

EnsLoss: Stochastic Calibrated Loss Ensembles for Preventing Overfitting in Classification

Ben Dai (CUHK)

ICML 2025

Background

The objective of binary classification is to categorize each instance into one of two classes.

- **Data:** $\mathbf{X} \in \mathbb{R}^d \rightarrow Y \in \{-1, +1\}$
- **Classifier:** $f(\mathbf{X}) : \mathbb{R}^d \rightarrow \mathbb{R}$
- **Predicted label:** $\hat{Y} = \text{sgn}(f(\mathbf{X}))$
- **Evaluation via Misclassification error (risk):**

$$R(f) = 1 - \text{Acc}(f) = \mathbb{E}(\mathbf{1}(Y f(\mathbf{X}) \leq 0)),$$

where $\mathbf{1}(\cdot)$ is an indicator function.

Aim. To obtain *the Bayes classifier* or the best classifier:

$$f^* := \arg \min R(f)$$

Background

Due to the **discontinuity** of the indicator function:

$$R(f) = 1 - \text{Acc}(f) = \mathbb{E}(\mathbf{1}(Y f(\mathbf{X}) \leq 0)),$$

the zero-one loss is usually replaced by a **convex** and **classification-calibrated** loss ϕ to facilitate the empirical computation (Lin, 2004; Zhang, 2004; Bartlett et al., 2006):

$$R_{\phi}(f) = \mathbb{E}(\phi(Y f(\mathbf{X})))$$

For example, the hinge loss for SVM, exponential loss for AdaBoost, and logistic loss for logistic regression all follow this framework.

Background

Due to the **discontinuity** of the indicator function:

$$R(f) = 1 - \text{Acc}(f) = \mathbb{E}(\mathbf{1}(Y f(\mathbf{X}) \leq 0)),$$

the zero-one loss is usually replaced by a **convex** and **classification-calibrated** loss ϕ to facilitate the empirical computation (Lin, 2004; Zhang, 2004; Bartlett et al., 2006):

$$R_\phi(f) = \mathbb{E}(\phi(Y f(\mathbf{X})))$$

For example, the hinge loss for SVM, exponential loss for AdaBoost, and logistic loss for logistic regression all follow this framework.

(If we optimize with respect to ϕ , will the resulting solution still be the function f^* that we need?)

That's why we need the loss ϕ to be **calibrated**?

Background

Definition 1 (Bartlett et al. (2006)). A loss function $\phi(\cdot)$ is **classification-calibrated**, if for every sequence of measurable function f_n and every probability distribution on $\mathcal{X} \times \{\pm 1\}$,

$$R_\phi(f_n) \rightarrow \inf_f R_\phi(f) \text{ implies that } R(f_n) \rightarrow \inf_f R(f).$$

A **calibrated** loss function ϕ guarantees that any sequence f_n that optimizes R_ϕ will eventually also optimize R , thereby ensuring **consistency** in maximizing classification accuracy.

Background

Definition 1 (Bartlett et al. (2006)). A loss function $\phi(\cdot)$ is **classification-calibrated**, if for every sequence of measurable function f_n and every probability distribution on $\mathcal{X} \times \{\pm 1\}$,

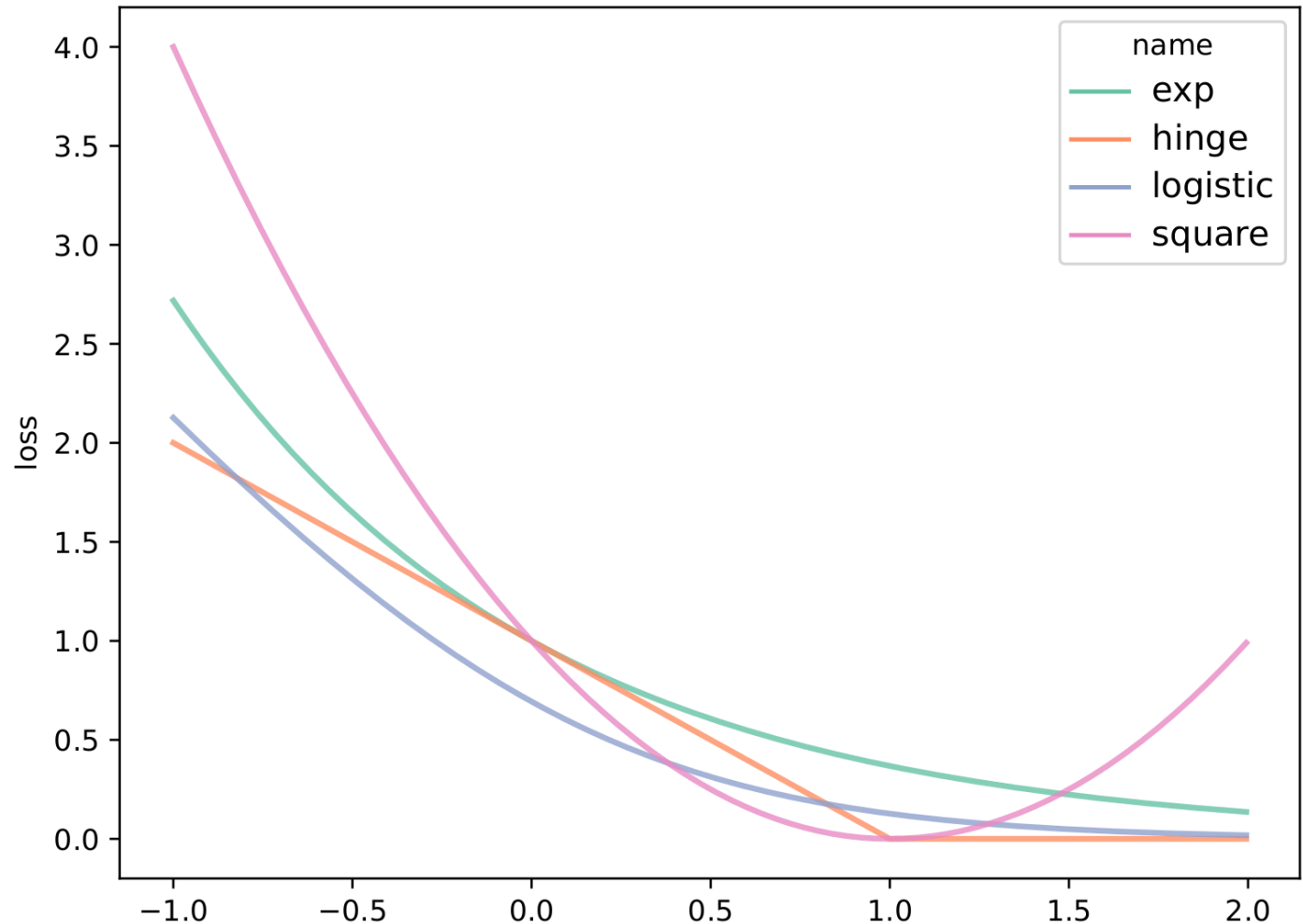
$$R_\phi(f_n) \rightarrow \inf_f R_\phi(f) \text{ implies that } R(f_n) \rightarrow \inf_f R(f).$$

A series of studies (Lin, 2004; Zhang, 2004; Bartlett et al., 2006) culminates in the following theorem for **iff conditions** of calibration:

Theorem 1 (Bartlett et al. (2006)) Let ϕ be convex. Then ϕ is classification-calibrated iff it is differentiable at 0 and $\phi'(0) < 0$.

Classification ERM framework

(i) Select a **convex** and **calibrated** (CC) loss function ϕ



Classification ERM framework

- (i) Select a **convex** and **calibrated** (CC) loss function ϕ
- (ii) Directly minimizes the **ERM** of R_ϕ to obtain f_n

$$\hat{f}_n = \arg \min_{f \in \mathcal{F}} \hat{R}_\phi(f), \quad \hat{R}_\phi(f) := \frac{1}{n} \sum_{i=1}^n \phi(y_i f(\mathbf{x}_i)).$$

(SGD is widely adopted for its scalability and generalization when dealing with large-scale datasets and DL models)

$$\begin{aligned} \theta^{(t+1)} &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \nabla_{\theta} \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i)) \\ &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \partial \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i)) \nabla_{\theta} f_{\theta^{(t)}}(\mathbf{x}_i), \end{aligned}$$

The **ERM** paradigm with **calibrated losses**, when combined with **ML/DL models** and optimized using **SGD**, has achieved tremendous success in numerous real-world applications.

EnsLoss: Calibrated Loss Ensembles

SGD + Fixed Loss

For each iteration:

- batch sampling from a training set;
- implement SGD on batch samples and *a fixed surrogate loss*.

SGD + Ensemble Loss (ENSLOSS; our)

For each iteration:

- batch sampling from a training set;
- ✚ randomly generate a new “valid” surrogate loss;
- ➔ implement SGD on batch samples and *the generated surrogate loss*.

Inspired by **Dropout**
(model ensemble over
one training process)

$$\theta^{(t+1)} = \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \nabla_{\theta} \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i))$$

EnsLoss: Calibrated Loss Ensembles

SGD + Fixed Loss

For each iteration:

- batch sampling from a training set;
- implement SGD on batch samples and *a fixed surrogate loss*.

SGD + Ensemble Loss (ENSLOSS; our)

For each iteration:

- batch sampling from a training set;
- ✚ randomly generate a new “valid” surrogate loss;
- ➔ implement SGD on batch samples and *the generated surrogate loss*.

Inspired by **Dropout**
(model ensemble over
one training process)

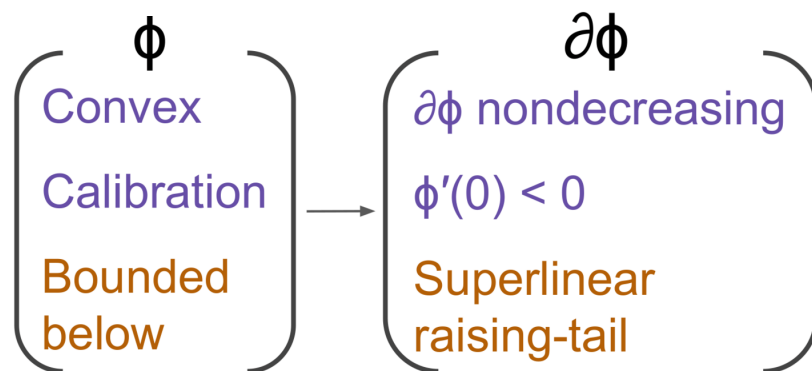
$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \nabla_{\theta} \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i)) \\ &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \partial \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i)) \nabla_{\theta} f_{\theta^{(t)}}(\mathbf{x}_i),\end{aligned}$$

EnsLoss: Calibrated Loss Ensembles

SGD + Ensemble Loss (ENSLOSS; our)

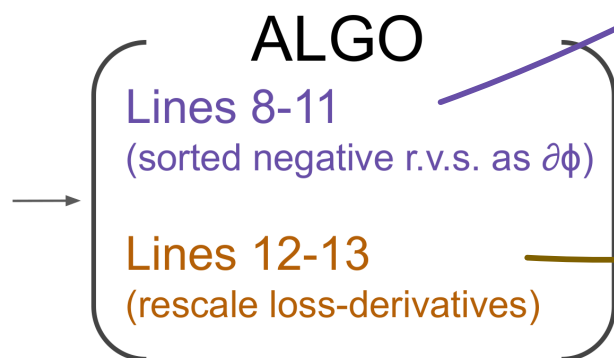
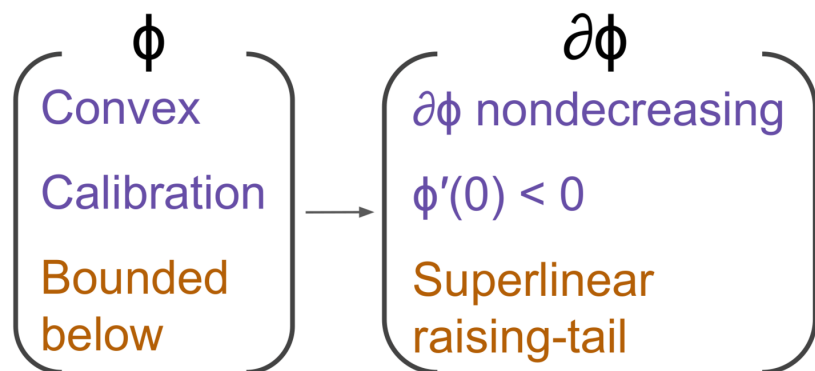
For each iteration:

- batch sampling from a training set;
- ✚ randomly generate a new “valid” surrogate loss;
- ➔ implement SGD on batch samples and *the generated surrogate loss*.



$$\begin{aligned}\theta^{(t+1)} &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \nabla_{\theta} \phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i)) \\ &= \theta^{(t)} - \gamma \frac{1}{B} \sum_{i \in \mathcal{I}_B} \boxed{\partial\phi(y_i f_{\theta^{(t)}}(\mathbf{x}_i))} \nabla_{\theta} f_{\theta^{(t)}}(\mathbf{x}_i),\end{aligned}$$

EnsLoss



Algorithm 1 (Minibatch) Calibrated ensemble SGD.

- 1: **Input:** a train set $\mathcal{D} = (\mathbf{x}_i, y_i)_{i=1}^n$, a minibatch size B ;
- 2: Initialize θ .
- 3: **for** number of epochs **do**
- 4: /* **Minibatch sampling** */
- 5: Sample a minibatch from \mathcal{D} *without* replacement:
 $\mathcal{B} = \{(\mathbf{x}_{i_1}, y_{i_1}), \dots, (\mathbf{x}_{i_B}, y_{i_B})\}$.
- 6: Compute $\mathbf{z} = (z_1, \dots, z_B)^\top$, where $z_b = y_{i_b} f_\theta(\mathbf{x}_{i_b})$
 for $b = 1, \dots, B$.
- 7: /* **Generate random RC loss-derivs** */
- 8: /* **calibration and convexity** */
- 9: Generate $\mathbf{g} = (g_1, \dots, g_B)^\top$, where $g_b \stackrel{iid}{\sim} -\xi$,
 where ξ is a *positive random variable* (accomplished
 through Algorithm 2)
- 10: Sort \mathbf{z} and \mathbf{g} decreasingly, that is

$$z_{\pi(1)} > \dots > z_{\pi(B)}, \quad g_{\sigma(1)} > \dots > g_{\sigma(B)};$$

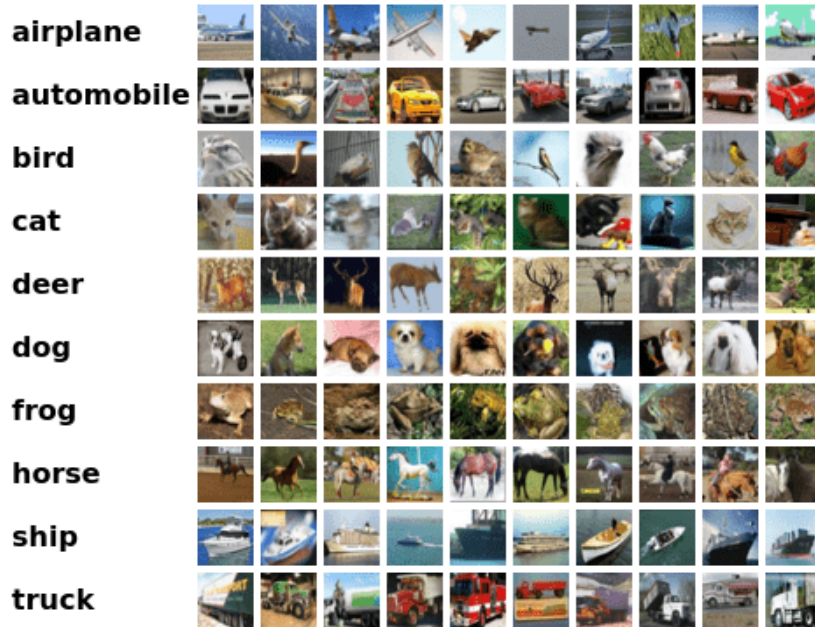
- 11: (the derivative corresponding to z_b is $g_{\sigma(\pi^{-1}(b))}$).
- 12: /* **bounded below** */
- 13: For $b = 1, \dots, B$,
- 14: /* **Update parameters** */
- 15: Compute gradients and update

$$\theta \leftarrow \theta - \frac{\gamma}{B} \sum_{b=1}^B y_{i_b} g_{\sigma(\pi^{-1}(b))} \nabla_{\theta} f_{\theta}(\mathbf{x}_{i_b})$$

- 16: **end for**
- 17: **Return** the estimated θ

Experiments

CIFAR

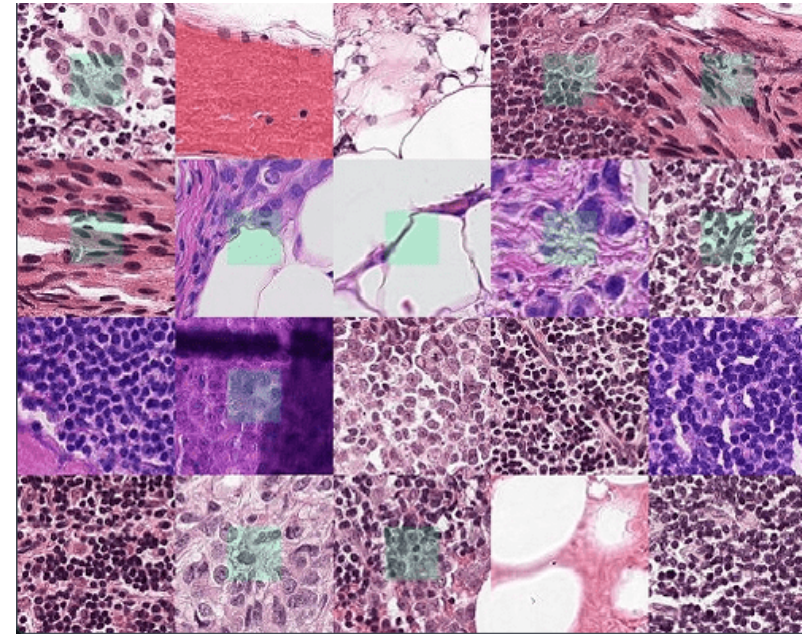


We construct binary CIFAR (CIFAR2), by selecting all possible pairs from CIFAR10, resulting:

$$10 \times 9 / 2 = 45$$

CIFAR2 datasets.

PCam



PCam is a binary image classification dataset comprising 327,680 96x96 images from histopathologic scans of lymph node sections

Experiments

OpenML

20 datasets found verified >1000 features >1000 instances binary class dense

Filter by Status Instances Features Target Format Tag

All Regression Binary classification Multi-class

Datasets

Datasets provide training data for machine learning models. OpenML datasets are uniformly formatted and come with rich meta-data to allow automated processing. You can sort or filter them by a range of different properties. [Learn more.](#)

Bioresponse Predict a biological response of molecules from their chemical properties. Each row in this data set represents a molecule. The first column contains experimental data describing an actual biological response; the molecule was seen to elicit this response (1), or not (0). The remaining columns represent molecular descriptors (d1 through d1776), these are calculated properties that can capture some of the characteristics of 48.7k 3.79k x 1.78k 4134 10 years ago v.1
OVA_Breast GEMLeR provides a collection of gene expression datasets that can be used for benchmarking gene expression oriented machine learning algorithms. They can be used for estimation of different quality metrics (e.g. accuracy, precision, area under ROC curve, etc.) for classification, feature selection or clustering algorithms. 2.87k 1.54k x 10.9k 1128 11 years ago v.1
OVA_Ovary GEMLeR provides a collection of gene expression datasets that can be used for benchmarking gene expression oriented machine learning algorithms. They can be used for estimation of different quality metrics (e.g. accuracy, precision, area under ROC curve, etc.) for classification, feature selection or clustering algorithms. 2.86k 1.54k x 10.9k 1166 11 years ago v.1
OVA_Uterus GEMLeR provides a collection of gene expression datasets that can be used for benchmarking gene expression oriented machine learning algorithms. They can be used for estimation of different quality metrics (e.g. accuracy, precision, area under ROC curve, etc.) for classification, feature selection or clustering algorithms. 2.86k 1.54k x 10.9k 1138 11 years ago v.1
OVA_Kidney GEMLeR provides a collection of gene expression datasets that can be used for benchmarking gene expression oriented machine learning algorithms. They can be used for estimation of different quality metrics (e.g. accuracy, precision, area under ROC curve, etc.) for classification, feature selection or clustering algorithms. 2.85k 1.54k x 10.9k 1134 11 years ago v.1

We applied a filtering:

$n \geq 1000$

$d \geq 1000$

at least one official run

resulting 14 datasets

EnsLoss is a more desirable option compared to fixed losses in image data; and it is a viable option worth considering in tabular data.

MODELS	BCE	EXP	HINGE	ENSLOSS
(Acc)				
ResNet34	76.91(0.52)	73.78(0.52)	77.20(0.18)	82.33(0.30)
ResNet50	77.23(0.51)	74.10(0.49)	77.96(0.34)	82.00(0.07)
VGG16	80.97(0.25)	77.11(0.50)	82.69(0.30)	85.77(0.35)
VGG19	81.58(0.25)	76.13(0.35)	82.77(0.41)	85.91(0.19)
(AUC)				
ResNet34	88.69(0.34)	83.30(0.57)	76.11(0.37)	92.24(0.13)
ResNet50	88.75(0.30)	83.51(0.46)	77.24(0.67)	92.07(0.49)
VGG16	93.35(0.26)	88.77(0.59)	86.18(0.56)	95.44(0.24)
VGG19	93.49(0.17)	87.89(0.46)	84.09(0.60)	95.51(0.14)

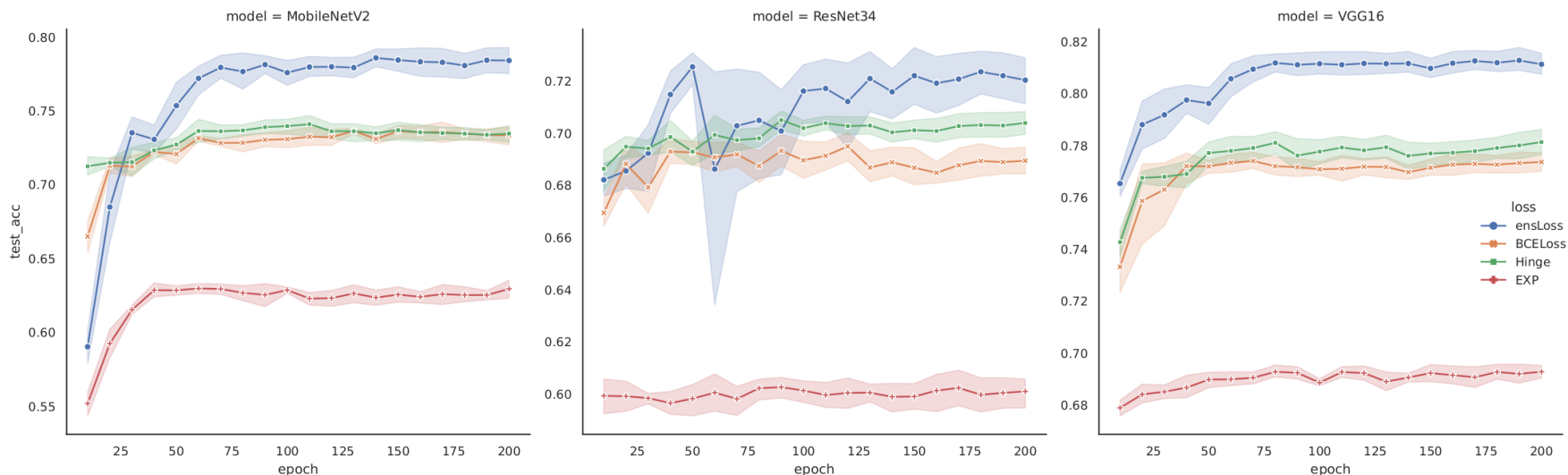
CIFAR2

(ENSLOSS) MODELS	(vs BCE) (better, no diff, worse)	(vs EXP) with p < 0.05	(vs HINGE) with p < 0.05
ResNet34	(41, 4, 0)	(45, 0, 0)	(36, 9, 0)
ResNet50	(42, 3, 0)	(45, 0, 0)	(43, 2, 0)
ResNet101	(39, 6, 0)	(45, 0, 0)	(40, 5, 0)
VGG16	(36, 9, 0)	(45, 0, 0)	(29, 16, 0)
VGG19	(36, 9, 0)	(45, 0, 0)	(27, 18, 0)
MobileNet	(45, 0, 0)	(45, 0, 0)	(44, 1, 0)
MobileNetV2	(45, 0, 0)	(45, 0, 0)	(45, 0, 0)

OpenML

(ENSLOSS) MODELS	(vs BCE) (better, no diff, worse)	(vs EXP) with p < 0.05	(vs HINGE) with p < 0.05
MLP(1)	(9, 4, 1)	(7, 5, 2)	(5, 4, 5)
MLP(3)	(7, 7, 0)	(8, 5, 1)	(9, 3, 2)
MLP(5)	(11, 3, 0)	(11, 2, 1)	(13, 0, 1)

Epoch-level performance



Compatibility of prevent-overfitting methods

REG	HP	BCE	EXP	HINGE	ENSLoss
NO REG; baseline	—	67.99(0.30)	60.09(0.19)	68.19(0.40)	69.52(1.38)
WEIGHTD	5e-5	67.64(0.14)	60.43(0.23)	68.26(0.65)	71.01(1.04)
	5e-4	67.59(0.35)	61.57(0.56)	67.57(0.28)	72.04(0.35)
	5e-3	68.00(0.31)	62.26(0.45)	68.26(0.35)	70.84(0.67)
DROPOUT	0.1	67.50(0.39)	60.70(0.34)	67.89(0.30)	72.48(0.22)
	0.2	68.13(0.54)	60.02(0.52)	67.78(0.44)	70.08(1.28)
	0.3	67.65(0.29)	59.70(0.46)	67.78(0.49)	72.44(0.68)
DATAUG	—	79.22(0.12)	58.96(0.31)	80.47(0.26)	83.00(0.25)

EnsLoss consistently outperforms the fixed losses across epochs; and it is compatible with other methods, and their combination yields additional improvement.

Summary

The primary motivation of **EnsLoss** behind consists of two components: “**ensemble**” and the “**CC**” of the loss functions.

This concept can be extensively applied to various ML problems, by identify the *specific conditions for loss consistency* or calibration.

Thank you!

