+1} = gradient_update(w_t, L_t);
  for $i = 1$ **to** M **do** // Using updated model $f_{w_{t+1}}$ on D_{val}, update bin statistics and γ
    Re-compute bin boundaries $B_{t+1,i}$ and corresponding $C_{val,t+1,i}$, $A_{val,t+1,i}$;
    if *CalFocal* **then**
      $\gamma_{t+1,i} = \exp(\lambda * (C_{val,t+1,i} - A_{val,t+1,i}))$;
    else if *AdaFocal* **then**
      $\gamma_{t+1,i} = \min\{\gamma_{\max},\ \gamma_{t,i} * \exp(C_{val,t+1,i} - A_{val,t+1,i})\}$;

Baseline methods. As baseline, we use the following calibration methods in the literature: MMCE Kumar et al. (2018), Brier loss Brier (1950), Label smoothing Müller et al. (2019), and sample-dependent focal loss (FLSD-53) Mukhoti et al. (2020). We also report the effect of temperature scaling Guo et al. (2017) when used on top of these calibration methods. For selecting the optimum temperature, we follow Mukhoti et al. (2020) and chose the temperature that produces the minimum ECE on the validation set as it gives a stronger baseline to compare against. The temperature is grid searched in the interval $(0, 10]$ with steps of 0.1.

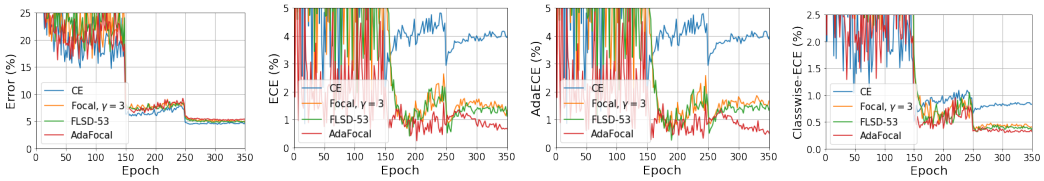


Figure 4: ResNet-50 trained on CIFAR-10 with cross entropy (CE), focal loss $\gamma = 3$, FLSD-53 and AdaFocal. (a) Error, (b) ECE, (c) AdaECE and (d) classwise-ECE. AdaFocal achieves the lowest calibration error while maintaining similar error performance.

Results. In Figure 4, we compare the performance of AdaFocal against cross entropy (CE), focal loss $\gamma = 3$ (FL-3), and FLSD-53 for ResNet-50 trained on CIFAR-10. We chose FL-3 and FLSD-53 as our competitive baseline as they were consistently shown to be better than MMCE, Brier Loss and Label smoothing in Mukhoti et al. (2020) across many datasets-model pairs. The figure plots the test set error along with the three calibration metrics: ECE, AdaECE and Classwise ECE. We see that throughout the training AdaFocal is better calibrated while achieving the same level of accuracy. This implies that during training AdaFocal is able to closely track the under/over-confidence of the model (using the validation set) and maintain a well calibrated model at all times. Also note that FLSD-53 behaves almost identically to FL-3; this is because if we look from the perspective of assigning different γs for different bins, FLSD-53 assigns $\gamma = 5$ for $p_n \in [0, 0.2)$ which is a very small interval and as shown for FL-3 in Figure 10 in Appendix B (see training-bin boundaries for bin-0 and bin-1), most of the samples lie above 0.2 during major part of the training making FL-3 and FLSD-53 almost the same in practice.

In Figure 5, we plot the calibration statistics from validation set used by AdaFocal at each epoch to train ResNet-50 on CIFAR-10. The values of calibration error (1) $E_{val,i} = C_{val,i} - A_{val,i}$, (2) γ_t, and (3) bin boundaries are shown for a few bins from lower, middle, and higher probability region. From the figure, we observe that in higher probability regions (bin 10, 12, 14) AdaFocal is almost perfectly calibrated, and in the lower bins, it achieves lower calibration error than FLSD-53. We also see how the associated γ_t for each bin evolves during training: it fluctuates during the initial phases

of training but later settles down to some appropriate value that continues to keep the calibration error in check.

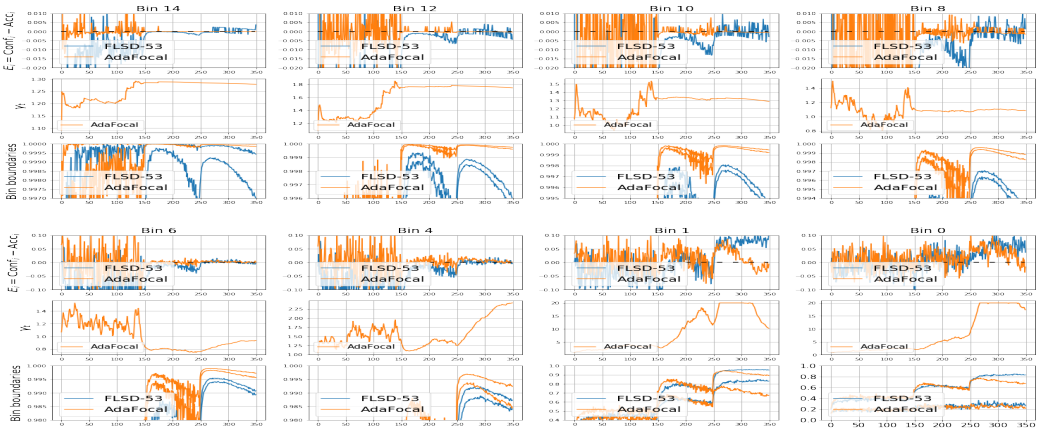


Figure 5: Calibration statistics from validation set used by AdaFocal at each epoch to train ResNet-50 on CIFAR-10. Each bin has three subplots: **top**: $E_{val,i} = C_{val,i} - A_{val,i}$, **middle**: evolution of γ_t, and **bottom**: bin boundaries. Black dotted line in top plot represent zero calibration error.

For rest of the experiments, AdaFocal's performance (averaged over 5 runs with different initialization seeds) is reported against the baseline methods in Table 1 (ECE) and Table 2 (Error). [2] AdaECE and classwise-ECE are reported in Appendix E, whereas the de-biased estimates namely $\text{ECE}_{\text{debias}}$ (with 15 and 30 bins), $\text{ECE}_{\text{EW}-\text{sweep}}$ (equal-width), and $\text{ECE}_{\text{EM}-\text{sweep}}$ (equal-mass) are reported in Appendix F. The error bars for ECE with mean and standard deviations computed over 5 runs are plotted in Figure 6 for some of the experiments. From Table 1 we observe that, apart from Tiny-ImageNet, AdaFocal outperforms all the baseline methods by a substantial margin especially if we compare the pre-temperature scaling results. With post-temperature scaling included as well, AdaFocal achieves the lowest calibration error in 6 out of the 10 experiments. Further, observe that in many cases temperature scaling on top of AdaFocal does not seem to offer any improvement at all (optimal temperature $= 1$). For the cases with some improvement, the optimal temperature is still very close to 1 indicating that AdaFocal innately produces highly calibrated models that may not require any additional post-processing.

Dataset	Model	Cross Entropy		Brier Loss		MMCE		LS-0.05		FLSD-53		AdaFocal	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.35	1.35(2.5)	1.82	1.08(1.1)	4.56	1.19(2.6)	2.96	1.67(0.9)	1.55	0.95(1.1)	0.8	**0.65**(1.08)
	ResNet-110	4.41	1.09(2.8)	2.56	1.25(1.2)	5.08	1.42(2.8)	2.09	2.09(1.0)	1.87	1.07(1.1)	0.8	**0.65**(1.06)
	Wide-ResNet-26-10	3.23	0.92(2.2)	1.25	1.25(1.0)	3.29	0.86(2.2)	4.26	1.84(0.8)	1.56	0.84(0.9)	**0.7**	0.7(1.0)
	DenseNet-121	4.52	1.31(2.4)	1.53	1.53(1.0)	5.1	1.61(2.5)	1.88	1.82(0.9)	1.22	1.22(1.0)	0.76	**0.66(1.02)**
CIFAR-100	ResNet-50	17.52	3.42(2.1)	6.52	3.64(1.1)	15.32	2.38(1.8)	7.81	4.01(1.1)	4.5	2.0(1.1)	**1.3**	1.3(1.0)
	ResNet-110	19.05	4.43(2.3)	7.88	4.65(1.2)	19.14	3.86(2.3)	11.02	5.89(1.1)	8.56	4.12(1.2)	**1.3**	1.3(1.0)
	Wide-ResNet-26-10	15.33	2.88(2.2)	4.31	2.7(1.1)	13.17	4.37(1.9)	4.84	4.84(1)	3.03	**1.64**(1.1)	1.92	1.92(1.0)
	DenseNet-121	20.98	4.27(2.3)	5.17	2.29(1.1)	19.13	3.06(2.1)	12.89	7.52(1.2)	3.73	**1.31(1.1)**	1.74	1.74(1.0)
Tiny-ImageNet	Resnet-50	15.32	5.48(1.4)	4.44	4.13(0.9)	13.01	5.55(1.3)	15.23	6.51(0.7)	**1.76**	1.76(1)	2.41	2.25(0.96)
20 Newsgroups	Global-pool CNN	17.92	2.39(3.4)	13.58	3.22(2.3)	15.48	6.78(2.2)	4.79	2.54(1.1)	6.92	**2.19(1.5)**	2.72	2.67(1.12)

Table 1: Test set ECE(%) for different methods (pre and post temperature scaling). Underlined values mark the lowest error among Pre-T results and bold marks the overall lowest in the row. Optimal temperatures are given in brackets. For AdaFocal, the values are averaged over 5 runs.

Dataset	Model	Cross Entropy	Brier Loss	MMCE	LS-0.05	FLSD-53	AdaFocal
CIFAR-10	ResNet-50	**4.95**	5.0	4.99	5.29	4.98	5.30
	ResNet-110	**4.89**	5.48	5.4	5.52	5.42	5.27
	Wide-ResNet-26-10	**3.86**	4.08	3.91	4.2	4.01	4.5
	DenseNet-121	**5.0**	5.11	5.41	5.09	5.46	5.2
CIFAR-100	ResNet-50	23.3	23.39	23.2	23.43	23.22	**22.60**
	ResNet-110	22.73	25.1	23.07	23.43	**22.51**	22.79
	Wide-ResNet-26-10	20.7	20.59	20.73	21.19	20.11	**20.07**
	DenseNet-121	24.52	23.75	24.0	24.05	22.67	**22.22**
Tiny-ImageNet	Resnet-50	49.81	53.2	51.31	**47.12**	49.06	48.26
20 Newsgroups	Global-pool CNN	26.68	27.06	27.23	**26.03**	27.98	28.53

Table 2: Test set error (%). The model with the lowest error is marked in bold.

Number of bins used by AdaFocal. All the results in the paper are reported for AdaFocal trained using 15 equal-mass (adaptive) bins to draw calibration statistics form the validation set. We

[2] While reproducing the baseline experiments in Mukhoti et al. (2020) we obtained very similar results, therefore, we simply borrow the exact values to maintain consistent comparison.

Dataset	Model	Cross Entropy		Brier Loss		MMCE		LS-0.05		FL-3		FLSD-53		AdaFocal
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T
CIFAR-10 / SVHN	ResNet-110	61.71	59.66	94.80	95.13	85.31	85.39	68.68	68.68	90.27	90.39	90.33	90.49	**96.09**
	Wide-ResNet-26-10	96.82	97.62	94.51	94.51	97.35	97.95	84.63	84.66	90.92	91.30	93.08	93.11	**96.63**
CIFAR-10 / CIFAR-10-C	ResNet-110	77.53	75.16	84.09	83.86	71.96	70.02	72.17	72.18	80.11	79.78	82.06	81.38	**84.96**
	Wide-ResNet-26-10	81.06	80.68	85.03	85.03	82.17	81.72	71.10	71.16	83.33	84.00	80.00	80.76	**89.52**

Table 3: AUROC (%) of models trained on CIFAR-10 as the in-distribution data and tested on SVHN and CIFAR-10-C as out-of-distribution data.

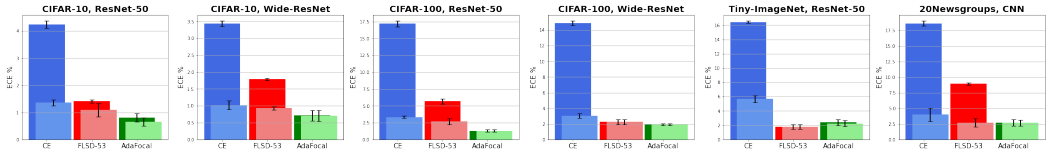
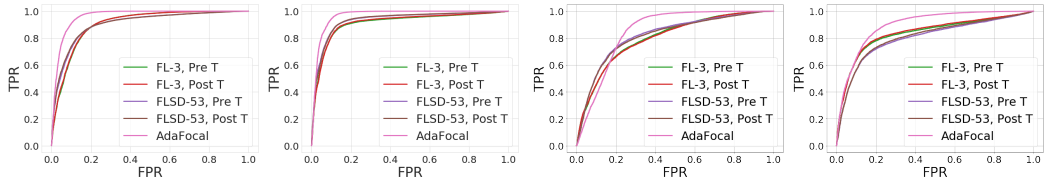


Figure 6: ECE error bars with mean and standard deviation computed over 5 runs with different initialization seed. The dark and light colors show pre and post temperature scaling results respectively.

experiment with 5, 10, 15, 20, 30, and 50 bins to train ResNet-50 on CIFAR-10, as reported in Appendix G, and find that the performance degrades when number of bins used is too few (5) or too high (≥ 30). The best results are obtained in the range 10 to 20 and, therefore, we use 15 bins in the paper with the additional reason of making the results comparable to Mukhoti et al. (2020) which also uses 15 bins.

Out-of-Distribution (OOD) detection. Following Mukhoti et al. (2020), we also report the performance of AdaFocal on an OOD detection task. We train ResNet-110 and Wide-ResNet26-10 on CIFAR-10 as the in-distribution data and test on SVHN Netzer et al. (2011) and CIFAR-10-C Hendrycks & Dietterich (2019) (with level 5 Gaussian noise corruption) as the out-of-distribution data. The entropy of the softmax is used as the measure of uncertainty and the results are reported in Table 3[3] and Figure 7. We observe that models trained with AdaFocal obtain the highest AUROC values compared to other focal losses, thus, further highlighting the benefits of AdaFocal and inherently calibrated models, as temperature scaling has been shown to be ineffective under distributional shift Snoek et al. (2019). The same is also observed in Table 3.



(a) SVHN: ResNet-110, WideResNet (b) CIFAR-10-C: ResNet-110, WideResNet

Figure 7: ROC for ResNet-110 and Wide-ResNet-26-10 trained on in-distribution CIFAR-10 and tested on out-of-distribution (a) SVHN and (b) CIFAR-10-C.

7 Conclusion

In this work, we first revisit the calibration properties of regular focal loss and highlight the downside of using a fixed common γ for all samples. Particularly, by studying the calibration behaviour of different samples in different probability region, we find that there's no single γ that achieves the best calibration over the entire region. We use this observation to motivate selecting γ independently for each sample based on the knowledge of network's under/over-confidence. Then we propose a calibration-aware adaptive focal loss called AdaFocal that accounts for such information and updates the γ_t at every time step based on γ_{t-1} from the previous step and the magnitude of network's under/over-confidence. We find AdaFocal to perform consistently better across different datasets and model pairs. Further, we find AdaFocal to produce innately calibrated models that most of the times do not substantially benefit from post-hoc processing through temperature scaling. We also find models trained with AdaFocal to exhibit significantly better out-of-distribution detection performance than the competing focal losses.

[3]For ResNet110 on CIFAR-10/SVHN, we were not able to reproduce the reported results of 96.74, 96.92 for FL-3 in Mukhoti et al. (2020). Instead we found those values to be 90.27, 90.39 and report them in Table 3.

Reproducibility For reproducibility, we have include in the supplementary material a zip file that contains the code base for running the experiments. For running particular experiments

- CIFAR-10, ResNet-50, Cross entropy: *python train.py –dataset cifar10 –model resnet50 –loss cross_entropy –num_bins 15 -e 400 –save-path experiments/cifar10_resnet50_ce*
- CIFAR-100, ResNet-50, Cross entropy: *python train.py –dataset cifar100 –model resnet50 –loss cross_entropy –num_bins 15 -e 400 –save-path experiments/cifar100_resnet50_ce*
- Tiny-ImageNet, ResNet-50, Cross entropy: *python train.py –dataset tiny_imagenet –model resnet50_ti –loss cross_entropy –num_bins 15 –first-milestone 40 –second-milestone 60 -e 100 -b 64 -tb 64 –dataset-root data/tiny-imagenet-200 –save-path experiments/tinyImageNet_resnet50_ce*
- 20 Newgroups, CNN, Cross entropy: *python main.py –loss cross_entropy –num-epochs 50 –num-bins 15 –save-path experiments/cnn_ce*

REFERENCES

Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 1950.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2017.

Kartik Gupta, Amir Rahimi, Thalaiyasingam Ajanthan, Thomas Mensink, Cristian Sminchisescu, and Richard Hartley. Calibration of neural networks using splines. In *International Conference on Learning Representations*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations*, 2019.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems*, 2019.

Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Aviral Kumar. 20 newsgroups mmce. `https://github.com/aviralkumar2907/MMCE`, 2018.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, Proceedings of Machine Learning Research, 2018.

Ken Lang. Newsweeder: Learning to filter netnews. In *in Proceedings of the 12th International Machine Learning Conference (ML95*, 1995.

M. Lin, Q. Chen, and S. Yan. Network in network. *CoRR*, 1312.4400, 2014.

Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

Jishnu Mukhoti. Focal calibration. `https://github.com/torrvision/focal_calibration`, 2020.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip HS Torr, and Puneet K Dokania. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems*, 2020.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? In *Advances in Neural Information Processing Systems*, 2019.

Mahdi Pakdaman Naeini and Gregory F Cooper. Binary classifier calibration using an ensemble of near isotonic regression models. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, 2016.

Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, AAAI'15, 2015.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

Khanh Nguyen and Brendan O'Connor. Posterior calibration and exploratory analysis for natural language processing models. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2015.

Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014.

John C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, 1999.

Rebecca Roelofs, Nicholas Cain, Jonathon Shlens, and Michael C. Mozer. Mitigating bias in calibration error estimation, 2021.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115 (3):211–252, 2015.

Jasper Snoek, Yaniv Ovadia, Emily Fertig, Balaji Lakshminarayanan, Sebastian Nowozin, D. Sculley, Joshua V. Dillon, Jie Ren, and Zachary Nado. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, 2019.

Juozas Vaicenavicius, David Widmann, Carl Andersson, Fredrik Lindsten, Jacob Roll, and Thomas Schön. Evaluating model calibration in classification. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89, pp. 3459–3467. PMLR, 16–18 Apr 2019.

Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *ICML*, 2001.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD*, 2002.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016.

APPENDICES

A CALIBRATION BEHAVIOUR OF FOCAL LOSS ACROSS DIFFERENT BINS

In the main paper, we showed the calibration behavior of different focal losses (for ResNet50 trained on CIFAR-10) for only a few bins. For completeness, the rest of the bins and their calibration error $E_i = C_{val,i} - A_{val,i}$ are shown in Figure 8 for focal losses with $\gamma = 0, 3, 4, 5$. We observe that there's no single γ that performs the best across all the bins. Rather, every bin seems to have a particular γ that achieves a better calibration.

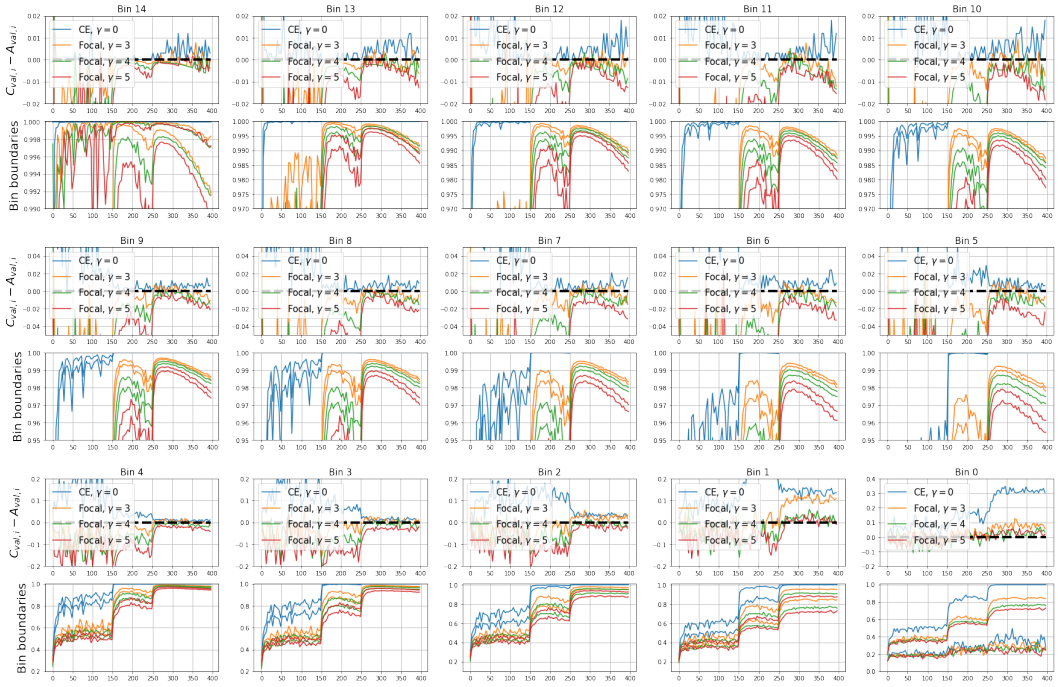


Figure 8: Bin specific calibration error $E_i = C_{val,i} - A_{val,i}$ of a ResNet50 model trained on CIFAR-10 using different focal losses $\gamma = 0, 3, 4, 5$. The statistics are reported on validations set ($5,000$ examples) using 15 equal-mass (adaptive) bins. The black horizontal line represents zero calibration error.

B CORRESPONDENCE BETWEEN CONFIDENCE OF TRAINING AND VALIDATION SAMPLES

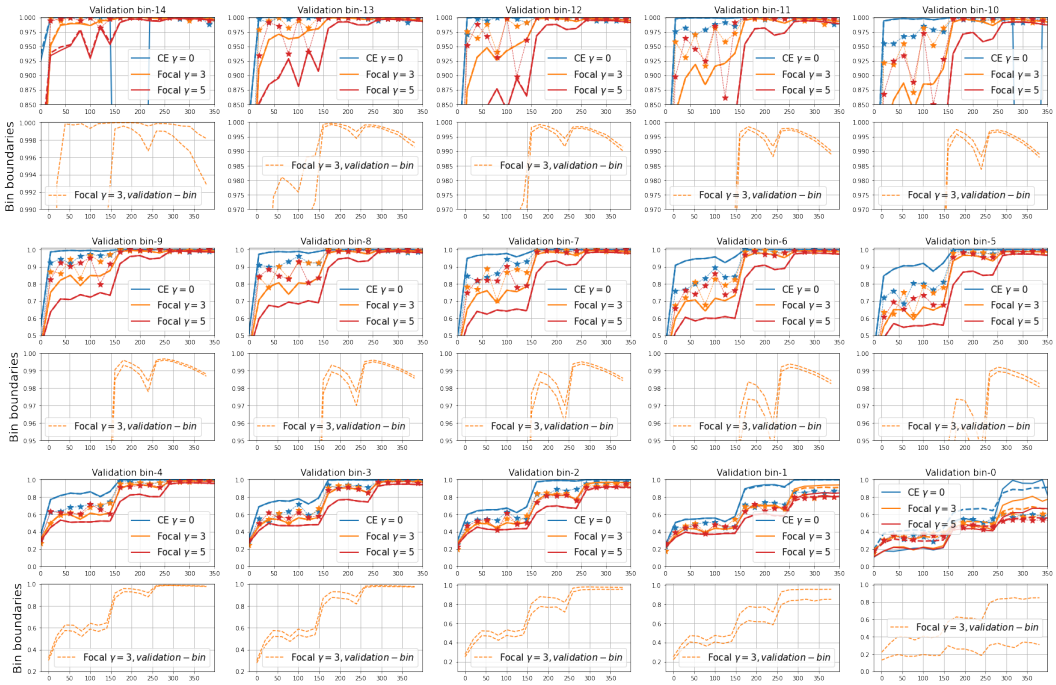


Figure 9: Common validation-bin for both validation and training samples. **Solid line**: $C_{train,true,i}$ in validation-bin i, **Dashed line**: $C_{val,top,i}$ and **Star-dashed line**: $A_{val,i}$ in validation-bin i.

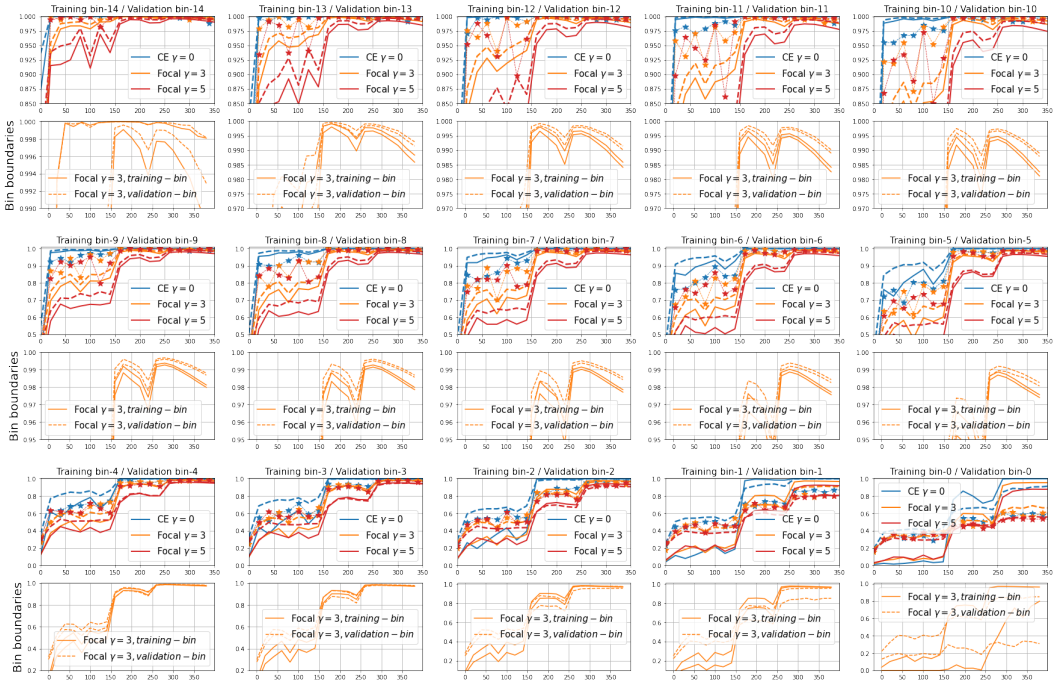


Figure 10: Independent training and validation-bins. **Solid line**: $C_{train,true,i}$ in training-bin i, **Dashed line**: $C_{val,top,i}$ and **Star-dashed line**: $A_{val,i}$ in validation-bin i. Second row shows the training and validation bin boundaries for focal loss $\gamma = 3$.

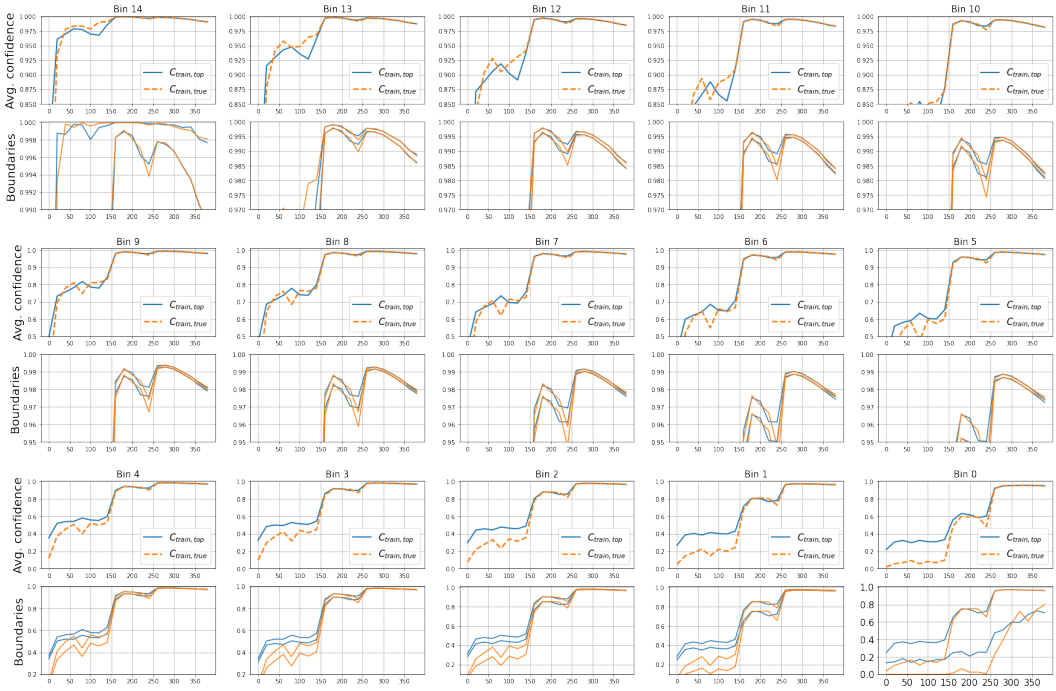
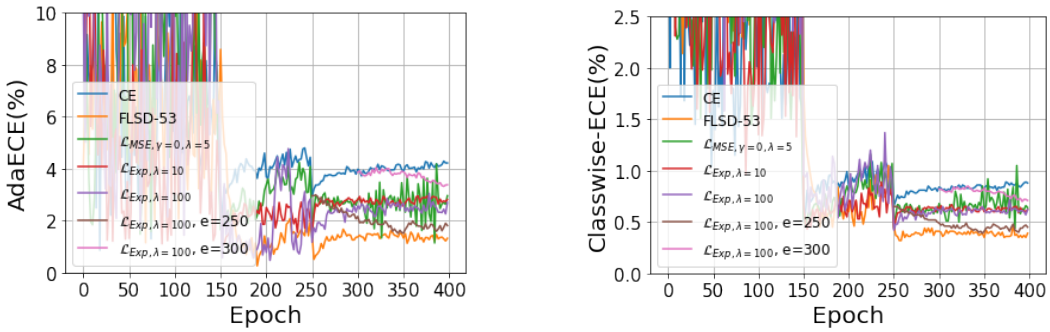


Figure 11: CIFAR-10, ResNet50 trained with focal loss $\gamma = 3$. It shows that $C_{train,true,i}$ and $C_{train,top,i}$ are mostly the same during major part of the training.

C CALFOCAL LOSS



(a) AdaECE

(b) Classwise-ECE

Figure 12: ResNet-50 trained on CIFAR-10 using using CalFocal loss.

Please ignore plots for $\mathcal{L}_{MSE}$ from these figures. They are not presented in any way in the main paper.

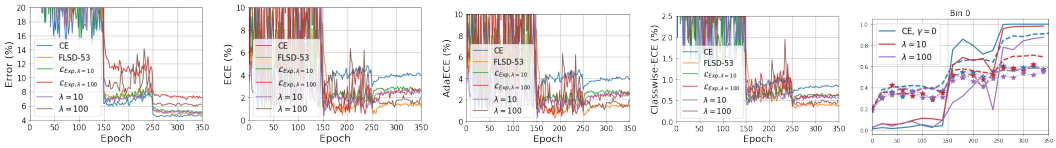


Figure 13: ResNet-50 trained on CIFAR-10 using CalFocal loss with common γ_b for all training sample in bin b. Legend $\mathcal{L}_{Exp,\lambda}$ represent the per sample CalFocal loss in Eq. 1 of the main paper. Legend λ represent the CalFocal loss with common γ_b for all training samples in bin b. **Please ignore plots for $\mathcal{L}_{MSE}$ from these figures.**

D DATASETS AND EXPERIMENTS

D.1 DATASET DESCRIPTION

CIFAR-10 Krizhevsky (2009): This dataset contains $60,000$ coloured images of size 32×32, which are equally divided into 10 classes. A split of $45,000/5,000/10,000$ images is used as train/validation/test sets respectively.

CIFAR-100 Krizhevsky (2009): This dataset contains $60,000$ coloured images of size 32×32, which are equally divided into 100 classes. A split of $45,000/5,000/10,000$ images is used as train/validation/test sets respectively.

Tiny-ImageNet Deng et al. (2009): It is a subset of the ImageNet dataset Russakovsky et al. (2015) with 64×64 dimensional images and 200 classes. It has 500 images per class in the training set and 50 images per class in the validation set.

20 Newsgroups Lang (1995): This dataset contains $20,000$ news articles, categorised evenly into 20 different newsgroups. Some of the newsgroups are very closely related to each other (e.g. comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (e.g misc.forsale / soc.religion.christian). We use a train/validation/test split of $15,098/900/3,999$ documents.

D.2 EXPERIMENTAL DETAILS

For all our experiments, we have used Nvidia Titan X Pascal GPU with 12GB memory.

CIFAR-10 and **CIFAR-100**: We use SGD with a momentum of 0.9 as our optimiser, and train the networks for 350 epochs, with a learning rate of 0.1 for the first 150 epochs, 0.01 for the next 100 epochs, and 0.001 for the last 100 epochs. We use a training batch size of 128. The training data is augmented by applying random crops and random horizontal flips.

Tiny-ImageNet: We use SGD with a momentum of 0.9 as our optimiser, and train the models for 100 epochs with a learning rate of 0.1 for the first 40 epochs, 0.01 for the next 20 epochs and 0.001 for the last 40 epochs. We use a training batch size of 64. Note that we use 50 samples per class (i.e. a total of 10000 samples) from the training set as the validation set. Hence, the training is only on 90000 images. We use the Tiny-ImageNet validation set as our test set.

20 Newsgroups: We train the Global Pooling CNN Network Lin et al. (2014) using the Adam optimiser, with learning rate 0.001, and default betas 0.9 and 0.999. We used Glove word embeddings Pennington et al. (2014) to train the network. We train the model for 50 epochs and use the model at the end to evaluate the performance.

All our experiments are implemented in PyTorch. The hyperparameters that are not explicitly mentioned above are set to their default values. For CIFAR-10/100 and Tiny-ImageNet, AdaFocal is implemented on top of the base code available from Mukhoti (2020). The code for 20 Newsgroups is implemented in PyTorch by adapting the code (TensorFlow) available from Kumar (2018).

The experimental results in the paper are reported for the model at the end of (1) CIFAR-10/100: 350 epochs, (2) Tiny-ImageNet: 100 epochs, and (3) 20 NewsGroups: 50 epochs.

E ADAECE AND CLASSWISE-ECE PERFORMANCE

Here, we compare the performance of AdaFocal against the baseline methods in terms of AdaECE and classwise-ECE in Table 4 and 5 respectively. For CIFAR-10/100, the values are reported for the model at the end of 350 epochs; for Tiny-ImageNet, at the end of 100 epochs; and for 20 NewsGroup dataset, at the end of 50 epochs. From these tables, we observe that AdaFocal outperforms all the baseline methods by a substantial margin, especially if we compare the pre-temperature scaling results.

Dataset	**Model**	**Cross Entropy**		**Brier Loss**		**MMCE**		**LS-0.05**		**FLSD-53**		**AdaFocal**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.33	2.14(2.5)	1.74	1.23(1.1)	4.55	2.16(2.6)	3.89	2.92(0.9)	1.56	1.26(1.1)	**0.56**	0.88(0.9)
	ResNet-110	4.4	1.99(2.8)	2.6	1.7(1.2)	5.06	2.52(2.8)	4.44	4.44(1)	2.07	1.67(1.1)	**0.44**	0.44(1.0)
	Wide-ResNet-26-10	3.23	1.69(2.2)	1.7	1.7(1)	3.29	1.6(2.2)	4.27	2.44(0.8)	1.52	1.38(0.9)	0.64	**0.42**(1.1)
	DenseNet-121	4.51	2.13(2.4)	2.03	2.03(1)	5.1	2.29(2.5)	4.42	3.33(0.9)	1.42	1.42(1)	**0.54**	0.54(1.0)
CIFAR-100	ResNet-50	17.52	3.42(2.1)	6.52	3.64(1.1)	15.32	2.38(1.8)	7.81	4.01(1.1)	4.5	2.0(1.1)	**1.72**	1.72(1.0)
	ResNet-110	19.05	5.86(2.3)	7.73	4.53(1.2)	19.14	4.85(2.3)	11.12	8.59(1.1)	8.55	3.96(1.2)	**1.57**	1.57(1.0)
	Wide-ResNet-26-10	15.33	2.89(2.2)	4.22	2.81(1.1)	13.16	4.25(1.9)	5.1	5.1(1)	2.75	**1.63**(1.1)	2.22	2.22(1.0)
	DenseNet-121	20.98	5.09(2.3)	5.04	2.56(1.1)	19.13	3.07(2.1)	12.83	8.92(1.2)	3.55	**1.24**(1.1)	1.54	1.54(1.0)
Tiny-ImageNet	ResNet-50	15.23	5.41(1.4)	4.37	4.07(0.9)	13.0	5.56(1.3)	15.28	6.29(0.7)	1.42	1.42(1)	**1.26**	1.26(1.0)
20 Newsgroups	Global-pool CNN	17.91	**2.23(3.4)**	13.57	3.11(2.3)	15.21	6.47(2.2)	4.39	2.63(1.1)	6.92	2.35(1.5)	2.38	2.38(1.0)

Table 4: $\text{AdaECE}(\%)$. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row.

Dataset	**Model**	**Cross Entropy**		**Brier Loss**		**MMCE**		**LS-0.05**		**FLSD-53**		**AdaFocal**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	0.91	0.45(2.5)	0.46	0.42(1.1)	0.94	0.52(2.6)	0.71	0.51(0.9)	0.42	0.42(1.1)	**0.29**	0.32(0.9)
	ResNet-110	0.91	0.50(2.8)	0.59	0.50(1.2)	1.04	0.55(2.8)	0.66	0.66(1)	0.48	0.44(1.1)	**0.33**	0.33(1.0)
	Wide-ResNet-26-10	0.68	0.37(2.2)	0.44	0.44(1)	0.70	0.35(2.2)	0.80	0.45(0.8)	0.41	0.31(0.9)	**0.27**	0.28(1.1)
	DenseNet-121	0.92	0.47(2.4)	0.46	0.46(1)	1.04	0.57(2.5)	0.60	0.50(0.9)	0.41	0.41(1)	**0.30**	0.30(1.0)
CIFAR-100	ResNet-50	0.38	0.22(2.1)	0.22	**0.20(1.1)**	0.34	0.21(1.8)	0.23	0.21(1.1)	**0.20**	0.20(1.1)	**0.20**	0.20(1.0)
	ResNet-110	0.41	0.21(2.3)	0.24	0.23(1.2)	0.42	0.22(2.3)	0.26	0.22(1.1)	0.24	0.21(1.2)	**0.19**	0.19(1.0)
	Wide-ResNet-26-10	0.34	0.20(2.2)	0.19	0.19(1.1)	0.31	0.20(1.9)	0.21	0.21(1)	**0.18**	0.19(1.1)	0.19	0.19(1.0)
	DenseNet-121	0.45	0.23(2.3)	0.20	0.21(1.1)	0.42	0.24(2.1)	0.29	0.24(1.2)	**0.19**	0.20(1.1)	0.20	0.20(1.0)
Tiny-ImageNet	ResNet-50	0.22	**0.16(1.4)**	**0.16**	**0.16(0.9)**	0.21	**0.16(1.3)**	0.21	0.17(0.7)	**0.16**	**0.16(1)**	**0.16**	0.16(1.0)
20 Newsgroups	Global-pool CNN	1.95	0.83(3.4)	1.56	**0.82(2.3)**	1.77	1.10(2.2)	0.93	0.91(1.1)	1.40	1.19(1.5)	1.02	1.02(1.0)

Table 5: $\text{ClasswiseECE}(\%)$. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row.

F DEBISED ESTIMATES OF ECE

Dataset	**Model**	**Cross Entropy**		**FLSD-53**		**AdaFocal**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.05	1.7(2.5)	1.62	1.62(1)	0.47	0.82(0.9)
	ResNet-110	4.38	2.2(2.7)	1.82	1.3(1.1)	0.32	0.32(1)
	Wide-ResNet-26-10	3.52	1.89(2.2)	2.01	1.5(0.9)	0.59	0.25(1.1)
	DenseNet-121	4.26	2.15(2.3)	1.56	1.93(0.9)	0.42	0.42(1)
CIFAR-100	ResNet-50	17.73	3.86(2.2)	5.52	2.92(1.1)	1.46	1.46(1)
	ResNet-110	19.44	6.01(2.3)	7.31	3.55(1.2)	1.35	1.35(1)
	Wide-ResNet-26-10	14.91	3.32(2.1)	2.53	2.53(1)	2.12	2.12(1)
	DenseNet-121	19.82	3.44(2.3)	2.29	2.12(1.1)	1.27	1.27(1)
Tiny-ImageNet	ResNet-50	16.16	5.44(1.5)	2.00	2.00(1)	0.84	0.84 (1)
20 Newsgroups	Global-pool CNN	18.36	5.23(4.1)	8.94	0.94(1.6)	1.84	1.84(1)

Table 6: $\mathrm{ECE}_{\mathrm{debias}}(\%)$ 15 bins. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row. Optimal temperature is selected based on the lowest ECE on the validation set.

Dataset	**Model**	**Cross Entropy**		**FLSD-53**		**AdaFocal**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.03	1.73(2.5)	1.57	1.57(1)	0.46	0.93(0.9)
	ResNet-110	4.38	2.21(2.7)	1.81	1.23(1.1)	0.26	0.26(1)
	Wide-ResNet-26-10	3.5	1.84(2.2)	1.98	1.5(0.9)	0.66	0.24(1.1)
	DenseNet-121	4.24	2.16(2.3)	1.6	1.95(0.9)	0.64	0.64(1)
CIFAR-100	ResNet-50	17.71	3.66(2.2)	5.52	2.99(1.1)	1.55	1.55(1)
	ResNet-110	19.43	6.31(2.3)	7.23	3.86(1.2)	1.17	1.17(1)
	Wide-ResNet-26-10	14.9	3.4(2.1)	2.46	2.46(1)	2.12	2.12(1)
	DenseNet-121	19.81	3.49(2.3)	2.18	2.31(1.1)	1.08	1.08(1)
Tiny-ImageNet	ResNet-50	16.11	5.5(1.5)	1.81	1.81(1)	0.64	0.64(1)
20 Newsgroups	Global-pool CNN	18.39	5.51(4.1)	8.98	1.89(1.6)	1.92	1.92(1)

Table 7: $\mathrm{ECE}_{\mathrm{debias}}(\%)$ 30bins. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row. Optimal temperature is selected based on the lowest ECE on the validation set.

Dataset	**Model**	**Cross Entropy**		**FLSD-53**		**AdaFocal**	
		Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.05	1.17(2.5)	1.31	1.31(1)	0.48	0.92(0.9)
	ResNet-110	4.38	1.34(2.7)	1.96	1.05(1.1)	0.5	0.5(1)
	Wide-ResNet-26-10	3.53	1.14(2.2)	1.22	0.92(0.9)	0.64	0.36(1.1)
	DenseNet-121	4.26	1.43(2.3)	0.94	1.9(0.9)	0.33	0.33(1)
CIFAR-100	ResNet-50	17.72	2.81(2.2)	5.54	2.17(1.1)	1.7	1.7(1)
	ResNet-110	19.44	3.74(2.3)	7.36	3.86(1.2)	1.54	1.54(1)
	Wide-ResNet-26-10	14.99	3.11(2.1)	2.69	2.69(1)	2.02	2.02(1)
	DenseNet-121	19.82	2.58(2.3)	2.24	2.34(1.1)	1.27	1.27(1)
Tiny-ImageNet	ResNet-50	16.19	5.56(1.5)	2.14	2.14(1)	1.37	1.37(1)
20 Newsgroups	Global-pool CNN	18.39	5.14(4.1)	9.03	1.48(1.6)	2.06	2.06(1)

Table 8: $\mathrm{ECE}_{\mathrm{EW-sweep}}(\%)$. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row. Optimal temperature is selected based on the lowest ECE on the validation set.

Dataset	Model	Cross Entropy		FLSD-53		AdaFocal	
		Pre T	Post T	Pre T	Post T	Pre T	Post T
CIFAR-10	ResNet-50	4.05	1.43(2.5)	1.54	1.54(1)	0.04	0.7(0.9)
	ResNet-110	4.38	1.34(2.7)	1.83	1.32(1.1)	0.4	0.4(1)
	Wide-ResNet-26-10	3.53	1.41(2.2)	1.64	1.55(0.9)	0.38	0.32(1.1)
	DenseNet-121	4.27	2.17(2.3)	1.58	1.98(0.9)	0.34	0.34(1)
CIFAR-100	ResNet-50	17.72	0.51(2.2)	5.51	2.36(1.1)	1.89	1.89(1)
	ResNet-110	19.44	3.71(2.3)	7.34	3.65(1.2)	1.58	1.58(1)
	Wide-ResNet-26-10	14.92	2.62(2.1)	2.62	2.62(1)	2.25	2.25(1)
	DenseNet-121	19.82	3.12(2.3)	2.25	2.31(1.1)	1.47	1.47(1)
Tiny-ImageNet	ResNet-50	16.19	5.44(1.5)	2.10	2.10(1)	1.6	1.6(1)
20 Newsgroups	Global-pool CNN	18.38	5.53(4.1)	8.95	2.13(1.6)	2.22	2.22(1)

Table 9: $\mathrm{ECE}_{\mathrm{EM-sweep}}(\%)$. Underline marks the lowest error among pre-temperature scaling values. Bold marks the overall lowest in the row. Optimal temperature is selected based on the lowest ECE on the validation set.

G NUMBER OF BINS USED TO TRAIN ADAFOCAL

Experiment details: CIFAR-10, ResNet50 trained for 350 epochs. The reported results below are without temperature scaling. Our method AdaFocal with 5, 10, 15, 20, 30, and 50 adaptive (equal mass) bins vs FLSD-53. Note here that there are two types of binning:

- During training: the binning that is performed on the validation set from where AdaFocal draws calibration related information to adjust the γ. This corresponds to the columns in the table below.
- During evaluation: once we have a trained model, the binning that is used to compute different ECE metrics. These correspond to the rows in the table below.

	FLSD-53	5bins	10bins	15bins	20bins	30bins	50bins
ECE(15bins)	1.35	0.76	0.53	0.51	0.6	0.82	1.16
AdaECE(15bins)	1.67	0.63	0.53	0.56	0.4	0.84	1.1
ECE_debias (15bins)	1.62	0.5	0.44	0.47	0.25	0.79	1.07
ECE_debias(30bins)	1.57	0.73	0.43	0.46	0.27	0.72	1.06
EW_ECE_sweep	1.31	0.66	0.43	0.48	0.48	0.8	1.08
EM_ECE_sweep	1.54	0.53	0.21	0.04	0.38	0.07	1.08

Table 10: Effect of number of bins used for AdaFocal.

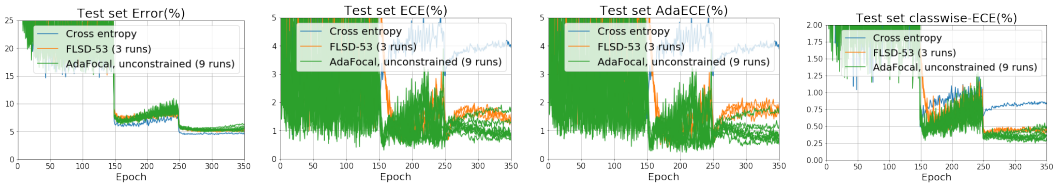
H MULTIPLE RUNS OF ADAFOCAL

Due to the stochastic nature of the experiments, AdaFocal γs may end up following different trajectories across different runs (initialization), which in turn might lead to variations in the final results. In this section, we look at the extent of such variations for (1) unconstrained γ (2) γ capped by $\gamma_{\max} = 20$ and (3) γ capped by $\gamma_{\max} = 50$. We study this for ResNet-50 trained multiple times on CIFAR-10 starting with a different random seed.

H.1 ADAFOCAL, UNCONSTRAINED γ

In Figure 14, we observe that AdaFocal with unconstrained γ does exhibit some variability across different runs: 7 out of 9 times it performs better than FLSD-53 whereas the other two times it is similar or slightly worse.

The above behaviour is mostly due to the variations in the trajectory of γs for lower bins, as shown in Figure 15. For higher bins, we see the γs settling to similar values, however for lower bins, as the γs are unconstrained they blow up to very high values.



(a) Error(%) (b) ECE (%) (c) AdaECE (%) (d) classwise-ECE(%)

Figure 14: Plots for ResNet-50 trained on CIFAR-10 using cross entropy (1 run), FLSD-53 (3 runs) and AdaFocal (9 runs). AdaFocal with unconstrained γ does exhibit some variability across different runs: 7 out of 9 times it is better than FLSD-53, two times it is similar or slightly worse.

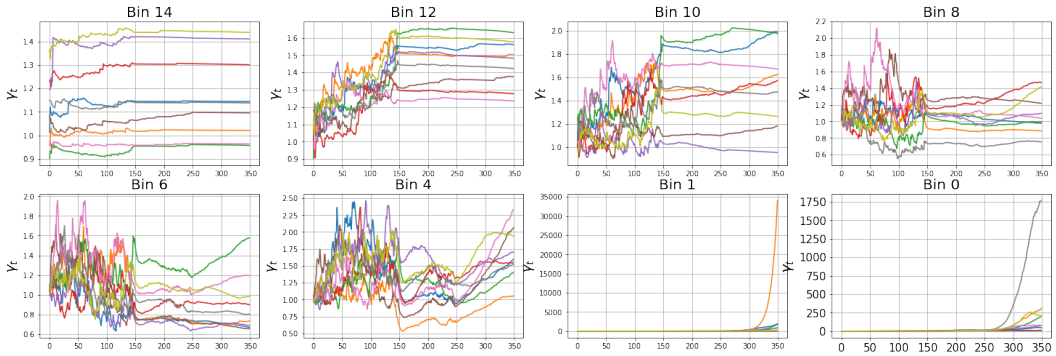
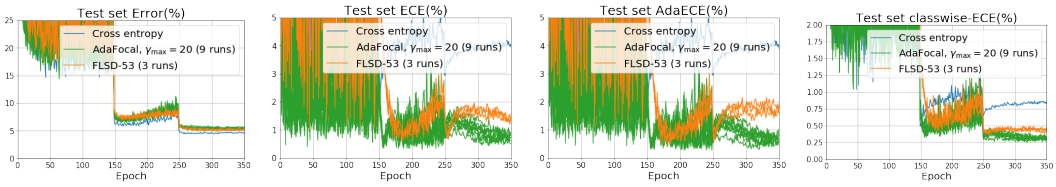


Figure 15: Evolution of γ_t for multiple runs of ResNet-50 trained on CIFAR-10 using unconstrained AdaFocal.

H.2 AdaFocal, $\gamma_{\max} = 20$

In Figure 16, we observe that AdaFocal with $\gamma_{\max} = 20$ is consistently (9 out of 9 times) better than FLSD-53. Figure 17 shows the evolution of γ across different runs.



(a) Error(%) (b) ECE (%) (c) AdaECE (%) (d) classwise-ECE(%)

Figure 16: Plots for ResNet-50 trained on CIFAR-10 using cross entropy (1 run), FLSD-53 (3 runs) and AdaFocal (9 runs). AdaFocal with $\gamma_{\max} = 20$ is consistently better than FLSD-53.

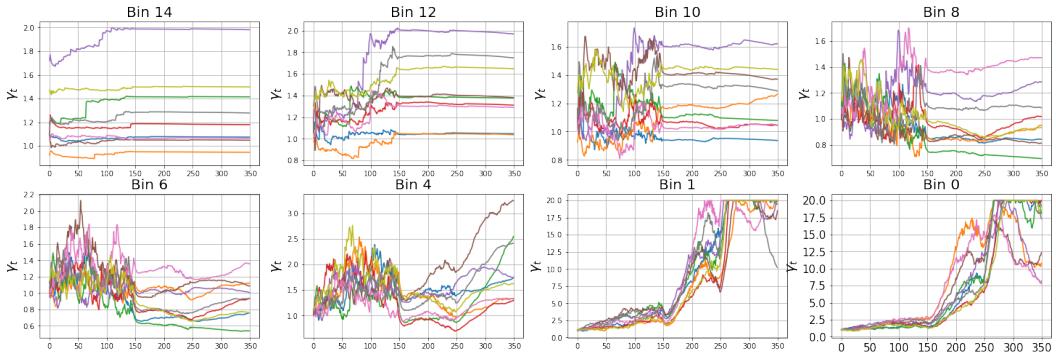


Figure 17: Evolution of γ_t for multiple runs of ResNet-50 trained on CIFAR-10 using AdaFocal $\gamma_{\max} = 20$.

I ERROR, ECE AND BIN STATISTICS PLOTS FOR REST OF THE EXPERIMENTS

Shown form next page onward.

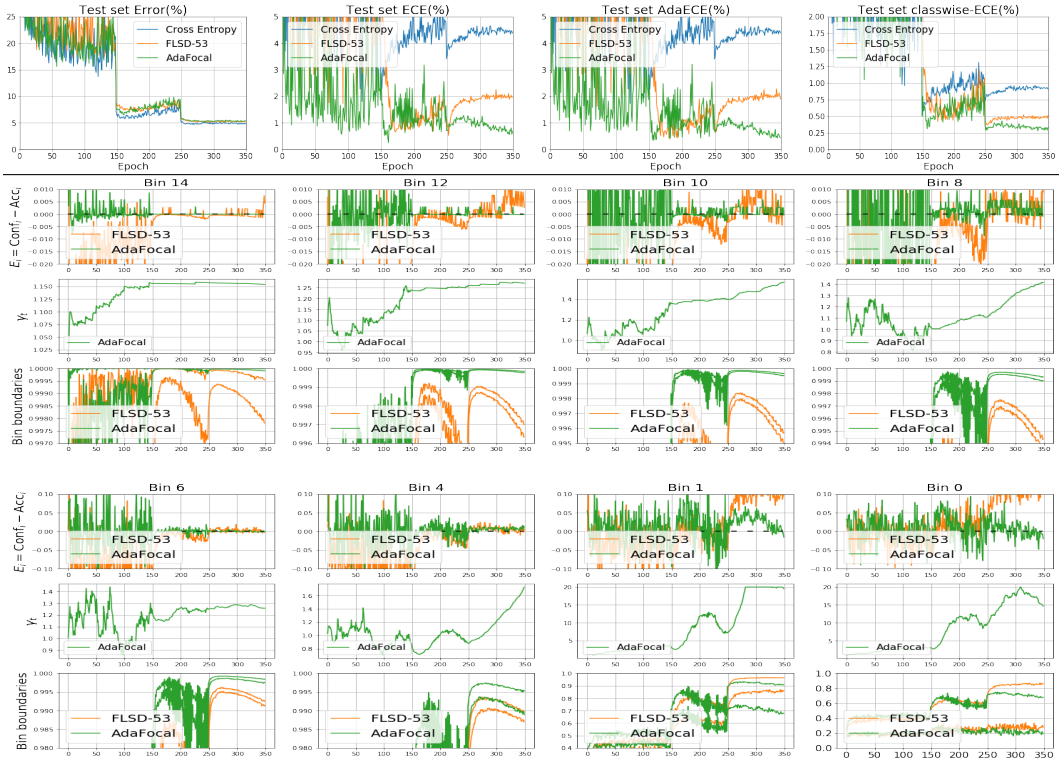


Figure 18: **CIFAR-10, ResNet-110**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

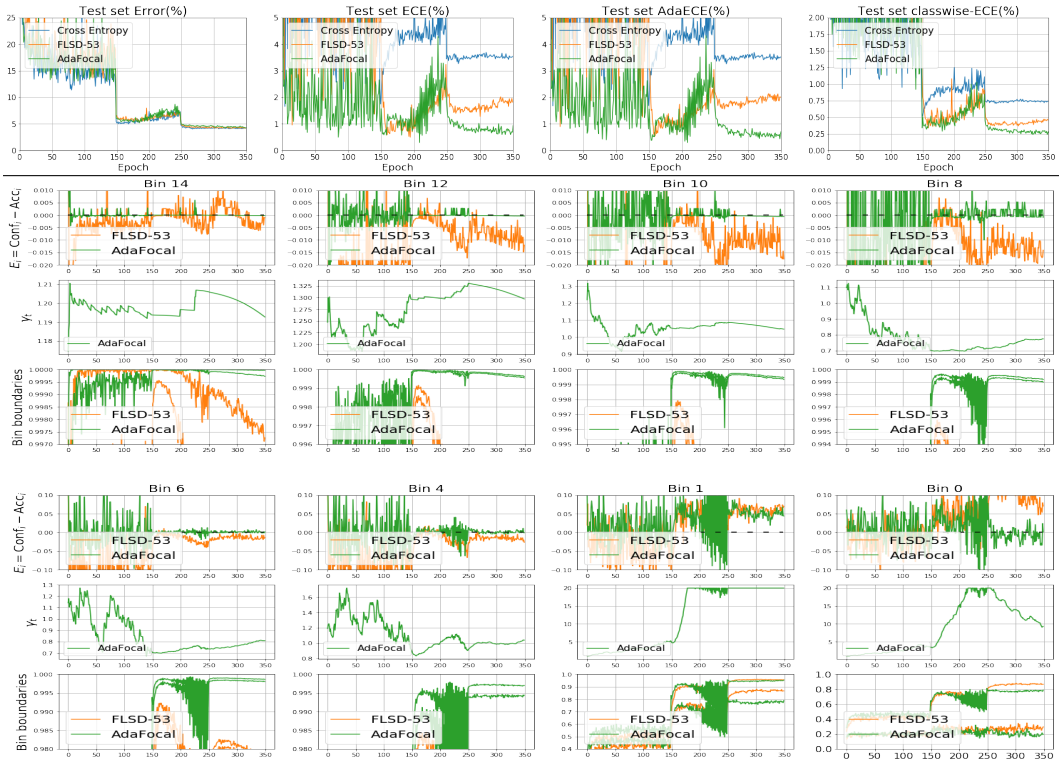


Figure 19: **CIFAR-10, Wide-ResNet**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

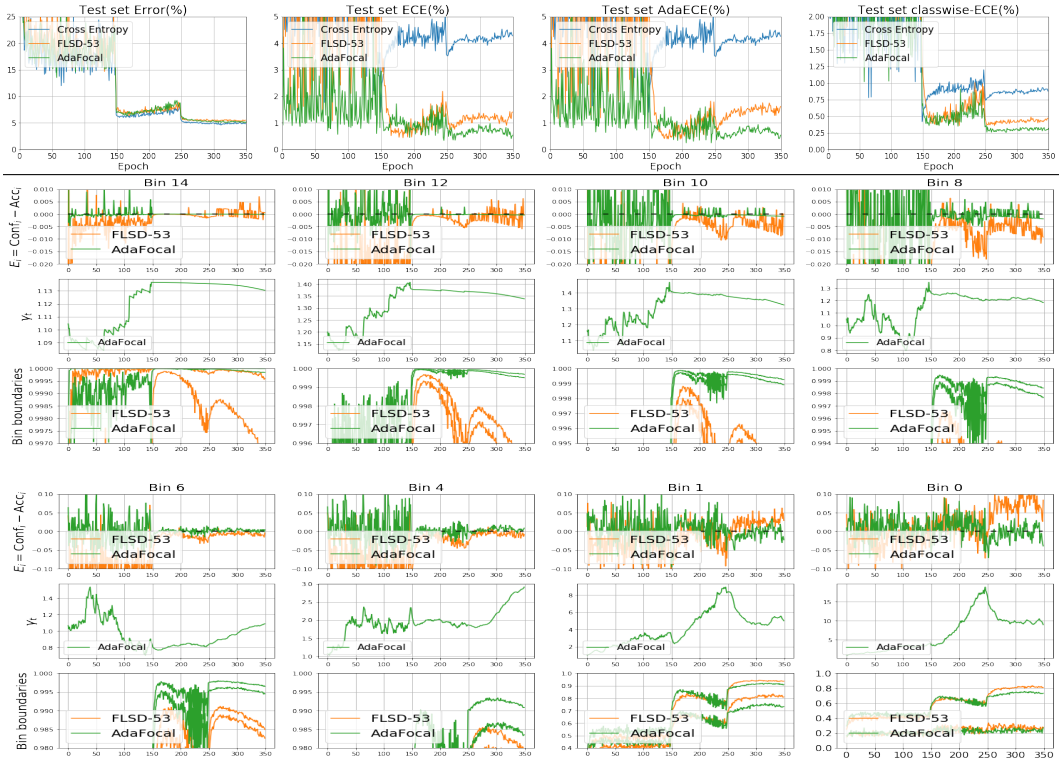


Figure 20: **CIFAR-10, DenseNet-121**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

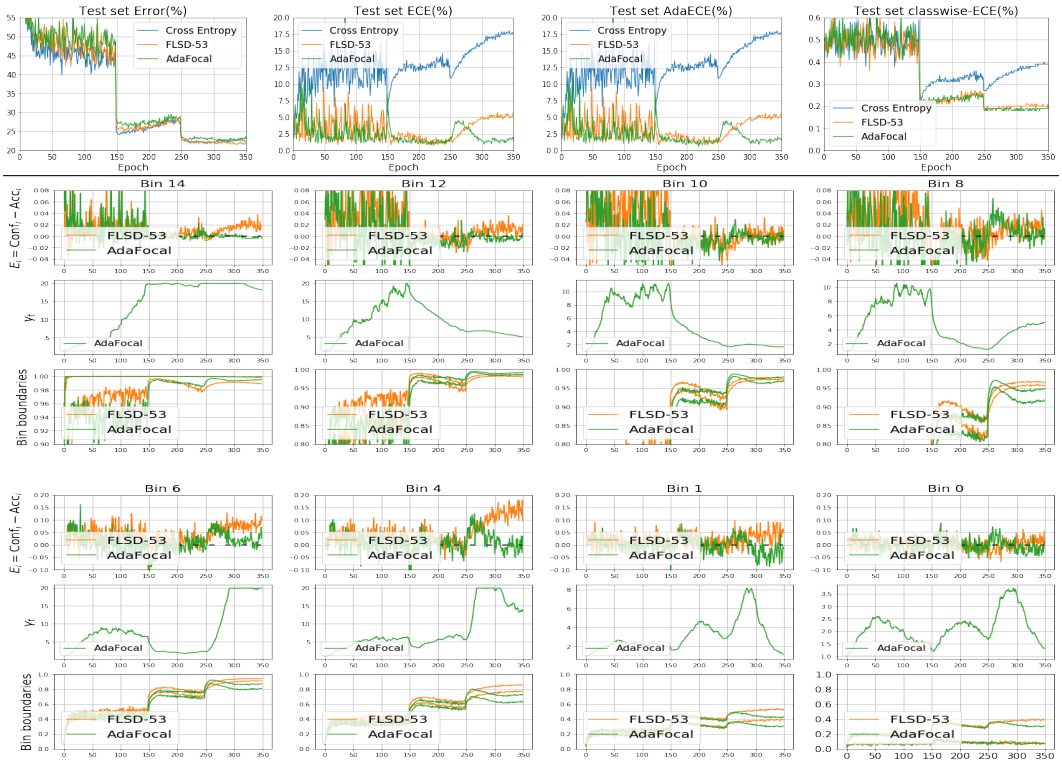


Figure 21: **CIFAR-100, ResNet-50**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

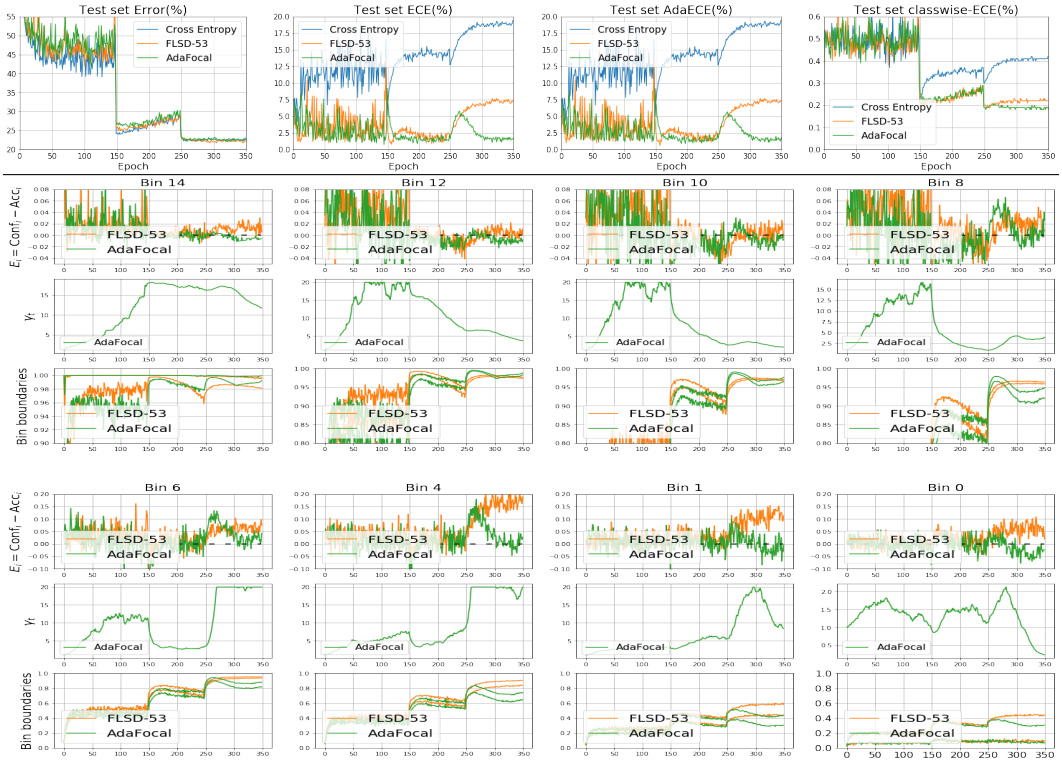


Figure 22: **CIFAR-100, ResNet-110**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

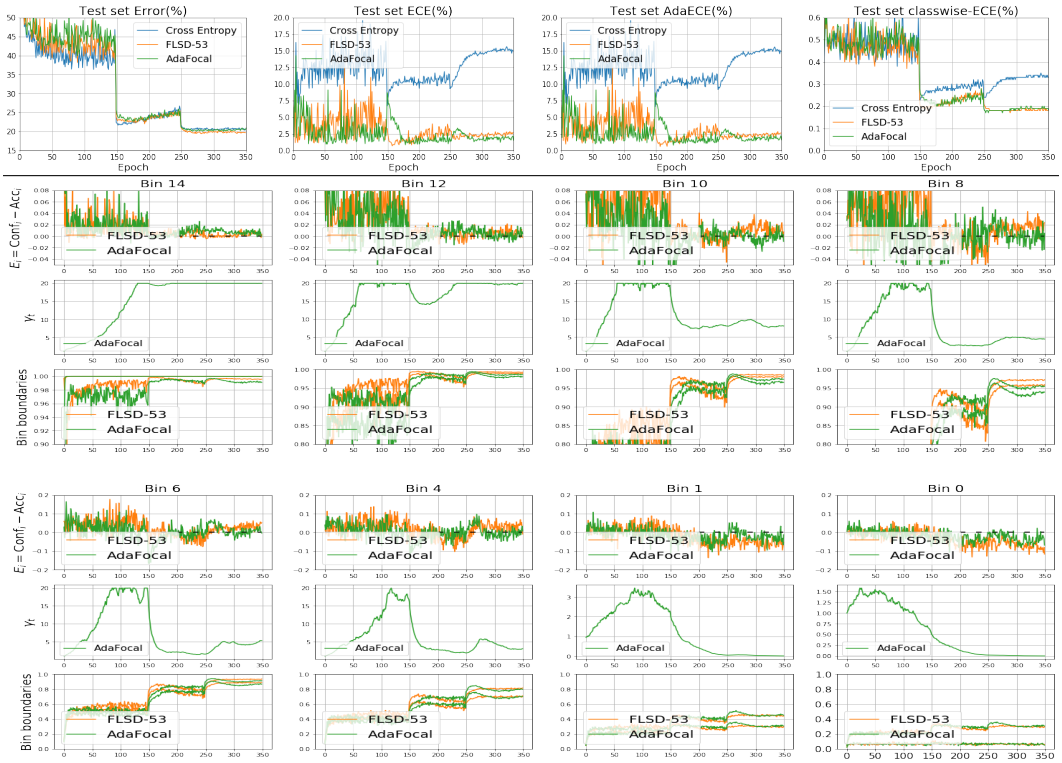


Figure 23: **CIFAR-100, Wide-ResNet**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

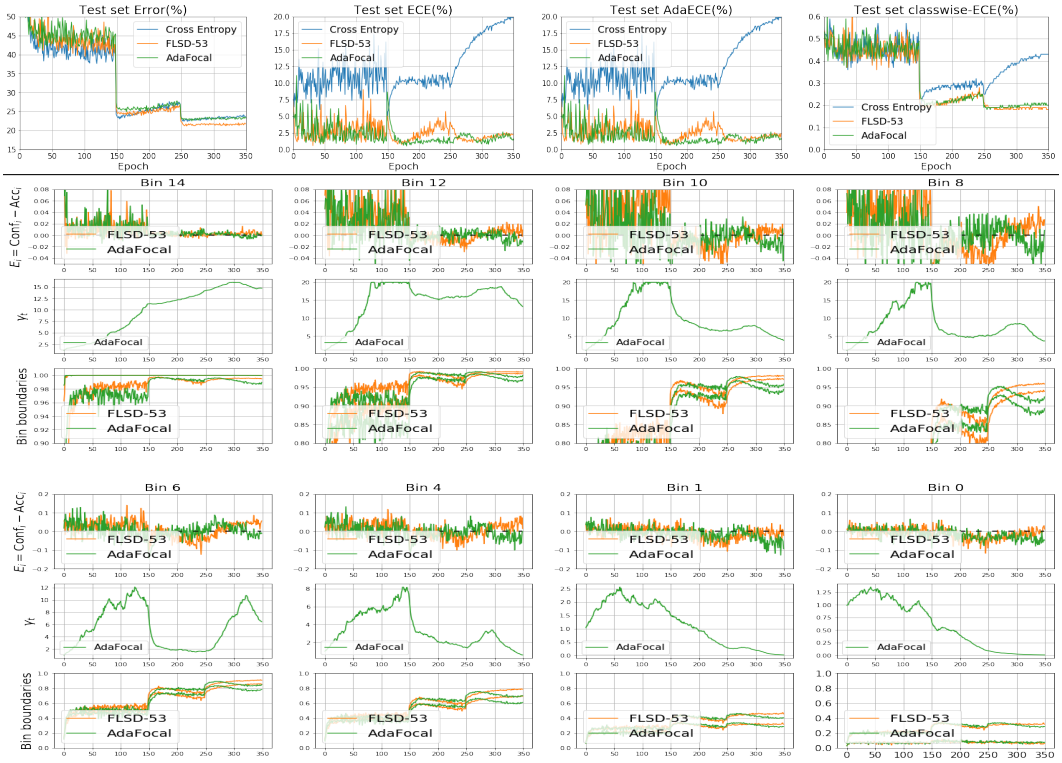


Figure 24: **CIFAR-100, DenseNet-121**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

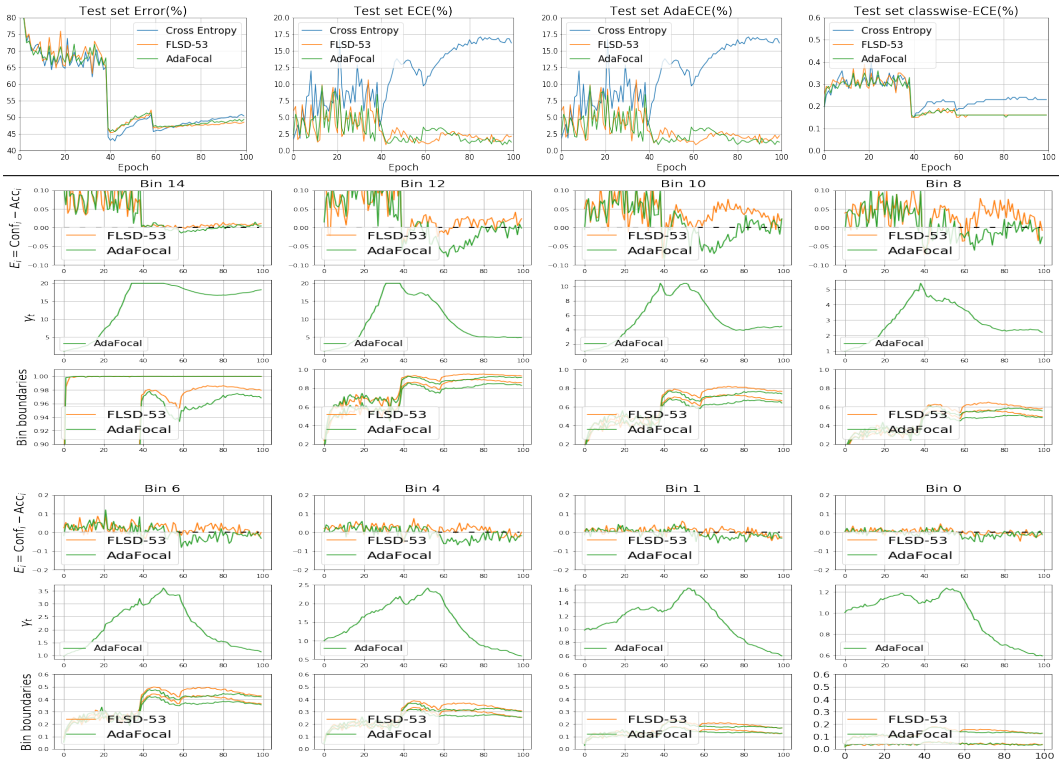


Figure 25: **Tiny-ImageNet, ResNet-50**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.

Figure 26: **20 Newsgroups, CNN**: Test set Error, ECE, AdaECE, classwise-ECE and Validation set bin information used by AdaFocal during training.