Do We Need Zero Training Loss After Achieving Zero Training Error?

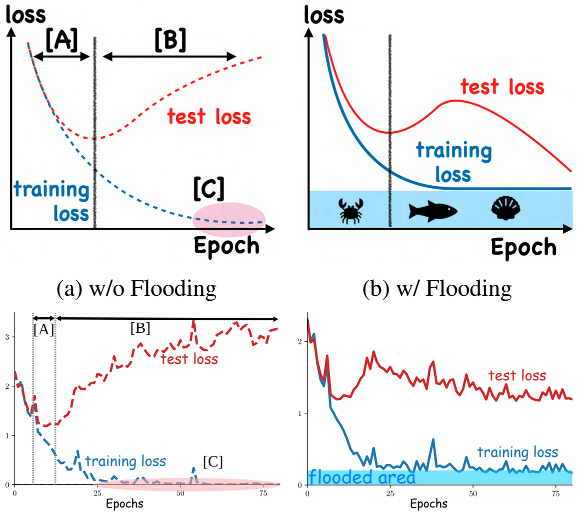
Takashi Ishida [1 2] **Ikko Yamane** [1] **Tomoya Sakai** [3] **Gang Niu** [2] **Masashi Sugiyama** [2 1]

Abstract

Overparameterized deep networks have the capacity to memorize training data with zero *training error*. Even after memorization, the *training loss* continues to approach zero, making the model overconfident and the test performance degraded. Since existing regularizers do not directly aim to avoid zero training loss, it is hard to tune their hyperparameters in order to maintain a fixed/preset level of training loss. We propose a direct solution called *flooding* that intentionally prevents further reduction of the training loss when it reaches a reasonably small value, which we call the *flood level*. Our approach makes the loss float around the flood level by doing mini-batched gradient descent as usual but gradient ascent if the training loss is below the flood level. This can be implemented with one line of code and is compatible with any stochastic optimizer and other regularizers. With flooding, the model will continue to "random walk" with the same non-zero training loss, and we expect it to drift into an area with a flat loss landscape that leads to better generalization. We experimentally show that flooding improves performance and, as a byproduct, induces a double descent curve of the test loss.

1. Introduction

"Overfitting" is one of the biggest interests and concerns in the machine learning community (Ng, 1997; Caruana et al., 2000; Belkin et al., 2018; Roelofs et al., 2019; Werpachowski et al., 2019). One way of identifying overfitting is to see whether the generalization gap, the test minus the training loss, is increasing or not (Goodfellow et al., 2016). We can further decompose the situation of the generalization gap increasing into two stages: The first stage is when both the training and test losses are decreasing, but the training loss is decreasing faster than the test loss ([A] in Fig. 1(a).) The next stage is when the training loss is decreasing, but the test loss is increasing.



(a) w/o Flooding (b) w/ Flooding

(c) C10 w/o Flooding (d) C10 w/ Flooding

Figure 1. (a) shows 3 different concepts related to overfitting. [A] shows the generalization gap increases, while the training and test losses decrease. [B] also shows the increasing gap, but the test loss starts to rise. [C] shows the training loss becoming (near-)zero. We avoid [C] by *flooding* the bottom area, visualized in (b), which forces the training loss to stay around a constant. This leads to a decreasing test loss once again. We confirm these claims in experiments with CIFAR-10 shown in (c)–(d).

Within stage [B], after learning for even more epochs, the training loss will continue to decrease and may become (near-)zero. This is shown as [C] in Fig. 1(a). If we continue training even after the model has memorized (Zhang et al., 2017; Arpit et al., 2017; Belkin et al., 2018) the training data completely with zero error, the training loss can easily become (near-)zero especially with overparametrized models. Recent works on overparametrization and double descent curves (Belkin et al., 2019; Nakkiran et al., 2020) have shown that learning until zero training error is meaningful to achieve a lower generalization error. However, whether zero training *loss* is necessary after achieving zero training *error* remains an open issue.

In this paper, we propose a method to make the training loss *float* around a small constant value, in order to prevent

[1]The University of Tokyo [2]RIKEN [3]NEC Corporation. Correspondence to: Takashi Ishida <ishida@ms.k.u-tokyo.ac.jp>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020. Copyright 2020 by the author(s).

the training loss from approaching zero. This is analogous to *flooding* the bottom area with water, and we refer to the constant value as the *flood level*. Note that even if we add flooding, we can still memorize the training data. Our proposal only forces the training *loss* to become positive, which does not necessarily mean the training *error* will become positive, as long as the flood level is not too large. The idea of flooding is shown in Fig. 1(b), and we show learning curves before and after flooding with a benchmark dataset in Fig. 1(c) and Fig. 1(d).[1]

Algorithm and implementation Our algorithm of flooding is surprisingly simple. If the original learning objective is J, the proposed modified learning objective $\widetilde{J}$ with flooding is

$$\widetilde{J}(\boldsymbol{\theta}) = |J(\boldsymbol{\theta}) - b| + b, \tag{1}$$

where $b > 0$ is the flood level specified by the user, and $\boldsymbol{\theta}$ is the model parameter.

The gradient of $\widetilde{J}$ w.r.t. $\boldsymbol{\theta}$ will point in the same direction as that of $J(\boldsymbol{\theta})$ when $J(\boldsymbol{\theta}) > b$ but in the opposite direction when $J(\boldsymbol{\theta}) < b$. This means that when the learning objective is above the flood level, there is a "gravity" effect with gradient *descent*, but when the learning objective is below the flood level, there is a "buoyancy" effect with gradient *ascent*. In practice, this will be performed with a mini-batch and will be compatible with any stochastic optimizers. It can also be used along with other regularization methods.

During flooding, the training loss will repeat going below and above the flood level. The model will continue to "random walk" with the same non-zero training loss, and we expect it to drift into an area with a flat loss landscape that leads to better generalization (Hochreiter & Schmidhuber, 1997; Chaudhari et al., 2017; Keskar et al., 2017; Li et al., 2018). In experiments, we show that during this period of random walk, there is an increase in flatness of the loss function (See Section 5.3).

This modification can be incorporated into existing machine learning code easily: Add one line of code for Eq. (1) after evaluating the original objective function $J(\boldsymbol{\theta})$. A minimal working example with a mini-batch in PyTorch (Paszke et al., 2019) is demonstrated below to show the additional one line of code:

```
outputs = model(inputs)
loss = criterion(outputs, labels)
flood = (loss-b).abs()+b  # This is it!
optimizer.zero_grad()
flood.backward()
optimizer.step()
```

It may be hard to set the flood level without expert knowledge on the domain or task. We can circumvent this situation easily by treating the flood level as a hyper-parameter. We may exhaustively evaluate the accuracy for the predefined hyper-parameter candidates with a validation dataset, which can be performed in parallel.

Previous regularization methods Many previous regularization methods also aim at avoiding *training too much* in various ways including restricting the parameter norm to become small by decaying the parameter weights (Hanson & Pratt, 1988), raising the difficulty of training by dropping activations of neural networks (Srivastava et al., 2014), avoiding the model to output a hard label by smoothing the training labels (Szegedy et al., 2016), and simply stopping training at an earlier phase (Morgan & Bourlard, 1990). These methods can be considered as indirect ways to control the training loss, by also introducing additional assumptions, e.g., the optimal model parameters are close to zero. Although making the regularization effect stronger would make it harder for the training loss to approach zero, it is still hard to maintain the right level of training loss till the end of training. In fact, for overparametrized deep networks, applying small regularization would not stop the training loss becoming (near-)zero, making it even harder to choose a hyper-parameter that corresponds to a specific level of loss.

Flooding, on the other hand, is a direct solution to the issue that the training loss becomes (near-)zero. Flooding intentionally prevents further reduction of the training loss when it reaches a reasonably small value, and the flood level corresponds to the level of training loss that the user wants to keep.

2. Backgrounds

In this section, we review regularization methods (summarized in Table 1), recent works on overparametrization and double descent curves, and the area of weakly supervised learning where similar techniques to flooding have been explored.

2.1. Regularization Methods

The name "regularization" dates back to at least Tikhonov regularization for the ill-posed linear least-squares problem (Tikhonov, 1943; Tikhonov & Arsenin, 1977). One example is to modify $\boldsymbol{X}^\top \boldsymbol{X}$ (where $\boldsymbol{X}$ is the design matrix) to become "regular" by adding a term to the objective function. ℓ_2 regularization is a generalization of the above example and can be applied to non-linear models. These methods implicitly assume that the optimal model parameters are close to zero.

It is known that weight decay (Hanson & Pratt, 1988), dropout (Srivastava et al., 2014), and early stopping (Mor-

[1] For the details of these experiments, see Appendix D.

Table 1. Conceptual comparisons of various regularizers. "Indep."/"tr." stands for "independent"/"training" and ✓/× stands for yes/no.

Regularization and other methods	Target tr. loss	Domain indep.	Task indep.	Model indep.	Main assumption
ℓ_2 regularization (Tikhonov, 1943)	×	✓	✓	✓	Optimal model params are close to 0
Weight decay (Hanson & Pratt, 1988)	×	✓	✓	✓	Optimal model params are close to 0
Early stopping (Morgan & Bourlard, 1990)	×	✓	✓	✓	Overfitting occurs in later epochs
ℓ_1 regularization (Tibshirani, 1996)	×	✓	✓	✓	Optimal model params are sparse
Dropout (Srivastava et al., 2014)	×	✓	✓	×	Co-adaptations do not generalize well
Batch normalization (Ioffe & Szegedy, 2015)	×	✓	✓	×	Internal covariate shift is harmful
Label smoothing (Szegedy et al., 2016)	×	✓	×	✓	True posterior is not a one-hot vector
Mixup (Zhang et al., 2018)	×	×	×	✓	Linear relationship between $\boldsymbol{x}$ and y
Image augment. (Shorten & Khoshgoftaar, 2019)	×	×	✓	✓	Input is invariant to the translations
Flooding (proposed method)	✓	✓	✓	✓	Learning until zero loss is harmful

gan & Bourlard, 1990) are equivalent to ℓ_2 regularization under certain conditions (Loshchilov & Hutter, 2019; Bishop, 1995; Goodfellow et al., 2016; Wager et al., 2013), implying that they have similar assumptions on the optimal model parameters. There are other penalties based on different assumptions such as the ℓ_1 regularization (Tibshirani, 1996) based on the sparsity assumption that the optimal model has only a few non-zero parameters.

Modern machine learning tasks are applied to complex problems where the optimal model parameters are not necessarily close to zero or are not sparse, and it would be ideal if we can properly add regularization effects to the optimization stage without such assumptions. Our proposed method does not have assumptions on the optimal model parameters and can be useful for more complex problems.

More recently, "regularization" has further evolved to a more general meaning including various methods that alleviate overfitting but do not necessarily have a step to regularize a singular matrix or add a regularization term to the objective function. For example, Goodfellow et al. (2016) defines regularization as "any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error." In this paper, we adopt this broader meaning of "regularization."

Examples of the more general regularization category include mixup (Zhang et al., 2018) and data augmentation methods like cropping, flipping, and adjusting brightness or sharpness (Shorten & Khoshgoftaar, 2019). These methods have been adopted in many state-of-the-art methods (Verma et al., 2019; Berthelot et al., 2019; Kolesnikov et al., 2020) and are becoming essential regularization tools for developing new systems. However, these regularization methods have the drawback of being domain-specific: They are designed for the vision domain and require some efforts when applying to other domains (Guo et al., 2019; Thulasidasan et al., 2019). Other regularizers such as label smoothing (Szegedy et al., 2016) is used for problems with class labels and harder to use with regression or ranking, meaning that they are task-specific. Batch normalization (Ioffe & Szegedy, 2015) and dropout (Srivastava et al., 2014) are designed for neural networks and are model-specific.

Although these regularization methods—both the *special* and *general* ones—already work well in practice and have become the de facto standard tools (Bishop, 2011; Goodfellow et al., 2016), we provide an alternative which is even more general in the sense that it has the potential to be domain-, task-, and model-independent.

That being said, we want to emphasize that the most important difference between flooding and other regularization methods is whether it is possible to target a specific level of training loss other than zero. While flooding allows the user to choose the level of training loss directly, it is hard to achieve this with other regularizers.

2.2. Double Descent Curves with Overparametrization

Recently, there has been increasing attention on the phenomenon of "double descent," named by Belkin et al. (2019), to explain the two regimes of deep learning: The first one (underparametrized regime) occurs where the model complexity is small compared to the number of samples, and the test error as a function of model complexity decreases with low model complexity but starts to increase after the model complexity is large enough. This follows the classical view of machine learning that excessive complexity leads to poor generalization. The second one (overparametrized regime) occurs when an even larger model complexity is considered. Then increasing the complexity only decreases test error, which leads to a double descent shape. The phase of decreasing test error often occurs after the training error becomes zero. This follows the modern view of machine learning that bigger models lead to better generalization.[2]

As far as we know, the discovery of double descent curves dates back to at least Krogh & Hertz (1992), where they theoretically showed the double descent phenomenon under

[2] `https://www.eff.org/ai/metrics`

a linear regression setup. Recent works (Belkin et al., 2019; Nakkiran et al., 2020) have shown empirically that a similar phenomenon can be observed with deep learning methods. Nakkiran et al. (2020) observed that the double descent curves for the *test error* can be shown not only as a function of model complexity, but also as a function of the epoch number.

To the best of our knowledge, the epoch-wise double descent curve was not observed for the *test loss* before but was observed in our experiments after using flooding with only about 100 epochs. Investigating the connection between epoch-wise double descent curves for the test loss and previous double descent curves (Krogh & Hertz, 1992; Belkin et al., 2019; Nakkiran et al., 2020) is out of the scope of this paper but is an important future direction.

2.3. Avoiding Over-Minimization of the Empirical Risk

It is commonly observed that the empirical risk goes below zero, and it causes overfitting (Kiryo et al., 2017) in *weakly supervised learning* when an equivalent form of the risk expressed with the given weak supervision is alternatively used (Natarajan et al., 2013; Cid-Sueiro et al., 2014; du Plessis et al., 2014; 2015; Patrini et al., 2017; van Rooyen & Williamson, 2018). Kiryo et al. (2017) proposed a gradient ascent technique to keep the empirical risk non-negative. This idea has been generalized and applied to other weakly supervised settings (Han et al., 2020; Ishida et al., 2019; Lu et al., 2020).

Although we also set a lower bound on the empirical risk, the motivation is different: First, while Kiryo et al. (2017) and others aim to fix the negative empirical risk to become non-negative, our original empirical risk is already non-negative. Instead, we are aiming to sink the original empirical risk by modifying it with a *positive* lower bound. Second, the problem settings are different. Weakly supervised learning methods require certain loss corrections or sample corrections (Han et al., 2020) before the non-negative correction, but we work on the original empirical risk without any setting-specific modifications.

Early stopping (Morgan & Bourlard, 1990) may be a naive solution to this problem where the empirical risk becomes too small. However, performance of early stopping highly depends on the training dynamics and is sensitive to the randomness in the optimization method and mini-batch sampling. This suggests that early stopping at the optimal epoch in terms of a single training path does not necessarily perform well in another round of training. This makes it difficult to use hyper-parameter selection techniques such as cross-validation that requires re-training a model more than once. In our experiments, we will demonstrate how flooding performs even better than early stopping.

3. Flooding: How to Avoid Zero Training Loss

In this section, we propose our regularization method, *flooding*. Note that this section and the following sections only consider multi-class classification for simplicity.

3.1. Preliminaries

Consider input variable $\boldsymbol{x} \in \mathbb{R}^d$ and output variable $y \in [K] := \{1, \ldots, K\}$, where K is the number of classes. They follow an unknown joint probability distribution with density $p(\boldsymbol{x}, y)$. We denote the score function by $\boldsymbol{g} : \mathbb{R}^d \to \mathbb{R}^K$. For any test data point $\boldsymbol{x}_0$, our prediction of the output label will be given by $\widehat{y}_0 := \arg\max_{z \in [K]} g_z(\boldsymbol{x}_0)$, where $g_z(\cdot)$ is the z-th element of $\boldsymbol{g}(\cdot)$, and in case of a tie, $\arg\max$ returns the largest argument. Let $\ell : \mathbb{R}^K \times [K] \to \mathbb{R}$ denote a loss function. ℓ can be the *zero-one loss*,

$$\ell_{01}(\boldsymbol{v}, z') := \begin{cases} 0 & \text{if} \quad \arg\max_{z \in \{1,\ldots,K\}} v_z = z', \\ 1 & \text{otherwise}, \end{cases}$$

where $\boldsymbol{v} := (v_1, \ldots, v_K)^\top \in \mathbb{R}^K$, or a surrogate loss such as the softmax cross-entropy loss,

$$\ell_{\mathrm{CE}}(\boldsymbol{v}, z') := -\log \frac{\exp(v_{z'})}{\sum_{z \in [K]} \exp(v_z)}.$$

For a surrogate loss ℓ, we denote the *classification risk* by

$$R(\boldsymbol{g}) := \mathbb{E}_{p(\boldsymbol{x},y)}[\ell(\boldsymbol{g}(\boldsymbol{x}), y)] \tag{2}$$

where $\mathbb{E}_{p(\boldsymbol{x},y)}[\cdot]$ is the expectation over $(\boldsymbol{x}, y) \sim p(\boldsymbol{x}, y)$. We use $R_{01}(\boldsymbol{g})$ to denote Eq. (2) when $\ell = \ell_{01}$ and call it the *classification error*.

The goal of multi-class classification is to learn $\boldsymbol{g}$ that minimizes the classification error $R_{01}(\boldsymbol{g})$. In optimization, we consider the minimization of the risk with a almost surely differentiable surrogate loss $R(\boldsymbol{g})$ instead to make the problem more tractable. Furthermore, since $p(\boldsymbol{x}, y)$ is usually unknown and there is no way to exactly evaluate $R(\boldsymbol{g})$, we minimize its empirical version calculated from the training data instead:

$$\widehat{R}(\boldsymbol{g}) := \frac{1}{n} \sum_{i=1}^{n} \ell(\boldsymbol{g}(\boldsymbol{x}_i), y_i), \tag{3}$$

where $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ are i.i.d. sampled from $p(\boldsymbol{x}, y)$. We call $\widehat{R}$ the *empirical risk*.

We would like to clarify some of the undefined terms used in the title and the introduction. The "train/test loss" is the empirical risk with respect to the surrogate loss function ℓ over the training/test data, respectively. We refer to the "training/test error" as the empirical risk with respect to ℓ_{01} over the training/test data, respectively (which is equal to one minus accuracy) (Zhang, 2004). Also see Guo et al.

(2017) for a discussion of the empirical differences of loss and error with neural networks.

Finally, we formally define the Bayes risk as

$$R^* := \inf_{\boldsymbol{h}} R(\boldsymbol{h}),$$

where the infimum is taken over all measurable, vector-valued functions $\boldsymbol{h}\colon \mathbb{R}^d \to \mathbb{R}^K$. The Bayes risk is often referred to as the *Bayes error* if the zero-one loss is used:

$$\inf_{\boldsymbol{h}} R_{01}(\boldsymbol{h}).$$

3.2. Algorithm

With flexible models, $\widehat{R}(\boldsymbol{g})$ w.r.t. a surrogate loss can easily become small if not zero, as we mentioned in Section 1; see [C] in Fig. 1(a). We propose a method that "floods the bottom area and sinks the original empirical risk" as in Fig. 1(b) so that the empirical risk cannot go below the flood level. More technically, if we denote the flood level as b, our proposed training objective with flooding is a simple fix.

Definition 1. *The* flooded empirical risk *is defined as*[3]

$$\widetilde{R}(\boldsymbol{g}) = |\widehat{R}(\boldsymbol{g}) - b| + b. \tag{4}$$

Note that when $b = 0$, then $\widetilde{R}(\boldsymbol{g}) = \widehat{R}(\boldsymbol{g})$. The gradient of $\widetilde{R}(\boldsymbol{g})$ w.r.t. model parameters will point to the same direction as that of $\widehat{R}(\boldsymbol{g})$ when $\widehat{R}(\boldsymbol{g}) > b$ but in the opposite direction when $\widehat{R}(\boldsymbol{g}) < b$. This means that when the learning objective is above the flood level, we perform gradient *descent* as usual (gravity zone), but when the learning objective is below the flood level, we perform gradient *ascent* instead (buoyancy zone).

The issue is that in general, we seldom know the optimal flood level in advance. This issue can be mitigated by searching for the optimal flood level b^* with a hyper-parameter optimization technique. In practice, we can search for the optimal flood level by performing the exhaustive search in parallel.

3.3. Implementation

For large scale problems, we can employ mini-batched stochastic optimization for efficient computation. Suppose that we have M disjoint mini-batch splits. We denote the empirical risk (3) with respect to the m-th mini-batch by $\widehat{R}_m(\boldsymbol{g})$ for $m \in \{1, \dots, M\}$. Our mini-batched optimization performs gradient descent updates in the direction of the gradient of $|\widehat{R}_{\mathrm{m}}(\boldsymbol{g}) - b| + b$. By the convexity of the absolute value function and Jensen's inequality, we have

$$\widetilde{R}(\boldsymbol{g}) \leq \frac{1}{M} \sum_{m=1}^{M} \left(|\widehat{R}_{\mathrm{m}}(\boldsymbol{g}) - b| + b \right).$$

This indicates that mini-batched optimization will simply minimize an upper bound of the full-batch case with $\widetilde{R}(\boldsymbol{g})$.

4. Does Flooding Generalize Better?

In this section, we show experimental results to demonstrate that adding flooding leads to better generalization. The implementation in this section and the next is based on PyTorch (Paszke et al., 2019) and demo code is available.[4] Experiments were carried out with NVIDIA GeForce GTX 1080 Ti, NVIDIA Quadro RTX 5000 and Intel Xeon Gold 6142.

4.1. Synthetic Datasets

The aim of synthetic experiments is to study the behavior of flooding with a controlled setup. We use three types of synthetic data: Two Gaussians, Sinusoid (Nakkiran et al., 2019), and Spiral (Sugiyama, 2015). See Appendix C.1 for the details of how these data were generated.

Settings For Two Gaussians, we use a one-hidden-layer feedforward neural network with 500 units in each hidden layer with the ReLU activation function (Nair & Hinton, 2010). We train the network for 1000 epochs with the logistic loss and vanilla gradient descent with 0.05 learning rate. The flood level is chosen from $b \in \{0, 0.01, 0.02, \dots, 0.40\}$.

For Sinusoid and Spiral, we use a four-hidden-layer feedforward neural network with 500 units in each hidden layer, the ReLU acitvation function (Nair & Hinton, 2010), and the batch normalization (Ioffe & Szegedy, 2015). We train the network for 500 epochs with the logistic loss and the Adam (Kingma & Ba, 2015) optimizer with 100 mini-batch size and learning rate of 0.001. The flood level is chosen from $b \in \{0, 0.01, 0.02, \dots, 0.20\}$. We tried adding label noise to training and validation dataset, by flipping 1%, 5%, or 10% of the labels randomly.

Note that the training with $b = 0$ is identical to the baseline method without flooding. We report the test accuracy of the flood level with the best validation accuracy. We first conduct experiments without early stopping, which means that the last epoch was chosen for all flood levels.

[3] Strictly speaking, Eq. (1) is different from Eq. (4), since Eq. (1) can ignore constant terms of the original empirical risk. We will refer to Eq. (4) for the flooding operator for the rest of the paper. Adding back b will not affect the gradient but will ensure $\widetilde{R}(\boldsymbol{g}) = \widehat{R}(\boldsymbol{g})$ when $\widehat{R}(\boldsymbol{g}) > b$.

[4] https://github.com/takashiishida/flooding

Table 2. Experimental results for the synthetic data. Sub-table (A) shows the results without early stopping. Sub-table (B) shows the results with early stopping. The better method is shown in **bold** in each of sub-tables (A) and (B). "BR" stands for the Bayes risk. For Two Gaussians, the distance between the positive and negative distributions is larger for larger m.

		(A) Without Early Stopping			(B) With Early Stopping		
Data	Setting	Without Flooding	With Flooding	Chosen b	Without Flooding	With Flooding	Chosen b
Two Gaussians	m: 1.0, BR: 0.14	87.96%	**92.25%**	0.28	91.63%	**92.25%**	0.27
Two Gaussians	m: 0.8, BR: 0.24	82.00%	**87.31%**	0.33	86.57%	**87.29%**	0.35
Sinusoid	Label Noise: 0.01	93.84%	**94.46%**	0.01	**92.54%**	**92.54%**	0.00
Sinusoid	Label Noise: 0.05	91.12%	**95.44%**	0.10	93.26%	**94.60%**	0.01
Sinusoid	Label Noise: 0.10	86.57%	**96.02%**	0.17	**96.70%**	**96.70%**	0.00
Spiral	Label Noise: 0.01	**98.96%**	97.85%	0.01	98.60%	**98.88%**	0.01
Spiral	Label Noise: 0.05	93.87%	**96.24%**	0.04	**96.58%**	95.62%	0.14
Spiral	Label Noise: 0.10	89.70%	**92.96%**	0.16	89.70%	**92.96%**	0.16

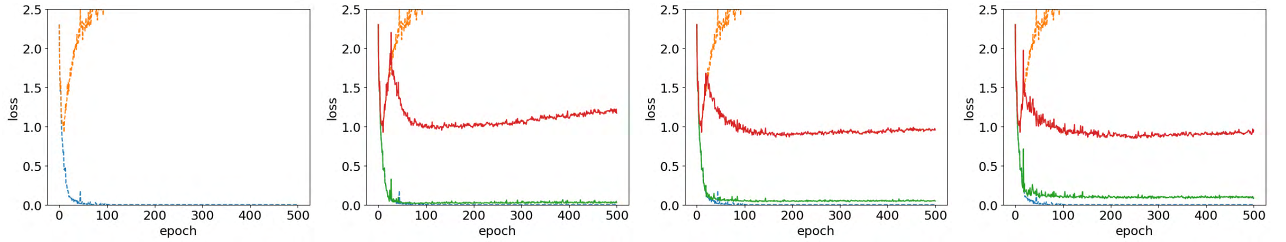
Table 3. Benchmark datasets. Reporting accuracy for all combinations of weight decay (✓, ×), early stopping (✓, ×) and flooding (✓, ×). W, E, and F stand for weight decay, early stopping, and flooding, respectively. "—" means that flood level of zero was optimal. "N/A" means that we skipped the experiments because zero weight decay was optimal in the case without flooding. The best and equivalent are shown in **bold** by comparing "F:✓" and "F:×" for two columns with the same W and E, e.g., the 1st and 5th columns out of the 8 columns. The best combination is in **red**.

Dataset	W:	×	×	✓	✓	×	×	✓	✓
	E:	×	✓	×	✓	×	✓	×	✓
	F:	×	×	×	×	✓	✓	✓	✓
MNIST		98.32%	98.30%	**98.51%**	98.42%	**98.46%**	**98.53%**	98.50%	**98.48%**
Fashion		89.34%	89.36%	N/A	N/A	—	—	N/A	N/A
Kuzushiji		91.63%	91.62%	91.63%	91.71%	**92.40%**	**92.12%**	**92.11%**	**91.97%**
CIFAR-10		**73.59%**	73.36%	73.65%	73.57%	73.06%	**73.44%**	—	**74.41%**
CIFAR-100		42.16%	42.33%	42.67%	42.45%	**42.50%**	**42.36%**	—	—
SVHN		92.38%	92.41%	93.20%	92.99%	**92.78%**	**92.79%**	—	**93.42%**

Results The results are summarized in Table 2. It is worth noting that for Two Gaussians, the chosen flood level b is larger for the smaller distance between the two distributions, which is when the classification task is harder and the Bayes risk becomes larger since the two distributions become less separated. See Appendix B for how we analytically derived the Bayes risk. This implies the positive correlation between the optimal flood level and the Bayes risk. In Theorem 1 in Appendix A, we show a related connection between the risk and the flood level.

From (A) in Table 2, we can see that the method with flooding often improves test accuracy over the baseline method without flooding. As we mentioned in the introduction, it can be harmful to keep training a model until the end without flooding. However, with flooding, the model at the final epoch has good prediction performance according to the results, which implies that flooding helps the late-stage training improve test accuracy.

We also conducted experiments with early stopping, meaning that we chose the model that recorded the best validation accuracy during training. The results are reported in sub-table (B) of Table 2. Compared with sub-table (A), we see that early stopping improves the baseline method without flooding in many cases. This indicates that training longer without flooding was harmful in our experiments. On the other hand, the accuracy for flooding combined with early stopping is often close to that with early stopping, meaning that training until the end with flooding tends to be already as good as doing so with early stopping. The table shows that flooding often improves or retains the test accuracy of the baseline method without flooding even after deploying early stopping. Flooding does not hurt performance but can be beneficial for methods used with early stopping.



(a) CIFAR10 (0.00) (b) CIFAR10 (0.03) (c) CIFAR10 (0.07) (d) CIFAR10 (0.20)

Figure 2. Learning curves of training and test loss for training/validation proportion of 0.8. For each row, the first figure is the learning curves without flooding. The 2nd, 3rd, and 4th figures show the results with different flood levels. Blue: training loss w/o flooding. Orange: test loss w/o flooding. Green: training loss w/ flooding. Red: test loss w/ flooding. See Fig. 6 in Appendix for other datasets.

4.2. Benchmark Experiments

We next perform experiments with benchmark datasets. Not only do we compare with the baseline without flooding, we also compare or combine with other general regularization methods, which are early stopping and weight decay.

Settings We use the following six benchmark datasets: MNIST, Fashion-MNIST, Kuzushiji-MNIST, CIFAR-10, CIFAR-100, and SVHN. The details of the benchmark datasets can be found in Appendix C.2. We split the original training dataset into training and validation data with with a proportion of $8 : 2$. We perform the exhaustive hyper-parameter search for the flood level with candidates from $\{0.00, 0.01, \ldots, 0.20\}$. The number of epochs is 500. Stochastic gradient descent (Robbins & Monro, 1951) is used with learning rate of 0.1 and momentum of 0.9. For MNIST, Fashion-MNIST, and Kuzushiji-MNIST, we use a one-hidden-layer feedforward neural network with 500 units and the ReLU activation function (Nair & Hinton, 2010). For CIFAR-10, CIFAR-100, and SVHN, we used ResNet-18 from He et al. (2016) with the default implementation provided in Paszke et al. (2019). We did not use any data augmentation or manual learning rate decay. We used the original labels and did not add label noise. We deployed early stopping in the same way as in Section 4.1.

We first ran experiments with the following candidates for the weight decay rate: $\{1 \times 10^{-5}, 1 \times 10^{-4}, 4 \times 10^{-4}, 7 \times 10^{-4}, 1 \times 10^{-3}, 4 \times 10^{-3}, 7 \times 10^{-3}\}$. We choose the weight decay rate with the best validation accuracy, for each dataset and each training/validation proportion. Then, fixing the weight decay to the chosen one, we ran experiments with flood level candidates from $\{0, 0.01, \ldots, 0.20\}$, to investigate whether weight decay and flooding have complementary effects, or if adding weight decay will diminish the accuracy gain of flooding.

Results We show the results in Table 3. Flooding gives better accuracy in many cases. Combining flooding with early stopping or with both early stopping and weight decay may lead to even better accuracy in some cases. As a byproduct, we were able to produce a double descent curve for the test loss with a relatively few number of epochs, e.g., in around 100 epochs, shown in Fig. 2.

5. *Why* Does Flooding Generalize Better?

In this section, we investigate four key properties of flooding to seek potential reasons for better generalization.

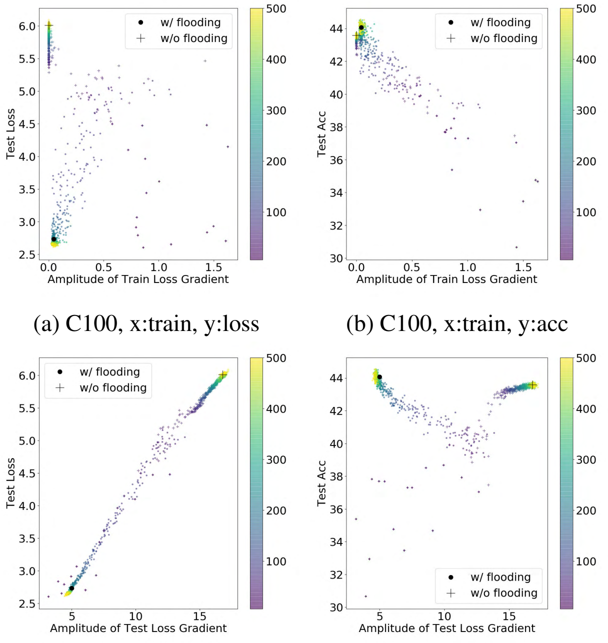
5.1. Performance and gradients

We visualize the relationship between test performance (loss or accuracy) and gradient amplitude of the training/test loss in Fig. 3, where the gradient amplitude is the ℓ_2 norm of the *filter-normalized gradient* of the loss. The filter-normalized gradient is the gradient appropriately scaled depending on the magnitude of the corresponding convolutional filter, similarly to (Li et al., 2018). More specifically, for each filter of every convolutional layer, we multiply the corresponding elements of the gradient by the norm of the filter. Note that a fully connected layer is a special case of convolutional layer and is also subject to this scaling.

For the sub-figures with gradient amplitude of training loss on the horizontal axis, "$\circ$" marks (with flooding) are often plotted on the right of "$+$" marks (without flooding), which implies that flooding prevents the model from staying at a local minimum. For the sub-figures with gradient amplitude of test loss on the horizontal axis, the method with flooding ("$\circ$") improves performance while the gradient amplitude becomes smaller. On the other hand, the performance with the method without flooding ("$+$") degenerates while the gradient amplitude of test loss keeps increasing.

5.2. Memorization

Can we maintain memorization (Zhang et al., 2017; Arpit et al., 2017; Belkin et al., 2018) even after adding flooding? We investigate if the trained model has zero training error (100% accuracy) for the flood level that was chosen with validation data. The results are shown in Fig. 4.



(a) C100, x:train, y:loss (b) C100, x:train, y:acc

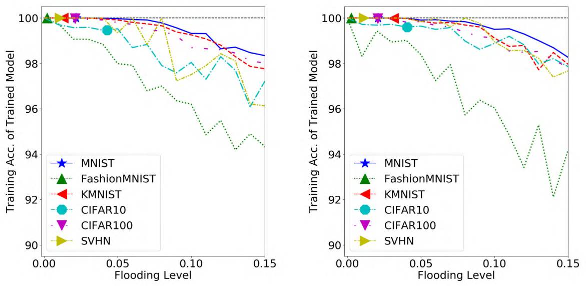
(c) C100, x:test, y:loss (d) C100, x:test, y:acc

Figure 3. Relationship between test performance (loss or accuracy) and amplitude of gradient (with training or test loss). Each point (“$\circ$” or “$+$”) in the figures corresponds to a single model at a certain epoch. We remove the first 5 epochs and plot the rest. “$\circ$” is used for the method with flooding and “$+$” is used for the method without flooding. The large black “$\circ$” and “$+$” show the epochs with early stopping. The color becomes lighter (purple → yellow) as the training proceeds. See Fig. 7 in Appendix for other datasets.

All datasets show downward curves, implying that the model will give up eventually on memorizing all training data as the flood level becomes higher. A more interesting observation is the position of the optimal flood level (the one chosen by validation accuracy which is marked with $\star$, $\triangle$, $\triangleleft$, $\circ$, $\triangledown$ or $\triangleright$). We can observe that the marks are often plotted at zero error. These results are consistent with recent empirical works that imply zero training error leads to lower generalization error (Belkin et al., 2019; Nakkiran et al., 2020), but we further demonstrate that zero training loss may be harmful under zero training error.

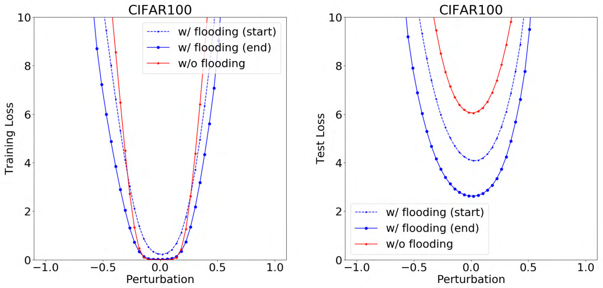
5.3. Flatness

We give a visualization of flatness (Li et al., 2018). We compare the flatness of the model right after the empirical risk with respect to a mini-batch becomes smaller than the flood level, $\widehat{R}_m(\boldsymbol{g}) < b$, for the first time (dotted blue) and the model after training (solid blue). We also compare them with the model trained by the baseline method without flooding, and training is finished (solid red).



(a) w/o early stopping (b) w/ early stopping

Figure 4. Vertical axis is the training accuracy and the horizontal axis is the flood level. Marks are placed on the flood level that was chosen based on validation accuracy.



(a) CIFAR-100 (train) (b) CIFAR-100 (test)

Figure 5. Visualization of flatness of training/test loss with respect to perturbation. Depicting the results for 3 models which is explained in Sec. 5.3. See Fig. 8 in Appendix for other datasets.

In Fig. 5, the test loss becomes lower and flatter during the training with flooding. Note that the training loss, on the other hand, continues to float around the flood level until the end of training after it enters the flooding zone. We expect that the model makes a random walk and escapes regions with sharp loss landscapes during the period. This may be a possible reason for better generalization results with our proposed method.

5.4. Theoretical Insight

We show that the mean squared error (MSE) of the flooded risk estimator is smaller than that of the original risk estimator without flooding, with the condition that the flood level is between the original training loss and the test loss. See Appendix A for formal discussions and the proof.

6. Conclusion

We proposed a novel regularization method called *flooding* that keeps the training loss to stay around a small constant value, to avoid zero training loss. In our experiments, the optimal flood level often maintained memorization of training data, with zero error. With flooding, our experiments confirmed that the test accuracy improves for various synthetic and benchmark datasets, and we theoretically showed that the MSE will be reduced under certain conditions.

As a byproduct, we were able to produce a double descent curve for the test loss when flooding was used. An important future direction is to study the relationship between this and the double descent curves from previous works (Krogh & Hertz, 1992; Belkin et al., 2019; Nakkiran et al., 2020).

Acknowledgements

We thank Chang Xu, Genki Yamamoto, Kento Nozawa, Nontawat Charoenphakdee, Voot Tangkaratt, and Yoshihiro Nagano for the helpful discussions. We also thank anonymous reviewers for helpful comments. TI was supported by the Google PhD Fellowship Program and JSPS KAKENHI 20J11937. MS and IY were supported by JST CREST Grant Number JPMJCR18A2 including AIP challenge program, Japan.

References

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *ICML*, 2017.

Belkin, M., Hsu, D. J., and Mitra, P. Overfitting or perfect fitting? Risk bounds for classification and regression rules that interpolate. In *NeurIPS*, 2018.

Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *PNAS*, 116:15850–15854, 2019.

Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., and Raffel, C. A. MixMatch: A holistic approach to semi-supervised learning. In *NeurIPS*, 2019.

Bishop, C. M. Regularization and complexity control in feed-forward networks. In *ICANN*, 1995.

Bishop, C. M. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2011.

Caruana, R., Lawrence, S., and Giles, C. L. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *NeurIPS*, 2000.

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-SGD: Biasing gradient descent into wide valleys. In *ICLR*, 2017.

Cid-Sueiro, J., García-García, D., and Santos-Rodríguez, R. Consistency of losses for learning from weak labels. In *ECML-PKDD*, 2014.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. Deep learning for classical Japanese literature. In *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2018.

du Plessis, M. C., Niu, G., and Sugiyama, M. Analysis of learning from positive and unlabeled data. In *NeurIPS*, 2014.

du Plessis, M. C., Niu, G., and Sugiyama, M. Convex formulation for learning from positive and unlabeled data. In *ICML*, 2015.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *ICML*, 2017.

Guo, H., Mao, Y., and Zhang, R. Augmenting data with mixup for sentence classification: An empirical study. In *arXiv:1905.08941*, 2019.

Han, B., Niu, G., Yu, X., Yao, Q., Xu, M., Tsang, I. W., and Sugiyama, M. SIGUA: Forgetting may make learning with noisy labels more robust. In *ICML*, 2020.

Hanson, S. J. and Pratt, L. Y. Comparing biases for minimal network construction with back-propagation. In *NeurIPS*, 1988.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural Computation*, 9:1–42, 1997.

Ioffe, S. and Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Ishida, T., Niu, G., Menon, A. K., and Sugiyama, M. Complementary-label learning for arbitrary losses and models. In *ICML*, 2019.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017.

Kingma, D. P. and Ba, J. L. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Kiryo, R., Niu, G., du Plessis, M. C., and Sugiyama, M. Positive-unlabeled learning with non-negative risk estimator. In *NeurIPS*, 2017.

Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., and Houlsby, N. Big transfer (BiT): General visual representation learning. In *arXiv:1912.11370v3*, 2020.

Krogh, A. and Hertz, J. A. A simple weight decay can improve generalization. In *NeurIPS*, 1992.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *ICLR*, 2019.

Lu, N., Zhang, T., Niu, G., and Sugiyama, M. Mitigating overfitting in supervised classification from two unlabeled datasets: A consistent risk correction approach. In *AISTATS*, 2020.

Morgan, N. and Bourlard, H. Generalization and parameter estimation in feedforward nets: Some experiments. In *NeurIPS*, 1990.

Nair, V. and Hinton, G. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Nakkiran, P., Kaplun, G., Kalimeris, D., Yang, T., Edelman, B. L., Zhang, F., and Barak, B. SGD on neural networks learns functions of increasing complexity. In *NeurIPS*, 2019.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. In *ICLR*, 2020.

Natarajan, N., Dhillon, I. S., Ravikumar, P. K., and Tewari, A. Learning with noisy labels. In *NeurIPS*, 2013.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NeurIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

Ng, A. Y. Preventing "overfitting" of cross-validation data. In *ICML*, 1997.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Patrini, G., Rozza, A., Menon, A. K., Nock, R., and Qu, L. Making deep neural networks robust to label noise: A loss correction approach. In *CVPR*, 2017.

Robbins, H. and Monro, S. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

Roelofs, R., Shankar, V., Recht, B., Fridovich-Keil, S., Hardt, M., Miller, J., and Schmidt, L. A meta-analysis of overfitting in machine learning. In *NeurIPS*, 2019.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6, 2019.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

Sugiyama, M. *Introduction to statistical machine learning*. Morgan Kaufmann, 2015.

Szegedy, C., Vanhoucke, V., Ioffe, S., and Shlens, J. Rethinking the inception architecture for computer vision. In *CVPR*, 2016.

Thulasidasan, S., Chennupati, G., Bilmes, J., Bhattacharya, T., and Michalak, S. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019.

Tibshirani, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58:267–288, 1996.

Tikhonov, A. N. On the stability of inverse problems. *Doklady Akademii Nauk SSSR*, 39:195–198, 1943.

Tikhonov, A. N. and Arsenin, V. Y. *Solutions of Ill Posed Problems*. Winston, 1977.

Torralba, A., Fergus, R., and Freeman, W. T. 80 million tiny images: A large data set for nonparametric object and scene recognition. In *IEEE Trans. PAMI*, 2008.

van Rooyen, B. and Williamson, R. C. A theory of learning with corrupted labels. *Journal of Machine Learning Research*, 18:1–50, 2018.

Verma, V., Lamb, A., Kannala, J., Bengio, Y., and Lopez-Paz, D. Interpolation consistency training for semi-supervised learning. In *IJCAI*, 2019.

Wager, S., Wang, S., and Liang, P. Dropout training as adaptive regularization. In *NeurIPS*, 2013.

Werpachowski, R., György, A., and Szepesvári, C. Detecting overfitting via adversarial examples. In *NeurIPS*, 2019.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.

Zhang, T. Statistical behavior and consistency of classification methods based on convex risk minimization. *The Annals of Statistics*, 32:56–85, 2004.

A. Analysis of Mean Squared Error

In the following theorem, we will show that the mean squared error (MSE) of the proposed risk estimator with flooding is smaller than that of the original risk estimator without flooding when the flood level is smaller than the population risk.

Theorem 1. *Fix any measurable vector-valued function $\boldsymbol{g}$. If the flood level b satisfies $b \leq R(\boldsymbol{g})$, we have*

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) \geq \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})).$$

If b further satisfies $b < R(\boldsymbol{g})$ and $\Pr[\widehat{R}(\boldsymbol{g}) < b] > 0$, the inequality will be strict:

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) > \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})).$$

Proof. If the flood level is b, then the proposed flooding estimator is

$$\widetilde{R}(\boldsymbol{g}) = |\widehat{R}(\boldsymbol{g}) - b| + b.$$

Since the absolute operator can be expressed with a max operator with $\max(a, b) = \frac{a+b+|a-b|}{2}$, the proposed estimator can be re-expressed as

$$\widetilde{R}(\boldsymbol{g}) = 2\max(\widehat{R}(\boldsymbol{g}), b) - \widehat{R}(\boldsymbol{g}) = A - \widehat{R}(\boldsymbol{g}).$$

For convenience, we used $A = 2\max(\widehat{R}(\boldsymbol{g}), b)$. From the definition of MSE,

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) = \mathbb{E}[(\widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2],$$

and

$$\begin{aligned}
\mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) &= \mathbb{E}[(\widetilde{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2] \\
&= \mathbb{E}[(A - \widehat{R}(\boldsymbol{g}) - R(\boldsymbol{g}))^2)] \\
&= \mathbb{E}[A^2] - 2\mathbb{E}[A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))] + \mathbb{E}[(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))^2].
\end{aligned}$$

We are interested in the sign of

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) = \mathbb{E}[-4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - A^2 + 2A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))].$$

Define the inside of the expectation as $B = -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - A^2 + 2A(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g}))$. B can be divided into two cases, depending on the outcome of the max operator:

$$\begin{aligned}
B &= \begin{cases} -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - 4\widehat{R}(\boldsymbol{g})^2 + 4\widehat{R}(\boldsymbol{g})(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) \geq b \\ -4\widehat{R}(\boldsymbol{g})R(\boldsymbol{g}) - 4b^2 + 4b(\widehat{R}(\boldsymbol{g}) + R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) < b \end{cases} \\
&= \begin{cases} 0 & \text{if} \quad \widehat{R}(\boldsymbol{g}) \geq b \\ -4(b - \widehat{R}(\boldsymbol{g}))(b - R(\boldsymbol{g})) & \text{if} \quad \widehat{R}(\boldsymbol{g}) < b. \end{cases}
\end{aligned}$$

In the latter case, B becomes positive when $\widehat{R}(\boldsymbol{g}) < b < R(\boldsymbol{g})$. Note that we do not need to consider $R(\boldsymbol{g}) < b < \widehat{R}(\boldsymbol{g})$ since $\widehat{R}(\boldsymbol{g}) < b$ must be satisfied in the latter case.

Therefore, if $b \leq R(\boldsymbol{g})$, we have

$$\begin{aligned}
\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) &= \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) \geq b] \Pr[\widehat{R}(\boldsymbol{g}) \geq b] + \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] \\
&= \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] \\
&\geq 0.
\end{aligned}$$

Furthermore, if $b < R(\boldsymbol{g})$ and $\Pr[\widehat{R}(\boldsymbol{g}) < b] > 0$, we have

$$\mathrm{MSE}(\widehat{R}(\boldsymbol{g})) - \mathrm{MSE}(\widetilde{R}(\boldsymbol{g})) = \mathbb{E}[B \mid \widehat{R}(\boldsymbol{g}) < b] \Pr[\widehat{R}(\boldsymbol{g}) < b] > 0.$$

□

B. Bayes Risk for Gaussian Distributions

In this section, we explain in detail how we derived the Bayes risk with respect to the surrogate loss in the experiments with Gaussian data in Section 4.1. Since we are using the logistic loss in the synthetic experiments, the loss of the margin is

$$\ell(yg(\boldsymbol{x})) = \log(1 + \exp(-yg(\boldsymbol{x}))),$$

where $g(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, because the synthetic experiments only consider binary classification. Take the derivative to derive,

$$\begin{aligned}
\frac{\partial \mathbb{E}[\ell(yg(\boldsymbol{x}))]}{\partial g(\boldsymbol{x})} = \frac{\mathbb{E}[\log(1 + \exp(-yg(\boldsymbol{x})))]}{\partial g(\boldsymbol{x})} &= \mathbb{E}\left[\left.\frac{-y \exp(-yg(\boldsymbol{x}))}{1 + \exp(-yg(\boldsymbol{x}))}\right| \boldsymbol{x}\right] p(\boldsymbol{x}) \\
&= \mathbb{E}\left[\left.\frac{-y}{1 + \exp(yg(\boldsymbol{x}))}\right| \boldsymbol{x}\right] p(\boldsymbol{x}) \\
&= \mathbb{E}\left[\left.\frac{y+1}{2}\frac{1}{\exp(-g(\boldsymbol{x})) + 1} + \frac{y-1}{2}\frac{-1}{\exp(g(\boldsymbol{x})) + 1}\right| \boldsymbol{x}\right] p(\boldsymbol{x}) \\
&= \mathbb{E}\left[\left.\frac{y+1}{2}\right| \boldsymbol{x}\right] \frac{1}{\exp(-g(\boldsymbol{x})) + 1} p(\boldsymbol{x}) + \mathbb{E}\left[\left.\frac{y-1}{2}\right| \boldsymbol{x}\right] \frac{-1}{\exp(g(\boldsymbol{x})) + 1} p(\boldsymbol{x}) \\
&= p(y = +1|\boldsymbol{x}) \frac{1}{\exp(-g(\boldsymbol{x})) + 1} p(\boldsymbol{x}) + p(y = -1|\boldsymbol{x}) \frac{-1}{\exp(g(\boldsymbol{x})) + 1} p(\boldsymbol{x}).
\end{aligned}$$

Set this to zero, divide by $p(\boldsymbol{x}) > 0$ to obtain,

$$\begin{aligned}
p(y = -1|\boldsymbol{x}) \frac{1}{\exp(-g(\boldsymbol{x})) + 1} &= p(y = +1|\boldsymbol{x}) \frac{1}{\exp(g(\boldsymbol{x})) + 1} \\
\exp(g(\boldsymbol{x})) &= \frac{p(y = +1|\boldsymbol{x})}{p(y = -1|\boldsymbol{x})} \\
g(\boldsymbol{x}) &= \log \frac{p(y = +1|\boldsymbol{x})}{p(y = -1|\boldsymbol{x})}.
\end{aligned}$$

Since we are interested in the surrogate loss under this classifier, we plug this into the logistic loss, to obtain the Bayes risk,

$$\mathbb{E}[\ell(yg(\boldsymbol{x}))] = \mathbb{E}\left[\log(1 + \frac{p(-y|\boldsymbol{x})}{p(y|\boldsymbol{x})})\right] = \mathbb{E}\left[\log(\frac{1}{p(y|\boldsymbol{x})})\right] = \mathbb{E}[-\log p(y|\boldsymbol{x})].$$

In the experiments in Section 4.1, we report the empirical version of this with the test dataset as the Bayes risk.

C. Details of Datasets

C.1. Synthetic Datasets

C.1.1. Two Gaussians Data

We used two 10-dimensional Gaussian distributions (one for each class) with covariance matrix identity and means $\mu_{\mathrm{P}} = [0, 0, \ldots, 0]^\top$ and $\mu_{\mathrm{N}} = [m, m, \ldots, m]^\top$, where $m \in \{0.8, 1.0\}$. The Bayes risk for $m = 1.0$ and $m = 0.8$ are 0.14 and 0.24, respectively, where proofs are shown in Appendix B. The training, validation, and test sample sizes are 25, 10000, and 10000 per class respectively.

C.1.2. Sinusoid Data

The sinusoid data (Nakkiran et al., 2019) are generated as follows. We first draw input data points from the 2-dimensional standard Gaussian distribution, i.e., $\boldsymbol{x} \sim N(\boldsymbol{0}_2, \boldsymbol{I}_2)$, where $\boldsymbol{0}_2$ is the two-dimensional zero vector, and $\boldsymbol{I}_2$ 2×2 identity matrix. Then put class labels based on

$$y = \mathrm{sign}(\boldsymbol{x}^\top \boldsymbol{w} + \sin(\boldsymbol{x}^\top \boldsymbol{w}')),$$

where $\boldsymbol{w}$ and $\boldsymbol{w}'$ are any two 2-dimesional vectors such that $\boldsymbol{w} \perp \boldsymbol{w}'$. The training, validation, and test sample sizes are 100, 100, and 20000, respectively.

C.1.3. Spiral Data

The spiral data (Sugiyama, 2015) is two-dimensional synthetic data. Let $\theta_1^+ := 0, \theta_2^+, \ldots, \theta_{n^+}^+ := 4\pi$ be equally spaced n^+ points in the interval $[0, 4\pi]$, and $\theta_1^- := 0, \theta_2^-, \ldots, \theta_{n^-}^- := 4\pi$ be equally spaced n^- points in the interval $[0, 4\pi]$. Let positive and negative input data points be

$$\begin{aligned} \boldsymbol{x}_i^+ &:= \theta_i[\cos(\theta_i), \sin(\theta_i)]^\top + \tau\boldsymbol{\nu}_i^+, \\ \boldsymbol{x}_i^- &:= (\theta + \pi)[\cos(\theta), \sin(\theta)]^\top + \tau\boldsymbol{\nu}_i^- \end{aligned}$$

for $i = 1, \ldots, n$, where τ controls the magnitude of the noise, $\boldsymbol{\nu}_i^+$ and $\boldsymbol{\nu}_i^-$ are i.i.d. distributed according to the two-dimensional standard normal distribution. Then, we make data for classification by $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n := \{(\boldsymbol{x}_i^+, +1)\}_{i^+=1}^{n^+} \cup \{(\boldsymbol{x}_i^-, -1)\}_{i^-=1}^{n^-}$, where $n := n^+ + n^-$. The training, validation, and test sample sizes are 100, 100, and 10000 per class respectively.

C.2. Benchmark Datasets

In the experiments in Section 4.2, we use 6 image benchmark datasets explained below.

- MNIST[5] (Lecun et al., 1998) is a 10 class dataset of handwritten digits: $1, 2 \ldots, 9$ and 0. Each sample is a 28 × 28 grayscale image. The number of training and test samples are 60,000 and 10,000, respectively.
- Fashion-MNIST[6] (Xiao et al., 2017) is a 10 class dataset of fashion items: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot. Each sample is a 28 × 28 grayscale image. The number of training and test samples are 60,000 and 10,000, respectively.
- Kuzushiji-MNIST[7] (Clanuwat et al., 2018) is a 10 class dataset of cursive Japanese ("Kuzushiji") characters. Each sample is a 28 × 28 grayscale image. The number of training and test samples are 60,000 and 10,000, respectively.
- CIFAR-10[8] is a 10 class dataset of various objects: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each sample is a colored image in 32 × 32 × 3 RGB format. It is a subset of the 80 million tiny images dataset (Torralba et al., 2008). There are 6,000 images per class, where 5,000 are for training and 1,000 are for test.
- CIFAR-100[9] is a 100 class dataset of various objects. Each class has 600 samples, where 500 samples are for training and 100 samples are for test. This is also a subset of the 80 million tiny images dataset (Torralba et al., 2008).
- SVHN[10] (Netzer et al., 2011) is a 10 class dataset of house numbers from Google Street View images, in 32 × 32 × 3 RGB format. 73257 digits are for training and 26032 digits are for testing.

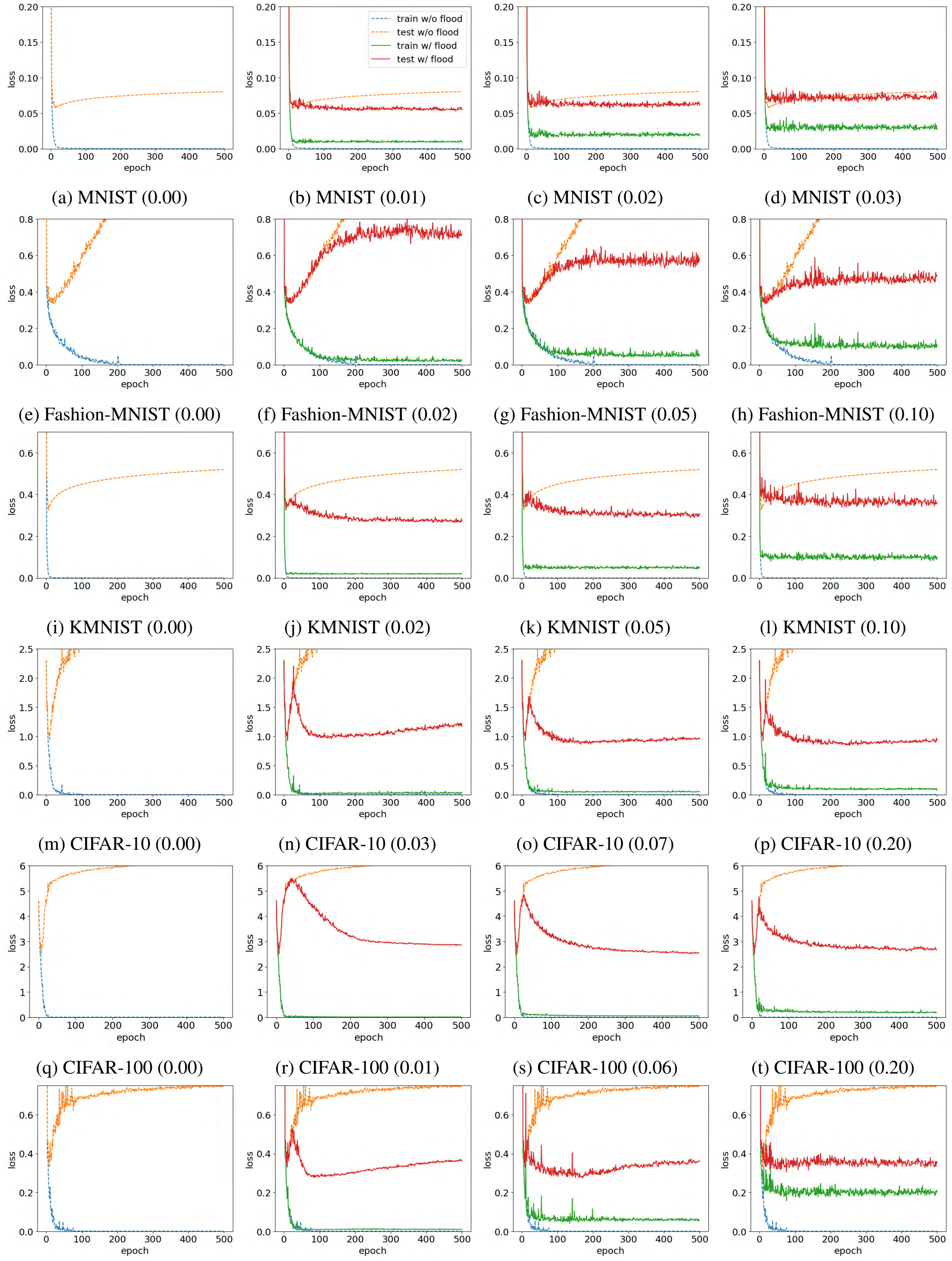
D. Learning Curves

In Fig. 6, we visualize learning curves for all datasets (including CIFAR-10 which we already visualized in Fig. 2). Note that Figure 1(c) shows the learning curves for the first 80 epochs for CIFAR-10 without flooding. Figure 1(d) shows the learning curves with flooding, when the flood level is 0.18.

E. Additional Figures for Section 5.1 and Section 5.3

See Fig. 7 and Fig. 8 for datasets that were not included in the main paper.

[5] http://yann.lecun.com/exdb/mnist/
[6] https://github.com/zalandoresearch/fashion-mnist
[7] https://github.com/rois-codh/kmnist
[8] https://www.cs.toronto.edu/~kriz/cifar.html
[9] https://www.cs.toronto.edu/~kriz/cifar.html
[10] http://ufldl.stanford.edu/housenumbers/



(a) MNIST (0.00)

(b) MNIST (0.01)

(c) MNIST (0.02)

(d) MNIST (0.03)

(e) Fashion-MNIST (0.00)

(f) Fashion-MNIST (0.02)

(g) Fashion-MNIST (0.05)

(h) Fashion-MNIST (0.10)

(i) KMNIST (0.00)

(j) KMNIST (0.02)

(k) KMNIST (0.05)

(l) KMNIST (0.10)

(m) CIFAR-10 (0.00)

(n) CIFAR-10 (0.03)

(o) CIFAR-10 (0.07)

(p) CIFAR-10 (0.20)

(q) CIFAR-100 (0.00)

(r) CIFAR-100 (0.01)

(s) CIFAR-100 (0.06)

(t) CIFAR-100 (0.20)

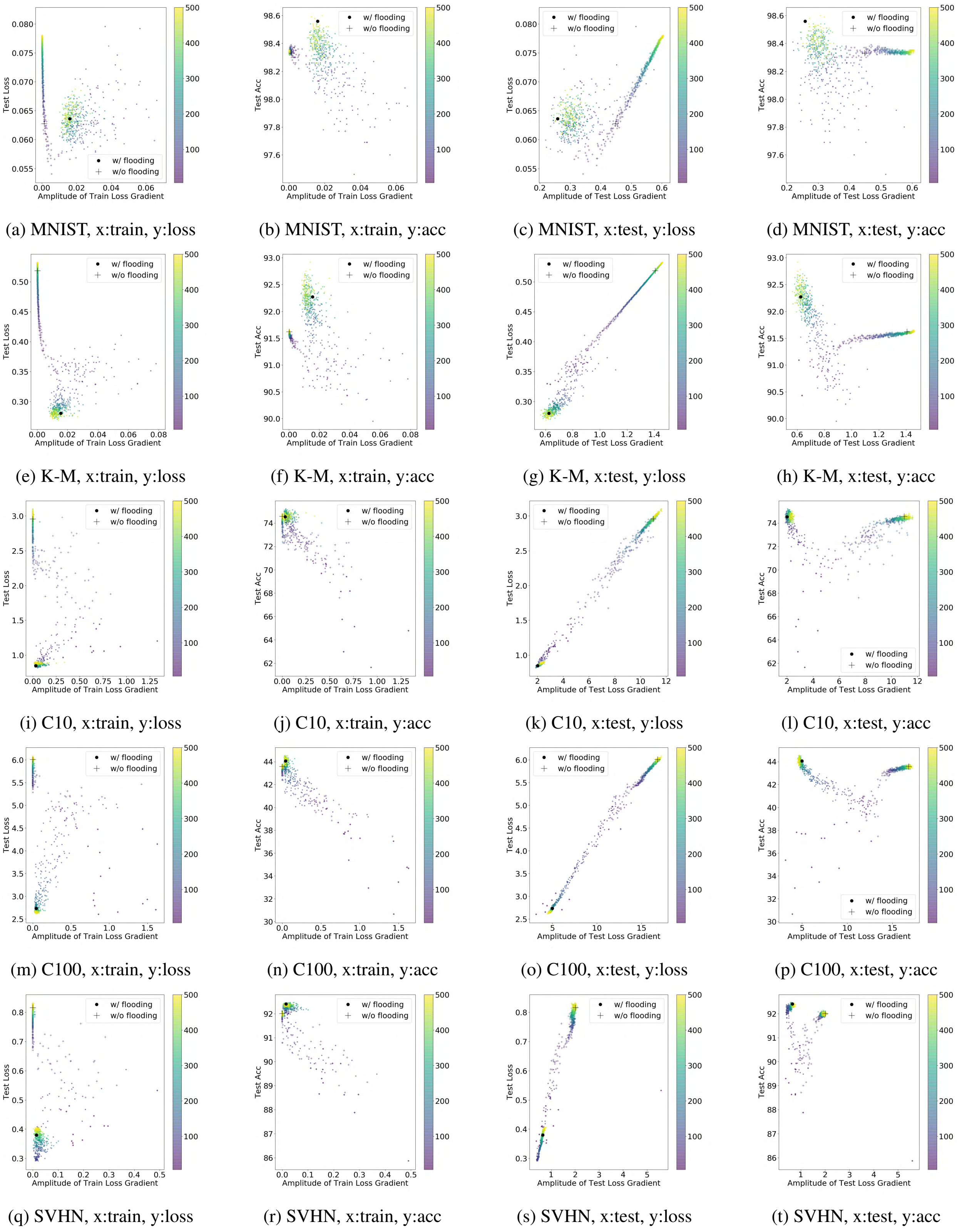
(u) SVHN (0.00)

(v) SVHN (0.01)

(w) SVHN (0.06)

(x) SVHN (0.20)

Figure 6. Learning curves of training and test loss. The first figure in each row is the learning curves without flooding. The 2nd, 3rd, and 4th figures show the results with different flood levels. The flood level increases towards the right-hand side. The two training losses are evaluated with the original empirical risk Eq. (3) and not the flooded empirical risk Eq. (4).



(a) MNIST, x:train, y:loss

(b) MNIST, x:train, y:acc

(c) MNIST, x:test, y:loss

(d) MNIST, x:test, y:acc

(e) K-M, x:train, y:loss

(f) K-M, x:train, y:acc

(g) K-M, x:test, y:loss

(h) K-M, x:test, y:acc

(i) C10, x:train, y:loss

(j) C10, x:train, y:acc

(k) C10, x:test, y:loss

(l) C10, x:test, y:acc

(m) C100, x:train, y:loss

(n) C100, x:train, y:acc

(o) C100, x:test, y:loss

(p) C100, x:test, y:acc

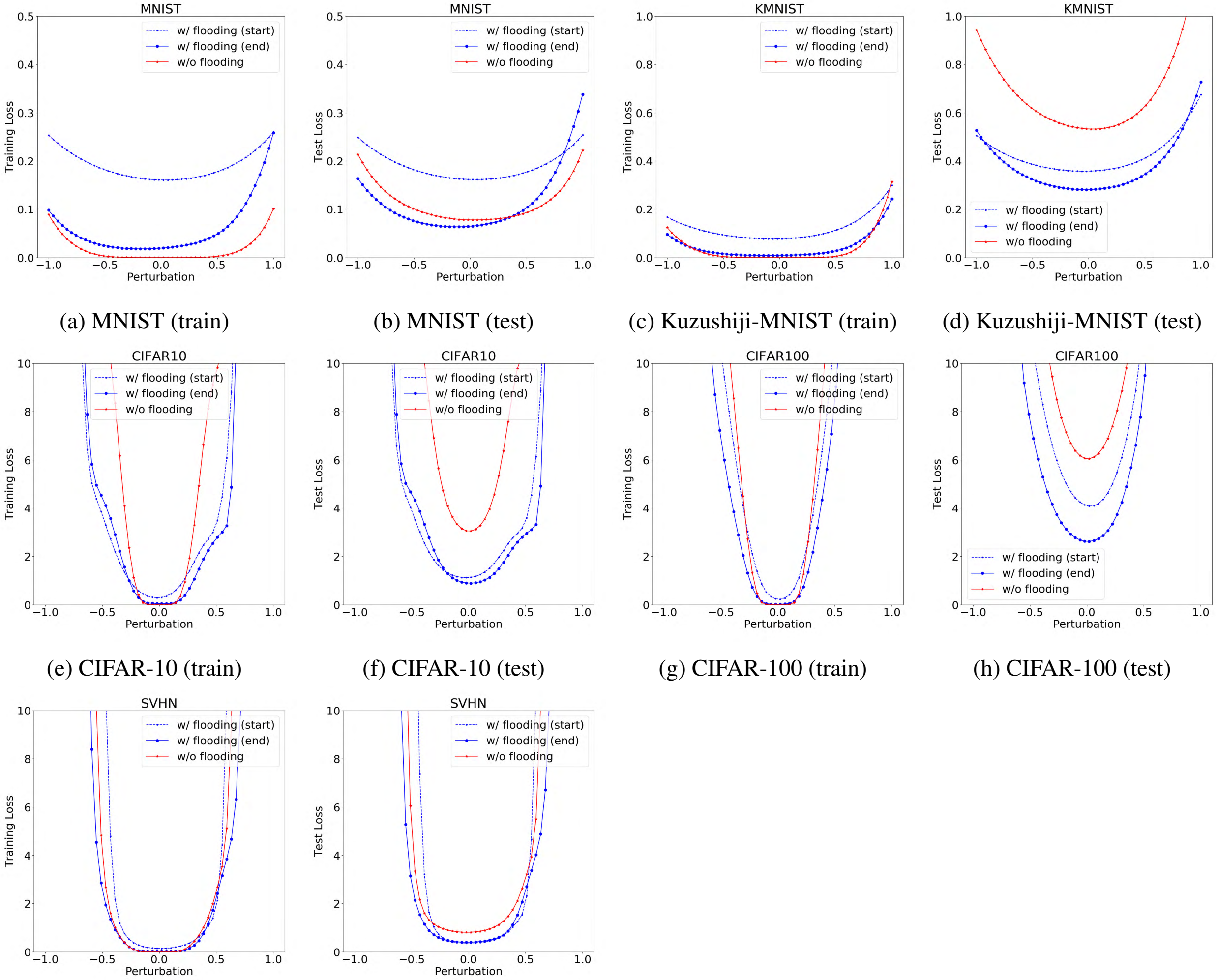
(q) SVHN, x:train, y:loss

(r) SVHN, x:train, y:acc

(s) SVHN, x:test, y:loss

(t) SVHN, x:test, y:acc

Figure 7. Relationship between test performance (loss or accuracy) and amplitude of gradient (with training or test loss). Each point ("○" or "+") in the figures corresponds to a single model at a certain epoch. We remove the first 5 epochs and plot the rest. "○" is used for the method with flooding and "+" is used for the method without flooding. The large black "○" and "+" show the epochs with early stopping. The color becomes lighter (purple → yellow) as the training proceeds. K-M, C10, and C100 stand for Kuzushiji-MNIST, CIFAR-10, and CIFAR-100.



(a) MNIST (train) (b) MNIST (test) (c) Kuzushiji-MNIST (train) (d) Kuzushiji-MNIST (test)

(e) CIFAR-10 (train) (f) CIFAR-10 (test) (g) CIFAR-100 (train) (h) CIFAR-100 (test)

(i) SVHN (train) (j) SVHN (test)

Figure 8. Visualization of flatness. We visualize the training/test loss with respect to perturbation. We depict the results for 3 models: the model when the empirical risk with respect to training data is below the flooding level for the first time during training (dotted blue), the model at the end of training with flooding (solid blue), and the model at the end of training without flooding (solid red).