

PENEX: AdaBoost-Inspired Neural Network Regularization

Klaus-Rudolf Kladny^{1,2}

Bernhard Schölkopf^{1,2,3}

Michael Muehlebach¹

¹ Max Planck Institute for Intelligent Systems, Tübingen, Germany

² Tübingen AI Center, Tübingen, Germany

³ ELLIS Institute Tübingen, Tübingen, Germany

Abstract

AdaBoost sequentially fits so-called weak learners to minimize an exponential loss, which penalizes mislabeled 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. We demonstrate both empirically and theoretically that PENEX implicitly maximizes margins of data points. Also, we show that gradient increments on PENEX implicitly parameterize weak learners in the boosting framework. Across computer vision and language tasks, we show that PENEX exhibits a *regularizing effect often better than established methods with similar computational cost*. Our results highlight PENEX’s potential as an AdaBoost-inspired alternative for effective training and fine-tuning of deep neural networks.

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. 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 maximization (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

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

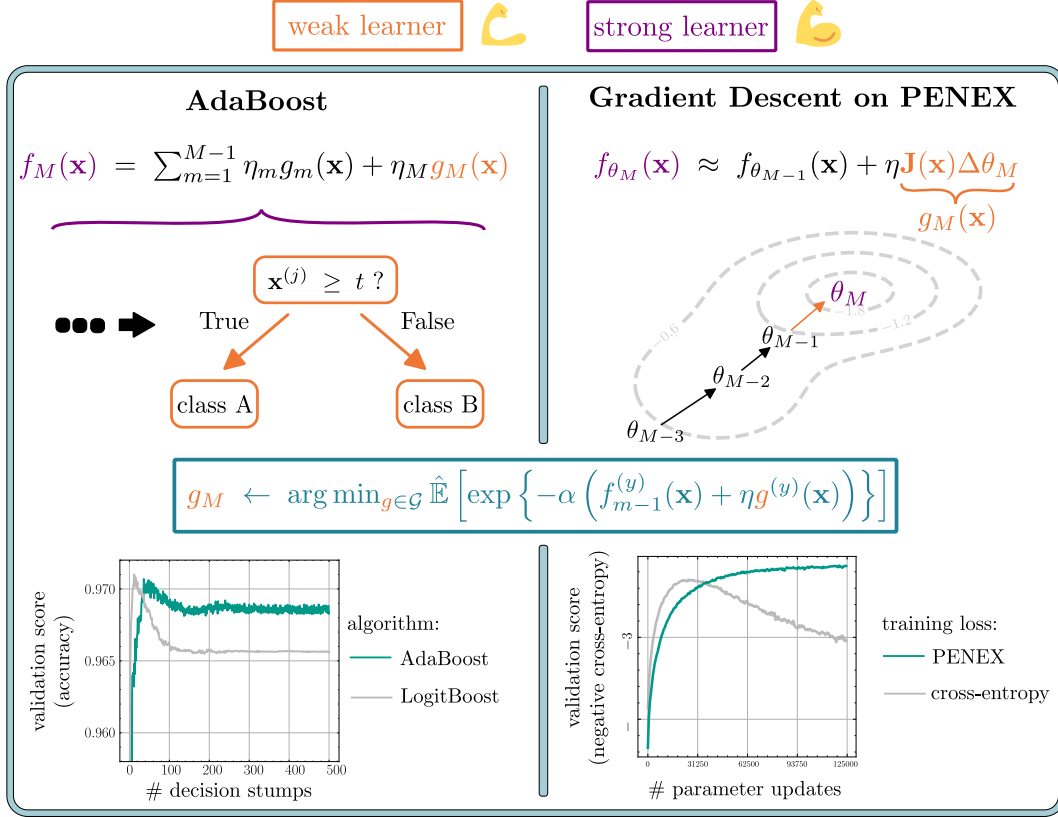


Figure 1: **Gradient Descent on PENEX as a Form of Implicit AdaBoost.** AdaBoost (left) builds a strong learner $f_M(\mathbf{x})$ (purple) by sequentially fitting weak learners such as decision stumps (orange) and linearly combining them. Gradient descent itself (right) can be thought of as an implicit form of boosting where weak learners correspond to $\mathbf{J}(\mathbf{x})\Delta\theta_m$ (orange), parameterized by parameter increments $\Delta\theta_m$. Combining many gradient descent steps can thus be interpreted as forming a strong learner $f_{\theta_M}(\mathbf{x})$ (green) as an approximate linear combination of weak learners. In both cases, each weak learner is obtained by minimizing the exponential loss within a fixed function class \mathcal{G} (more details in Sec. 2.3).

penalty on miss-classifications, this margin is maximized and thus, the model concentrates more on hard-to-learn examples, which play a similar role to support vectors (Cortes & Vapnik, 1995). This principle is demonstrated in Fig. 2. Maximizing the margin for many training data points, in turn, guarantees good generalization (e.g., Schapire et al. (1998)).

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 Fisher consistent (Fisher, 1922), i.e., it is a proper loss for estimating the Bayes-optimal classifier.

We also prove formally that PENEX effectively maximizes margins with high probability, thereby providing a theoretical foundation for PENEX’s effectivity. 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 that small gradient-based parameter increments implicitly parameterize a form of weak learner, as visualized in Fig. 1. 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 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:

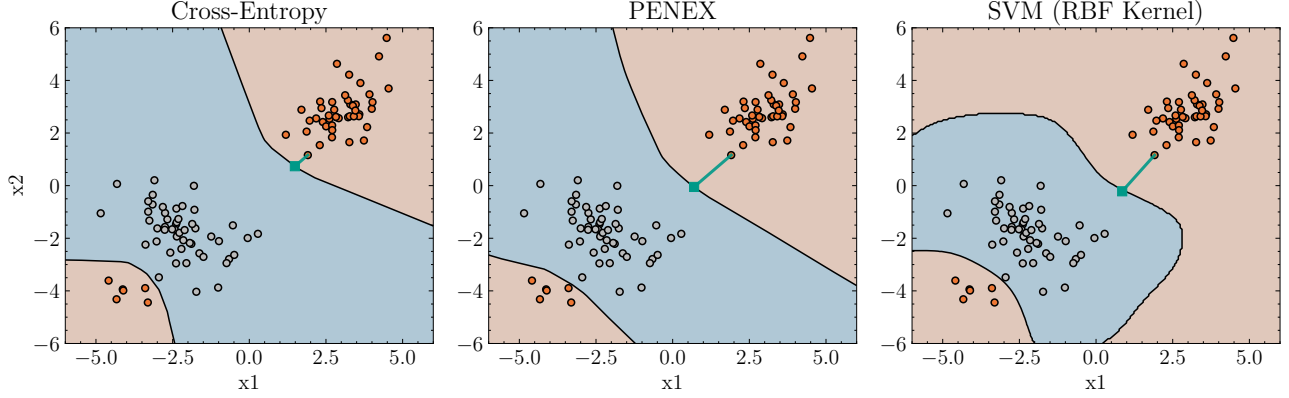


Figure 2: **Comparison of Margins.** Neural networks trained with PENEX (center) tend to implicitly maximize margins (here, geometric margins, indicated for an example point in green), in a similar way to support vector machines (right), here trained with a RBF kernel (Schölkopf & Smola, 2002, p. 46). Training neural networks 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 Fisher consistency (Prop. 2.1) and its margin maximization quality (Thm. 2.1).
- We demonstrate the relationship between performing gradient descent on PENEX and AdaBoost. Specifically, we can interpret the former as an implicit form of AdaBoost, where small parameter increments correspond to parameters of weak learners (Fig. 1; Sec. 2.3).
- We empirically compare the regularizing effect of PENEX with ablations and other regularizers, on diverse tasks ranging from computer vision to fine-tuning of a language model. Our results show that PENEX typically outperforms existing regularization techniques with similar computational cost, in low data regimes (Sec. 4).
- Our empirical results 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. Thereafter (Sec. 2.3), we provide insights into the relationship between minimizing PENEX via gradient descent and multi-class AdaBoost (Zhu et al., 2009). 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. Finally,

we discuss limitations and future work in Sec. 5 and conclude in Sec. 6.

2 Penalized Exponential Loss (PENEX)

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

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

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):

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

The intuition behind the constraint in (2) is that it acts as a barrier preventing the $f^{(j)}$ from diverging without bounds. 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 over $f^{(j)}$, specifically

$$\mathcal{L}_{\text{PENEX}}(f; \alpha, \rho) := \mathcal{L}_{\text{EX}}(f; \alpha) + \rho \hat{\mathbb{E}}[\text{SE}(\mathbf{x})], \quad (3)$$

where

$$\text{SE}(\mathbf{x}) := \sum_{j=1}^K \exp\{f^{(j)}(\mathbf{x})\}$$

is the SumExp and $\rho > 0$ 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 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 (Fisher Consistency). *For any $\rho > 0$ and $\alpha > 0$, the penalized exponential loss (3) is Fisher consistent, meaning that the minimizer of the population equivalent², called f_* , recovers the Bayes-optimal classifier in the sense that*

$$P(y | \mathbf{x}) \propto \exp \left\{ (1 + \alpha) f_*^{(y)}(\mathbf{x}) \right\},$$

i.e., the $f_*^{(y)}(\mathbf{x})$ recover the true logits (up to constant shift) at temperature $(1 + \alpha)^{-1}$.³

A proof for Prop. 2.1 is supplied in Sec. A.1. While other loss functions such as cross-entropy $\mathcal{L}_{\text{CE}}(f) := -\hat{\mathbb{E}}[\log(\hat{P}(y | \mathbf{x}))]$ with $\hat{P}(y | \mathbf{x}) \propto \exp\{f^{(y)}(\mathbf{x})\}$ are Fisher consistent as well, the solutions on finite datasets typically differ, as can be seen in Fig. 3: When the “correct” logit grows negative ($f^{(1)}(\mathbf{x}) \rightarrow -\infty$), the gradient of PENEX keeps decreasing. The cross-entropy gradient, however, levels out at -1 .

Popular regularization methods for deep learning are ℓ_2 regularization, confidence penalty and label smoothing. These methods can be interpreted as adding a regularization term to cross-entropy (Meister et al., 2020) and, in contrast to PENEX, fail Fisher consistency:

Observation 2.1. *Consider the family of loss functions obtained by adding a regularization term $\Omega(f)$ to the cross-entropy loss:*

$$\mathcal{L}_{\text{reg}}(f; \lambda) = \mathcal{L}_{\text{CE}}(f) + \lambda \Omega(f). \quad (4)$$

In general, $\mathcal{L}_{\text{reg}}(f; \lambda)$ is not Fisher consistent for $\hat{P}(y | \mathbf{x}) \propto \exp\{f^{(y)}(\mathbf{x})\}$.

We demonstrate a proof of Obs. 2.1 in Sec. A.2. A key aspect in the generalization ability of PENEX lies

²This means that we exchange $\hat{\mathbb{E}}$ by \mathbb{E} in (3).

³Based on this result, in practice, we always rescale logits during inference (see Sec. C.1).

in an implicit margin-maximization property. 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, that is

$$m_f(\mathbf{x}, y) := f^{(y)}(\mathbf{x}) - \max_{j \neq y} f^{(j)}(\mathbf{x}). \quad (5)$$

The next theorem shows that PENEX effectively aims at maximizing the margin for many data points:

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

$$\mathbb{P}(m_f(\mathbf{x}, y) \leq \gamma) \leq e^{\frac{\gamma\alpha}{\alpha+1}} \rho^{-\frac{\alpha}{\alpha+1}} \mathbb{E}[\mathcal{L}_{\text{PENEX}}(f; \alpha, \rho)]. \quad (6)$$

A proof is provided in Sec. A.4. Intuitively, Thm. 2.1 says that the probability of a small margin for a data point is low, if PENEX has a small value for a model f , in expectation. Not only does Thm. 2.1 explain the margin-maximization property of PENEX (Fig. 2), but also it gives us a hint on how to reasonably choose ρ as a function of α :

Proposition 2.2 (Optimal Penalty Parameter). *The parameter ρ that minimizes the upper bound in (6) is*

$$\rho_*(\alpha, f) = \alpha \frac{\mathbb{E}[\mathcal{L}_{\text{EX}}(f; \alpha)]}{\mathbb{E}[\sum_{j=1}^K f^{(j)}(\mathbf{x})]}. \quad (7)$$

This result follows from realizing that the upper bound in (6) is convex and differentiable in ρ , thus we can solve for ρ_* by setting its derivative to zero.

2.2 Practical Implementation

In practice, the optimal penalty parameter in (7) cannot be computed directly, so we rely on batch statistics for the estimation. To further 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 function object and *does not require a custom training loop*. We refer to Sec. C for more details regarding the implementation.

2.3 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

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

These weak learners g_m , all together with their linear factors η_m , are fitted in a greedy fashion⁴ according to

⁴Also known as *stagewise additive modeling using a multi-class exponential loss function* (SAMME).

Algorithm 1: PENEX Training

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

```

1 for  $t \leftarrow 1, 2, \dots, T$  do
2    $\mathcal{D}_b \stackrel{i.i.d.}{\sim} \mathcal{D}$  // sample mini-batch
3    $\rho' \leftarrow \alpha \frac{\mathcal{L}_{\text{EX}}^{\mathcal{D}_b}(f; \alpha)}{\mathbb{E}_{\mathcal{D}_b} [\sum_{j=1}^K f^{(j)}(\mathbf{x})] + \epsilon}$  // (7)
4   if  $t = 1$  then
5      $\rho_0 \leftarrow \rho'$ 
6    $\rho'_t \leftarrow (1 - \beta)\rho_{t-1} + \beta\rho'$  // EMA
7    $\rho_t \leftarrow \min\{\max\{\rho'_t, \rho_{\min}\}, \rho_{\max}\}$  // clip
8    $\theta_t \leftarrow \text{optim}(\mathcal{L}_{\text{PENEX}}^{\mathcal{D}_b}(\theta_{t-1}; \alpha, \rho_t))$ 
9 return  $\theta_T$ 
    
```

the CONEX objective (2)

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

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

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

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, i.e., single-split decision trees (e.g., Hastie et al. (2017, p.362)). For completeness, we show the full AdaBoost algorithm in Sec. B.

PENEX is Implicit AdaBoost. We now proceed to demonstrate formally that gradient descent on PENEX is analogous to AdaBoost in that gradient descent steps 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:

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

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

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

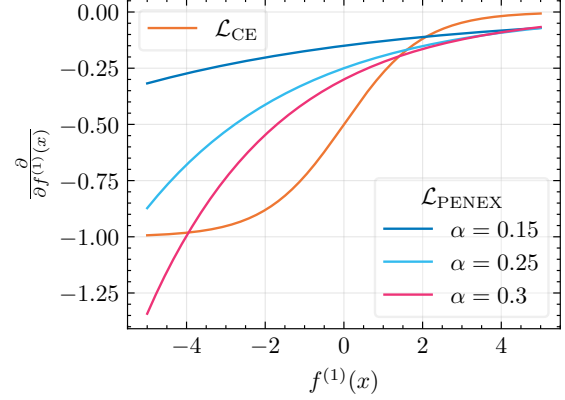


Figure 3: **CE vs. PENEX.** We consider the binary case ($K = 2$) with $f^{(2)}(x) \equiv 0$, for a single x and $y = 1$. PENEX penalizes errors more than cross-entropy.

where $\mathcal{G}_{\text{PENEX}}$ is the set of functions given in Sec. A.3. The following proposition shows that a small gradient descent step on PENEX indeed does solve (10):

Proposition 2.3. *In the limit $\eta \rightarrow 0$, for some $\mathcal{G}_{\text{PENEX}}$ and $\rho > 0$, the solution of (10) converges to the negative rescaled gradient of PENEX. That is,*

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

A proof for Prop. 2.3 is provided in Sec. A.3.

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 also cover related works in the field of boosting within deep learning.

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 of work by Soudry et al. (2018); Ji & Telgarsky (2019); Gunasekar et al. (2018); Nacson et al. (2019) developed theoretical results based on margin theory for performing gradient descent on the binary exponential loss (1) with linear models. This

contrasts the focus of the present work, which specifically considers multi-class problems and highly non-linear neural networks. Our ablation studies indicate that the existing formulation of the exponential loss studied in the listed works is not amenable to effective neural network training (see Sec. E.2).

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 highly 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.

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. 2.3) and is given as the Jacobian, multiplied by a small parameter increment (10), instead of existing approaches that consider explicitly combining entire models to gradually build more complex ones.

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?

Ablation studies (2) are presented in Sec. E.2, parameter analysis (5) is performed in Sec. E.3, all other questions are answered in the main text.

4.1 Setup

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

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. **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. **3)** Noisy CIFAR-10: CIFAR-10 with 10% of training labels flipped uniformly at random. The test set does not have flipped labels. **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 ≈ 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. **2)** Cross-Entropy (CE): Corresponds to \mathcal{L}_{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.*

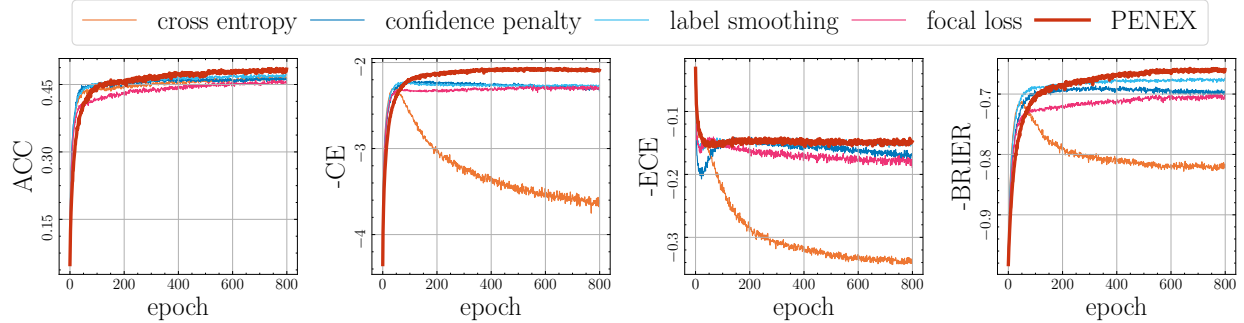


Figure 4: **Performance Analysis on CIFAR-100.** Larger means better. Results are computed from validation data. All hyperparameters have been tuned individually. PENEX (thick red) is an effective regularizer with often better generalization than other common regularization techniques (thin), and shows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

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. C 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. D.11).

4.2 Results

PENEX often achieves regularization performance that outperforms other regularization techniques (though taking a bit longer to converge), as can be seen in Fig. 1, showing multiple validation curves for CIFAR-100 (analogous plots for all others datasets are shown in Sec. E.1) as a function of the number of training epochs. We also provide test results in Tab. 1 that confirm the results from Fig. 1 on multiple datasets and domains. Remarkably, PENEX also outperforms other regularizers in the setting of artificial label noise. The only experiment where PENEX performs 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.

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 (and common regularization techniques; see Sec. 4.2). 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 favourably in deep neural networks 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). 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.

Convergence Speed & Instability. For large values of α or large values of the learning rate, PENEX typically takes longer to converge and loss curves becomes more unsteady, compared to other meth-

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).

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	-0.032 ± 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	0.651 ± 0.000	0.970 ± 0.006
	-ECE	-0.112 ± 0.002	-0.183 ± 0.003	-0.147 ± 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	-0.589 ± 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	0.974 ± 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	-0.042 ± 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	-0.093 ± 0.000	-0.051 ± 0.005
	-CE	-0.661 ± 0.010	-0.905 ± 0.008	-2.341 ± 0.022	-0.939 ± 0.050	-1.578 ± 0.000	-0.092 ± 0.014
	-BRIER	-0.313 ± 0.005	-0.423 ± 0.004	-0.723 ± 0.005	-0.291 ± 0.006	-0.482 ± 0.000	-0.042 ± 0.008
PENEX	ACC	0.793 ± 0.004	0.766 ± 0.004	0.460 ± 0.005	0.833 ± 0.004	0.404 ± 0.000	0.968 ± 0.006
	-ECE	-0.109 ± 0.002	-0.131 ± 0.002	-0.147 ± 0.003	-0.100 ± 0.003	-0.096 ± 0.000	-0.034 ± 0.006
	-CE	-0.646 ± 0.012	-0.716 ± 0.009	-2.140 ± 0.018	-1.200 ± 0.089	-3.011 ± 0.000	-0.124 ± 0.025
	-BRIER	-0.299 ± 0.005	-0.332 ± 0.004	-0.685 ± 0.004	-0.251 ± 0.006	-0.738 ± 0.000	-0.055 ± 0.011

ods (Fig. 4, Sec. E.3). However, PENEX typically achieves better generalization in the long run, for larger α . We believe that adaptive scheduling of α could play an important role in optimizing training efficiency and stability, in addition to using smaller learning rates.

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

Domain Shift, Corruption Analyses, and Fairness. This study prioritizes theoretical analysis, empirical results and core robustness results (label noise). Evaluating PENEX under domain shift and adversarial corruptions, as well as fairness considerations are important next steps; we leave a comprehensive study to future work.

6 Conclusion

We introduced the *penalized exponential loss* (PENEX), a margin-promoting objective inspired by multi-class AdaBoost and designed for modern neural network training. Beyond its theoretical footing, PENEX reliably regularizes training and, in low-data settings, it typically surpasses existing methods.

These results suggest PENEX is a strong default choice for practitioners and a promising basis for future work.

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. 18
- Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999. 1
- 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. 1
- Leo Breiman. Arcing Classifiers. *Annals of Statistics*, 26:123–40, 1996a. 1
- Leo Breiman. Bias, Variance, and Arcing Classifiers. *Tech. Rep. 460, Statistics Department, University of California, Berkeley*, 1996b. 1

- 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. [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. [6](#)
- 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. [7](#)
- 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. [6](#), [23](#)
- Harris Drucker. Improving Regressors using Boosting Techniques. In *International Conference on Machine Learning*, volume 97, 1997. [8](#)
- Harris Drucker and Corinna Cortes. Boosting Decision Trees. *Advances in Neural Information Processing Systems*, 8, 1995. [1](#)
- 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>. [18](#)
- 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. [7](#)
- 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](#)
- 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](#), [7](#)
- Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001. [5](#), [7](#)
- John D. Garrett. garrettj403/SciencePlots. *Zenodo*, September 2021. doi: 10.5281/zenodo.4106649. URL <http://doi.org/10.5281/zenodo.4106649>. [18](#)
- 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. [6](#)
- 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. [1](#), [7](#)
- 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. [5](#)
- 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. [6](#)
- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2017. [1](#), [5](#)

- Magnus R. Hestenes. Multiplier and Gradient Methods. *Journal of Optimization Theory and Applications*, 4:303–320, 1969. 4, 20, 25
- 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. 5
- 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. 6
- Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, 2015. 4, 23
- 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. 22
- Alex Krizhevsky. Learning Multiple Layers of Features from Tiny Images. 2009. 6
- 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. 6, 24
- 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. 6
- Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. *International Conference on Learning Representations*, 2019. 23
- 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
- David J.C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003. 1
- 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. 5, 7
- 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, 4
- 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. 5
- 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. 1
- 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. 18
- Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey Hinton. Regularizing Neural Networks by Penalizing Confident Output Distributions. *International Conference on Learning Representations*, 2017. 2, 6, 24

- 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. 7
- Gunnar Rätsch, Takashi Onoda, and K.-R. Müller. Soft Margins for AdaBoost. *Machine Learning*, 42: 287–320, 2001. 1, 7
- Saharon Rosset, Ji Zhu, and Trevor Hastie. Margin Maximizing Loss Functions. *Advances in Neural Information Processing Systems*, 16, 2003. 1
- 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. 1
- 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. 1, 2
- 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. 8
- 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. 5
- 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. 16
- 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, 6, 24
- 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
- Shuhei Watanabe. Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance. *arXiv preprint arXiv:2304.11127*, 2023. 7
- 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. 1
- 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. 6
- 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. 7

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

Checklist

1. For all models and algorithms presented, check if you include:
 - (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes
 - (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. Not Applicable
 - (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes, anonymized source code, including instructions for optimal reproducibility, is included in the supplementary material.
2. For any theoretical claim, check if you include:
 - (a) Statements of the full set of assumptions of all theoretical results. Yes
 - (b) Complete proofs of all theoretical results. Yes
 - (c) Clear explanations of any assumptions. Yes
3. For all figures and tables that present empirical results, check if you include:
 - (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes
 - (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes
 - (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes
 - (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
 - (a) Citations of the creator If your work uses existing assets. Yes
 - (b) The license information of the assets, if applicable. Yes
 - (c) New assets either in the supplemental material or as a URL, if applicable. Yes
 - (d) Information about consent from data providers/curators. Not Applicable
 - (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable
5. If you used crowdsourcing or conducted research with human subjects, check if you include:
 - (a) The full text of instructions given to participants and screenshots. Not Applicable
 - (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable
 - (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable

PENEX: AdaBoost-Inspired Neural Network Regularization Supplementary Materials

Appendix

Table of Contents

A Proofs	14
A.1 Proof for Prop. 2.1	14
A.2 Proof for Obs. 2.1	14
A.3 Proof for Prop. 2.3	15
A.4 Proof for Thm. 2.1	16
B Statewise Additive Modeling using a Multi-class Exponential loss	17
C Practical Implementation of PENEX	17
C.1 Inference	17
C.2 Default Value for Sensitivity α	17
C.3 Default Values for Alg. 1	17
C.4 PyTorch-like Code	18
D Experimental Details	18
D.1 Libraries	18
D.2 Dataset & Model Licenses	18
D.3 Computational Resources	19
D.4 Ablation Implementations	20
D.5 Metrics	20
D.6 Data Preprocessing	21
D.7 Train/Validation Splits	22
D.8 Architectures	23
D.9 Training	23
D.10 Hyperparameter Tuning	24
D.11 Hyperparameters for ImageNet	24
E Additional Experiments	25
E.1 All Validation Results	25
E.2 Ablation Studies	25
E.3 Sensitivity Parameter Analysis	25

A Proofs

A.1 Proof for Prop. 2.1

Proof. We consider the following blueprint:

1. We show that PENEX is Fisher consistent, conditionally on any \mathbf{x} and for any $\rho > 0$.
2. We show that PENEX is also unconditionally Fisher consistent.

Step 1 The population-level equivalent of (3), conditionally on \mathbf{x} , is

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

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:

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

which leads to

$$P(y|\mathbf{x}) = \frac{\rho}{\alpha} \exp \left\{ (1 + \alpha) f_*^{(y)}(\mathbf{x}) \right\}.$$

Due to re-normalization, the factor ρ/α is canceled out and hence, the result follows.

Step 2 First, we note that the specific choice of ρ does not matter for the result shown in Step 1, due to re-normalization. Consequently, we can pick a single, arbitrary $\rho > 0$ that achieves Fisher consistency for all \mathbf{x} . We now consider the unconditional population equivalent of PENEX (3),

$$\mathbb{E} \mathcal{L}_{\text{PENEX}}(f) = \mathbb{E}_{\mathbf{x}} \left[\mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}(f(\mathbf{x})) \right]. \quad (13)$$

It is easy to see that its minimizer is given by the function f_* that minimizes the conditional PENEX $\mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}(f(\mathbf{x}))$ for each $\mathbf{x} \in \text{supp}(P)$ (11). We consider an arbitrary function f and see that

$$\mathbb{E}[\mathcal{L}_{\text{PENEX}}(f)] - \mathbb{E}[\mathcal{L}_{\text{PENEX}}(f_*)] = \int_{\mathbf{x} \in \text{supp}(P)} \left[\mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}(f(\mathbf{x})) - \mathcal{L}_{\text{PENEX}}^{y|\mathbf{x}}(f_*(\mathbf{x})) \right] p(\mathbf{x}) d\mathbf{x} \geq 0,$$

which concludes the proof. \square

A.2 Proof for Obs. 2.1

Proof. For simplicity, we assume that \mathbf{x} can only take one single value, such that estimating $P(Y|X)$ simplifies to estimating the parameters of a multinoulli/categorical distribution, $P(y)$. Furthermore, we note that we can make statements about the estimated f by solving for \hat{P} instead, since $\hat{P}(y) \propto \exp \{ f^{(y)} \}$ constrains f up to an additive constant. We consider a more explicit notation for the objective (4),

$$\mathcal{L}(\hat{P}) = H(P \| \hat{P}) + \lambda \Omega(\hat{P}), \quad (14)$$

where $H(P \| \hat{P})$ denotes cross-entropy between P and \hat{P} and $\lambda > 0$ is regularization strength. With $H(P)$ denoting Shannon entropy of P , we note that we can rewrite cross-entropy as

$$H(P \| \hat{P}) = H(P) + \text{KL}(P \| \hat{P}),$$

and see that the unique optimum for the trivial case $\lambda = 0$ is given as $\hat{P}_* = P$, i.e., Fisher consistency holds. However, this is generally not the case if $\lambda > 0$. To see this, we construct the Lagrangian

$$\mathcal{L}(\hat{P}) = \mathcal{L}(\hat{P}) + \lambda' \left(\sum_{y=1}^K \hat{P}(y) - 1 \right),$$

with Lagrange multiplier λ' . By the stationarity condition, it must hold for all y that

$$-\frac{P(y)}{\hat{P}(y)} \Big|_{\hat{P}(y)=\hat{P}_*(y)} + \lambda \frac{\partial \Omega}{\partial \hat{P}(y)} \Big|_{\hat{P}(y)=\hat{P}_*(y)} + \lambda' = 0. \quad (15)$$

At $\hat{P} = P$, (15) simplifies to

$$\lambda \frac{\partial \Omega}{\partial \hat{P}(y)} = 1 - \lambda',$$

which means that P can only be optimal if

$$\frac{\partial \Omega}{\partial \hat{P}(y)} \Big|_{\hat{P}(y)=P(y)} = \text{const.}, \quad \forall y \in [K]. \quad (16)$$

It is easy to see, however, that (16) is generally not the case. For example, in the case of a confidence penalty $\Omega(\hat{P}) = -H(\hat{P})$,

$$\frac{\partial \Omega}{\partial \hat{P}(y)} = 1 + \log(\hat{P}(y))$$

and so (16) is true if and only if $P(y)$ is the uniform distribution, i.e., $P(y) = K^{-1}$, $\forall y$. This is only true in trivial cases. The same argument applies to other choices of $\Omega(\hat{P})$. \square

A.3 Proof for Prop. 2.3

We choose the set of functions to be

$$\mathcal{G}_{\text{PENEX}} := \left\{ \mathbf{J}(\mathbf{x})\Delta\theta \mid \hat{\mathbb{E}} \left[\sum_{j=1}^K \exp \left\{ f_{\theta_{m-1}}^{(j)}(\mathbf{x}) + \eta(\mathbf{J}(\mathbf{x})\Delta\theta_m)^{(j)} \right\} \right] \leq \epsilon, \|\Delta\theta\|_2 = 1 \right\}, \quad (17)$$

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

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

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

for the objective function and

$$\hat{\mathbb{E}} \left[\sum_{j=1}^K \exp \left\{ f_{\theta_{m-1}}^{(j)}(\mathbf{x}) \right\} \right] + \eta \hat{\mathbb{E}} \left[\sum_{j=1}^K \exp \left\{ f_{\theta_{m-1}}^{(j)}(\mathbf{x}) \right\} \mathbf{J}(\mathbf{x})^{(j)} \right] \Delta\theta + h_2(\eta, \Delta\theta)\eta \leq \epsilon$$

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

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

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

$$\begin{aligned} v(\eta) &:= \min_{\Delta\theta} -\alpha \hat{\mathbb{E}} \left[\exp \left\{ -\alpha f_{\theta_{m-1}}^{(y)}(\mathbf{x}) \right\} \mathbf{J}(\mathbf{x})^{(y)} \right] \Delta\theta + h_1(\eta, \Delta\theta) \\ &\text{subject to } \hat{\mathbb{E}} \left[\sum_{j=1}^K \exp \left\{ f_{\theta_{m-1}}^{(j)}(\mathbf{x}) \right\} \mathbf{J}(\mathbf{x})^{(j)} \right] \Delta\theta + h_2(\eta, \Delta\theta) \leq 0 \\ &\quad \|\Delta\theta\|_2^2 = 1. \end{aligned} \quad (18)$$

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 (18)) is nonempty for small enough η . We further note that $v(\eta)$, the minimum value in (18), is continuous for small η , see for example (Still, 2018, Ch. 5) and consider any sequence $\eta_k > 0$ and corresponding $\Delta\theta_{mk}$, $k = 1, 2, \dots$ where $\eta_k \rightarrow 0$ 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 $\Delta\theta_{mk} \rightarrow \Delta\theta_\infty$. We now show that $\Delta\theta_\infty$ is a minimizer of (10) (or equivalently (18)) for $\eta = 0$. The limit point $\Delta\theta_\infty$ satisfies $\|\Delta\theta_\infty\|_2^2 = 1$ and the first constraint

$$\hat{\mathbb{E}} \left[\sum_{j=1}^K \exp \left\{ f_{\theta_{m-1}}^{(j)}(\mathbf{x}) \right\} \mathbf{J}(\mathbf{x})^{(j)} \right] \Delta\theta_\infty \leq 0,$$

and is therefore feasible for the minimization in (18) when $\eta = 0$. We conclude from the continuity of v that $v(\eta_k) \rightarrow v(0)$ 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 (18) for $\eta = 0$. These can be expressed as

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

where $\bar{\rho} \geq 0$ is related to a multiplier of the first constraint in (18). 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.

A.4 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 $\alpha := \frac{\beta}{1-\beta}$ ⁵ 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

$$\mathbb{E}[\mathcal{L}_{\text{EX}}(f; \alpha)]^{1-\beta} \cdot (\rho \mathbb{E}[\text{SE}(f)])^\beta \rho^{-\beta} \leq ((1-\beta)\mathbb{E}[\mathcal{L}_{\text{EX}}(f; \alpha)] + \beta\rho\mathbb{E}[\text{SE}(f)])\rho^{-\beta} \quad (19)$$

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 $\beta = \frac{\alpha}{\alpha+1}$. □

⁵We 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.

B Stategewise 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

1 Initialize sample weights $w_1(i) \leftarrow \frac{1}{n}$ for $i = 1, \dots, n$

2 **for** $m = 1$ **to** M **do**

3 Fit weak classifier g_m using weights w_m

4 Compute weighted error $\epsilon_m \leftarrow \sum_{i=1}^n w_m(i) \cdot \mathbb{1}\{g_m(\mathbf{x}_i) \neq y_i\}$

5 Compute classifier weight $\eta_m \leftarrow \log\left(\frac{1-\epsilon_m}{\epsilon_m}\right) + \log(K-1)$

6 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

7 **end**

8 Combine weak classifiers: $f_M(\mathbf{x}) = \arg \max_{k \in [K]} \sum_{m=1}^M \eta_m g_m(\mathbf{x})$

C Practical Implementation of PENEX

C.1 Inference

Motivated by the Fisher consistency result (Prop. 2.1), we always rescale logits that are optimized on PENEX by $1 - \alpha$. Specifically, to obtain a prediction \hat{P}_{PENEX} , we always use

$$\hat{P}_{\text{PENEX}}(y | \mathbf{x}) := \frac{\exp\{(1 - \alpha)f^{(y)}(\mathbf{x})\}}{\sum_{j=1}^K \exp\{(1 - \alpha)f^{(j)}(\mathbf{x})\}}, \quad (20)$$

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

C.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

$$\alpha = 0.1. \quad (21)$$

If PENEX is used without tuning, we generally recommend (21) 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 maximization.

C.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 ρ_{\min}	10^{-6}
Upper penalty bound ρ_{\max}	100.0

```

1  import torch
2  import torch.nn as nn
3
4
5  def exp_loss(logits, targets, alpha):
6      """Exponential loss on the true-class score fy(x)."""
7      y = targets.long()
8      fy = logits.gather(1, y.unsqueeze(1)).squeeze(1)
9      return torch.exp(-alpha * fy)
10
11 class PENEX(nn.Module):
12     """Ultra-minimal readable version."""
13     def __init__(self, alpha=0.1, rho_min=1e-6, rho_max=100.0, ema=0.1):
14         super().__init__()
15         self.alpha = alpha
16         self.rho_min = rho_min
17         self.rho_max = rho_max
18         self.ema = ema
19         self.rho = None # running estimate of rho
20
21     def forward(self, logits, targets):
22         # Base loss
23         base = exp_loss(logits, targets, self.alpha)
24         # Penalty term; uses torch.logsumexp() for numerical stability
25         sumexp = torch.exp(torch.logsumexp(logits, dim=1))
26
27         # Update rho without tracking gradients
28         with torch.no_grad():
29             est = self.alpha * (base.mean() / (sumexp.mean() + 1e-12))
30             est = est.clamp(self.rho_min, self.rho_max)
31             self.rho = est if self.rho is None else (1 - self.ema) * self.rho + self.ema * est
32
33         loss = base + self.rho * sumexp
34         return loss.mean()
    
```

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

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

C.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. C.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.

D Experimental Details

D.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.

D.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

D.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

D.4 Ablation Implementations

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

$$h(\mathbf{x}; \theta) := \mathbf{1}^\top f_\theta(\mathbf{x})$$

and see that

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

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

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

the optimum of which converges to the constrained solution for $\rho \rightarrow \infty$ (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 $\lambda_m \in \mathbb{R}$:

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

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

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

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.

D.5 Metrics

Accuracy. We measure the accuracy of a classifier f as

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

Expected Calibration Error. Formally, it is given as

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

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

$$I_m := \left(\frac{m-1}{M}; \frac{m}{M} \right]$$

and

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

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

Brier Score. We compute the Brier score as

$$\text{BRIER}(f) := \frac{1}{N} \sum_{i=1}^N \sum_{j=1}^K \left(\hat{P}(y = j | f(\mathbf{x}_i)) - \mathbf{y}_i^{(j)} \right)^2,$$

where

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

is a one-hot encoding of the label.

D.6 Data Preprocessing

D.6.1 Computer Vision

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

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.

D.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.

D.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

D.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. 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	$256 \rightarrow \text{n.out}$	n.out

In the final layer, $\text{n.out} = 10$, $\text{n.out} = 100$ or $\text{n.out} = 9$, respectively. We do so for all methods except for ablation 3) (see Sec. 4.1), where $\text{n.out} = 9$, $\text{n.out} = 99$ or $\text{n.out} = 8$, 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 $3 \times 64 \times 64$).

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 $768 \rightarrow \text{n.out}$	n.out

In the final layer, $\text{n.out} = 1000$.

D.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:

PENEX: AdaBoost-Inspired Neural Network Regularization

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-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

D.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.

D.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

E Additional Experiments

E.1 All Validation Results

The additional plots shown in Fig. 5 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. D.9) and leading to worse generalization as a consequence.

E.2 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. D.4.

We perform all ablation studies for the CIFAR-10 dataset only. The results, displayed in Fig. 6, 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. D.10).

E.3 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.

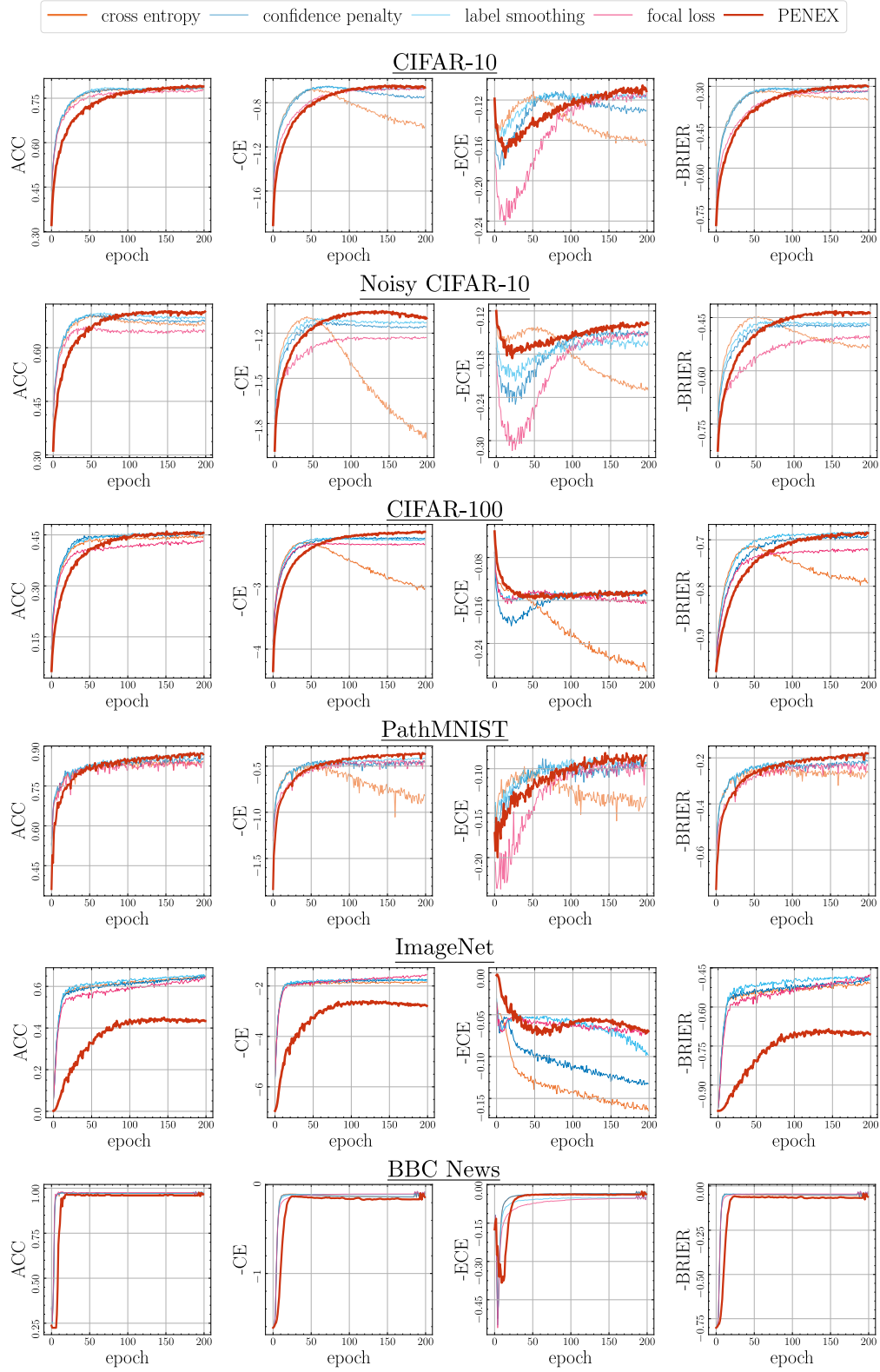


Figure 5: **All Validation Curves.** Larger means better. Validation curves over 200 epochs for all experiments, similar to Fig. 4.

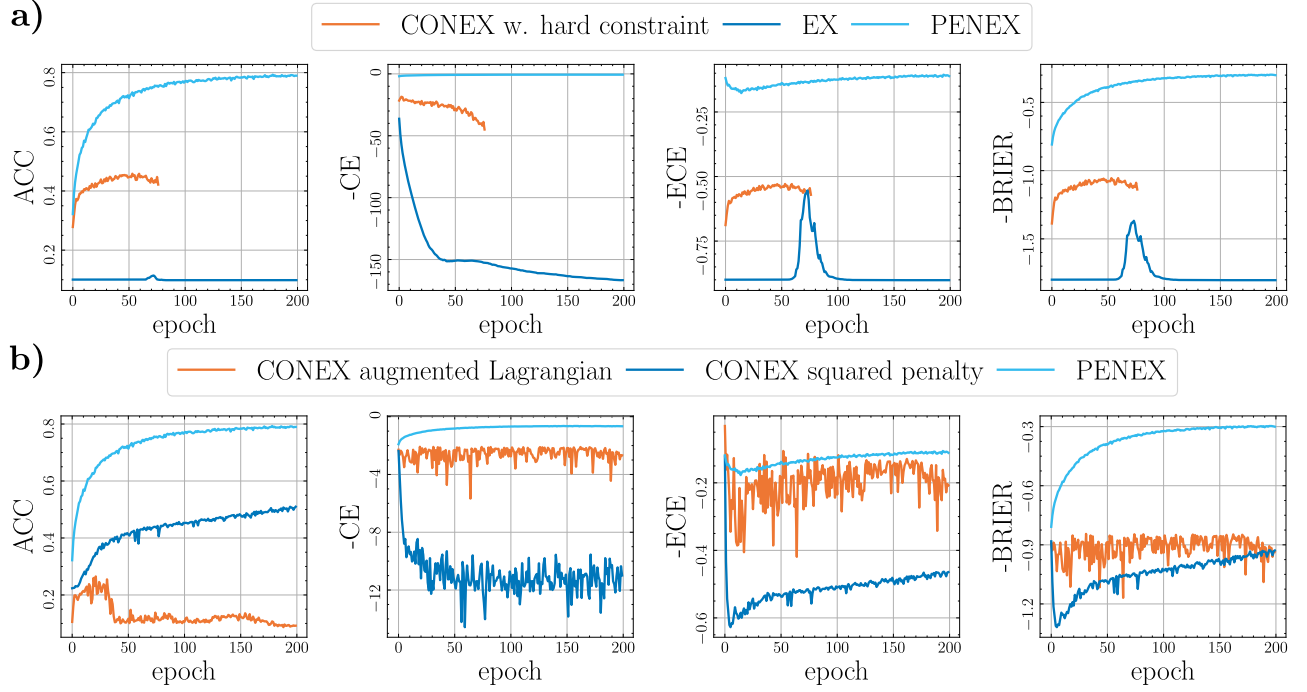


Figure 6: **Ablation Studies on CIFAR-10.** Larger means better. **a)** Optimizing the exponential loss without any constraint or with the constraint encoded into the model architectures break down early on during training (no results are shown for the hard constraint method starting at around epoch 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. D.10).

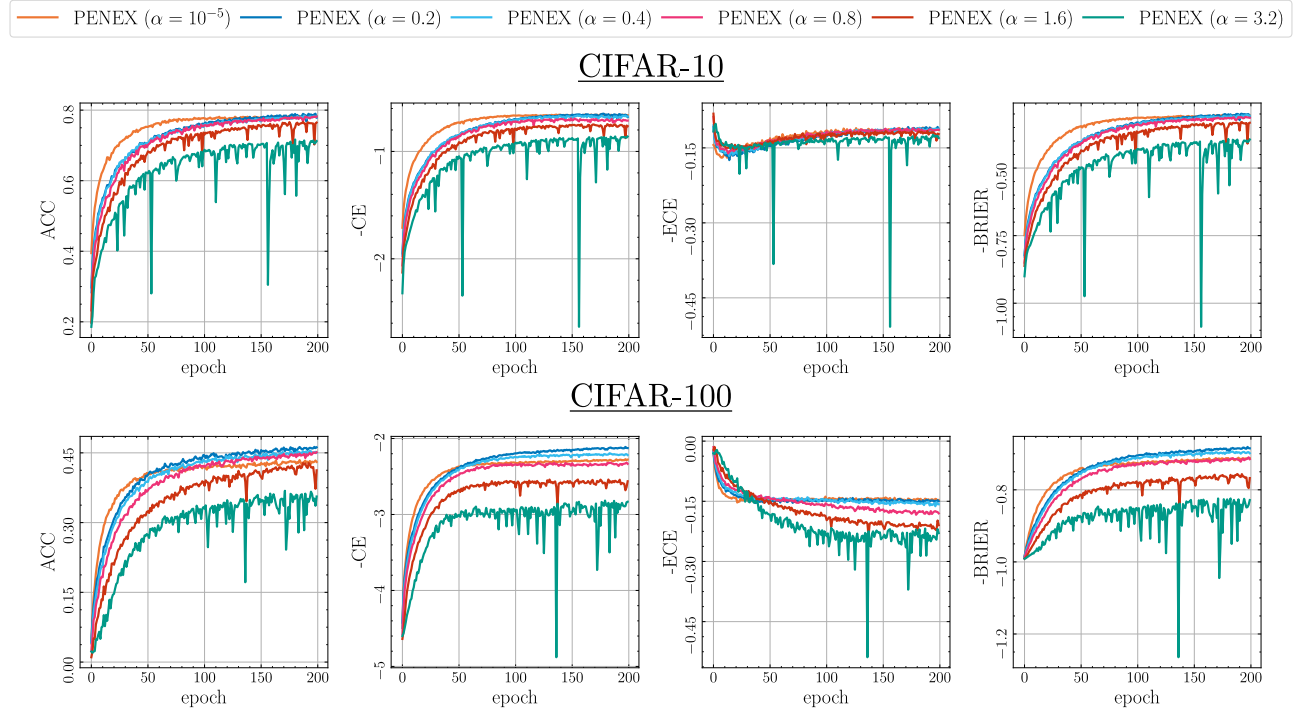


Figure 7: **Effects of the Sensitivity Parameter on Validation Curves.** Smaller α typically lead to faster convergence, but worse generalization toward the end of the training duration (as can be seen more clearly for CIFAR-100). In addition, for very large $\alpha = 3.2$, training curves tend to become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