A Details about Datasets

Stack Overflow It contains 6,633 sequences with 480,414 events where an event is the acquisition of badges received by users. The maximum number of sequence length is 736 and the number of marks is 22. The dataset is provided by Du et al. (2016); we use the first folder, following Shchur et al. (2020) and Bae et al. (2023).

Reddit It contains 10,000 sequences with 532,026 events where an event is posting in Reddit. The maximum number of sequence length is 736 and the number of marks is 22. Marks represent sub-reddit categories.

Uber It contains 791 sequences with 701,579 events where an event is pick-up of customers. The maximum number of sequence length is 2,977 and there is no marks. It is processed and provided by Bae et al. (2023).

Brazilian Houses It contains information of 10,962 houses to rent in Brazil in 2020 with 13 features. The target is the rent price for each house in Brazilian Real. According to OpenML (Vanschoren et al., 2013) where we obtained this dataset, since the data is web-scrapped, there are some values in the dataset that can be considered outliers.

Wine Quality It contains 6,497 samples with 11 features and the quality of wine is numerically labeled as targets. This dataset is also obtained from OpenML (Vanschoren et al., 2013).

SST-2 The Stanford Sentiment Treebank (SST-2) is a dataset containing fully annotated parse trees, enabling a comprehensive exploration of how sentiment influences language composition. Comprising 11,855 individual sentences extracted from film reviews, this dataset underwent parsing using the Stanford parser, resulting in a collection of 215,154 distinct phrases.

B Additional Results on Image Classification

Datasets We use ImageNet100 (Tian et al., 2020) for image classification with random crop, horizontal flip, and color jitter as augmentation. We also add 30% of label noise as done in Section 4.3.

Implementation We train ResNet34 (He et al., 2016) on the dataset with L_2 regularization (with a weight of 0.0001). All methods are trained for 200 epochs with early stopping using SGD. We use a multi-step learning rate scheduler with an initial learning rate of 0.1 and decay coefficient of 0.5, applied at every 25 epochs. The optimal flood levels are selected based on validation performance with a grid search on $\{0.01, 0.02..., 0.1, 0.15, 0.2..., 0.3\}$ for Flood and iFlood, and $\{0.05, 0.1..., 0.95\}$ for AdaFlood. We use a single ResNet34 auxiliary network where its last FC layer is randomly initialized and fine-tuned on held-out sets with $n = 10$ splits.

Results Table 4 (Left) compares flooding methods on ImageNet100 dataset with and without 30% of label noise. We report test accuracies along with expected calibration error (ECE) on the right. Although Flood and iFlood do not improve the performance over the unregularized model, AdaFlood improves the performance by about 0.80% over the unregularized baseline. Given the size of the dataset, the gap is not marginal. This gap is even larger than that we observed in SVHN and CIFAR datasets Table 2. We conjecture it is because ImageNet contains more noisy samples. It is well-known that there are many ImageNet images containing multiple objects although the label says there is only one object (Beyer et al., 2020).

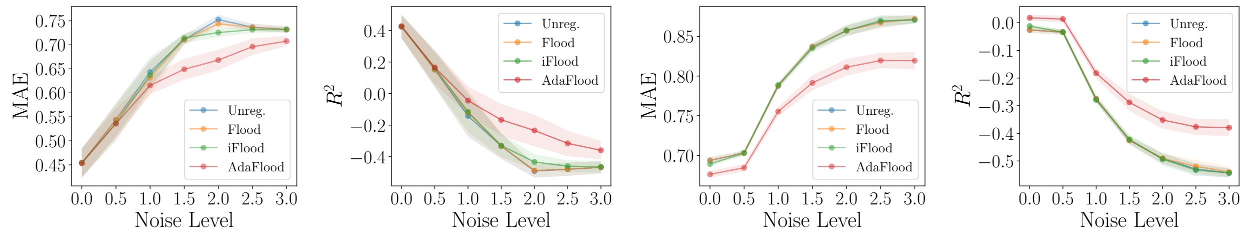
C Additional Results on Tabular Regression

Datasets We use NYC Taxi Tip dataset from OpenML (Vanschoren et al., 2013), one of the largest tabular dataset used in Grinsztajn et al. (2022), for regression tasks. NYC Taxi Tip dataset contains $581,835$ rows and 9 features. As the name of the dataset implies the target variable is “tip amount". To increase the importance of other features, the creator of the dataset deliberately ignores “fare amount" or “trip distance".

Method	Mislabled Sample Rate	
	0%	30%
Unreg.	81.00 / 6.64	68.12 (17.23)
Flood	81.18 (6.44)	68.19 (17.71)
iFlood	81.04 (6.86)	68.24 (20.67)
AdaFlood	**81.79**(**4.81**)	**69.22**(**17.45**)

Method	Noise Level		
	0.0	1.5	3.0
Unreg.	0.2373(0.3335)	0.3707 (-0.0409)	0.3910 (-0.0978)
Flood	0.2373(0.3335)	0.3707 (-0.0409)	0.3904 (-0.0980)
iFlood	0.2370(0.3374)	0.3652 (-0.0255)	0.3902 (-0.0986)
AdaFlood	**0.2369**(**0.3348**)	**0.3520**(**0.0250**)	**0.3465**(**0.0119**)

Table 4: Comparison of flooding methods on ImageNet100 (Left) and NYC Taxi Tip (Right) datasets with and without label noise. The numbers on the right on ImageNet100 represents ECE metric whereas on NYC Taxi Tip, they are R^2 scores.



(a) Brazilian House (MAE) (b) Brazilian House (R^2) (c) Wine Quality (MAE) (d) Wine Quality (R^2)

Figure 8: Additional results in various metrics on tabular datasets with noise and bias

Implementation As with Section 4.3, we use a model tailored for tabular dataset proposed by (Grinsztajn et al., 2022) and add errors sampled from a skewed normal distribution, with skewness parameter ranging from 0.0 to 3.0.

Results Table 4 (Right) compares flooding methods on NYC Taxi Tip dataset (Grinsztajn et al., 2022) with and without noises. We report mean square error (MSE) and R2 score on the right. Note that R2 score is usually in between 0 and 1 but when predictions are bad, it can go below 0.

From the table, we can observe that all flooding methods perform similar to the unregularized baseline when there is no noise. Although it continues for Flood and iFlood even under noisy settings, AdaFlood significantly outperforms (lower MSE and higher R^2 scores) the other methods when noise level is 1.5 and 3.0. In particular, while R2 scores of other methods go below 0, it does not happen with AdaFlood, which demonstrates the robustness of AdaFlood even for the large-scale dataset like NYC Taxi Tip. It is consistent with the results we provided in Section 4.3.

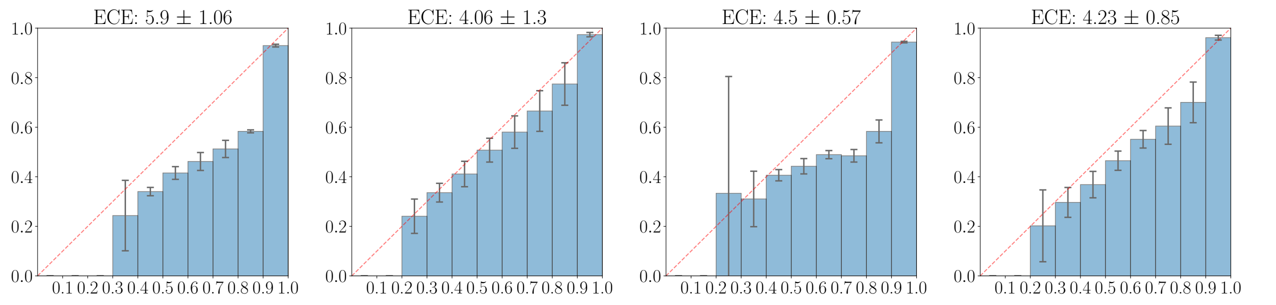
For additional information, in Figure 8, we also provide experiment results in MAE and R^2 on Brazilian House and Wine Quality datasets of which results in MSE is provided in Figures 5a and 5b.

D Additional Results on Calibration

We provide calibration results on CIFAR10 in Figure 9. As briefly mentioned in Section 4.4, since model predictions are generally accurate and quite confident, the gap between different methods are not as significant as on CIFAR100 shown in Figure 6.

E Initialization of the Main Model using an Auxiliary Network

Even though we efficiently fine-tune an auxiliary network to compute flood levels, it may be still too expensive to train both the auxiliary network f^{aux} and main model f. To reduce computation further, we may utilize a pre-trained auxiliary network when we use the same architecture for the auxiliary and main model. Instead of randomly initialize the main model, we can initialize the parameters of the main model using the parameters from the pre-trained auxiliary network (before fine-tuning).



(a) Unregularized (b) Flood (c) iFlood (d) AdaFlood

Figure 9: Calibration results of flooding methods with 10 bins on CIFAR10.

Random Init	Freezing first k layers			
	$k = 0$	$k = 1$	$k = 2$	$k = 3$
90.83 / 3.52	91.30 / 2.50	90.90 / 3.44	90.68 / 3.96	89.99 / 4.69

Table 5: Comparison between random initialization and initialization with the pre-trained auxiliary network. ECE metrics are reported on the right.

We conduct an experiment to validate if it actually saves computation without hurting performance on CIFAR10 with ResNet18. We first trained an auxiliary network on the whole training set. The pre-trained auxiliary network is then used for both computing θ through the fine-tuning step described in Section 3.3 and initializing the main model. When we "fine-tune" the main model, we freeze the first k layers (among four layers of ResNet18) to save computation, and randomly initialize the last fully connected layer to have the model forget some information.

Table 5 shows that if we initialize the parameters of the main model with those of the pre-trained auxiliary model without freezing any layers, it performs better than random initialization. As we freeze more layers to save computation, the performance gradually goes down compared to that of random initialization but it is still comparable up to $k = 2$ case.

F Ablation study: Various Architectures for Auxiliary Networks

	VGG11	VGG19	ResNet18 Small	ResNet18
# of params	9.8 M	21.6 M	2.8 M	11.2 M
AdaFlood	90.74 ± 0.13	91.09 ± 0.06	90.73 ± 0.12	90.82 ± 0.08

Table 6: Comparison of various architectures for auxiliary networks in AdaFlood.

To investigate the robustness of AdaFlood in terms of the choice of architectures for auxiliary networks, we conduct an ablation study on AdaFlood with various architectures for auxiliary networks: VGG11, VGG19, ResNet18 small and ResNet18. Here, we use a ResNet18 for the main model, and utilize the efficient fine-tuning method to train auxiliary networks. We report the mean and standard error of three runs in Table 6.

With VGG11 where its number of parameters is slightly less than ResNet18, the mean test accuracy of the main model is lower but the gap is marginal. With VGG19 which is larger than ResNet18, the performance slightly improves. We also try with a smaller variant of ResNet18 for the auxiliary network where its number of parameters is a quarter of the original ResNet18. Even with this significantly smaller architecture, the mean test accuracy of the main model is only slightly worse than much larger models e.g. VGG11 and ResNet18, which implies that what is important from the auxiliary network is the relative magnitude of losses (or flood levels) not the absolute values of losses.

NTPP	Method	Uber	Reddit	Stack Overflow
Intensity-free	Flood	{4.0, 4.0, 3.0}	{-1.0, -5.0, -3.0}	{0.0, 3.0, 1.0}
	iFlood	{4.0, 0.0, 1.0}	{-2.0, -3.0, -4.0}	{2.0, 2.0, 3.0}
	AdaFlood	{0.2, 0.0, 0.3}	{0.0, 0.1, 0.1}	{0.1, 0.4, 0.2}
THP$^+$	Flood	{0.0, -5.0, 0.0}	{-1.0, -2.0, -15.0}	{0.0, 2.0, 1.0}
	iFlood	{-1.0, 4.0, 3.0}	{-4.0, -4.0, -5.0}	{0.0, 0.0, -1.0}
	AdaFlood	{0.0, 0.1, 0.3}	{0.0, 0.0, 0.1}	{0.3, 0.3, 0.3}

Table 7: Choice of flood levels for (i)Flood, and γ for AdaFlood on TPP datasets with three seeds.

Method	SVHN		CIFAR10		CIFAR100	
	w/o L_2 reg.	w/ L_2 reg.	w/o L_2 reg.	w/ L_2 reg.	w/o L_2 reg.	w/ L_2 reg.
Flood	{0.04, 0.02, 0.03}	{0.02, 0.03, 0.04}	{0.02, 0.04, 0.05}	{0.07, 0.02, 0.01}	{0.01, 0.01, 0.01}	{0.01, 0.01, 0.05}
iFlood	{0.05, 0.09, 0.02}	{0.05, 0.06, 0.06}	{0.07, 0.04, 0.07}	{0.07, 0.03, 0.01}	{0.45, 0.35, 0.30}	{0.30, 0.40, 0.65}
AdaFlood	{0.75, 0.85, 0.70}	{0.35, 0.50, 0.65}	{0.95, 0.70, 0.65}	{0.75, 0.95, 0.65}	{0.40, 0.50, 0.45}	{0.50, 0.50, 0.55}

Table 8: Choice of flood levels for (i)Flood, and γ for AdaFlood on image classification with three seeds.

G Theoretic Intuition for Efficient Training of Auxiliary Networks

In this section, we provide theoretic intuition for efficient training of auxiliary networks. Following (Lee et al., 2019), we approximate the predictions of a neural network f on a test sample x_i

$$f(x_i) \approx \sum_{j=1}^{n} \alpha_j \mathrm{eNTK}(x_i, x_j) \tag{11}$$

where eNTK stands for empirical neural tangent kernel (NTK) following (Mohamadi et al., 2023) and $\{x_j\}_{j=1}^n$ denotes data from a training set. Equation (11) says we can approximate a prediction on x_i as an interpolation of $\mathrm{eNTK}(x_i, \cdot)$ with some weights α.

Suppose $f(x) = V\phi(x)$ where $\phi(x) \in \mathbb{R}^h$ denotes a feature from the penultimate layer and $V \in \mathbb{R}^{k\times h}$ denotes the weights of the last fully connected layer (k being the number of classes), consisting of $v_j \in \mathbb{R}^h$ for j-th row. If $v_{j,i}$, i-th entry of v_j, is from $N(0, \sigma^2)$, then Mohamadi et al. (2023) haven shown that,

$$\mathrm{eNTK}_w(x_1, x_2)_{jj'} = v_j^T \mathrm{eNTK}^{\phi}_{w\setminus V}(x_1, x_2) v_{j'} + \mathbb{1}(j = j')\phi(x_1)^T \phi(x_2) \tag{12}$$

where w denotes a set of all the model parameters and $w \setminus V$ means a set of all the parameters except the last fully connected layer.

With this frame, we can approximate the predictions from the efficiently trained auxiliary network denoted as f_{tune}, as follows,

$$f_{\text{tune}}(x_i) \approx \sum_{j\neq i} \alpha_j^{\text{tune}} \left(v^T \mathrm{eNTK}^{\phi,\text{trained}}_{w\setminus V}(x_j, x_i) v + \phi^{\text{trained}}(x_j)^T \phi^{\text{trained}}(x_i) \right). \tag{13}$$

Here, superscript "trained" means the model parameters are pre-trained on the whole training set.

On the other hand, the original "direct" training algorithm of auxiliary networks trains an auxiliary network from scratch, on the training set excluding the samples that we measure the difficulty on. Similarly, the predictions of a single auxiliary network denoted as f_{direct} is approximated as,

$$f_{\text{direct}}(x_i) \approx \sum_{j\neq i} \alpha_j^{\text{direct}} \left(v^T \mathrm{eNTK}^{\phi,\text{untrained}}_{w\setminus V}(x_j, x_i) v + \phi^{\text{untrained}}(x_j)^T \phi^{\text{untrained}}(x_i) \right) \tag{14}$$

where superscript "untrained" means the model parameters are randomly initialized.

In **NTK regime** where a neural network has infinite-width, the terms in the parentheses (the term except α's) are the same for Equation (13) and Equation (14). Therefore, the difficulty measures from the efficient training of a single auxiliary network and direct training of multiple auxiliary networks are equivalent in highly-overparameterized regime.

H Choice of Flood Levels

In Table 7 and Table 8, we report the choice of flood levels for Flood and iFlood, and γ for AdaFlood (recall that γ is the hyperparameter for correct function adjusting interpolation level) on TPP and image classification tasks with three different random seeds.

One interesting observation is that flood levels for iFlood on CIFAR100 is significantly larger than on the other datasets. It is because CIFAR100 is particularly harder than the other two datasets. However, even though a flood level is sometimes high *e.g.* 0.65, it is still a reasonable choice because 0.65 flood level implies that the model's highest predicted probabilities do not deviate much from $0.52 \sim \exp(-0.65)$. With this high regularization, the model is not too overconfident for its predictions.