PENEX: AdaBoost-Inspired Neural Network Regularization

Klaus-Rudolf Kladny[1,2] **Bernhard Schölkopf**[1,2,3]

Michael Muehlebach[1,4]

[1] Max Planck Institute for Intelligent Systems, Tübingen
[2] Tübingen AI Center
[3] ELLIS Institute Tübingen
[4] Vienna University of Economics and Business
{kkladny,bs,michaelm}@tue.mpg.de

Abstract

AdaBoost sequentially fits so-called weak learners to minimize an exponential loss, which penalizes misclassified data points more severely than other loss functions like cross-entropy. Paradoxically, AdaBoost generalizes well in practice as the number of weak learners grows. In the present work, we introduce *Penalized Exponential Loss* (PENEX), a new formulation of the multi-class exponential loss that is theoretically grounded and, in contrast to the existing formulation, amenable to optimization via first-order methods, making it a practical objective for training neural networks. We demonstrate that PENEX effectively increases margins of data points, which can be translated into a generalization bound. Empirically, across computer vision and language tasks, PENEX improves neural network generalization in *low-data regimes*, matching and in some settings outperforming established regularizers at comparable computational cost. Our results highlight the potential of the exponential loss beyond its application in AdaBoost.

1 Introduction

Regularization techniques improve generalization, typically at the cost of reduced in-sample performance (Goodfellow et al., 2016). There exists a variety of regularizers such as ℓ_2 regularization (Tikhonov, 1943; Foster, 1961), early stopping (e.g., Bishop (1995)), RKHS norm penalization (e.g., Schölkopf & Smola (2002)), adversarial training (Szegedy et al., 2014), dropout (Srivastava et al., 2014) and entropy regularization (Meister et al., 2020).

A seemingly unrelated branch of literature to regularization is boosting (Schapire, 1990): The method of generating a *strong learner* by sequentially fitting *weak learners*. A weak learner can be thought of as an inaccurate "rule-of-thumb" that barely does better than random guessing. A prime example of a weak learner is a single-split decision tree ("decision stump"), which classifies based on a single attribute split (e.g., Hastie et al. (2017, p. 362); left part of Fig. 1). The goal of boosting is to train weak learners in order to gradually refine the predictions arising from a linear combination over all weak learners that have been fit so far (resulting in a single, strong learner). One of the best-known boosting algorithms for classification is AdaBoost (Freund & Schapire, 1995), which has been called the "best off-the-shelf classifier in the world" by Leo Breiman (Friedman et al., 2000). Although originally derived from the probably approximately correct learning framework (Valiant, 1984), Breiman (1999) famously gave AdaBoost an interpretation in the language of empirical risk minimization (Vapnik, 2000) by showing that AdaBoost minimizes an exponential loss function.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

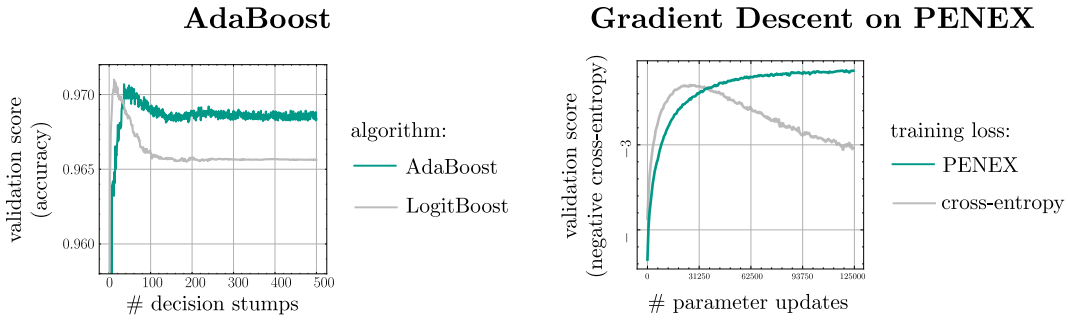

Figure 1: **AdaBoost vs. Gradient Descent on PENEX.** AdaBoost (left) builds a strong learner by sequentially fitting weak learners such as decision stumps and linearly combining them. Prior work has shown that this algorithm tends to be resilient to "overfitting", meaning that validation performance does not worsen as many decision stumps are added. PENEX (right) extends this phenomenon to deep neural network training by re-formulating the exponential loss into a formulation that is amenable to optimization via first-order methods. Just as for AdaBoost, the neural network does not overfit and keeps generalizing better in contrast to using cross-entropy loss.

This loss has been criticized for its lack of robustness with respect to outliers and misclassified data points (Bishop, 2006, p. 662). Counterintuitively, empirical studies have demonstrated that AdaBoost resists "overfitting" in practice as the number of weak learners increases (Drucker & Cortes, 1995; Breiman, 1996a; Friedman et al., 2000), thereby seemingly defying the Occam's razor principle by which simpler models should be favored over complex ones (e.g., MacKay (2003, p. 343)). To resolve the paradox, various authors have developed theoretical arguments, most of them based on margin properties (Schapire et al., 1998; Rätsch et al., 2001; Rosset et al., 2003; Grove & Schuurmans, 1998; Onoda et al., 1998; Rosset et al., 2004):[1] The margin is the minimum distance of a data point to any decision boundary. Due to the large penalty on miss-classifications, the model concentrates more on hard-to-learn examples, which play a similar role to support vectors (Cortes & Vapnik (1995), Fig. 2). Thus, the tail of the margin distribution is lowered. A light tail of the margin distribution, in turn, guarantees good generalization (Mohri et al., 2018, Thm. 9.2).

Motivated by the works above, we take the view that AdaBoost does not generalize well *in spite* of the exponential loss, but *because* of it, in the sense that it acts as a regularizer. The main contribution of the present work is to effectively translate this regularizing quality to training deep neural network classifiers. Specifically, we introduce the *Penalized Exponential Loss* (PENEX), which reformulates the objective of the multi-class exponential loss (Zhu et al., 2009) to a neural network-friendly variant by replacing a hard sum constraint by a SumExp penalty. We show that PENEX is self-calibrating, meaning that at the population level its optimal logits can be rescaled in closed form to recover the true class-probability distribution, and that it is Fisher consistent (Fisher, 1922), i.e., its optimal logits induce the Bayes-optimal classifier (Lin, 2004; Zhu et al., 2009). We also prove that PENEX bounds the tail of the margin distribution, thereby providing a theoretical foundation for PENEX's generalization ability.[2] We propose to minimize PENEX via gradient descent, which, at first sight, appears to contrast the idea of sequentially fitting weak learners in the boosting framework. However, we show in Sec. A that small gradient-based parameter increments implicitly parameterize a form of weak learner. This result draws an analogy in that training on PENEX keeps improving on generalization as the model is trained for many epochs, just as how AdaBoost keeps improving on generalization as many decision stumps are added.

Empirically, we observe that PENEX performs considerably better than the standard formulation of the exponential loss (Zhu et al., 2009). In low data regimes, PENEX often improves upon the performance of popular regularizers such as label smoothing (Szegedy et al., 2016) and confidence penalty (Pereyra et al., 2017) across computer vision and language modeling tasks.

We summarize our main contributions as follows:

[1] Though alternative theories based on bias-variance theory (Breiman, 1996a,b; Bauer & Kohavi, 1999) and self-averaging properties (Wyner et al., 2017) exist.

[2] We include a broader discussion about how this result differs from the known result by Schapire et al. (1998) in Sec. C.

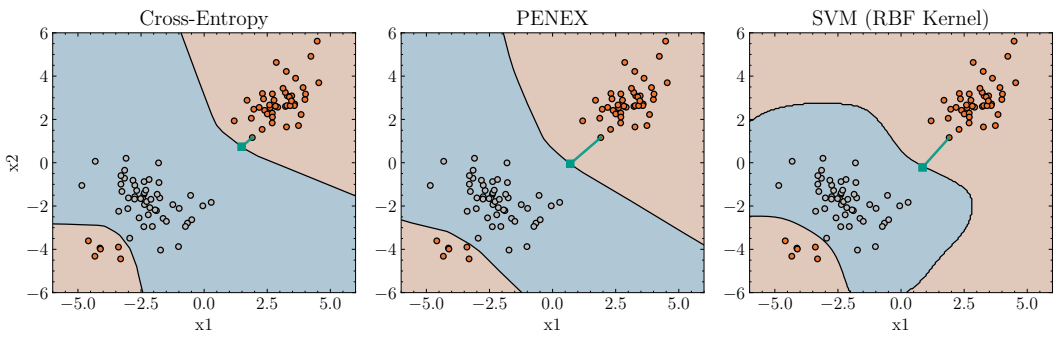

Figure 2: **Comparison of Margins.** Neural networks trained with PENEX (center) tend to implicitly increase the smallest margins (here, geometric margins, indicated for example point in green), in a similar way to support vector machines (right), here trained with a RBF kernel (Schölkopf & Smola, 2002, p. 46). Training via cross-entropy loss (left), in contrast, typically leads to smaller margins.

- We introduce PENEX (Sec. 2), an unconstrained alternative to the optimization problem proposed by Zhu et al. (2009) as stated in (2) below. We demonstrate favorable theoretical properties of PENEX, specifically self-calibration (Prop. 2.1), Fisher consistency (Cor. 2.1) and its margin-promoting quality (Thm. 2.1). The latter can be combined with an existing result by Mohri et al. (2018) to provide a generalization bound (Cor. 2.2).
- We show that the optimal penalty parameter can be inferred from Thm. 2.1, in closed form. This insight motivates an online algorithm for estimating the optimal penalty parameter (Alg. 1) and therefore eliminates one tuning parameter (Sec. 2.1, Sec. 2.2).
- We empirically compare the regularizing effect of PENEX with the classical formulation of the exponential loss and other regularizers, on diverse tasks ranging from computer vision to fine-tuning of a language model. Our results show that PENEX greatly outperforms the classical formulation of the exponential loss and often improves upon existing regularization techniques with similar computational cost, in low data regimes (Sec. 4).
- Our empirical results effectively translate the well-known observation that AdaBoost generalizes well with increasing numbers of weak learners to deep learning.

Overview. Sec. 2 introduces our main contribution, the PENEX loss function. We demonstrate desirable properties of PENEX in Sec. 2.1 and practical implementation in Sec. 2.2. In Sec. 3, we provide an overview of related works. This section is followed by experiments in Sec. 4 that demonstrate the advantages of our approach on computer vision and language models, in comparison to other regularizers and the existing formulation of the exponential loss (Sec. 4.3). Finally, we discuss limitations and future work in Sec. 5 and conclude in Sec. 6.

2 Penalized Exponential Loss (PENEX)

For class logits $f : \mathcal{X} \mapsto \mathbb{R}^K$ mapping from input space $\mathcal{X}$ to K classes, the exponential loss (EX) is given as

$$\mathcal{L}_{\text{EX}}\left(f;\, \alpha\right) \;:=\; \hat{\mathbb{E}}\left[\exp\left\{-\alpha f^{(y)}(\mathbf{x})\right\}\right], \tag{1}$$

where $\hat{\mathbb{E}}[z] = n^{-1}\sum_{i=1}^{n} z_i$ denotes the empirical mean over a given labeled training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ and $\alpha > 0$ controls sensitivity. A multi-class version of the exponential loss, which we call linearly-constrained exponential loss (CONEX), has been derived by Zhu et al. (2009):

$$\mathcal{L}_{\text{EX}}\left(f;\, (K-1)^{-1}\right) \quad \text{subject to} \quad \sum_{j=1}^{K} f^{(j)}(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \mathcal{X}. \tag{2}$$

The intuition behind the constraint in (2) is that it acts as a barrier preventing the $f^{(j)}$ from diverging without bounds: without the constraint, the exponential loss (1) can be decreased indefinitely by sending all logits to infinity. In the present work, we propose an alternative to CONEX, which

we refer to as penalized exponential loss (PENEX). It penalizes EX (1) by a SumExp $\mathrm{SE}(\mathbf{x}) \coloneqq \sum_{j=1}^{K} \exp\{f^{(j)}(\mathbf{x})\}$ over $f^{(j)}$, specifically

$$\mathcal{L}_{\text{PENEX}}(f;\, \alpha, \rho) \coloneqq \mathcal{L}_{\text{EX}}(f;\, \alpha) + \rho \hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\{f^{(j)}(\mathbf{x})\}\Big], \tag{3}$$

where $\rho > 0$ is a penalty parameter. The penalty in (3) fulfills a similar purpose as the zero-sum constraint in (2), but acts differently: PENEX penalizes individual logits for being large rather than requiring logits to "cancel each other out". The penalty is an alternative means of making the exponential loss (1) well-posed. Therefore, the penalty in (3) is not an optional "regularization term" as commonly used in methods such as weight decay, but rather acts as a barrier term preventing divergence of the optimization. The main benefit of PENEX (3) over CONEX (2) is that PENEX avoids constraints, which would require more elaborate algorithms like the augmented Lagrangian method (Hestenes, 1969). Instead, (3) poses just an objective function, which can be optimized directly using gradient-based algorithms like Adam (Kingma & Ba, 2015).

2.1 Theoretical Properties

The following proposition provides a theoretical ground for PENEX:

Proposition 2.1 (Self-Calibration)**.** *For any fixed $\rho > 0$ and $\alpha > 0$, and for any $\mathbf{x}$ in the support of the distribution of $\mathbf{x}$, the minimizer of the population equivalent of (3), called f_*:*

$$f_* \coloneqq \arg\min_{f} \mathbb{E}[\mathcal{L}_{\text{PENEX}}(f;\, \alpha, \rho)], \tag{4}$$

is unique and recovers the true class probabilities $P(y \mid \mathbf{x})$ via

$$P(y \mid \mathbf{x}) \;\propto\; \exp\Big\{(1+\alpha)\, f_*^{(y)}(\mathbf{x})\Big\},$$

i.e., the $f_*^{(y)}(\mathbf{x})$ recover the log-posteriors at temperature $(1+\alpha)^{-1}$, up to a constant shift.

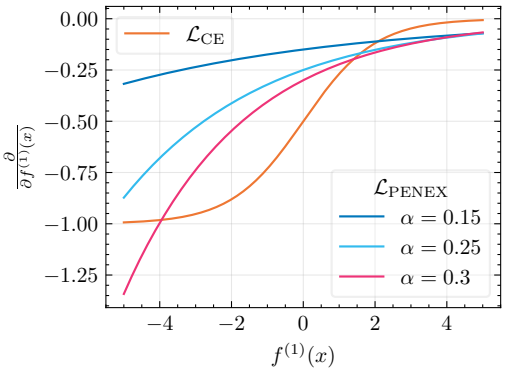

Figure 3: **CE vs. PENEX.** We consider the binary case ($K = 2$) with $f^{(2)}(x) \equiv 0$, for a single x and $y = 1$. PENEX penalizes errors more than cross-entropy.

A proof for Prop. 2.1 is supplied in Sec. B.1. Prop. 2.1 shows that PENEX provides a closed-form temperature re-scaling to recover the true class probabilities, on the population level. In addition, Prop. 2.1 lays the ground for showing that PENEX is Fisher consistent:

Corollary 2.1 (Fisher Consistency)**.** *Under the assumptions of Prop. 2.1, PENEX (3) is Fisher consistent according to the definition of Zhu et al. (2009), i.e., f_* (4) induces the Bayes-optimal classifier via*

$$\arg\max_i P(i \mid \mathbf{x}) \;=\; \arg\max_i f_*^{(i)}(\mathbf{x}).$$

Cor. 2.1 follows directly from Prop. 2.1 since $(1+\alpha) > 0$ and so

$$\arg\max_i f_*^{(i)}(\mathbf{x}) \quad = \quad \arg\max_i \exp\Big\{(1+\alpha)\, f_*^{(i)}(\mathbf{x})\Big\} \quad = \quad \arg\max_i P(i \mid \mathbf{x})\,.$$

While other loss functions such as cross-entropy $\mathcal{L}_{\text{CE}}(f) \coloneqq -\hat{\mathbb{E}}\big[\log\big(\hat{P}(y \mid \mathbf{x})\big)\big]$ with $\hat{P}(y \mid \mathbf{x}) \propto \exp\{f^{(y)}(\mathbf{x})\}$ are Fisher consistent as well, the solutions on finite datasets usually differ, as can be inferred from Fig. 3: When the "correct" logit grows negative ($f^{(1)}(\mathbf{x}) \to -\infty$), the gradient of PENEX keeps decreasing. The cross-entropy gradient, however, levels out at -1.

A key aspect in the generalization ability of PENEX lies in a margin-emphasizing quality. In the present work, we define the margin $m(\mathbf{x}, y)$ of an example $(\mathbf{x}, y)$ as the minimum difference of the true logit to any "false" logit,

$$m_f(\mathbf{x}, y) \;\coloneqq\; f^{(y)}(\mathbf{x}) \;-\; \max_{j \neq y} f^{(j)}(\mathbf{x}).$$

The next theorem shows that PENEX effectively aims at avoiding small margins:

Theorem 2.1 (Margin Tail Bound)**.** *For any $\gamma > 0$, it holds that*

$$\hat{\mathbb{P}}(m_f(\mathbf{x}, y) \leq \gamma) \leq e^{\gamma \frac{\alpha}{\alpha+1}} \rho^{-\frac{\alpha}{\alpha+1}} \mathcal{L}_{\text{PENEX}}(f;\, \alpha, \rho), \tag{5}$$

where $\hat{\mathbb{P}}$ denotes empirical probability.

A proof is provided in Sec. B.3. Intuitively, Thm. 2.1 says that the fraction of data points with a small margin is low, if PENEX has a small value for a model f. Defining the expected miss-classification error $\mathcal{R}$ as

$$\mathcal{R}(f) \;\coloneqq\; \mathbb{E}[\mathbb{1}\{\arg\max_i f^{(i)}(\mathbf{x}) \neq y\}], \tag{6}$$

we can combine Thm. 2.1 with a known result from statistical learning (Mohri et al., 2018, Thm. 9.2) to obtain the following generalization bound:

Corollary 2.2 (Generalization Bound)**.** *Assume $f \in \mathcal{F}$ and denote Rademacher complexity of $\mathcal{F}$ by $\mathfrak{R}$. Then, for any $\delta \in (0; 1), \rho > 0, \gamma > 0$, it holds with probability at least $1 - \delta$ that*

$$\mathcal{R}(f) \;\leq\; e^{\gamma \frac{\alpha}{\alpha+1}} \rho^{-\frac{\alpha}{\alpha+1}} \mathcal{L}_{\text{PENEX}}(f;\, \alpha, \rho) \;+\; \frac{4K}{\gamma} \mathfrak{R}(\mathcal{F}) \;+\; \sqrt{\frac{\log(1/\delta)}{2n}}.$$

We interpret Cor. 2.2 by minimizing the right-hand side over $\gamma > 0$. The choice of γ mediates a trade-off: the capacity term $\frac{4K}{\gamma}\mathfrak{R}(\mathcal{F})$ decreases as $1/\gamma$, whereas the leading term increases as $\exp(\gamma \frac{\alpha}{\alpha+1})$. Consequently, the bound suggests an intermediate margin threshold γ_*. Notably, achieving a smaller PENEX value decreases the prefactor of the exponential term, which shifts γ_* toward larger values and thereby reduces the capacity term $\frac{4K}{\gamma_*}\mathfrak{R}(\mathcal{F})$.

In addition to providing a generalization bound, Thm. 2.1 suggests a principled choice of ρ:

Proposition 2.2 (Optimal Penalty Parameter)**.** *The parameter ρ that minimizes the upper bound in (5) is*

$$\rho_*(\alpha, f) \;=\; \alpha \frac{\hat{\mathbb{E}}[\mathcal{L}_{\text{EX}}(f; \alpha)]}{\hat{\mathbb{E}}[\text{SE}(f; \mathbf{x})]}. \tag{7}$$

This result follows from realizing that the upper bound in (5) is convex and differentiable in ρ, thus we can solve for ρ_* by setting its derivative to zero. Prop. 2.2 has a practically relevant consequence in that it avoids the necessity of tuning ρ, as we demonstrate in the following section (Sec. 2.2).

2.2 Practical Implementation

Prop. 2.2 motivates dynamically estimating ρ_* during training. In practice, however, the optimal penalty parameter in (7) is expensive to compute directly, because f changes during training and so ρ_* would have to be re-computed on the entire data set after each parameter update. Thus, we rely on online batch statistics for estimation. To reduce variance, we use exponential moving averaging (EMA) and clipping, as shown in Alg. 1. The weights for EMA and clipping are fixed for all experiments, i.e., we do not treat them as tuning parameters. We note that Alg. 1 can be implemented in a stateful loss object and *does not require a custom training loop*. We refer to Sec. E for more details regarding the implementation.

Also, the practical implementation implements the fixed temperature-rescaling suggested by Prop. 2.1. That is, during inference, estimates of class logits are always multiplied by $1 + \alpha$ before feeding them into the softmax function (more details are provided in Sec. E.1).

3 Related Work

The exponential loss is strongly associated with AdaBoost. Thus, besides discussing prior works on gradient-based exponential loss minimization and margin maximization, we cover related works on the intersection of boosting and deep learning.

Algorithm 1: PENEX Training

Input: EMA factor $\beta > 0$; sensitivity $\alpha > 0$; clipping parameters $0 < \rho_{\min} < \rho_{\max}$; horizon $T \in \mathbb{N}^+$; initialization θ_0; optimizer `optim`; training set $\mathcal{D}$; $\epsilon > 0$;

for $t \leftarrow 1, 2, \ldots, T$ **do**
- $\mathcal{D}_\mathrm{b} \overset{i.i.d.}{\sim} \mathcal{D}$ // sample mini-batch
- $\rho' \leftarrow \alpha \frac{\mathcal{L}_{\mathrm{EX}}^{\mathcal{D}_\mathrm{b}}(f;\alpha)}{\hat{\mathbb{E}}_{\mathcal{D}_\mathrm{b}}[\mathrm{SE}(f;\mathbf{x})] + \epsilon}$ // (7)
- **if** $t = 1$ **then**
  - $\rho_0 \leftarrow \rho'$
- $\rho'_t \leftarrow (1-\beta)\rho_{t-1} + \beta\rho'$ // EMA
- $\rho_t \leftarrow \min\{\max\{\rho'_t, \rho_{\min}\}, \rho_{\max}\}$ // clip
- $\theta_t \leftarrow \texttt{optim}\left(\mathcal{L}_{\mathrm{PENEX}}^{\mathcal{D}_\mathrm{b}}(\theta_{t-1};\alpha,\rho_t)\right)$

return θ_T

Exponential Loss and Gradient Descent. Prior works have highlighted a relationship between AdaBoost and gradient descent. Specifically, Mason et al. (1999) discovered that binary AdaBoost can be thought of as an approximate gradient descent on the exponential loss, where the weak learners minimize an inner product with the gradient at the specific step. Friedman (2001) has made a similar observation and subsequently generalized this method to general loss functions, now commonly known as gradient boosting. Our work takes the reverse direction that gradient descent can also be interpreted as a form of boosting. Another line work by Soudry et al. (2018); Ji & Telgarsky (2019); Gunasekar et al. (2018); Nacson et al. (2019); Cai et al. (2025) developed theoretical results based on margin theory for performing gradient descent on the binary exponential loss (1) with linear/near-homogeneous models. This contrasts the focus of the present work, which specifically considers multi-class problems, without imposing assumptions on the underlying neural networks. Also, our ablation studies indicate that the existing formulation of the exponential loss studied in these listed works is not amenable to effective neural network training (see Sec. 4.3).

Margin Maximization in Deep Learning. The idea of explicitly maximizing the classification margin in the context of deep learning has been explored in prior work: One line of work studies the effects of the spectral norm of the network's Jacobian matrix (Sokolić et al., 2017) or network depth (Sun et al., 2016) on the classification margin to motivate regularization techniques. Elsayed et al. (2018) introduce a large-margin objective that enforces margins at selected intermediate layers. The method adds several hyperparameters and requires computing per-example gradient norms with respect to those activations (scaling with the number of layers and top-k classes), making the method computationally demanding on deeper architectures. Another line of work by Jiang et al. (2019) shows that the distribution of the margin correlates with the generalization gap, which inspired Lyu et al. (2022) to develop a loss that penalizes deviations of the margin from its expected value. While similarly lightweight to PENEX, the objective of Lyu et al. (2022) does not recover the full multiclass probability distribution, thereby leaving the problem of calibration un-resolved (see Sec. G.5 for experimental evidence).

Boosting in Deep Learning. Tolstikhin et al. (2017); Grover & Ermon (2018); Giaquinto & Banerjee (2020) construct a "strong" generative model by greedily fitting multiple "weak" generative models in a boosting-like fashion. Similarly, but with classifiers instead of generative models, dense (Schwenk & Bengio, 1997) and convolutional neural networks (Drucker et al., 1992; LeCun et al., 1995; Taherkhani et al., 2020; Sun et al., 2021) have been used as weak learners for Boosting. In the context of language models, boosting has further been applied to prompt generation (Pitis et al., 2023; Zhang et al., 2024), text classification (Hou et al., 2023; Huang et al., 2020), text-summary based prediction (Manikandan et al., 2023) and machine teaching (Agrawal et al., 2024). All of these studies differ from the present work in what is considered as a "weak learner". In the present work, this notion only arises implicitly (Sec. A) and is given as the Jacobian, multiplied by a small parameter increment (10), instead of existing approaches that consider explicitly combining entire models (such as decision trees and neural networks) to gradually build more complex ones.

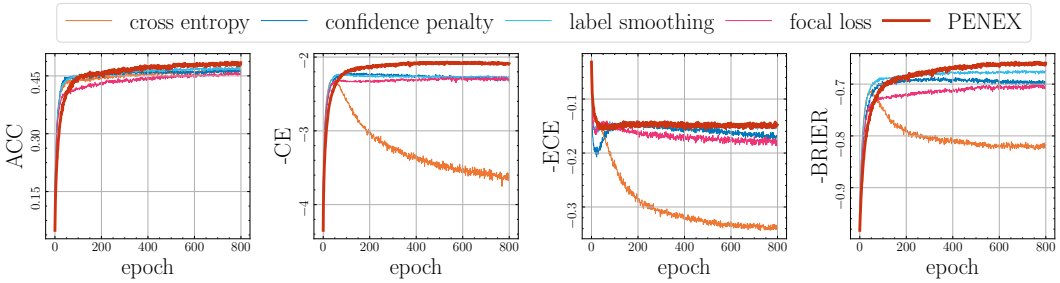

Figure 4: **Performance Analysis on CIFAR-100.** Larger means better. Results are computed from validation data. All hyperparameters have been tuned individually. PENEX (thick red) is an effective regularizer with often better generalization than other common regularization techniques (thin), and shows no signs of "overfitting" like cross-entropy training (orange)

4 Experiments

We aim to answer the following questions: **1)** How does PENEX compare with other regularization methods? **2)** How does optimizing PENEX (3) compare to optimizing CONEX (2) via constrained optimization? **3)** How does PENEX perform under label noise? **4)** How does PENEX perform for fine-tuning foundation models? **5)** How does the value of α affect training? **6)** How do other penalty formulations compare with the SumExp penalty term? **7)** How does Alg. 1 compare with tuning a fixed ρ? **8)** How sensitive is Alg. 1 with respect to EMA and clipping parameters?

Experimental results addressing questions **5)**-**8)** are shown in Sec. G.6, Sec. G.7, Sec. G.8 and Sec. G.9, respectively. All other questions are answered in the main text.

Reproducibility. All code can be found on GitHub at https://github.com/rudolfwilliam/penex.

4.1 Setup

In the main text, we provide a high-level description of the experimental setup. More details can be found in Sec. F.

Baselines. For fairness of comparison, we restrict ourselves to baselines that can be expressed as a single, architecture-invariant loss function that does not require additional forward or backward passes (more details in Sec. G.1). **1)** Cross-entropy (CE). **2)** Confidence penalty (Pereyra et al. (2017)), **3)** Label smoothing (Szegedy et al. (2016)), and **4)** Focal loss (Lin et al., 2018). The latter three are well-established regularization techniques.

Tasks. **1-2)** CIFAR-10/100 (Krizhevsky, 2009): Image classification with either 10 or 100 classes, using a convolutional neural network. An additional experiment with a residual neural network (He et al., 2016) is shown in Sec. G.3. **3)** Noisy CIFAR-10: CIFAR-10 with 10% of training labels flipped uniformly at random. The test set does not have flipped labels. Further noise settings are covered in Sec. G.4. **4)** Pathology Image Classification (PathMNIST; Kather et al. (2019); Yang et al. (2023)): Recognizing 9 different tissue types from $10,000$ non-overlapping image patches from hematoxylin & eosin stained histological images, using a convolutional neural network. **5)** ImageNet (Deng et al. (2009)): Recognizing 1000 different classes from $\approx$ 14 million images, using a vision transformer (Dosovitskiy et al., 2021). **6)** News Classification (BBC News; Greene & Cunningham (2006)): Fine-tuning a pretrained RoBERTa-base language transformer encoder (Liu et al., 2019) for news article classification with 5 classes.

Metrics. **1)** Accuracy (ACC): Proportion of correctly classified points ($1 - \hat{\mathcal{R}}(f)$; see (6)). **2)** Cross-Entropy (CE): Corresponds to $\mathcal{L}_{\text{CE}}$, which (for classification) is equivalent to the *negative log-likelihood*. **3)** Brier Score (BRIER; Brier (1950)): Mean squared difference between predicted class probabilities and one-hot encoded labels. **4)** Expected Calibration Error (ECE; Guo et al. (2017)): Measures the discrepancy between a model's predicted confidence and its actual accuracy. *We change the sign in some metrics, such that a larger value indicates better performance for all metrics.*

Table 1: **Test Set Performance.** Larger means better, best is bold. Mean test result ± 1 standard deviation after 200 training epochs (for ImageNet: 300 epochs). Standard deviation is obtained by running 100 bootstrap evaluations (Efron & Tibshirani, 1993). Hyperparameters are tuned on validation data (except for ImageNet, see Sec. F.11).

Method	Metric	CIFAR-10	Noisy CIFAR-10	CIFAR-100	PathMNIST	ImageNet	BBC News
CE	ACC	0.785 ± 0.004	0.724 ± 0.004	0.443 ± 0.004	0.826 ± 0.004	0.648 ± 0.000	0.967 ± 0.007
	-ECE	-0.162 ± 0.003	-0.179 ± 0.003	-0.287 ± 0.003	-0.151 ± 0.004	-0.200 ± 0.000	$\mathbf{-0.032 \pm 0.006}$
	-CE	-1.004 ± 0.024	-1.125 ± 0.019	-3.072 ± 0.034	-2.018 ± 0.130	-2.175 ± 0.000	-0.109 ± 0.024
	-BRIER	-0.346 ± 0.006	-0.424 ± 0.006	-0.794 ± 0.006	-0.300 ± 0.007	-0.536 ± 0.000	-0.051 ± 0.011
label smoothing	ACC	0.789 ± 0.004	0.747 ± 0.004	0.451 ± 0.005	0.829 ± 0.004	$\mathbf{0.651 \pm 0.000}$	0.970 ± 0.006
	-ECE	-0.112 ± 0.002	-0.183 ± 0.003	$\mathbf{-0.147 \pm 0.002}$	-0.109 ± 0.002	-0.159 ± 0.000	-0.033 ± 0.006
	-CE	-0.657 ± 0.011	-0.889 ± 0.008	-2.292 ± 0.019	$\mathbf{-0.589 \pm 0.012}$	-2.180 ± 0.000	-0.115 ± 0.022
	-BRIER	-0.300 ± 0.005	-0.384 ± 0.004	-0.692 ± 0.004	-0.255 ± 0.005	-0.537 ± 0.000	-0.049 ± 0.010
confidence penalty	ACC	0.786 ± 0.004	0.733 ± 0.004	0.449 ± 0.006	0.828 ± 0.004	0.641 ± 0.000	$\mathbf{0.974 \pm 0.006}$
	-ECE	-0.130 ± 0.002	-0.149 ± 0.003	-0.152 ± 0.002	-0.110 ± 0.003	-0.161 ± 0.000	-0.050 ± 0.005
	-CE	-0.731 ± 0.015	-0.866 ± 0.009	-2.254 ± 0.018	-0.917 ± 0.047	-1.972 ± 0.000	-0.094 ± 0.015
	-BRIER	-0.317 ± 0.005	-0.385 ± 0.004	-0.695 ± 0.005	-0.262 ± 0.005	-0.518 ± 0.000	$\mathbf{-0.042 \pm 0.008}$
focal loss	ACC	0.778 ± 0.004	0.708 ± 0.004	0.428 ± 0.005	0.803 ± 0.004	0.647 ± 0.000	0.970 ± 0.006
	-ECE	-0.117 ± 0.002	-0.165 ± 0.003	-0.161 ± 0.003	-0.112 ± 0.003	$\mathbf{-0.093 \pm 0.000}$	-0.051 ± 0.005
	-CE	-0.661 ± 0.010	-0.905 ± 0.008	-2.341 ± 0.022	-0.939 ± 0.050	$\mathbf{-1.578 \pm 0.000}$	$\mathbf{-0.092 \pm 0.014}$
	-BRIER	-0.313 ± 0.005	-0.423 ± 0.004	-0.723 ± 0.005	-0.291 ± 0.006	$\mathbf{-0.482 \pm 0.000}$	$\mathbf{-0.042 \pm 0.008}$
PENEX	ACC	$\mathbf{0.793 \pm 0.004}$	$\mathbf{0.766 \pm 0.004}$	$\mathbf{0.460 \pm 0.005}$	$\mathbf{0.833 \pm 0.004}$	0.590 ± 0.000	0.968 ± 0.006
	-ECE	$\mathbf{-0.109 \pm 0.002}$	$\mathbf{-0.131 \pm 0.002}$	$\mathbf{-0.147 \pm 0.003}$	$\mathbf{-0.100 \pm 0.003}$	-0.107 ± 0.000	-0.034 ± 0.006
	-CE	$\mathbf{-0.646 \pm 0.012}$	$\mathbf{-0.716 \pm 0.009}$	$\mathbf{-2.140 \pm 0.018}$	-1.200 ± 0.089	-1.853 ± 0.000	-0.124 ± 0.025
	-BRIER	$\mathbf{-0.299 \pm 0.005}$	$\mathbf{-0.332 \pm 0.004}$	$\mathbf{-0.685 \pm 0.004}$	$\mathbf{-0.251 \pm 0.006}$	-0.553 ± 0.000	-0.055 ± 0.011

Hyperparameter Tuning. Each method's hyperparameters (except for the learning rate, which we keep fixed for all methods) are tuned on the validation set using the tree-structured Parzen estimator (e.g., Watanabe (2023)). We minimize cross-entropy loss over a fixed amount of epochs. For PENEX, we always estimate $\rho^*(\alpha)$ according to Alg. 1 (see Sec. E for details). Thus, *PENEX only requires tuning the sensitivity* α. Tuning is performed individually for each experiment and method, except for ImageNet, where we use default parameters for each method (see Sec. F.11).

4.2 Main Results

PENEX often achieves regularization performance that outperforms other regularization techniques (though taking a bit longer to converge), as can be seen in Fig. 4, showing multiple validation curves for CIFAR-100 (analogous plots for all others datasets are shown in Sec. G.2) as a function of the number of training epochs. We also provide test results in Tab. 1 that confirm the results from Fig. 4 on multiple datasets, domains and metrics. Remarkably, PENEX also outperforms other regularizers in the setting of artificial label noise. The only experiment where PENEX performs slightly worse than other methods is the ImageNet experiment. However, ImageNet is a large data set and other regularization methods do also not benefit generalization much in this setting.

4.3 Ablation Studies

We compare PENEX with the following ablations: **1–3)** Optimizing CONEX (2) via **1)** the augmented Lagrangian method (Hestenes, 1969); or **2)** the squared penalty method. **3)** Fixing $f^{(K)}(\mathbf{x}) \equiv -\sum_{j=1}^{K-1} f^{(j)}(\mathbf{x})$. **4)** Optimizing EX (1) directly, neither with constraint nor penalty. Details for the implementation of each baseline are provided in Sec. F.4.

We perform all ablation studies for the CIFAR-10 dataset only. The results, displayed in Fig. 5, confirm the claim that PENEX is favorable over the CONEX formulation (2) in the context of training deep neural networks. Enforcing the zero-sum constraint of CONEX in the model architecture (ablation **3)**) and directly minimizing the exponential loss (1) without any constraint (ablation **4)**) break down entirely: Using no penalty or constraint leads to all logits diverging without bounds, while the hard constraint method results in `NaN` logits around epoch 80. Only the penalty and augmented Lagrangian approach for optimizing CONEX work at all, while still performing much worse than PENEX, in addition to requiring the tuning of at least one more hyperparameter (see Sec. F.10).

Overall, the ablation studies show that the PENEX formulation of the exponential loss is not merely a minor refinement over CONEX, but crucial for stable and effective neural network training.

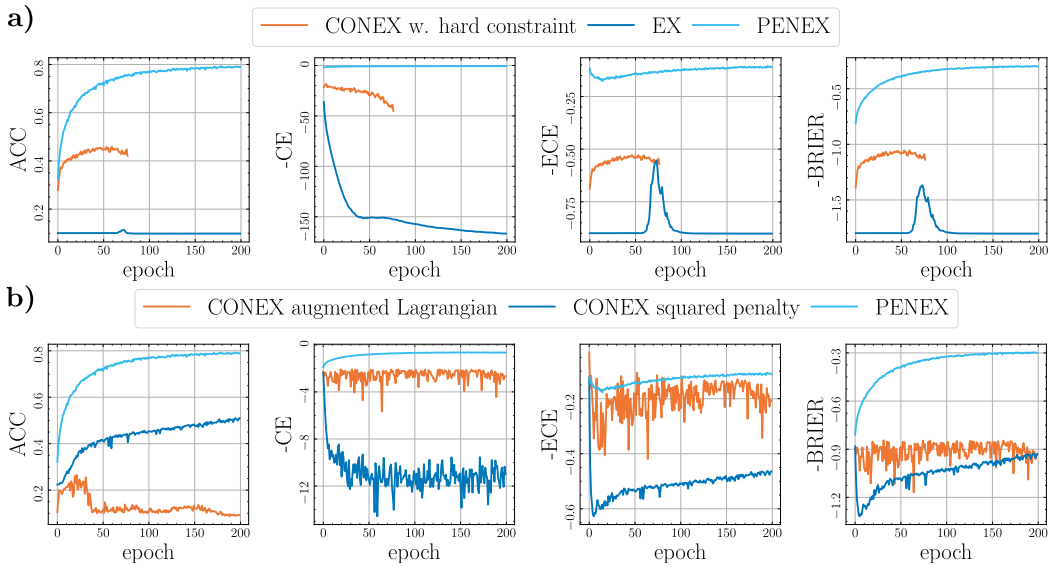

Figure 5: **Ablation Studies on CIFAR-10.** Larger means better. **a)** Optimizing the exponential loss without any constraint or with the constraint encoded into the model architectures break down early on during training (no results are shown for the hard constraint method starting at around epoch 80 because of `NaN` model outputs). **b)** While constrained optimization algorithms based on CONEX (2) are trainable in principle, they reach worse optima and suffer from less stable training, in addition to requiring tuning more parameters (see Sec. F.10).

5 Discussion & Limitations

The Secret Sauce of AdaBoost. Conventional wisdom treats AdaBoost's exponential loss as a flaw, inspiring variants that swap the exponential for the logistic loss (Friedman et al., 2000) and generalizations to arbitrary loss functions (Mason et al., 1999; Friedman, 2001). Our evidence suggests the opposite. PENEX typically outperforms standard cross-entropy training. Remarkably, the resulting accuracy gap is wider than the well-documented (and usually modest) difference between AdaBoost and LogitBoost, two algorithms that typically achieve similar performance (e.g., Friedman et al., 2000). Explaining why the formulation introduced in the present work behaves so favorably in deep neural networks specifically remains an open direction for future work.

Label Noise. High noise levels in the training labels have been regarded to be the Achilles' heel of AdaBoost (Dietterich, 2000; Quinlan, 1996; Grove & Schuurmans, 1998; Rätsch et al., 2001), which does not translate to the present work: Neural networks trained with PENEX handle artificial label noise even better than other regularizers (see Sec. 4.2 and Sec. G.4). This phenomenon may be related to the well-known ability of deep neural networks to fit random labels well (Zhang et al., 2017). Further improving the resilience of PENEX with respect to training label corruptions, for instance by incorporating ideas by Rätsch et al. (2001); Freund (1999), may nevertheless be a promising direction for future work.

Training Set Size. The only experiment where PENEX does not perform better or equally well in comparison to other methods is the ImageNet experiment, indicating that PENEX is advantageous only in low-data regimes, such as CIFAR-100. We believe that developing remedies for this issue is another relevant direction for future research.

Beyond Classification. We propose PENEX as a loss function for classification problems. Future work may explore adaptions of PENEX to regression problems. Specifically, one may draw inspiration from existing works such as the ones by Drucker (1997); Solomatine & Shrestha (2004) who proposed boosting techniques for regression problems.

Adaptive Penalty Parameter. A limitation of the adaptive choice of ρ in Alg. 1 is that it does not retain the identification of the global additive logit offset established for the fixed-ρ objective in Prop. 2.1. In the idealized setting where ρ is recomputed exactly at every step without EMA, a common shift of all logits only changes the magnitude, but not the direction, of the resulting gradient. Consequently, the adaptive dynamics identify the logits only up to a common additive constant. This ambiguity does not affect the margin-tail and generalization results (Thm. 2.1 and Cor. 2.2), because they are invariant to a constant logit shift. An experiment with a tuned and fixed ρ in Sec. G.8 furthermore demonstrates that, if tuned, a fixed ρ can lead to even better results. The adaptive algorithm (Alg. 1) remains a competitive heuristic if tuning is not feasible.

6 Conclusion

We introduced the *penalized exponential loss* (PENEX), a margin-promoting objective inspired by multi-class AdaBoost and designed for neural network training. We provided practical and theoretical motivation for its form and showed empirically that PENEX acts as a training regularizer: across various settings, it improves generalization over standard cross-entropy and the traditional formulation of the exponential loss. In low-data regimes, PENEX often surpasses established regularization techniques like label smoothing. Beyond offering a practical loss function, our results also highlight a broader takeaway: in the form presented here, the exponential loss can be useful beyond the setting of the AdaBoost algorithm. In particular, PENEX demonstrates that the margin-emphasizing behavior that underpins AdaBoost can be translated into modern neural optimization. Additionally, PENEX challenges the dominant paradigm in which a loss function is constructed as a well-behaved log-likelihood (e.g., cross-entropy) plus a separate "regularization term," because its exponential form *intrinsically regularizes* by encouraging large margins, rather than treating regularization as an add-on to a base log-likelihood. Overall, we hope PENEX provides a simple and practical alternative to cross-entropy, particularly in limited-data regimes, and that it encourages further work on exponential-loss objectives beyond classical boosting.

Acknowledgments and Disclosure of Funding

Michael Muehlebach is supported by the German Research Foundation (DFG) through the Emmy Noether Programme (project no. 456587626). The authors acknowledge support from the FWF Cluster of Excellence BilAI. Klaus-Rudolf Kladny acknowledges support from the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

References

Aakriti Agrawal, Mucong Ding, Zora Che, Chenghao Deng, Anirudh Satheesh, John Langford, and Furong Huang. EnsemW2S: Can an Ensemble of LLMs be Leveraged to Obtain a Stronger LLM? *arXiv preprint arXiv:2410.04571*, 2024. 6

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. *International Conference on Knowledge Discovery and Data Mining*, 2019. 21

Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999. 2

Christopher M. Bishop. Regularization and Complexity Control in Feed-forward Networks. *International Conference on Artificial Neural Networks*, 1995. 1

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. 2

Leo Breiman. Arcing Classifiers. *Annals of Statistics*, 26:123–40, 1996a. 2

Leo Breiman. Bias, Variance, and Arcing Classifiers. *Tech. Rep. 460, Statistics Department, University of California, Berkeley*, 1996b. 2

Leo Breiman. Prediction Games and Arcing Algorithms. *Neural Computation*, 11(7):1493–1517, 1999. 1

Glenn W. Brier. Verification of Forecasts Expressed in Terms of Probability. *Monthly Weather Review*, 78(1):1–3, 1950. 7

Yuhang Cai, Kangjie Zhou, Jingfeng Wu, Song Mei, Michael Lindsey, and Peter L. Bartlett. Implicit Bias of Gradient Descent for Non-Homogeneous Deep Networks. *International Conference on Machine Learning*, 2025. 6

Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine Learning*, 20(3):273–297, 1995. 2

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. 7

Thomas G. Dietterich. An Experimental Comparison of Three Methods for Constructing Ensembles of Decision Trees: Bagging, Boosting, and Randomization. *Machine Learning*, 40:139–157, 2000. 9

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *International Conference on Learning Representations*, 2021. 7, 27

Harris Drucker. Improving Regressors using Boosting Techniques. In *International Conference on Machine Learning*, volume 97, 1997. 9

Harris Drucker and Corinna Cortes. Boosting Decision Trees. *Advances in Neural Information Processing Systems*, 8, 1995. 2

Harris Drucker, Robert Schapire, and Patrice Simard. Improving Performance in Neural Networks Using a Boosting Algorithm. *Advances in Neural Information Processing Systems*, 5, 1992. 6

Bradley Efron and Robert J. Tibshirani. An Introduction to the Bootstrap. *Monographs on Statistics and Applied Probability*, 57:1–436, 1993. 8

Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. Large Margin Deep Networks for Classification. *Advances in Neural Information Processing Systems*, 31, 2018. 6

William Falcon and The PyTorch Lightning team. PyTorch Lightning. *Apache-2.0*, 2019. doi: 10.5281/zenodo.3828935. URL https://github.com/Lightning-AI/lightning. 21

Ronald A. Fisher. On the Mathematical Foundations of Theoretical Statistics. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 1922. 2

Manus Foster. An application of the Wiener-Kolmogorov smoothing theory to matrix inversion. *Journal of the Society for Industrial and Applied Mathematics*, 9(3):387–392, 1961. 1

Yoav Freund. An adaptive version of the boost by majority algorithm. In *Conference on Computational Learning Theory*, pp. 102–113, 1999. 9

Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *European Conference on Computational Learning Theory*, pp. 23–37, 1995. 1, 19

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive Logistic Regression: A Statistical View of Boosting. *The Annals of Statistics*, 28(2):337–407, 2000. 1, 2, 9

Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. 6, 9

John D. Garrett. garrettj403/SciencePlots. *Zenodo*, September 2021. doi: 10.5281/zenodo.4106649. URL http://doi.org/10.5281/zenodo.4106649. 21

Robert Giaquinto and Arindam Banerjee. Gradient Boosted Normalizing Flows. *Advances in Neural Information Processing Systems*, 33:22104–22117, 2020. 6

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*, volume 1. MIT Press, 2016. 1

Derek Greene and Pádraig Cunningham. Practical Solutions to the Problem of Diagonal Dominance in Kernel Document Clustering. In *International Conference on Machine Learning*, pp. 377–384, 2006. 7

Adam J. Grove and Dale Schuurmans. Boosting in the limit: Maximizing the margin of learned ensembles. *AAAI Conference on Artificial Intelligence*, pp. 692–699, 1998. 2, 9

Aditya Grover and Stefano Ermon. Boosted Generative Models. *AAAI Conference on Artificial Intelligence*, 32(1), 2018. 6

Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing Implicit Bias in Terms of Optimization Geometry. In *International Conference on Machine Learning*, pp. 1832–1841, 2018. 6

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017. 7

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2017. 1, 17

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016. 7, 29

Magnus R. Hestenes. Multiplier and Gradient Methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. 4, 8, 23

Bairu Hou, Joe O'connor, Jacob Andreas, Shiyu Chang, and Yang Zhang. Promptboosting: Black-box text classification with ten forward passes. *International Conference on Machine Learning*, pp. 13309–13324, 2023. 6

Tongwen Huang, Qingyun She, and Junlin Zhang. BoostingBERT: Integrating multi-class boosting into BERT for NLP tasks. *arXiv preprint arXiv:2009.05959*, 2020. 6

Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2019. 6

Yiding Jiang, Dilip Krishnan, Hossein Mobahi, and Samy Bengio. Predicting the Generalization Gap in Deep Networks with Margin Distributions. *International Conference on Learning Representations*, 2019. 6

Jakob Nikolas Kather, Johannes Krisam, Pornpimol Charoentong, Tom Luedde, Esther Herpel, Cleo-Aron Weis, Timo Gaiser, Alexander Marx, Nektarios A. Valous, Dyke Ferber, et al. Predicting survival from colorectal cancer histology slides using deep learning: A retrospective multicenter study. *PLoS medicine*, 16(1), 2019. 7

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015. 4, 27

Klaus-Rudolf Kladny, Bernhard Schölkopf, and Michael Muehlebach. Conformal Generative Modeling with Improved Sample Efficiency through Sequential Greedy Filtering. *International Conference on Learning Representations*, 2025. 25

Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. 7

Yann LeCun, Lawrence D. Jackel, Léon Bottou, Corinna Cortes, John S. Denker, Harris Drucker, Isabelle Guyon, Urs A. Muller, Eduard Sackinger, Patrice Simard, et al. Learning Algorithms for Classification: A Comparison on Handwritten Digit Recognition. *Neural Networks: The Statistical Mechanics Perspective*, 261(276):2, 1995. 6

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal Loss for Dense Object Detection. *Conference on Computer Vision and Pattern Recognition*, pp. 2980–2988, 2018. 7, 28

Yi Lin. A note on margin-based loss functions in classification. *Statistics & Probability Letters*, 68 (1):73–82, 2004. 2

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint arXiv:1907.11692*, 2019. 7

Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *International Conference on Learning Representations*, 2019. 27

Shen-Huan Lyu, Lu Wang, and Zhi-Hua Zhou. Improving generalization of deep neural networks by leveraging margin distribution. *Neural Networks*, 151:48–60, 2022. 6, 31

David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 2

Hariharan Manikandan, Yiding Jiang, and J Zico Kolter. Language models are weak learners. *Advances in Neural Information Processing Systems*, 36:50907–50931, 2023. 6

Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Boosting Algorithms as Gradient Descent. *Advances in Neural Information Processing Systems*, 12, 1999. 6, 9

Clara Meister, Elizabeth Salesky, and Ryan Cotterell. Generalized Entropy Regularization or: There's Nothing Special about Label Smoothing. *Annual Meeting of the Association for Computational Linguistics*, pp. 6870–6886, 2020. 1

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. MIT press, 2018. 2, 3, 5

Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, Pedro Henrique Pamplona Savarese, Nathan Srebro, and Daniel Soudry. Convergence of Gradient Descent on Separable Data. In *International Conference on Artificial Intelligence and Statistics*, pp. 3420–3428, 2019. 6

T. Onoda, G. Rätsch, and K.-R. Müller. An asymptotic analysis of AdaBoost in the binary classification case. *International Conference on Artificial Neural Networks*, pp. 195–200, 1998. 2

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019. 21

Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. *International Conference on Learning Representations*, 2017. 2, 7, 28

Silviu Pitis, Michael R Zhang, Andrew Wang, and Jimmy Ba. Boosted Prompt Ensembles for Large Language Models. *arXiv preprint arXiv:2304.05970*, 2023. 6

J. R. Quinlan. Boosting First-Order Learning. *International Workshop on Algorithmic Learning Theory*, pp. 143–155, 1996. 9

Gunnar Rätsch, Takashi Onoda, and K.-R. Müller. Soft Margins for AdaBoost. *Machine Learning*, 42:287–320, 2001. 2, 9

Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin Maximizing Loss Functions. *Advances in Neural Information Processing Systems*, 16, 2003. 2

Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a Regularized Path to a Maximum Margin Classifier. *The Journal of Machine Learning Research*, 5:941–973, 2004. 2

Robert E. Schapire. The Strength of Weak Learnability. *Machine Learning*, 5:197–227, 1990. 1

Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. *The Annals of Statistics*, 26(5):1651–1686, 1998. 2, 19

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002. 1, 3

Holger Schwenk and Yoshua Bengio. AdaBoosting Neural Networks: Application to on-line Character Recognition. *International Conference on Artificial Neural Networks*, pp. 967–972, 1997. 6

Jure Sokolić, Raja Giryes, Guillermo Sapiro, and Miguel R. D. Rodrigues. Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, 2017. 6

Dimitri P. Solomatine and Durga L. Shrestha. AdaBoost.RT: a Boosting Algorithm for Regression Problems. In *International Joint Conference on Neural Networks*, volume 2, pp. 1163–1168, 2004. 9

Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The Implicit Bias of Gradient Descent on Separable Data. *Journal of Machine Learning Research*, 19 (70):1–57, 2018. 6

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, 15:1929–1958, 2014. 1

Georg Still. *Lectures on Parametric Optimization: An Introduction*. Preprint, University of Twente, 2018. 18

Ke Sun, Zhanxing Zhu, and Zhouchen Lin. AdaGCN: Adaboosting Graph Convolutional Networks into Deep Models. *International Conference on Learning Representations*, 2021. 6

Shizhao Sun, Wei Chen, Liwei Wang, Xiaoguang Liu, and Tie-Yan Liu. On the Depth of Deep Neural Networks: A Theoretical View. In *AAAI Conference on Artificial Intelligence*, volume 30, 2016. 6

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *International Conference on Learning Representations*, 2014. 1

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. *Conference on Computer Vision and Pattern Recognition*, pp. 2818–2826, 2016. 2, 7, 28

Aboozar Taherkhani, Georgina Cosma, and T. Martin McGinnity. AdaBoost-CNN: An adaptive boosting algorithm for convolutional neural networks to classify multi-class imbalanced datasets using transfer learning. *Neurocomputing*, 404:351–366, 2020. 6

A. N. Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pp. 195–198, 1943. 1

Ilya O Tolstikhin, Sylvain Gelly, Olivier Bousquet, Carl-Johann Simon-Gabriel, and Bernhard Schölkopf. AdaGAN: Boosting Generative Models. *Advances in Neural Information Processing Systems*, 30, 2017. 6

Leslie G. Valiant. A Theory of the Learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 1

Vladimir Vapnik. *The Nature of Statistical Learning Theory*. Springer, 2000. 1

Junhui Wang. Boosting the Generalized Margin in Cost-Sensitive Multiclass Classification. *Journal of Computational and Graphical Statistics*, 22(1):178–192, 2013. 20

Shuhei Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. *arXiv preprint arXiv:2304.11127*, 2023. 8

Jiaheng Wei, Zhaowei Zhu, Hao Cheng, Tongliang Liu, Gang Niu, and Yang Liu. Learning with Noisy Labels Revisited: A Study Using Real-World Human Annotations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TBWA6PLJZQm. 29

Abraham J. Wyner, Matthew Olson, Justin Bleich, and David Mease. Explaining the Success of AdaBoost and Random Forests as Interpolating Classifiers. *Journal of Machine Learning Research*, 18(48):1–33, 2017. 2

Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. MedMNIST v2-A large-scale lightweight benchmark for 2D and 3D biomedical image classification. *Scientific Data*, 10:41, 2023. 7

Chenrui Zhang, Lin Liu, Chuyuan Wang, Xiao Sun, Hongyu Wang, Jinpeng Wang, and Mingchen Cai. Prefer: Prompt Ensemble Learning via Feedback-Reflect-Refine. *AAAI Conference on Artificial Intelligence*, 38:19525–19532, 2024. 6

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *International Conference on Learning Representations*, 2017. 9

Ji Zhu, Hui Zou, Saharon Rosset, and Trevor Hastie. Multi-class AdaBoost. *Statistics and Its Interface*, 2:349–360, 2009. 2, 3, 4

Appendix

Table of Contents

A Relationship to Multi-Class AdaBoost **17**

B Proofs **17**
B.1 Proof for Proposition 2.1 17
B.2 Proof for Proposition A.1 18
B.3 Proof for Theorem 2.1 19

C Differences to Schapire et al. (1998) **19**

D Stagewise Additive Modeling using a Multi-class Exponential loss **20**

E Practical Implementation of PENEX **20**
E.1 Inference 20
E.2 Default Value for Sensitivity α 20
E.3 Default Values for Alg. 1 21
E.4 PyTorch-like Code 21

F Experimental Details **21**
F.1 Libraries 21
F.2 Dataset & Model Licenses 21
F.3 Computational Resources 22
F.4 Ablation Implementations 23
F.5 Metrics 24
F.6 Data Preprocessing 24
F.7 Train/Validation Splits 25
F.8 Architectures 26
F.9 Training 27
F.10 Hyperparameter Tuning 28
F.11 Hyperparameters for ImageNet 28

G Additional Experiments **29**
G.1 Runtime Comparison 29
G.2 All Validation Results 29
G.3 Alternative Model Architecture 29
G.4 Additional Noisy Label Experiments 29
G.5 Comparison with a Margin-Based Objective 31
G.6 Sensitivity Parameter Analysis 32
G.7 Different Penalties 32
G.8 Fixed Penalty Parameter 33
G.9 Ablating EMA and Clipping Parameters 33

A Relationship to Multi-Class AdaBoost

The essential ingredient of AdaBoost are so-called weak learners g_m, for all $m \in [M]$ that constitute a strong learner f_M when considering their linear combination

$$f_M^{(y)}(\mathbf{x}) \;=\; \sum_{m=1}^{M} \eta_m g_m^{(y)}(\mathbf{x}),$$

for class $y \in [K]$. These weak learners g_m, all together with their linear factors η_m, are fitted in a greedy fashion[3] according to the CONEX objective (2)

$$(g_m, \eta_m) \;\leftarrow\; \arg\min_{g \in \mathcal{G}_{\text{CONEX}},\, \eta > 0} \mathcal{L}_{\text{EX}}\Big(f_{m-1} + \eta g;\, \alpha\Big), \tag{8}$$

where $\alpha = (K-1)^{-1}$ and $\mathcal{G}_{\text{CONEX}}$ is the class of functions

$$\begin{aligned} \mathcal{G}_{\text{CONEX}} \;&:=\; \big\{g : \mathcal{X} \to \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}\big\}, \\ \mathbf{e}_i \;&:=\; \big(\hat{\mathbf{y}}^{(1)}, \ldots, \hat{\mathbf{y}}^{(K)}\big), \hat{\mathbf{y}}^{(i)} = 1, \\ &\quad\; \hat{\mathbf{y}}^{(j)} = -(K-1)^{-1} \;\; (j \neq i), \end{aligned}$$

which ensures that the linear constraint $\sum_j f_m^{(j)}(\mathbf{x}) = \sum_j \hat{\mathbf{y}}^{(j)} = 0$ is fulfilled, for all $\mathbf{x} \in \mathcal{X}$ (2). In practice, the g_m are typically represented by decision stumps (e.g., Hastie et al. (2017, p.362)). For completeness, we show the full AdaBoost algorithm in Sec. D.

PENEX is Implicit AdaBoost. We now proceed to demonstrate formally that gradient descent on PENEX is analogous to AdaBoost in that gradient descent steps on a neural network parameterization f_θ with parameters θ can be thought of as constrained parameters of weak learners that minimize the exponential loss. To see this, we consider a linearization about the previous parameters θ_{m-1}, which assumes additivity as an approximation:

$$f_{\theta_m}(\mathbf{x}) \;\approx\; f_{\theta_{m-1}}(\mathbf{x}) \;+\; \eta \mathbf{J}(\mathbf{x}) \Delta\theta_m, \tag{9}$$

where $\mathbf{J}(\mathbf{x})$ denotes the Jacobian of $f_{\theta_{m-1}}$ with respect to θ_{m-1} at $\mathbf{x}$. This approximation (9) gives rise to an optimization problem closely related to that of AdaBoost (8), given as

$$\Delta\theta_m(\eta) \;\leftarrow\; \arg\min_{\Delta\theta : g_{\Delta\theta} \in \mathcal{G}_{\text{PENEX}}} \mathcal{L}_{\text{EX}}\left(f_{\theta_{m-1}} + \eta g_{\Delta\theta};\, \alpha\right), \tag{10}$$

where $\mathcal{G}_{\text{PENEX}}$ is the set of functions given in Sec. B.2. The following proposition shows that a small gradient descent step on PENEX indeed does solve (10):

Proposition A.1. *In the limit $\eta \to 0$, for some $\mathcal{G}_{\text{PENEX}}$ and $\rho > 0$, the solution of (10) converges to the negative rescaled gradient of PENEX. That is,*

$$\lim_{\eta \to 0} \Delta\theta_m(\eta) \;\propto\; -\nabla_\theta \mathcal{L}_{\text{PENEX}}(\theta_{m-1};\, \alpha).$$

A proof for Prop. A.1 is provided in Sec. B.2.

B Proofs

B.1 Proof for Prop. 2.1

Proof. The population-level equivalent of (3), conditionally on $\mathbf{x}$, is

$$\mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}(f(\mathbf{x})) \;:=\; \mathbb{E}_{y|\mathbf{x}}\Big[\exp\Big\{-\alpha f^{(y)}(\mathbf{x})\Big\}\Big] \;+\; \rho \sum_{j=1}^{K} \exp\Big\{f^{(j)}(\mathbf{x})\Big\}. \tag{11}$$

We note that (11) is strictly convex in $f(\mathbf{x})$ and differentiable. Hence, the unique minimum of (11), called $f_*(\mathbf{x})$, is characterized by its gradient being zero:

$$\frac{\partial \mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}}{\partial f_*^{(y)}(\mathbf{x})} \;=\; 0, \tag{12}$$

[3] Also known as *stagewise additive modeling using a multi-class exponential loss function* (SAMME).

which leads to

$$P(y \mid \mathbf{x}) \;=\; \frac{\rho}{\alpha} \exp\Big\{(1+\alpha) f_*^{(y)}(\mathbf{x})\Big\}.$$

Due to re-normalization, the factor ρ/α is canceled out. Hence, for every $\mathbf{x}$ that has non-zero support, the true conditional distribution $P(y \mid \mathbf{x})$ is recovered. □

B.2 Proof for Prop. A.1

We choose the set of functions to be

$$\mathcal{G}_{\text{PENEX}} \;\coloneqq\; \left\{ \mathbf{J}(\mathbf{x})\Delta\theta \;\middle|\; \hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\Big\{ f^{(j)}_{\theta_{m-1}}(\mathbf{x}) + \eta(\mathbf{J}(\mathbf{x})\Delta\theta_m)^{(j)} \Big\}\Big] \le \epsilon,\; \|\Delta\theta\|_2 = 1 \right\}, \quad (13)$$

where $\epsilon \coloneqq \hat{\mathbb{E}}[\sum_{j=1}^{K} \exp\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\}]$.

Proof. We start by performing a Taylor approximation of the objective function and the constraints around $\eta = 0$ in (10). This yields

$$\hat{\mathbb{E}}\left[\exp\left\{-\alpha f^{(y)}_{\theta_{m-1}}(\mathbf{x})\right\}\right] - \alpha\eta\hat{\mathbb{E}}\left[\exp\left\{-\alpha f^{(y)}_{\theta_{m-1}}(\mathbf{x})\right\}\mathbf{J}(\mathbf{x})^{(y)}\right]\Delta\theta + h_1(\eta,\Delta\theta)\eta$$

for the objective function and

$$\hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\Big\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\Big\}\Big] + \eta\hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\Big\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\Big\}\mathbf{J}(\mathbf{x})^{(j)}\Big]\Delta\theta + h_2(\eta,\Delta\theta)\eta \le \varepsilon$$

for the constraints, where $h_1(\eta,\Delta\theta)$, $h_2(\eta,\Delta\theta)$ are continuous functions that satisfy $\lim_{\eta\to 0} h_1(\eta,\Delta\theta) = 0$ and $\lim_{\eta\to 0} h_2(\eta,\Delta\theta) = 0$, and where, by assumption,

$$\epsilon = \hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\}\Big].$$

In addition, we used the notation $\mathbf{J}(\mathbf{x})^{(j)}$ to refer to the jth row of the Jacobian matrix $\mathbf{J}(\mathbf{x})$. The optimization in (10) can therefore be rewritten as

$$\begin{aligned} v(\eta) \quad \coloneqq \quad & \min_{\Delta\theta} -\alpha\hat{\mathbb{E}}\left[\exp\left\{-\alpha f^{(y)}_{\theta_{m-1}}(\mathbf{x})\right\}\mathbf{J}(\mathbf{x})^{(y)}\right]\Delta\theta + h_1(\eta,\Delta\theta) \\ & \text{subject to} \quad \hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\Big\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\Big\}\mathbf{J}(\mathbf{x})^{(j)}\Big]\Delta\theta + h_2(\eta,\Delta\theta) \le 0 \\ & \qquad\quad \|\Delta\theta\|_2^2 = 1. \end{aligned} \quad (14)$$

For small η, the first constraint reduces to a half-space passing through the origin. This implies that the constraint can be strictly satisfied for $\eta = 0$ and therefore, due to continuity, the feasible set in (10) (or equivalently (14)) is nonempty for small enough η. We further note that $v(\eta)$, the minimum value in (14), is continuous for small η, see for example (Still, 2018, Ch. 5) and consider any sequence $\eta_k > 0$ and corresponding $\Delta\theta_{mk}$, $k = 1, 2, \ldots$ where $\eta_k \to 0$ and $\Delta\theta_{mk}$ belongs to the corresponding set of minimizers of (10). Our goal is to show that any limit point of the sequence belongs to the set of minimizer of (9) with $\eta = 0$. To that extend, we first note that $\Delta\theta_{mk}$ remains bounded, since $\|\Delta\theta_{mk}\|_2^2 = 1$. Consider any limit point $\Delta\theta_\infty$ of $\Delta\theta_{mk}$ and pass to a subsequence $\Delta\theta_{mk}$ such that $\Delta\theta_{mk} \to \Delta\theta_\infty$. We now show that $\Delta\theta_\infty$ is a minimizer of (10) (or equivalently (14)) for $\eta = 0$. The limit point $\Delta\theta_\infty$ satisfies $\|\Delta\theta_\infty\|_2^2 = 1$ and the first constraint

$$\hat{\mathbb{E}}\Big[\sum_{j=1}^{K} \exp\Big\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\Big\}\mathbf{J}(\mathbf{x})^{(j)}\Big]\Delta\theta_\infty \le 0,$$

and is therefore feasible for the minimization in (14) when $\eta = 0$. We conclude from the continuity of v that $v(\eta_k) \to v(0)$ and therefore $\Delta\theta_\infty$ is a minimizer of (10) for $\eta = 0$. Hence the entire

sequence $\Delta\theta_{mk}$ converges to the set of minimizers of (10) for $\eta = 0$. As a result, $\Delta\theta_\infty$ satisfies the Karush-Kuhn-Tucker conditions corresponding to (14) for $\eta = 0$. These can be expressed as

$$\Delta\theta_\infty \propto \alpha\hat{\mathbb{E}}\Big[\sum_{j=1}^{K}\exp\left\{-\alpha f^{(j)}_{\theta_{m-1}}(\mathbf{x})\right\}\mathbf{J}(\mathbf{x})^{(j)}\Big] - \bar{\rho}\hat{\mathbb{E}}\Big[\sum_{j=1}^{K}\exp\left\{f^{(j)}_{\theta_{m-1}}(\mathbf{x})\right\}\mathbf{J}(\mathbf{x})^{(j)}\Big],$$

where $\bar{\rho} \geq 0$ is related to a multiplier of the first constraint in (14). This means that $\Delta\theta_\infty \propto -\nabla_\theta\mathcal{L}_{\text{PENEX}}(\theta_{m-1};\alpha)$ with $\rho = \bar{\rho}$, which concludes the proof.

B.3 Proof for Thm. 2.1

We first apply a Chernoff bound to see that

$$\mathbb{P}(m_f(\mathbf{x},y) \leq \gamma) \;\leq\; \mathbb{E}\left[\exp\left\{\beta\left(\gamma + \max_{j\neq y} f^{(j)}(\mathbf{x}) - f^{(y)}(\mathbf{x})\right)\right\}\right],$$

for any $\beta \in (0,1)$. Then, we note that for any $(\mathbf{x},y)$

$$\max_{j\neq y} f^{(j)}(\mathbf{x}) \;\leq\; \log\left(\sum_{j=1}^{K}\exp\left\{f^{(j)}(\mathbf{x})\right\}\right),$$

which we combine with Hölder's inequality for $\alpha \coloneqq \frac{\beta}{1-\beta}$[4] to see that

$$\mathbb{P}(m_f(\mathbf{x},y) \leq \gamma) \;\leq\; e^{\beta\gamma}\mathbb{E}[\mathcal{L}_{\text{EX}}(f;\alpha)]^{1-\beta}\cdot\mathbb{E}[\text{SE}(f)]^{\beta}.$$

We can now apply the weighted AM-GM inequality to get

$$\mathbb{E}[\mathcal{L}_{\text{EX}}(f;\alpha)]^{1-\beta}\cdot(\rho\mathbb{E}[\text{SE}(f)])^{\beta}\rho^{-\beta} \;\leq\; ((1-\beta)\mathbb{E}[\mathcal{L}_{\text{EX}}(f;\alpha)] + \beta\rho\mathbb{E}[\text{SE}(f)])\rho^{-\beta} \tag{15}$$

and further, because $\beta \in (0,1)$,

$$(1-\beta)\mathbb{E}[\mathcal{L}_{\text{EX}}(f;\alpha)] + \beta\rho\mathbb{E}[\text{SE}(f)] \;\leq\; \mathbb{E}[\mathcal{L}_{\text{EX}}(f;\alpha)] + \rho\mathbb{E}[\text{SE}(f)] \;=\; \mathbb{E}[\mathcal{L}_{\text{PENEX}}(f;\alpha,\rho)].$$

Thus, we have

$$\mathbb{P}(m_f(\mathbf{x},y) \leq \gamma) \;\leq\; e^{\beta\gamma}\rho^{-\beta}\mathbb{E}[\mathcal{L}_{\text{PENEX}}(f;\alpha,\rho)].$$

The result follows from replacing β by $\beta = \frac{\alpha}{\alpha+1}$. □

C Differences Between Thm. 2.1 and the Result by Schapire et al. (1998)

The earlier work by Schapire et al. (1998) shows that AdaBoost effectively increases margins. We recapitulate the Theorem 5 of Schapire et al. (1998), using the notation from the present work:

Theorem C.1 (Theorem 5 of Schapire et al. (1998))**.** *Consider a binary classification problem ($K = 2$) with $f^{(2)}(\mathbf{x}) \equiv -f^{(1)}(\mathbf{x})$ and assume that $f^{(1)}(\mathbf{x}) = \sum_{m=1}^{M}\eta_m g_m(\mathbf{x})$ with weighted training errors*

$$\epsilon_m \;\coloneqq\; \mathbb{P}_{(\mathbf{x}_i,y_i)\sim\hat{P}_m}(\mathbb{1}\{f^{(y_i)}(\mathbf{x}_i) < 0\}),$$

where the probability mass of the weighted empirical distribution $\hat{P}_m$ is given as

$$\hat{P}_m(\mathbf{x}_i,y_i) \;\coloneqq\; \frac{\hat{P}_{m-1}(\mathbf{x}_i,y_i)\exp\{-\eta_m f_m^{(y_i)}(\mathbf{x}_i)\}}{Z_m} \tag{16}$$

and

$$\eta_m \coloneqq \frac{1}{2}\log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) \tag{17}$$

with uniform initialization $\hat{P}_1 = \mathcal{U}([K])$. Then, for any $\gamma > 0$, it holds that

$$\hat{\mathbb{P}}(f_M^{(y)}(\mathbf{x}) \leq \gamma) \leq 2^M\prod_{m=1}^{M}\sqrt{\epsilon_m^{1-\gamma}(1-\epsilon_m)^{1+\gamma}},$$

for f_M fitted via binary AdaBoost (Freund & Schapire, 1995).

[4] We recall that $\beta \in (0,1)$, so $\frac{1}{\beta} \geq 1$ and $\frac{1}{1-\beta} \geq 1$, which is a condition for Hölder's inequality.

A generalization of Thm. C.1 to multi-class AdaBoost has been provided by Wang (2013). These earlier results differ from our result (Thm. 2.1) in three fundamental ways[5]:

1. Thm. C.1 makes a strong structural assumption about the classifier and provides a bound in terms of the weighted training errors of the individual "intermediate learners" f_m. Our result, in contrast, makes no assumptions about f.
2. Thm. C.1 assumes that f is fitted via the AdaBoost algorithm. Thm. 2.1 makes no assumption about how f is trained.
3. We consider a new formulation of the exponential loss that differs from the standard exponential loss in that the class scores/logits $f^{(y)}$ do not need to sum up to zero (concretely, the $f^{(2)}(\mathbf{x}) \equiv -f^{(1)}(\mathbf{x})$ assumption in Thm. C.1). Instead, PENEX incorporates a penalty to prevent the logits from diverging (see (3)). The margin result from Thm. C.1 cannot be directly translated to the PENEX formulation.

D Stagewise Additive Modeling using a Multi-class Exponential loss

The standard multi-class AdaBoost solves (8) via a specific algorithm called SAMME, shown in Alg. 2.

Algorithm 2: SAMME (Multi-class AdaBoost)

Input: Training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i \in \mathcal{X}$ and $y_i \in [K]$
Output: Strong classifier f_M

Initialize sample weights $w_1(i) \leftarrow \frac{1}{n}$ for $i = 1, \dots, n$
for $m = 1$ **to** M **do**
  Fit weak classifier g_m using weights w_m
  Compute weighted error $\epsilon_m \leftarrow \sum_{i=1}^n w_m(i) \cdot \mathbb{1}\{g_m(\mathbf{x}_i) \neq y_i\}$
  Compute classifier weight $\eta_m \leftarrow \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) + \log(K-1)$
  Update sample weights:

$$w_{m+1}(i) \leftarrow \frac{w_m(i)\exp(-\eta_m \mathbb{1}\{g_m(\mathbf{x}_i) \neq y_i\})}{Z_m},$$

  where Z_m is the partition function
end
Combine weak classifiers: $f_M(\mathbf{x}) = \arg\max_{k\in[K]} \sum_{m=1}^M \eta_m g_m(\mathbf{x})$

E Practical Implementation of PENEX

E.1 Inference

Motivated by Prop. 2.1, we rescale logits that are optimized on PENEX by $1+\alpha$. Specifically, to obtain a prediction $\hat{P}_{\text{PENEX}}$, we always use

$$\hat{P}_{\text{PENEX}}(y \,|\, \mathbf{x}) \;:=\; \frac{\exp\{(1+\alpha) f^{(y)}(\mathbf{x})\}}{\sum_{j=1}^K \exp\{(1+\alpha) f^{(j)}(\mathbf{x})\}}, \tag{18}$$

where $f^{(j)}(\mathbf{x})$ are the logits obtained by optimizing $\mathcal{L}_{\text{PENEX}}$ (3).

E.2 Default Value for Sensitivity α

While the optimal choice of sensitivity α depends on various factors such as training set size, model architecture, amount of classes and label noise, we notice that the tuned value for α is typically close to

$$\alpha \;=\; 0.1. \tag{19}$$

[5] We only refer to Thm. C.1 from hereon, because the same arguments can be made for the results presented by Wang (2013)

If PENEX is used without tuning, we generally recommend (19) as the default choice. If, however, there is substantial label noise, we recommend a smaller value. If there is little label noise, α may be chosen larger, leading to more aggressive margin increase.

E.3 Default Values for Alg. 1

For all experiments conducted in the present work, we use the following hyper parameter configuration:

Parameter	Value
EMA factor β	0.1
Lower penalty bound $\rho_{\min}$	10^{-6}
Upper penalty bound $\rho_{\max}$	100.0

In practice, we observe in all conducted experiments that results typically do not differ much if these parameters are chosen differently.

E.4 PyTorch-like Code

As mentioned in Sec. 2.2, PENEX does not require a custom training loop in order to implement Alg. 1. A PyTorch-like loss module is displayed in Sec. E.4, with only 34 lines of code (including comments). It is easy to see that it can be integrated seamlessly into any standard training loop.

F Experimental Details

F.1 Libraries

For our implementation, we use `PyTorch Lightning` (Falcon & The PyTorch Lightning team, 2019), a framework built on top of `PyTorch` (Paszke et al., 2019). For all experiments involving language models, we use the `transformers`, `datasets` and `evaluate` Hugging Face libraries. For hyperparameter tuning, we use the `Optuna` library (Akiba et al., 2019). We use `SciencePlots` (Garrett, 2021) for appealing plots and `weights & biases` for development and analysis.

F.2 Dataset & Model Licenses

The licenses of each dataset and pretrained model used in the present work are listed in the following table:

Dataset	License
CIFAR-10	MIT
PathMNIST	CC BY 4.0
BBC News	Apache 2.0
RoBERTa	MIT
ImageNet	Non-commercial research/educational

```python
import torch
import torch.nn as nn


def exp_loss(logits, targets, alpha):
    """Exponential loss on the true-class score fy(x)."""
    y  = targets.long()
    fy = logits.gather(1, y.unsqueeze(1)).squeeze(1)
    return torch.exp(-alpha * fy)

class PENEX(nn.Module):
    """Ultra-minimal readable version."""
    def __init__(self, alpha=0.1, rho_min=1e-6, rho_max=100.0, ema=0.1):
        super().__init__()
        self.alpha   = alpha
        self.rho_min = rho_min
        self.rho_max = rho_max
        self.ema     = ema
        self.rho     = None  # running estimate of rho

    def forward(self, logits, targets):
        # Base loss
        base   = exp_loss(logits, targets, self.alpha)
        # Penalty term; uses torch.logsumexp() for numerical stability
        sumexp = torch.exp(torch.logsumexp(logits, dim=1))

        # Update rho without tracking gradients
        with torch.no_grad():
            est = self.alpha * (base.mean() / (sumexp.mean() + 1e-12))
            est = est.clamp(self.rho_min, self.rho_max)
            self.rho = est if self.rho is None else (1 - self.ema) * self.rho +
            ↪  self.ema * est

        loss = base + self.rho * sumexp
        return loss.mean()
```

Listing 1: Simplified PyTorch-like code for the PENEX loss.

F.3 Computational Resources

All experiments are run on a HTCondor cluster. All models are trained with a single GPU and CPU (except for ImageNet, which is trained with 4 GPUs and 32 CPUs), where we adjust the type of GPU for each experiment. Specifically, all experiments can be run on the following hardware:

Dataset	GPU
CIFAR-10	NVIDIA A100-SXM4-40GB
Noisy CIFAR-10	NVIDIA A100-SXM4-40GB
CIFAR-100	NVIDIA A100-SXM4-40GB
PathMNIST	NVIDIA A100-SXM4-40GB
BBC News	NVIDIA A100-SXM4-40GB
ImageNet	NVIDIA A100-SXM4-80GB

We note, however, that all listed results should be achievable on different hardware (as we do not measure properties like runtime). In addition, all experiments are run with a single worker (as recommended by PyTorch for the used setup).

The amount of CPU memory also differs between experiments:

Dataset	Memory (in megabytes)
CIFAR-10	8096
Noisy CIFAR-10	8096
CIFAR-100	16192
PathMNIST	8096
BBC News	16192
ImageNet	150000

We furthermore report the approximate runtime for a single run of an experiment (200 epochs):

Dataset	Approx. time per run (in minutes)
CIFAR-10 Training	60
CIFAR-100 Training	120
PathMNIST Training	18
BBC News Training	60
ImageNet Training	3600

F.4 Ablation Implementations

To take the zero-sum constraint in CONEX (2) into account, we first define the constraint

$$h(\mathbf{x};\, \theta) \coloneqq \mathbf{1}^\top f_\theta(\mathbf{x})$$

and see that

$$h(\mathbf{x};\, \theta) = 0,\ \forall \mathbf{x} \in \operatorname{supp}(P) \iff \mathbb{E}[h(\mathbf{x};\, \theta)^2] = 0.$$

Squared Penalty. We penalize deviations from the penalty constraint via a quadratic penalty. To this end, we construct the loss function

$$\mathcal{L}(\theta_m) \;=\; \mathcal{L}_{\text{EX}}(\theta_m) + \frac{\rho}{2}\Big(\hat{\mathbb{E}}\left[h(\mathbf{x};\, \theta_m)^2\right]\Big)^2,$$

the optimum of which converges to the constrained solution for $\rho \to \infty$ (for a fixed learning rate η). In practice, however, choosing large ρ leads to an ill-conditioned loss surface, which is why a trade-off must be made in the choice of ρ.

Augmented Lagrangian. The augmented Lagrangian method (Hestenes, 1969) considers an objective that arises from (2) by introducing a penalty term with parameter $\rho > 0$ and the dual variable $\lambda_m \in \mathbb{R}$:

$$\mathscr{L}(\theta_m) \;\coloneqq\; \mathcal{L}_{\text{EX}}(\theta_m) \;+\; \frac{\rho}{2}\Big(\hat{\mathbb{E}}\left[h(\mathbf{x};\, \theta_m)^2\right]\Big)^2 \;+\; \lambda_m\Big(\hat{\mathbb{E}}\left[h(\mathbf{x};\, \theta_m)^2\right]\Big).$$

The algorithm updates the primal (θ_m) and dual (λ_m) variables in an alternating fashion:

$$\theta_m \leftarrow \theta_{m-1} - \eta \nabla_\theta \mathscr{L}(\theta_{m-1}), \qquad \lambda_m \leftarrow \lambda_{m-1} \;+\; \frac{\rho}{\nu}\Big(\hat{\mathbb{E}}\left[h(\mathbf{x};\, \theta_{m-1})^2\right]\Big),$$

where $\nu > 0$ is an inverse scaling factor.

Note: In practice, computing $\hat{\mathbb{E}}$ would require loading the entire training set into memory. We thus resort to sub-sampling $\hat{\mathbb{E}}$ from batches.

F.5 Metrics

Accuracy. We measure the accuracy of a classifier f as

$$\text{ACC}(f) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}\Big\{y_i = \arg\max_j f^{(j)}(\mathbf{x}_i)\Big\}.$$

Expected Calibration Error. Formally, it is given as

$$\text{ECE}(f) \coloneqq \sum_{m=1}^{M} \frac{|B_m|}{n}\big|\text{ACC}(f;\, B_m) - \text{CONF}(f;\, B_m)\big|,$$

where B_m is the set of data points whose prediction confidence falls into the interval

$$I_m \coloneqq \Big(\frac{m-1}{M};\, \frac{m}{M}\Big]$$

and

$$\text{CONF}(f) \coloneqq \frac{1}{N}\sum_{i=1}^{N} \hat{P}(y_i \mid f(\mathbf{x}_i)).$$

We use the default $m = 15$, as implemented by the torchmetrics library.

Brier Score. We compute the Brier score as

$$\text{BRIER}(f) \coloneqq \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K}\Big(\hat{P}(y = j \mid f(\mathbf{x}_i)) - \mathbf{y}_i^{(j)}\Big)^2,$$

where

$$\mathbf{y}_i^{(j)} \coloneqq \begin{cases} 1, & \text{if } \; j = y_i \\ 0, & \text{else,} \end{cases}$$

is a one-hot encoding of the label.

F.6 Data Preprocessing

F.6.1 Computer Vision

We do not perform data augmentation for any computer vision task, except for ImageNet.

CIFAR-10/CIFAR-100/PathMNIST. We perform standard normalization for these datasets, with the following mean and standard deviation values

$$\mu = (0.5, 0.5, 0.5),\; \sigma = (0.5, 0.5, 0.5),$$

where we have three values for each parameter because standardization is applied per channel (there are three channels). For PathMNIST in particular, prior to standardization, we load the dataset at a resolution of 64×64 and further scale it down to 32×32.

ImageNet. We also perform normalization, but instead with the following (standard) values

$$\mu = (0.485, 0.456, 0.406),\; \sigma = (0.229, 0.224, 0.225).$$

In addition, as common with vision transformers, we apply the following data augmentations: images are first resized to 256 pixels along the shorter side, then randomly cropped to 224×224 pixels with a scale factor uniformly sampled between 0.8 and 1.0, followed by a random horizontal flip with probability 0.5.

F.6.2 Fine-Tuning RoBERTa

There exist five labels in total corresponding to the categories tech, business, sport, entertainment and politics. To prevent memory overload, the input length is limited to a maximum length of 512 tokens and text is clipped if this length is exceeded (this is fairly common in other works, e.g., Kladny et al. (2025)). An example text input for the label *business* is demonstrated in the following:

Input

```
china aviation seeks rescue deal scandal-hit jet fuel supplier
china aviation oil has offered to repay its creditors $220m
(117m) of the $550m it lost on trading in oil futures.  the
firm said it hoped to pay $100m now and another $120m over
eight years. with assets of $200m and liabilities totalling
$648m  it needs creditors  backing for the offer to avoid
going into bankruptcy. the trading scandal is the biggest to
hit singapore since the $1.2bn collapse of barings bank in
1995. chen jiulin  chief executive of china aviation oil (cao)
 was arrested by at changi airport by singapore police on 8
december. he was returning from china  where he had headed
when cao announced its trading debacle in late-november. the
firm had been betting heavily on a fall in the price of oil
during october  but prices rose sharply instead.  among the
creditors whose backing cao needs for its restructuring plan
are banking giants such as barclay s capital and sumitomo
mitsui  as well as south korean firm sk energy. of the
immediate payment  the firm - china s biggest jet fuel
supplier - said it would be paying $30m out of its own
resources. the rest would come from its parent company  china
aviation oil holding company in beijing. the holding company
owned by the chinese government  holds most of cao s
singapore-listed shares. it cut its holding from 75% to 60% on
20 october.
```

In addition to clipping, we sub-sample a training set of size 200 to enhance the difficulty of the classification task.

F.7 Train/Validation Splits

In our experiments, we always construct an additional validation set from the training set. We do this in order to ensure that the test task provides an unbiased estimate of the desired model scores. The split ratios are determined by the size of the training set (we choose a larger ratio for ImageNet, because this data set is very large) provided in the following table:

Method	Training/Validation Split Ratio
CIFAR-10	0.8
CIFAR-100	0.8
PathMNIST	0.8
ImageNet	0.9
BBC News	0.8

F.8 Architectures

We generally do not apply much regularization within the architectures (i.e., we do not use batch normalization) in order to better compare the effects of the used loss-based regularizers.

CIFAR-10/100 & PathMNIST (experiments from the main text). The architecture used is provided in the following table:

Layer Type	Parameters	Output Shape
Input	-	$3 \times 32 \times 32$
Conv2d	32 filters, 3×3, padding = 1	$32 \times 32 \times 32$
ReLU	-	$32 \times 32 \times 32$
MaxPool2d	2×2, stride = 2	$32 \times 16 \times 16$
Conv2d	64 filters, 3×3, padding = 1	$64 \times 16 \times 16$
ReLU	-	$64 \times 16 \times 16$
MaxPool2d	2×2, stride = 2	$64 \times 8 \times 8$
Conv2d	128 filters, 3×3, padding = 1	$128 \times 8 \times 8$
ReLU	-	$128 \times 8 \times 8$
MaxPool2d	2×2, stride = 2	$128 \times c \times c$
Flatten	-	2048
Dropout	p = 0.5	2048
Linear	$2048 \rightarrow 256$	256
ReLU	-	256
Linear	$256 \rightarrow$ `n_out`	`n_out`

In the final layer, `n_out` $= 10$, `n_out` $= 100$ or `n_out` $= 9$, respectively. We do so for all methods except for ablation 3) (see Sec. 4.1), where `n_out` $= 9$, `n_out` $= 99$ or `n_out` $= 8$, respectively. Furthermore, we note that $c = 4$ for CIFAR-10/100 (because the images have shape $3 \times 32 \times 32$) and $c = 8$ for PathMNIST (images have shape $3 \times 64 \times 64$).

CIFAR-10 & CIFAR-100 (additional ResNet experiment in Sec. G.3).

Layer Type	Parameters	Output Shape
Input	–	$3 \times 32 \times 32$
Conv2d + BN + ReLU	64, 3×3, s=1, p=1	$64 \times 32 \times 32$
Layer 1: BasicBlock$\times 2$	3×3 convs, stride 1	$64 \times 32 \times 32$
Layer 2: BasicBlock$\times 2$	first block: s=2; shortcut 1×1 second block: stride 1	$128 \times 16 \times 16$ $128 \times 16 \times 16$
Layer 3: BasicBlock$\times 2$	first block: s=2; shortcut 1×1 second block: stride 1	$256 \times 8 \times 8$ $256 \times 8 \times 8$
Layer 4: BasicBlock$\times 2$	first block: s=2; shortcut 1×1 second block: stride 1	$512 \times 4 \times 4$ $512 \times 4 \times 4$
AvgPool2d	4×4 global pool	$512 \times 1 \times 1$
Flatten	–	512
Linear	$512 \rightarrow 10$	10

ImageNet. The architecture used corresponds to ViT-Base (Dosovitskiy et al., 2021) and is provided in the following table:

Layer Type	**Parameters**	**Output Shape**
Input	-	$3 \times 224 \times 224$
Patch Embedding	Patch size 16×16, embedding dim 768	196×768
Add CLS token	-	197×768
Add Positional Encoding	-	197×768
Dropout	p = 0.1	197×768
Transformer Encoder (12 blocks)		
Multi-Head Self-Attention	heads = 12, dim_head = 64	197×768
Add & Layer Norm	-	197×768
MLP	hidden dim = 3072, dropout = 0.1	197×768
Add & Layer Norm (repeated 12 times)	-	197×768
Pooling	CLS token (pool = 'cls')	768
MLP Head	Linear $768 \to$ `n_out`	`n_out`

In the final layer, `n_out` $= 1000$.

F.9 Training

For all methods, we use either the Adam optimizer (Kingma & Ba, 2015) or the AdamW optimizer (Loshchilov & Hutter, 2019) with default parameters. This means that both optimizer are used with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. All parameters are given in the following table:

Method	Learning Rate	Optimizer	# Training Epochs	Batch Size (effective)	Learning Rate Scheduling	Gradient Clip Value	# Gradient Acc. Steps	Precision
CIFAR-10	$1 \cdot 10^{-4}$	Adam	200	64	constant	5.0	1	medium
CIFAR-10 (Sec. G.3)	$1 \cdot 10^{-1}$	SGD	200	128	cosine	5.0	1	medium
CIFAR-100	$1 \cdot 10^{-4}$	Adam	200	64	constant	15.0	1	medium
PathMNIST	$1 \cdot 10^{-4}$	Adam	200	64	constant	10.0	1	medium
ImageNet	$1 \cdot 10^{-5}$ for PENEX and $1 \cdot 10^{-3}$ for others	Adam	300	2048	cosine & 10 epochs warmup	1.0	1	medium
BBC News	$5 \cdot 10^{-6}$	AdamW	200	8	linear (no warmup)	5.0	2	high

F.10 Hyperparameter Tuning

For each method, we use the tree-structured Parzen estimator with 50 trials per method, using the standard parameters from the `Optuna` library. We list the parameter ranges for hyperparameter tuning that were used for each baseline:

Method	Hyperparameter	Lower Bound	Upper Bound
PENEX (CIFAR-10/100, PathMNIST)	Sensitivity α	0	1
PENEX (BBC News)	Sensitivity α	0	3
Label Smoothing	Smoothing Parameter ϵ	0	1
Confidence Penalty	Penalty Parameter λ	0	10
Focal Loss	Focusing parameter γ	0	5
CONEX + Squared Penalty	Penalty Parameter ρ	0	10^3
CONEX + AL	Penalty Parameter ρ	0	10^3
	Inverse Scaling Factor ν	1	10^3

We note that we do not tune the learning rate, but fix it to a fixed value per experiment. The reason is that it would allow methods that tend to overfit to delay their overfitting beyond the considered range of tuning epochs instead of alleviating overfitting. For each experiment and method, the amount of epochs per tuning run is identical to the training horizon for the reported validation and test results.

F.11 Hyperparameters for ImageNet

For reasons of computational requirements, we use all methods with their default parameters, as specified in the original works (see Sec. 4.1 for reference). Specifically, we use the following values:

Method	Parameter Setting
PENEX	Sensitivity $\alpha = 0.01$
Label Smoothing	Smoothing Parameter $\epsilon = 0.1$ (see Szegedy et al. (2016))
Focal Loss	Focusing Parameter $\gamma = 2$ (see Lin et al. (2018))
Confidence Penalty	Penalty Parameter $\lambda = 0.1$ (see Pereyra et al. (2017)); we use the reported value for CIFAR-10 because no ImageNet experiment is conducted

G Additional Experiments

G.1 Runtime Comparison

To quantify the computational overhead of PENEX relative to standard objectives, we benchmark the wall-clock time of a single forward loss evaluation for several classification losses in PyTorch. We consider a $K = 10$ class problem and generate synthetic logits $\mathbf{z} \in \mathbb{R}^{64\times 10}$ and labels $y \in \{0, \dots, 9\}^{64}$, corresponding to a batch size of 64. For each method we draw $N = 100{,}000$ independent batches of logits and labels from a fixed random seed and measure the time required for a single loss computation using `time.time()`. The results can be seen in the following table and show that PENEX even tends to be slightly faster to evaluate than other regularization techniques:

Method	Avg. time [s]	Std [s]
CE	1.9×10^{-5}	5×10^{-6}
label smoothing	7.4×10^{-5}	5×10^{-6}
confidence penalty	2.0×10^{-4}	6×10^{-6}
focal loss	1.07×10^{-4}	5×10^{-6}
PENEX	6.8×10^{-5}	7×10^{-6}

G.2 All Validation Results

The additional plots shown in Fig. 6 support that PENEX performs at least as good as all considered baseline regularizers and typically performs even better. The only clear exception is the ImageNet experiment, where we can observe difficulties in training stability, requiring a smaller learning rate (see Sec. F.9) and leading to worse generalization as a consequence.

G.3 Alternative Model Architecture

To demonstrate that PENEX does not only work well for a single architecture, we generate test results for CIFAR-10 and CIFAR-100 with the ResNet-18 architecture (He et al., 2016) described in Sec. F.8, with the training procedure described in Sec. F.9. We use the default parameters for each method and additional weight decay of $5 \cdot 10^{-4}$ and momentum of 0.9, for all methods. The results are provided in the following table:

Metric	CE	label smoothing	confidence penalty	focal loss	PENEX
			CIFAR-10		
ACC	0.947 ± 0.002	$\mathbf{0.949 \pm 0.002}$	0.947 ± 0.002	0.945 ± 0.002	0.948 ± 0.002
$-$ECE	-0.042 ± 0.001	-0.096 ± 0.001	-0.039 ± 0.002	-0.094 ± 0.001	$\mathbf{-0.037 \pm 0.001}$
$-$CE	-0.197 ± 0.009	-0.259 ± 0.007	$\mathbf{-0.181 \pm 0.007}$	-0.223 ± 0.004	-0.193 ± 0.008
$-$BRIER	-0.084 ± 0.003	-0.088 ± 0.003	$\mathbf{-0.082 \pm 0.003}$	-0.084 ± 0.002	-0.083 ± 0.003
			CIFAR-100		
ACC	0.571 ± 0.005	0.571 ± 0.005	0.549 ± 0.005	0.558 ± 0.005	$\mathbf{0.601 \pm 0.005}$
$-$ECE	-0.163 ± 0.003	$\mathbf{-0.107 \pm 0.003}$	-0.157 ± 0.004	-0.112 ± 0.003	-0.112 ± 0.002
$-$CE	-1.696 ± 0.023	-1.678 ± 0.019	-1.799 ± 0.024	-1.583 ± 0.017	$\mathbf{-1.511 \pm 0.017}$
$-$BRIER	-0.588 ± 0.006	-0.569 ± 0.006	-0.612 ± 0.007	-0.577 ± 0.005	$\mathbf{-0.532 \pm 0.005}$

G.4 Additional Noisy Label Experiments

To evaluate whether the robustness of PENEX extends beyond the noise setting considered in the main experiments Sec. 4.1, we conduct several additional experiments covering both synthetic and real-world label noise on CIFAR-10 with the main ConvNet implementation.

Setup. First, we consider uniform label noise at two additional noise levels, 20% and 40%. Second, we evaluate on real-world human annotation noise using the standardized CIFAR-10N benchmark introduced by Wei et al. (2022). Unlike synthetically corrupted labels, these labels arise from actual human annotations and therefore provide a complementary evaluation under more realistic noise patterns. Finally, we consider class-asymmetric label noise on CIFAR-10. Following a fixed

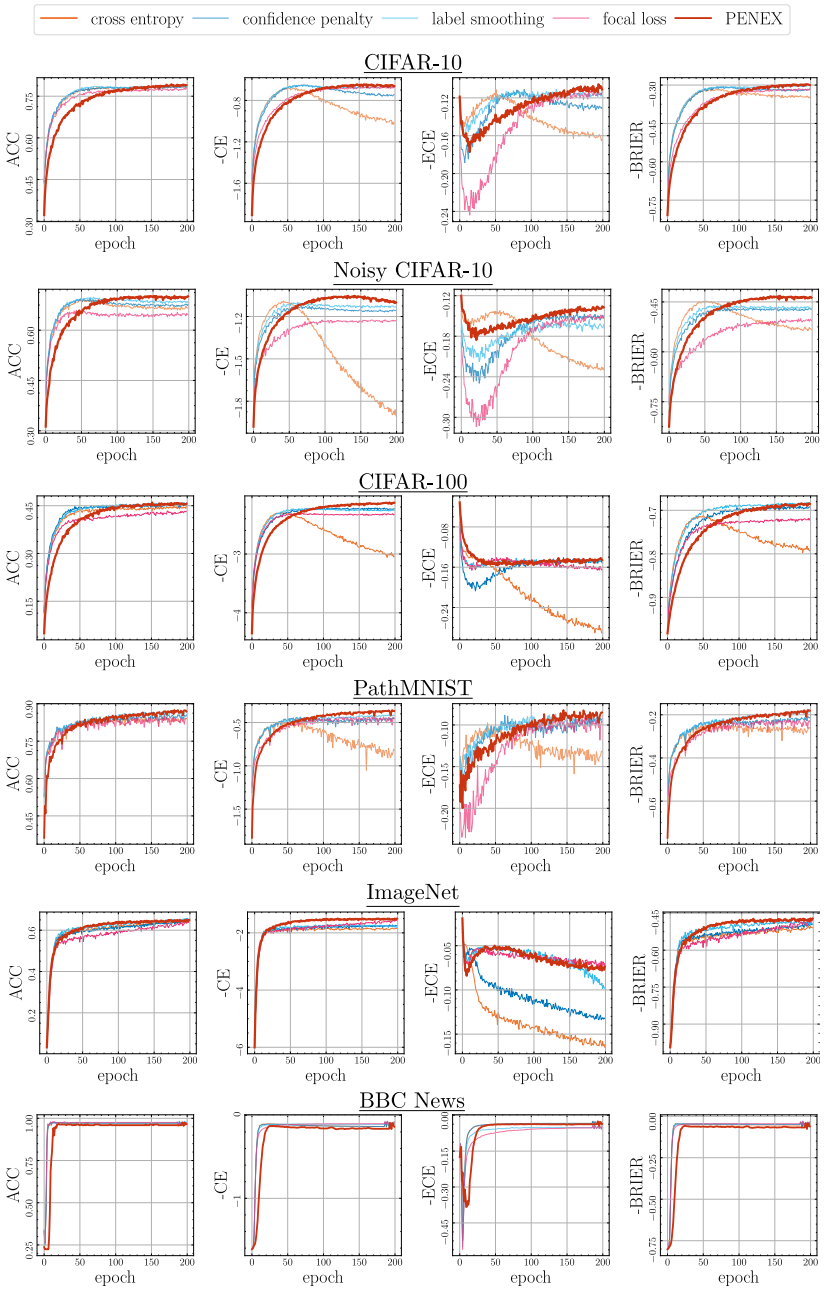

Figure 6: **All Validation Curves.** Larger means better. Validation curves over 200 epochs for all experiments, similar to Fig. 4.

class-dependent corruption scheme, bird labels are changed to airplane, truck to automobile, cat to dog, dog to cat, and deer to horse, each with probability given by the noise rate, while the remaining classes are left unchanged. This setting introduces structured, class-dependent label corruption and therefore differs substantially from uniform random noise.

Metric	CE	label smoothing	confidence penalty	focal loss	PENEX
			20% Uniform Label Noise		
ACC	0.682 ± 0.004	0.697 ± 0.005	0.685 ± 0.004	0.664 ± 0.005	$\mathbf{0.702 \pm 0.004}$
$-$ECE	-0.181 ± 0.003	-0.185 ± 0.003	$\mathbf{-0.151 \pm 0.002}$	-0.169 ± 0.003	-0.160 ± 0.003
$-$CE	-1.208 ± 0.020	-1.026 ± 0.009	-1.011 ± 0.009	-1.061 ± 0.009	$\mathbf{-0.916 \pm 0.009}$
$-$BRIER	-0.472 ± 0.006	-0.444 ± 0.004	-0.446 ± 0.004	-0.484 ± 0.004	$\mathbf{-0.417 \pm 0.004}$
			40% Uniform Label Noise		
ACC	0.569 ± 0.004	0.590 ± 0.005	0.563 ± 0.005	0.565 ± 0.005	$\mathbf{0.605 \pm 0.005}$
$-$ECE	-0.197 ± 0.003	-0.171 ± 0.003	$\mathbf{-0.155 \pm 0.003}$	-0.171 ± 0.004	-0.177 ± 0.003
$-$CE	-1.529 ± 0.019	-1.338 ± 0.009	-1.389 ± 0.011	-1.371 ± 0.009	$\mathbf{-1.245 \pm 0.009}$
$-$BRIER	-0.606 ± 0.006	-0.573 ± 0.004	-0.593 ± 0.004	-0.605 ± 0.004	$\mathbf{-0.552 \pm 0.004}$
			Real-World Human Annotation Noise		
ACC	0.697 ± 0.005	$\mathbf{0.717 \pm 0.004}$	0.703 ± 0.004	0.678 ± 0.004	$\mathbf{0.717 \pm 0.005}$
$-$ECE	-0.192 ± 0.003	-0.179 ± 0.003	-0.149 ± 0.003	-0.160 ± 0.002	$\mathbf{-0.135 \pm 0.002}$
$-$CE	-1.229 ± 0.022	-0.959 ± 0.008	-0.941 ± 0.010	-0.955 ± 0.008	$\mathbf{-0.842 \pm 0.010}$
$-$BRIER	-0.464 ± 0.007	-0.418 ± 0.003	-0.419 ± 0.004	-0.455 ± 0.004	$\mathbf{-0.396 \pm 0.004}$
			10% Asymmetric Noise		
ACC	0.759 ± 0.004	$\mathbf{0.767 \pm 0.004}$	0.759 ± 0.004	0.734 ± 0.005	0.765 ± 0.004
$-$ECE	-0.172 ± 0.003	-0.185 ± 0.003	-0.156 ± 0.002	-0.142 ± 0.003	$\mathbf{-0.120 \pm 0.002}$
$-$CE	-1.087 ± 0.023	-0.813 ± 0.007	-0.792 ± 0.008	-0.780 ± 0.009	$\mathbf{-0.721 \pm 0.012}$
$-$BRIER	-0.382 ± 0.006	-0.351 ± 0.004	-0.353 ± 0.004	-0.380 ± 0.004	$\mathbf{-0.329 \pm 0.004}$

Table 3: Robustness to different forms of label noise on CIFAR-10. We report results for 20% and 40% uniform label noise, real-world human annotation noise from CIFAR-10N, and 10% class-asymmetric label noise. Larger is better, best is bold.

Results. The results, provided in Tab. 3, show that PENEX is consistently robust across all of the tested noise settings, outperforming most baselines on most metrics. Under both 20% and 40% uniform label noise, PENEX achieves the highest accuracy as well as the best cross-entropy and Brier scores, while remaining competitive in terms of ECE. The same trend extends to real-world human annotation noise, where PENEX attains the joint-highest accuracy and the best ECE, cross-entropy, and Brier scores. Finally, under asymmetric label noise, PENEX again obtains the best ECE, cross-entropy, and Brier scores, while its accuracy remains close to the best-performing baseline.

G.5 Comparison with a Margin-Based Objective

Setup. To complement the comparisons against standard classification objectives, we additionally evaluate PENEX against mdNet (Lyu et al., 2022), a representative margin-based training objective designed to improve generalization by explicitly shaping the margin distribution. We evaluate all methods on CIFAR-10 using the same experimental protocol as in the main experiments. Cross-entropy (CE) is included as a reference baseline. The results are given in the following table:

Metric	CE	mdNet	PENEX
ACC	0.785 ± 0.004	0.786 ± 0.004	$\mathbf{0.793 \pm 0.004}$
$-$ECE	-0.162 ± 0.003	-0.419 ± 0.003	$\mathbf{-0.109 \pm 0.002}$
$-$CE	-1.004 ± 0.024	-1.196 ± 0.005	$\mathbf{-0.646 \pm 0.012}$
$-$BRIER	-0.346 ± 0.006	-0.525 ± 0.002	$\mathbf{-0.299 \pm 0.005}$

Table 4: Comparison with the margin-based objective mdNet on CIFAR-10. Cross-entropy (CE) is included as a reference. Larger is better, best is bold.

Results. As shown in Tab. 4, mdNet achieves accuracy comparable to standard cross-entropy training, but performs substantially worse on all probability-sensitive metrics when its logits are trans-

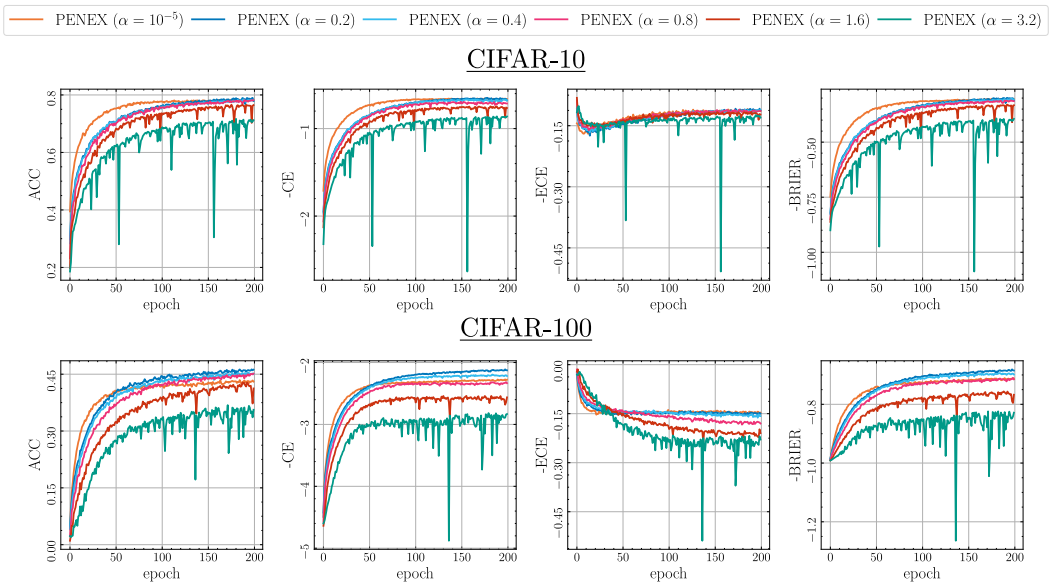

Figure 7: **Effects of the Sensitivity Parameter on Validation Curves.** Smaller α typically lead to faster convergence, but worse generalization toward the end of the training duration (as can be seen more clearly for CIFAR-100). In addition, for very large $\alpha = 3.2$, training curves tend to become less stable.

formed using the standard softmax. In contrast, PENEX achieves the best result across all four metrics, including both accuracy and calibration. This comparison highlights an important distinction between the two approaches. While mdNet operates directly on the classification margin, PENEX additionally admits a closed-form link between its population-optimal logits and the complete class-probability vector, as established in Prop. 2.1. Consequently, PENEX combines the generalization benefits associated with margin-based training with principled probability estimation.

G.6 Sensitivity Parameter Analysis

The experimental results demonstrated in Fig. 7 show how the sensitivity parameter α of PENEX affects all metrics considered in the present work, for six different values $\alpha \in \{10^{-5}, 0.2, 0.4, 0.8, 1.6, 3.2\}$. Fig. 7 shows that larger values of α lead to slower improvements initially, but often better generalization toward the end of training. In addition, for very large α, training curves become more unstable, likely due to steeper areas in the loss surface.

G.7 Different Penalties

Setup. To investigate whether the particular penalty formulation used by PENEX is important for its empirical performance, we compare the proposed SumExp penalty against two alternative formulations: a squared Euclidean penalty and a LogSumExp penalty. We evaluate all three variants using the same experimental protocol as in the main experiments. As throughout the paper, larger values are better for all reported metrics.

Metric	Squared Euclidean Penalty	LogSumExp Penalty	SumExp Penalty (PENEX)
ACC	0.765 ± 0.004	0.757 ± 0.004	$\mathbf{0.793 \pm 0.004}$
$-$ECE	-0.664 ± 0.004	-0.127 ± 0.002	$\mathbf{-0.109 \pm 0.002}$
$-$CE	-2.294 ± 0.000	-0.709 ± 0.010	$\mathbf{-0.646 \pm 0.012}$
$-$BRIER	-0.898 ± 0.000	-0.343 ± 0.005	$\mathbf{-0.299 \pm 0.005}$

Table 5: Comparison of different penalty formulations. The SumExp penalty used by PENEX outperforms the squared Euclidean and LogSumExp alternatives across all reported metrics. Larger is better, best is bold.

Results. As shown in Tab. 5, the alternative penalties yield viable classifiers, but neither matches the performance of the SumExp penalty used by PENEX. In particular, the LogSumExp penalty performs considerably better than the squared Euclidean alternative on the probability-sensitive metrics, yet remains consistently behind PENEX in accuracy, ECE, cross-entropy, and Brier score. The squared Euclidean penalty performs especially poorly in terms of calibration and probabilistic prediction quality. These results indicate that the empirical gains of PENEX are not merely a consequence of augmenting the exponential loss with an arbitrary logit penalty, but depend on the particular SumExp formulation. Moreover, the theoretical properties established for PENEX, including its self-calibration and margin-tail results, do not readily extend to these alternative penalties.

G.8 Fixed Penalty Parameter

Setup. In the main experiments, we estimate ρ automatically rather than treating it as an additional hyperparameter (see Alg. 1). To assess the cost of this design choice, we compare the default version of PENEX against a variant in which ρ is fixed and tuned using the validation set. This comparison isolates whether explicitly tuning ρ can provide additional performance beyond the automatic estimator used throughout the main experiments.

Metric	Default PENEX	Tuned fixed ρ
ACC	0.786 ± 0.004	$\mathbf{0.797 \pm 0.004}$
$-$ECE	-0.114 ± 0.002	$\mathbf{-0.108 \pm 0.002}$
$-$CE	-0.676 ± 0.014	$\mathbf{-0.623 \pm 0.013}$
$-$BRIER	-0.304 ± 0.005	$\mathbf{-0.288 \pm 0.005}$

Table 6: Comparison between the automatic estimator for ρ used by default in PENEX and a validation-tuned fixed ρ. Larger is better, best is bold.

Results. As shown in Tab. 6, explicitly tuning a fixed value of ρ yields better performance across all four metrics. This improvement, however, comes at the cost of introducing a second loss hyperparameter that must be selected using validation data. Since the main baselines considered in our experiments have only one tuned loss parameter, we use the automatic estimator for ρ in the main comparison and tune only the remaining loss parameter. The automatic estimator therefore reduces the tuning burden while retaining competitive performance, whereas validation-tuning ρ provides an additional performance gain when the extra tuning cost is acceptable.

G.9 Ablating EMA and Clipping Parameters

Setup. We additionally study the sensitivity of PENEX to two implementation choices used by the automatic estimator for penalty parameter ρ: clipping and exponential moving averaging (EMA). We compare the default configuration against a variant without clipping and a variant using $\beta = 0.01$. This ablation tests whether the performance of PENEX depends sensitively on either the clipping bounds or the EMA factor.

Metric	Default PENEX	No clipping	$\beta = 0.01$
ACC	0.786 ± 0.004	0.786 ± 0.004	0.785 ± 0.004
$-$ECE	-0.114 ± 0.002	-0.114 ± 0.002	-0.113 ± 0.002
$-$CE	-0.676 ± 0.014	-0.676 ± 0.014	-0.674 ± 0.013
$-$BRIER	-0.304 ± 0.005	-0.304 ± 0.005	-0.306 ± 0.005

Table 7: Ablation of clipping and the EMA factor used by the automatic ρ estimator. Performance remains nearly unchanged when clipping is removed or when the EMA factor is set to $\beta = 0.01$.

Results. The three configurations in Tab. 7 perform almost identically across all reported metrics. Removing clipping leaves the reported results unchanged, while changing the EMA factor to $\beta = 0.01$ produces only negligible differences. These results indicate that neither the clipping bounds

nor the EMA factor constitute sensitive hyperparameters of PENEX. In particular, the stability of the automatic ρ estimator does not appear to rely on careful tuning of these implementation choices.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: See abstract and Sec. 1.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: A discussion & limitations section (Sec. 5) is included.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: A proof for each theoretical result is supplied in Sec. B.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: In addition to source code, detailed descriptions for optimal reproducibility are supplied.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Source code is supplied. All used datasets are publicly available and referenced.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: A detailed description of all hyperparameters is supplied in Sec. F.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Standard deviations (and a description of how they are estimated) are supplied for all experimental results.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Information about computational resources is available in Sec. F.3.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: There are no conflicts with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: The contribution, which is a new loss function and its properties, is quite general and does not serve a specific purpose that would render such a discussion necessary.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: There is no high risk of misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All sources of both data and models are credited, according to their license and terms (see Sec. 4). Licenses are explicitly listed in Sec. F.2.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: No new assets are released.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing and no research with human subjects is involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: No crowdsourcing and no research with human subjects is involved.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.