Supplementary Material
Bridging the Gap: Unifying the Training and Evaluation of Neural Network Binary Classifiers

1 Theoretical Grounding

This section provides the proofs mentioned in Section 4 of the main paper.

1.1 Lipschitz Continuity of Metrics Based on Soft-Set Confusion Matrix Values

Theorem 4.1. *The linear Heaviside function approximation $\mathcal{H}^l$ is Lipschitz continuous with Lipschitz constant $M = \max\{m_1, m_2, m_3\}$.*

Proof. Recall that $\mathcal{H}^l$ is piecewise linear, consisting of three line segments of slopes:

$$m_1 = \frac{\delta}{\tau - \frac{\tau_m}{2}} \qquad m_2 = \frac{1-2\delta}{\tau_m} \qquad m_3 = \frac{\delta}{1 - \tau - \frac{\tau_m}{2}}$$

where $\tau_m = \min\{\tau, 1-\tau\}$.

Consider any fixed τ. $\mathcal{H}^l$ is continuous, because each piecewise linear component is continuous, and we can computationally verify that $\mathcal{H}^l(p, \tau)$ is defined to be continuous at the two points $p = \tau \pm \frac{\tau_m}{2}$. Let $M = \max\{m_1, m_2, m_3\}$. Then, we show that the slope of any secant line must be nonnegative and bounded by M.

For simplicity, let $f(p) = \mathcal{H}^l(p, \tau)$. Consider any two points $0 \leq p_1, p_2 \leq 1$, and assume without loss of generality that $p_1 \leq p_2$. If $p_1, p_2 \leq \tau - \frac{\tau_m}{2}$, then $|f(p_2) - f(p_1)| = m_1|p_2 - p_1|$. If $\tau - \frac{\tau_m}{2} \leq p_1, p_2 \leq \tau + \frac{\tau_m}{2}$, then $|f(p_2) - f(p_1)| = m_2|p_2 - p_1|$. Furthermore, if $\tau + \frac{\tau_m}{2} \leq p_1, p_2$, then $|f(p_2) - f(p_1)| = m_3|p_2 - p_1|$. In all three of these cases, $|f(p_2) - f(p_1)| \leq M|p_2 - p_1|$.

Otherwise, if p_1 and p_2 do not both lie within the bounds of same singular line segment, then there are three more cases. If $p_1 \leq \tau - \frac{\tau_m}{2} \leq p_2 \leq \tau + \frac{\tau_m}{2}$, then since $p_1 \leq p_2$ and f is nondecreasing:

$$\begin{aligned}
|f(p_2) - f(p_1)| &= \left|f(p_2) - f\left(\tau - \frac{\tau_m}{2}\right)\right| + \left|f\left(\tau - \frac{\tau_m}{2}\right) - f(p_1)\right| \\
&= m_2\left|p_2 - \left(\tau - \frac{\tau_m}{2}\right)\right| + m_1\left|\left(\tau - \frac{\tau_m}{2}\right) - p_1\right| \\
&\leq M|p_2 - p_1|
\end{aligned}$$

Similarly, if $\tau - \frac{\tau_m}{2} \leq p_1 \leq \tau + \frac{\tau_m}{2} \leq p_2$, then:

$$\begin{aligned}
|f(p_2) - f(p_1)| &= \left|f(p_2) - f\left(\tau + \frac{\tau_m}{2}\right)\right| + \left|f\left(\tau + \frac{\tau_m}{2}\right) - f(p_1)\right| \\
&= m_3\left|p_2 - \left(\tau + \frac{\tau_m}{2}\right)\right| + m_2\left|\left(\tau + \frac{\tau_m}{2}\right) - p_1\right| \\
&\leq M|p_2 - p_1|
\end{aligned}$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Finally, if $p_1 \leq \tau - \frac{\tau_m}{2}$ and $p_2 \geq \tau + \frac{\tau_m}{2}$, then:

$$\begin{aligned}|f(p_2) - f(p_1)| &= \left|f(p_2) - f\left(\tau + \frac{\tau_m}{2}\right)\right| + \left|f\left(\tau + \frac{\tau_m}{2}\right) - f\left(\tau - \frac{\tau_m}{2}\right)\right| + \left|f\left(\tau - \frac{\tau_m}{2}\right) - f(p_1)\right| \\ &= m_3\left|p_2 - \left(\tau + \frac{\tau_m}{2}\right)\right| + m_2\left|\left(\tau + \frac{\tau_m}{2}\right) - \left(\tau - \frac{\tau_m}{2}\right)\right| + m_1\left|\left(\tau - \frac{\tau_m}{2}\right) - p_1\right| \\ &\leq M|p_2 - p_1|\end{aligned}$$

This exhausts all cases, and so for all p_1, p_2, we have $|\mathcal{H}^l(p_2, \tau) - \mathcal{H}^l(p_1, \tau)| = |f(p_2) - f(p_1)| \leq M|p_2 - p_1|$. Thus, $\mathcal{H}^l$ is Lipschitz continuous with Lipschitz constant M. □

Theorem 4.2. *Every entry of the soft-sets confusion matrix based on the Heaviside approximations are Lipschitz continuous in the output of a neural network.*

Proof. Without loss of generality, we prove that the $|TP_s|$ entry is Lipschitz continuous in the input vector $p = (p_1, \ldots, p_n)$ corresponding to outputs of the neural network given inputs $x = (x_1, \ldots, x_n)$ and labels $y = (y_1, \ldots, y_n)$. The Lipschitz continuity of all the other entries ($|FN_s|$, $|FP_s|$, and $|TN_s|$) follows similarly by symmetry. For any sample tuple (p_i, y_i, τ), recall that:

$$tp_s(p_i, y_i, \tau) = \begin{cases} \mathcal{H}^l(p_i, \tau) & y_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

where the final value $|TP_s|$ is calculated from the summation $|TP_s| = \sum_{i=1}^{n} tp_s(p_i, y_i, \tau)$ over all sample tuples (p_i, y_i, τ). Since $tp_s(p_i, y_i, \tau)$ evaluates to 0 if $y_i \neq 1$, this sum is equivalent to $|TP_s| = \sum_{y_i=1} \mathcal{H}^l(p_i, \tau)$.

By Theorem 4.1, we know that $\mathcal{H}^l(p_i, \tau)$ is Lipschitz continuous in p_i over the domain $p_i \in [0, 1]$. Any function that takes as input a vector is Lipschitz continuous if it is Lipschitz continuous in each entry of its input. It follows that $|TP_s| = \sum_{y_i=1} \mathcal{H}^l(p_i, \tau)$ is Lipschitz continuous in the input vector p.

By a similar reasoning, it follows that the other entries $|FN_s|$, $|FP_s|$, and $|TN_s|$ are all also Lipschitz continuous in the input vector p. □

1.2 Approximation of Confusion-Matrix Based Metrics with Soft Sets

Following the ideas in Section 4.2 of the main paper, we generalize the proof for F_1-Score and show that, under similar assumptions, convergence holds for certain other objective functions. Specifically, our proof holds for all continuous objective functions that can be expressed solely in terms of the ratios of entries of the confusion matrix to n, the size of the dataset. Most metrics, including Accuracy, Arithmetic Mean, F_β-Score, Jaccard, and Geometric Mean all satisfy this property. Our generalized proof assumes a fixed threshold value τ, but we also show that convergence holds for AUROC, which is computed over a range of threshold values.

As before, consider a dataset of size n with $\{x_1, ..., x_n\}$ examples and $\{y_1, ..., y_n\}$ labels. Suppose a network outputs a probability p_i that the label $y_i = 1$. Since the specific outputs p_i are unknown and may change across iterations, we again assume that p_i is a random variable. When computing the entries of our confusion matrix, p_i is passed through the Heaviside function H, which generates an output $\hat{y}_i^H = H(p_i, \tau)$, where $\hat{y}_i^H \in \{0, 1\}$. Then $|TP| = \sum_{y_i=1} \hat{y}_i^H$ and $|FP| = \sum_{y_i=0} \hat{y}_i^H$. On the other hand, under our proposed approximation, p_i is passed through the Heaviside approximation $\mathcal{H}$, which generates an output $\hat{y}_i^{\mathcal{H}} = \mathcal{H}(p_i, \tau)$. In this case, $\hat{y}_i^{\mathcal{H}}$ can take on any real value between 0 and 1. Then $|TP_s| = \sum_{y_i=1} \hat{y}_i^{\mathcal{H}}$ and $|FP_s| = \sum_{y_i=0} \hat{y}_i^{\mathcal{H}}$ are entries of the soft set confusion matrix.

1.2.1 Continuous Generalization and Accuracy

Consider a network trained on a dataset with rn positive and $(1 - r)n$ negative elements, where $r \in [0, 1]$ is some constant. Let a loss function ℓ be continuous in the ratio of each entry of the confusion matrix to n, the size of the dataset. Hence, ℓ can be written as a function of $\frac{|TP|}{n}$, $\frac{|FP|}{n}$, $\frac{|TN|}{n}$, and $\frac{|FN|}{n}$. But since $|TP| + |FN| = nr$ and $|FP| + |TN| = (1 - r)n$, $\frac{|FN|}{n} = r - \frac{|TP|}{n}$

and $\frac{|TN|}{n} = (1-r) - \frac{|FP|}{n}$. Thus, $\frac{|FN|}{n}$ and $\frac{|TN|}{n}$ can be represented in terms of $\frac{|TP|}{n}$ and $\frac{|FP|}{n}$, so $\ell\left(\frac{|TP|}{n}, \frac{|FP|}{n}\right)$ can be expressed as a continuous function on just $\frac{|TP|}{n}$ and $\frac{|FP|}{n}$.

Thus, the loss computed using the Heaviside step function can be expressed as:

$$\ell = \ell\left(\frac{1}{n}\sum_{y_i=1}\hat{y}_i^H, \frac{1}{n}\sum_{y_i=0}\hat{y}_i^H\right) \tag{1}$$

The loss when computed from soft sets is then:

$$\ell^s = \ell\left(\frac{1}{n}\sum_{y_i=1}\hat{y}_i^{\mathcal{H}}, \frac{1}{n}\sum_{y_i=0}\hat{y}_i^{\mathcal{H}}\right) \tag{2}$$

As in Section 4.2 of the main paper, suppose a classifier correctly classifies any positive example as a true positive with probability u and any negative example as a false positive with probability v. Also, assume that all classifications are independent. Because the loss function ℓ is calculated with discrete $\hat{y}_i^H$, we assume that the classifier will classify examples as a random variable $\hat{y}_i^H \sim \text{Bernoulli}(uy_i + v(1-y_i))$. Thus, $\hat{y}_i^H \sim \text{Bernoulli}(u)$ if $y_i = 1$, and $\hat{y}_i^H \sim \text{Bernoulli}(v)$ if $y_i = 0$.

Since ℓ^s can take on continuous values in $[0, 1]$, we consider that $\hat{y}_i^{\mathcal{H}}$ is a random variable drawn from a Beta distribution, which has support $[0, 1]$. In particular, assume $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_u y_i + \alpha_v(1-y_i), \beta_u y_i + \beta_v(1-y_i))$. Hence, $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_u, \beta_u)$ if $y_i = 1$, and $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_v, \beta_v)$ if $y_i = 0$. Let $\frac{\alpha_u}{\alpha_u+\beta_u} = u$ and $\frac{\alpha_v}{\alpha_v+\beta_v} = v$, so for any i, $\mathbb{E}\left[\hat{y}_i^{\mathcal{H}}\right] = u$ if $y_i = 1$, and $\mathbb{E}\left[\hat{y}_i^{\mathcal{H}}\right] = v$ if $y_i = 0$.

Under the above assumptions, both ℓ and ℓ^s have the same average classification correctness: for any given i, $\mathbb{E}\left[\hat{y}_i^H\right] = \mathbb{E}\left[\hat{y}_i^{\mathcal{H}}\right]$. Also, there exists some $\alpha_u, \beta_u, \alpha_v, \beta_v$ such that the distributions of $\hat{y}_i^H = H(p_i, \tau)$ and $\hat{y}_i^{\mathcal{H}} = \mathcal{H}(p_i, \tau)$ can both hold simultaneously under the same network for all i.

If we let $\sum_{y_i=1}\hat{y}_i^H = U \sim \text{Binomial}(nr, u)$ and $\sum_{y_i=0}\hat{y}_i^H = V \sim \text{Binomial}(n(1-r), v)$ be the independent random variables denoting the number of true positives and false positives in a sequence of n independent predictions, then by the Strong Law of Large Numbers, $\frac{1}{nr}U \xrightarrow{\text{a.s.}} u$ and $\frac{1}{n(1-r)}V \xrightarrow{\text{a.s.}} v$ both converge with probability 1 as $n \to \infty$. Hence, $\frac{U}{n} \xrightarrow{\text{a.s.}} ru$ and $\frac{V}{n} \xrightarrow{\text{a.s.}} (1-r)v$. We therefore have, from the Continuous Mapping Theorem, that as $n \to \infty$:

$$\ell = \ell\left(\frac{1}{n}\sum_{y_i=1}\hat{y}_i^H, \frac{1}{n}\sum_{y_i=0}\hat{y}_i^H\right) = \ell\left(\frac{U}{n}, \frac{V}{n}\right) \xrightarrow{\text{a.s.}} \ell(ru, (1-r)v) \tag{3}$$

For ℓ^s, let $U^s = \sum_{y_i=1}\hat{y}_i^{\mathcal{H}}$ and $V^s = \sum_{y_i=0}\hat{y}_i^{\mathcal{H}}$ be the independent random variables denoting the total amount of true positives and false positives in the soft set case. Then, ℓ^s from Eq. (2) becomes:

$$\ell^s = \ell\left(\frac{1}{n}\sum_{y_i=1}\hat{y}_i^{\mathcal{H}}, \frac{1}{n}\sum_{y_i=0}\hat{y}_i^{\mathcal{H}}\right) = \ell\left(\frac{U^s}{n}, \frac{V^s}{n}\right) \tag{4}$$

But by the Strong Law of Large Numbers, $\frac{1}{nr}U^s \xrightarrow{\text{a.s.}} \frac{\alpha_u}{\alpha_u+\beta_u} = u$. Similarly, $\frac{1}{n(1-r)}V^s \xrightarrow{\text{a.s.}} \frac{\alpha_v}{\alpha_v+\beta_v} = v$ also converges with probability 1 as $n \to \infty$. Thus, $\frac{U^s}{n} \xrightarrow{\text{a.s.}} ru$ and $\frac{V^s}{n} \xrightarrow{\text{a.s.}} (1-r)v$. By the Continuous Mapping Theorem:

$$\ell^s = \ell\left(\frac{U^s}{n}, \frac{V^s}{n}\right) \xrightarrow{\text{a.s.}} \ell(ru, (1-r)v) \tag{5}$$

Thus, ℓ and ℓ^s both converge almost surely to the same value as $n \to \infty$. Since $\ell(ru, (1-r)v)$ is a finite constant, $\mathbb{E}[\ell], \mathbb{E}[\ell^s] \to \ell(ru, (1-r)v)$ by the Bounded Convergence Theorem. This means that the ℓ^s value is an asymptotically unbiased estimator for the expected loss ℓ, and we expect average loss values to converge to ℓ^s as $n \to \infty$, under our setup.

Since Accuracy $= \frac{|TP|+|TN|}{n}$ is a continuous function on $\frac{|TP|}{n}$ and $\frac{|TN|}{n}$, it follows that Accuracy computed from soft-sets, Accuracy^s, and Accuracy computed using the Heaviside step function both converge almost surely to the same value under this setup as $n \to \infty$. Similarly, $\mathbb{E}[\text{Accuracy}^s]$ and $\mathbb{E}[\text{Accuracy}]$ both converge to the same expectation as $n \to \infty$.

1.2.2 AUROC

Let the true positive rate, $\text{TPR}(\tau)$, and false positive rate, $\text{FPR}(\tau)$, be functions of the threshold value τ. Then AUROC is defined as $\text{AUROC} = \int_{x=0}^{1} \text{TPR}(\text{FPR}^{-1}(x))\,dx$. In our case of binary classification, the choice of threshold value τ can range from 0 to 1.

Consider a network trained on a dataset with rn positive and $(1-r)n$ negative elements, where $r \in [0,1]$ is some constant. Previously, we treated the probabilities u and v as constants since τ was fixed. Now, assume that $u = u(\tau)$ and $v = v(\tau)$ are both parameterized by τ. Both u and v must be nonincreasing in τ because increasing the threshold value will never increase the number of positives classified. Furthermore, we must have $u(0) = v(0) = 1$ and $u(1) = v(1) = 0$ as the boundary conditions. Since τ is no longer fixed, we also let $\hat{y}_i^H(\tau) = H(p_i, \tau)$ and $\hat{y}_i^{\mathcal{H}}(\tau) = \mathcal{H}(p_i, \tau)$ both be parameterized by τ and be nonincreasing in τ.

In practice, since binary classification has discrete true positive and false positive rates, we approximate/calculate AUROC by choosing threshold values $0 = \tau_0 < \tau_1 < \ldots < \tau_k = 1$ where $\tau_i = \frac{i}{k}$ for all $0 \le i \le k$ and then using the trapezoidal rule to approximate the area under the ROC curve with the intervals defined by the points $1 = \text{FPR}(\tau_0) \ge \ldots \ge \text{FPR}(\tau_k) = 0$. Since the true positive rate and false positive rate are both monotone in τ with $\text{TPR}(0) = \text{FPR}(0) = 1$ and $\text{TPR}(1) = \text{FPR}(1) = 0$, the length of the intervals all approach 0, and so this approximation using the trapezoidal rule converges to the AUROC value as $k \to \infty$.

Consider some fixed value k and let $0 = \tau_0 < \tau_1 < \ldots < \tau_k = 1$ where $\tau_i = \frac{i}{k}$. Then under this approximation with the trapezoidal rule, we have that the AUROC is:

$$\text{AUROC} = \frac{1}{2}\sum_{i=1}^{k}(\text{FPR}(\tau_{i-1}) - \text{FPR}(\tau_i))(\text{TPR}(\tau_i) + \text{TPR}(\tau_{i-1})) \tag{6}$$

For true AUROC, note that as before that for any given τ, $\text{TPR}(\tau) = \frac{1}{nr}\sum_{y_i=1}\hat{y}_i^H(\tau)$. Because this is calculated with discrete $\hat{y}_i^H$, we assume that the classifier will classify examples as a random variable $\hat{y}_i^H(\tau) \sim \text{Bernoulli}(u(\tau)\cdot y_i + v(\tau)\cdot(1-y_i))$. Thus, $\hat{y}_i^H(\tau) \sim \text{Bernoulli}(u(\tau))$ if $y_i = 1$ and $\hat{y}_i^H(\tau) \sim \text{Bernoulli}(v(\tau))$ if $y_i = 0$. Hence, for any τ, $\text{TPR}(\tau) = \frac{1}{nr}\sum_{y_i=1}\hat{y}_i^H(\tau) \sim \frac{1}{nr}\text{Binomial}(nr, u(\tau))$ is its marginal distribution. Similarly $\text{FPR}(\tau) = \frac{1}{n(1-r)}\sum_{y_i=0}\hat{y}_i^H(\tau) \sim \frac{1}{n(1-r)}\text{Binomial}(n(1-r), v(\tau))$ for all τ.

However, both $\text{TPR}(\tau)$ and $\text{FPR}(\tau)$ must be nonincreasing functions in τ. To enforce monotonicity so that $\text{TPR}(\tau_0) \ge \text{TPR}(\tau_1) \ge \cdots \ge \text{TPR}(\tau_k)$ and $\text{FPR}(\tau_0) \ge \cdots \ge \text{FPR}(\tau_k)$, we construct their joint distribution as follows. First, consider n uniformly distributed i.i.d. random variables $X_1, \ldots, X_n \sim \text{Uniform}(0,1)$. Then for all i and all $0 \le j \le k$, we let:

$$\hat{y}_i^H(\tau_j) = \begin{cases} 1 & X_i \le u(\tau_j)\cdot y_i + v(\tau_j)\cdot(1-y_i) \\ 0 & X_i > u(\tau_j)\cdot y_i + v(\tau_j)\cdot(1-y_i) \end{cases} \tag{7}$$

Indeed, under this construction, the marginal distributions $\text{TPR}(\tau_j) \sim \frac{1}{nr}\text{Binomial}(nr, u(\tau_j))$ and $\text{FPR}(\tau_j) \sim \frac{1}{n(1-r)}\text{Binomial}(n(1-r), v(\tau_j))$ both hold. We now, however, also have that $\text{TPR}(\tau_0) \ge \cdots \ge \text{TPR}(\tau_k)$ and $\text{FPR}(\tau_0) \ge \cdots \ge \text{FPR}(\tau_k)$.

Since each marginal distribution $\text{TPR}(\tau_i) \sim \frac{1}{nr}\text{Binomial}(nr, u(\tau_i))$, by the Law of Large Numbers, for all $0 \le i \le k$, $\text{TPR}(\tau_i) \xrightarrow{\text{a.s.}} u(\tau_i)$. Similarly, $\text{FPR}(\tau_i) \xrightarrow{\text{a.s.}} v(\tau_i)$. Thus, by the Continuous Mapping Theorem, the true AUROC is:

$$\begin{aligned} \text{AUROC} &= \frac{1}{2}\sum_{i=1}^{k}(\text{FPR}(\tau_{i-1}) - \text{FPR}(\tau_i))(\text{TPR}(\tau_i) + \text{TPR}(\tau_{i-1})) \\ &\xrightarrow{\text{a.s.}} \frac{1}{2}\sum_{i=1}^{k}(v(\tau_{i-1}) - v(\tau_i))\cdot(u(\tau_i) + u(\tau_{i-1})) \end{aligned} \tag{8}$$

Now, consider AUROC computed from soft sets (which we denote as AUROC^s). AUROC^s can take on continuous values in $[0,1]$, so we consider that $\hat{y}_i^{\mathcal{H}}$ is a random variable drawn from a Beta

distribution, which has support $[0, 1]$. In particular, for any τ, assume $\hat{y}_i^{\mathcal{H}}(\tau) \sim \text{Beta}(\alpha_u(\tau) \cdot y_i + \alpha_v(\tau) \cdot (1 - y_i), \beta_u(\tau) \cdot y_i + \beta_v(\tau) \cdot (1 - y_i))$. Hence, $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_u(\tau), \beta_u(\tau))$ if $y_i = 1$, and $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_v(\tau), \beta_v(\tau))$ if $y_i = 0$. Let $\frac{\alpha_u(\tau)}{\alpha_u(\tau)+\beta_u(\tau)} = u(\tau)$ and $\frac{\alpha_v(\tau)}{\alpha_v(\tau)+\beta_v(\tau)} = v(\tau)$, so for any i, $\mathbb{E}\left[\hat{y}_i^{\mathcal{H}}(\tau)\right] = u(\tau)$ if $y_i = 1$, and $\mathbb{E}\left[\hat{y}_i^{\mathcal{H}}(\tau)\right] = v(\tau)$ if $y_i = 0$.

Once again, we show that monotonicity can be enforced across the marginal distributions for corresponding true positive and false positive rates computed from soft sets: $\text{TPR}^s(\tau) = \frac{1}{nr} \sum_{y_i=1} \hat{y}_i^{\mathcal{H}}(\tau)$ and $\text{FPR}^s(\tau) = \frac{1}{n(1-r)} \sum_{y_i=0} \hat{y}_i^{\mathcal{H}}(\tau)$ among the threshold values $\tau_0, \ldots, \tau_k$. It is well known that for any constants $\alpha, \beta, \beta' > 0$ with $\beta > \beta'$, the distribution $\text{Beta}(\alpha, \beta')$ stochastically dominates $\text{Beta}(\alpha, \beta)$: specifically that the cumulative distribution function of $\text{Beta}(\alpha, \beta)$ lies above the cumulative distribution function of $\text{Beta}(\alpha, \beta')$ at every point $x \in (0, 1)$. Let $\alpha_u(\tau) = \alpha_u$ and $\alpha_v(\tau) = \alpha_v$ be constant, with $\beta_u(\tau)$ and $\beta_v(\tau)$ nondecreasing in τ such that $\frac{\alpha_u}{\alpha_u+\beta_u(\tau)} = u(\tau)$ and $\frac{\alpha_v}{\alpha_v+\beta_v(\tau)} = v(\tau)$. Then, for all i, the distribution $\text{Beta}(\alpha_u \cdot y_i + \alpha_v \cdot (1 - y_i), \beta_u(\tau') \cdot y_i + \beta_v(\tau') \cdot (1 - y_i))$ stochastially dominates $\text{Beta}(\alpha_u \cdot y_i + \alpha_v \cdot (1 - y_i), \beta_u(\tau) \cdot y_i + \beta_v(\tau) \cdot (1 - y_i))$ if $\tau > \tau'$. We can therefore construct a coupling among the distributions together by considering i.i.d. random variables $X_1', \ldots, X_n' \sim \text{Uniform}(0, 1)$ and letting $\hat{y}_i^{\mathcal{H}}(\tau)$ be the value corresponding to the X_i'-th percentile of $\text{Beta}(\alpha_u \cdot y_i + \alpha_v \cdot (1 - y_i), \beta_u(\tau) \cdot y_i + \beta_v(\tau) \cdot (1 - y_i))$. This construction still maintains the marginal distributions $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_u, \beta_u(\tau))$ if $y_i = 1$ and $\hat{y}_i^{\mathcal{H}} \sim \text{Beta}(\alpha_v, \beta_v(\tau))$ if $y_i = 0$. However, under this construction, $\text{TPR}^s(\tau_0) \geq \cdots \geq \text{TPR}^s(\tau_k)$ and $\text{FPR}^s(\tau_0) \geq \cdots \geq \text{FPR}^s(\tau_k)$ also hold.

By the Law of Large Numbers, for any $0 \leq i \leq k$, $\text{TPR}^s(\tau_i) = \frac{1}{nr} \sum_{y_i=1} \hat{y}_i^{\mathcal{H}}(\tau_i) \xrightarrow{\text{a.s.}} \frac{\alpha_u(\tau_i)}{\alpha_u(\tau_i)+\beta_u(\tau_i)} = u(\tau_i)$. Similarly, $\text{FPR}^s(\tau_i) \xrightarrow{\text{a.s.}} \frac{\alpha_v(\tau_i)}{\alpha_v(\tau_i)+\beta_v(\tau_i)} = v(\tau_i)$. Thus, by the Continuous Mapping Theorem:

$$\begin{aligned} \text{AUROC}^s &= \frac{1}{2} \sum_{i=1}^{k} (\text{FPR}^s(\tau_{i-1}) - \text{FPR}^s(\tau_i))(\text{TPR}^s(\tau_i) + \text{TPR}^s(\tau_{i-1})) \\ &\xrightarrow{\text{a.s.}} \frac{1}{2} \sum_{i=1}^{k} (v(\tau_{i-1}) - v(\tau_i)) \cdot (u(\tau_i) + u(\tau_{i-1})) \end{aligned} \tag{9}$$

Thus, $\text{AUROC}, \text{AUROC}^s \xrightarrow{\text{a.s.}} \frac{1}{2} \sum_{i=1}^{k} (v(\tau_{i-1}) - v(\tau_i)) \cdot (u(\tau_i) + u(\tau_{i-1}))$ both converge almost surely to the same value. Since AUROC is bounded between 0 and 1, by the Bounded Convergence Theorem, $\mathbb{E}[\text{AUROC}], \mathbb{E}[\text{AUROC}^s] \to \frac{1}{2} \sum_{i=1}^{k} (v(\tau_{i-1}) - v(\tau_i)) \cdot (u(\tau_i) + u(\tau_{i-1}))$.

2 Heaviside Function Approximation

2.1 Sigmoid Approximation: Visual Analysis of Trade-offs when Searching for Optimal k

As described in Section 3.1 of the main paper, a tradeoff must be made when choosing an appropriate value for k in the sigmoid approximation. As k increases, the approximation becomes closer to the Heaviside step function when $\tau = 0.5$, as shown in Figure 1. However, the range of values with zero gradient increases, as shown in Figure 2.

It is important to note that for soft-set membership calculation, as k decreases, the number of τ values over which the approximation diverges from the Heaviside step function at the limit increases. In other words, as k decreases, the number of τ values increases for which the sigmoid approximation does not adhere to the following limits:

$$\lim_{p \to 0} \mathcal{H}(p, \tau) = 0 \quad \forall \tau \qquad \lim_{p \to 1} \mathcal{H}(p, \tau) = 1 \quad \forall \tau \tag{10}$$

Figure 1 illustrates that for $k = 50$, the approximation does approach the limits in Eq. (10) for all values of τ. Contrarily, for k=1, the approximation does not approach the limits in Eq. (10) for any values of τ.

For our experiments, we searched for the best value of k in $\{1, 10, 20, 50\}$ and found that $k = 10$ led to the best performance.

2.2 Derivation of the Linear Heaviside Approximation

Our proposed linear Heaviside approximation, $\mathcal{H}^l$, is formulated to ensure adherence to the properties described in Section 3.1 of the main paper.

We concern ourselves with only the range over [0,1] since we expect the input to represent a probability in [0,1].

For $0 \leq p \leq 1$, we start by specifying the endpoints to ensure Eq. (10) above, and we define the point at $p = \tau$ to ensure the property $\mathcal{H}^l(p = \tau, \tau) = 0.5$:

$$\mathcal{H}^l(p, \tau) = \begin{cases} 0 & \text{if } p = 0 \\ 1 & \text{if } p = 1 \\ 0.5 & \text{if } p = \tau \end{cases} \tag{11}$$

A natural next step would be to solve for a three point linearly interpolated function using the points defined in Eq. (11). However, we find that in practice this is difficult to optimize, perhaps due to the fact that the gradient of one segment becomes much greater than the other as τ approaches 0 or 1. Thus we instead define two more points relative to τ with the introduction of the parameter δ, which

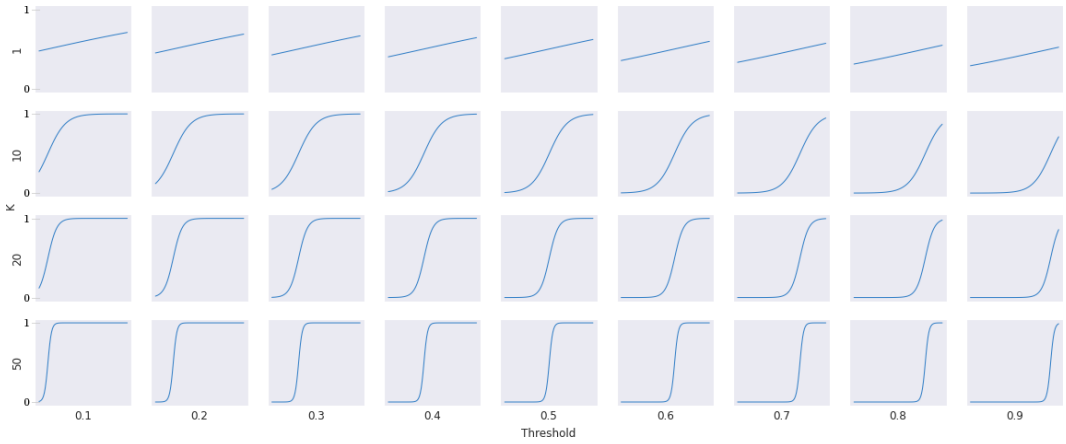

Figure 1: Sigmoid approximation of the Heaviside step function by k and τ. With increased values of k, the approximation becomes closer to the Heaviside step function. As k decreases, the number τ values over which the approximation diverges from the Heaviside step function at the limit increases.

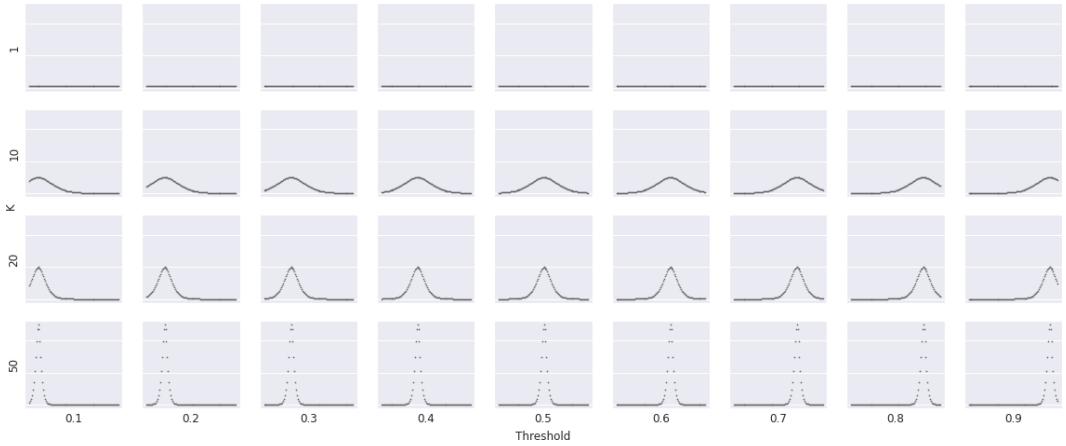

Figure 2: Derivative of the sigmoid approximation of the Heaviside step function by k and τ. As k increases, the range of values with zero gradient increases.

makes optimization via backpropagation more stable. We specify these two points to be equal to δ and $1-\delta$ when the inputs are halfway the distance between τ and the nearest endpoint (0 or 1).

Let τ_m be the distance between τ and the nearest endpoint of $\mathcal{H}^l$:

$$\tau_m = \min\{\tau, 1-\tau\} \tag{12}$$

Then, we define the two new points at which $\mathcal{H}^l(p,\tau)$ is defined as:

$$\mathcal{H}^l(p,\tau) = \begin{cases} \delta & \text{if } p = \tau - \frac{\tau_m}{2} \\ 1-\delta & \text{if } p = \tau + \frac{\tau_m}{2} \end{cases} \tag{13}$$

The parameter δ is chosen such that the function, $\mathcal{H}^l$ remains non-decreasing, limiting the range of δ to $[0, 0.5]$. Note that, as shown in Figure 3 (left), a choice of $\delta = 0$ would result in gradient equal to zero between $p = 0$ and $p = \tau - \frac{\tau_m}{2}$ as well as between $p = \tau + \frac{\tau_m}{2}$ and $p = 1$. As shown in Figure 3 (middle), a choice of $\delta = 0.5$ would result in a gradient equal to zero between $p = \tau - \frac{\tau_m}{2}$ and $p = \tau + \frac{\tau_m}{2}$. We empirically determined that a value in the range of $0.1 \leq \delta \leq 0.2$ works well in practice. A linear approximation with $\delta = 0.1$ is shown in Figure 3 (right).

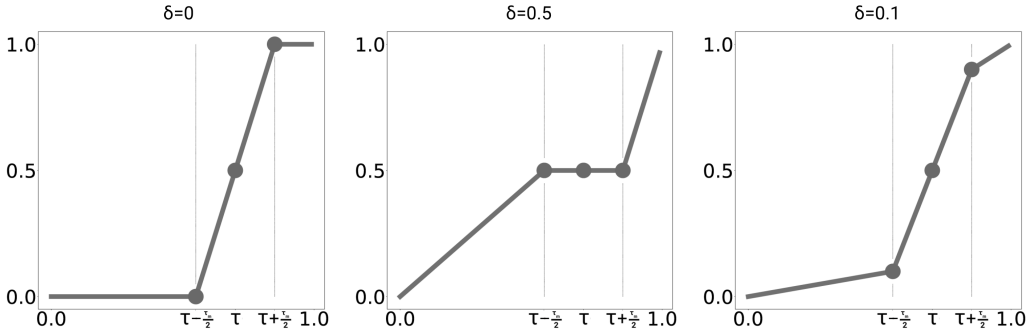

Figure 3: Linear Heaviside approximation $\mathcal{H}^l$ with δ values of 0 (left), 0.5 (middle), and 0.1 (right).

In total, the following five points for $\mathcal{H}^l$ have now been defined as follows:

$$\mathcal{H}^l(p,\tau) = \begin{cases} 0 & \text{if } p = 0 \\ \delta & \text{if } p = \tau - \frac{\tau_m}{2} \\ 0.5 & \text{if } p = \tau \\ 1-\delta & \text{if } p = \tau + \frac{\tau_m}{2} \\ 1 & \text{if } p = 1 \end{cases} \tag{14}$$

The property of $\mathcal{H}^l(p=\tau,\tau) = 0.5$ is now satisfied by our formulation. We can use the points to solve for three line segments that fully define $\mathcal{H}^l$. We denote the slope and intercept of each segment from left to right as m_1, m_2, m_3 and b_1, b_2, b_3 respectively. These slopes, from left to right, are:

$$m_1 = \frac{\delta}{\tau - \frac{\tau_m}{2}} \qquad m_2 = \frac{1-2\delta}{\tau_m} \qquad m_3 = \frac{\delta}{1-\tau-\frac{\tau_m}{2}}$$

We can solve for the first intercept, b_1, with $p = \tau - \frac{\tau_m}{2}$:

$$\begin{aligned} \delta &= m_1(\tau - \frac{\tau_m}{2}) + b_1 \\ b_1 &= \delta - (\frac{\delta}{\tau - \frac{\tau_m}{2}})(\tau - \frac{\tau_m}{2}) \\ b_1 &= 0 \end{aligned}$$

We then solve for the second intercept, b_2, with $p = \tau$:

$$\begin{aligned} 0.5 &= m_2\tau + b_2 \\ b_2 &= 0.5 - m_2\tau \end{aligned}$$

Finally, we solve for the third intercept, b_3, with $p = \tau + \frac{\tau_m}{2}$:

$$\begin{aligned} 1 - \delta &= m_3(\tau + \frac{\tau_m}{2}) + b_3 \\ b_3 &= 1 - \delta - m_3(\tau + \frac{\tau_m}{2}) \end{aligned}$$

Therefore the linear approximation is:

$$\mathcal{H}^l = \begin{cases} p \cdot m_1 & \text{if } p < \tau - \frac{\tau_m}{2} \\ p \cdot m_3 + (1 - \delta - m_3(\tau + \frac{\tau_m}{2})) & \text{if } p > \tau + \frac{\tau_m}{2} \\ p \cdot m_2 + (0.5 - m_2\tau) & \text{otherwise} \end{cases} \tag{15}$$

3 Computational Efficiency

At training time, an objective function with our proposed method has a runtime linear with regard to the number of samples. This is a large improvement over the adversarial method of Fathony and Kolter [2] which has, at best, cubic runtime. In practice however, optimizing for a metric with $\mathcal{H}$ over all samples in each batch could lead to increased run time due to the number of constant-time operations required to compute the metric. To mitigate this and further minimize run time, we observe that our proposed $\mathcal{H}$ can be replaced with an reasonably-sized $O(1)$ lookup table by truncating p to several decimal places and precomputing $\mathcal{H}$ for values of p and τ over the range $[0, 1]$. For example, if an interval between values of τ is set to 0.1 and p is truncated at two decimal places, the lookup table has only 1,000 elements. Using 8-bit storage, this table consumes 1kB of memory.

4 Experimental Setup

4.1 Datasets

We report results of experiments on four binary classification datasets of tabular data with varying levels of class imbalance applicable to a wide range of domains (Section 5.1 of the main paper). We also report results of experiments on two different binary classification datasets created from a commonly used image dataset (Section 5.2 of the main paper) and results of experiments on graph data (Section 5.3 of the main paper).

For tabular datasets, features were centered and scaled to unit variance, and the data was split into separate train (64%), test (20%) and validation (16%) sets. For image datasets, the data was normalized and split into separate train (67%), test (17%), and validation (17%) sets.

4.1.1 Synthetic Datasets

Two synthetic datasets were generated, named "Synthetic 33%" "Synthetic 5%" with a positive to negative sample balance of 33% and 4.76%, respectively. Each dataset was formed by creating two isotropic gaussian blobs and removing a randomly-sampled proportion of the positive data points.

4.1.2 CocktailParty Dataset

The CocktailParty Dataset[1] [15] consists of annotations for the locations and orientations of six people in a physical space. The task is to predict whether or not two individuals are part of the same

[1] https://tev.fbk.eu/technologies/cocktailparty-dataset-multi-view-dataset-social-behavior-analysis

conversational group. We use a total of 22 spatial properties describing the locations and orientations of the two individuals as well as those of the other four individuals around them. These features were extracted for every pair of individuals in every frame of the original dataset, resulting in 4800 samples being obtained from 320 frames (320×15 possible pairings = 4800 data points). The spatial coordinates of each data point were chosen such that the individuals in the pair lie along the x-axis with the origin halfway between them. The final dataset has a 30.29% positive class balance.

4.1.3 Adult Data Set

The Adult Data Set[2] from the UCI Machine Learning Repository [1] consists of data extracted from the 1994 Census database with the intended task of predicting whether a person makes more than \$50K per year. This dataset has 14 features of which 1 feature, indicating the number of people represented by the data point, was removed as it has no bearing on the labeled outcome. This dataset contains 48842 points of which 11687 are positive resulting in a 23.93% positive class balance.

4.1.4 Mammography Dataset

The binary classification data for microcalcifications in the Mammography dataset [13], available from OpenML,[3] is composed of 6 features, all of which were considered in our experiments. This dataset has 11183 total samples, 260 of which are positive making it imbalanced. The dataset has only 2.32% positive-class examples.

4.1.5 Kaggle Credit Card Fraud Detection

The Kaggle Credit Card Fraud Detection dataset [12] consists of transaction data for European card users over two days in September of 2013. It has data for 284807 total transactions and 492 instances of fraud. These positive samples, corresponding to cases of fraud, result in a 0.17% positive-sample balance. The Kaggle dataset has 28 unnamed features as well as two more named features: transaction time and amount. The time feature was removed to avoid learning correlation between time step and label. Amount was log-scaled due to the wide range in values resulting in a total of 29 features.

4.1.6 Image Datasets

The CIFAR-10 dataset[4] [7] contains sixty thousand images evenly distributed across ten mutually exclusive classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. Each image is a three-channel color (RGB) image with a size of 32x32 pixels. We created two binary datasets from CIFAR-10, discussed below.

CIFAR-10-Transportation: CIFAR-10-Transportation was created to achieve 40% class balance. If the image was labeled as an airplane, automobile, ship, or truck, it was considered to be in the positive class.

CIFAR-10-Frog: CIFAR-10-Frog was created to achieve 10% class balance. Only images labeled as a frog were considered the positive class.

4.2 Architecture and Training

Our experiments aim to fairly evaluate our method using different objective functions. Therefore, the same network architecture and training scheme was used unless otherwise noted. The binary classifier for tabular data was a feedforward neural network consisting of three fully connected layers of 32 units, 16 units, and 1 unit. The first two layers were activated via Rectified Linear Unit (ReLU) [9] and followed by dropout [4]. The final layer was a sigmoid-activated single-unit output. For image datasets, the Tiny Darknet [11] architecture was used. The ADAM optimizer [6] was used for training with $lr = 0.001$ and batch size of 1024 for tabular datasets, and $lr = 0.0001$ and batch size of 128 for image datasets. The same batch size and learning rate were used for all methods except [2], which used hyperparameters (e.g., batch size of 20) suggested by the first author through personal communication. Early stopping was used to terminate training after the validation loss stopped

[2] https://archive.ics.uci.edu/ml/machine-learning-databases/adult/

[3] https://www.openml.org/d/310

[4] https://www.cs.toronto.edu/ kriz/cifar.html

decreasing over a sliding window of 100 epochs. Each model optimized via a specific loss was trained using PyTorch. We provide open source implementations of our proposed approach. Training systems used either an NVIDIA Titan X or RTX 2080ti GPU with an Intel i7 3.7GHz processor and 32GB of RAM, with the exception of the adversarial approach to optimize F_1-Score [2], which had to be run on a CPU due to the author's provided implementation.

4.3 Hyperparameters

Our method introduces two new hyperparameters: τ, and k or δ. We empirically determined values of $\tau = 0.5$, $k = 10$ for $\mathcal{H}^s$, and $\delta = 0.1$ for $\mathcal{H}^l$, which were used for all experiments. Given the results of these experiments on a wide range of problems, we believe these values work well in practice and, therefore, do not require extra effort for tuning them.

The incorporation of an approximation for the Heaviside step function introduces the k or δ parameter, depending on which approximation is chosen. For the sigmoid approximation, as k decreases, the derivative of $\mathcal{H}^s$ becomes smoother, facilitating gradient descent. However, in the limit, the approximation may no longer approach H as discussed previously in Section 2.1 of the Supplementary Material. Similar to the parameter k in the sigmoid approximation, we define a slope parameter, δ, for our proposed $\mathcal{H}^l$. A larger δ provides a smoother derivative but further deviation from H. However, unlike $\mathcal{H}^s$, $\mathcal{H}^l$ always approaches H in the limit. Therefore, the particular choice of δ is less crucial within a reasonable range. We empirically determined that $0.1 \leq \delta \leq 0.2$ works well in practice.

In order to determine an appropriate batch size, we trained and evaluated models on Accuracy and F_1-Score using our method over 10 trials. Both the sigmoid and the piecewise linear Heaviside approximations were used. We considered batch sizes in $\{128, 1024, 2048, 4096\}$ and results are shown in Table 2. Training batch size had a minimal effect on final classifier performance in our experiments and the impact of the choice of approximation varied with class imbalance. The performance of $\mathcal{H}^l$ and $\mathcal{H}^s$ were similar for the more balanced dataset (Synthetic 33%). However, in the case of imbalanced data (Synthetic 5%), the performance of the $\mathcal{H}^l$ was greater than of $\mathcal{H}^s$ when optimizing F_1 over the soft-set confusion matrix and evaluating with the F_1-Score metric. For optimizing Accuracy with our approach, performance evaluated on F_1-Score was zero, due to the network's incentive to maximize Accuracy by predicting only dominant-class samples [3]. We believe that the advantage of the $\mathcal{H}^l$ in some cases is due to its adherence to the key properties mentioned in Section 3.1 of the main paper. For the experiments on tabular data, we chose a batch size of 1024.

5 Experiments on Tabular Data with Sigmoid Heaviside Approximation

Section 5.1 of the main paper presented results with our method by approximating the Heaviside approximation with our proposed linearly interpolated function. Table 1 present results using the sigmoid approximation.

6 Additional Results

6.1 Batch Size

Table 2 provides the results of our experiments exploring the effect of batch size on the performance of our method, using both approximations, compared to the BCE baseline. This experiment informed our choice of the batch size hyperparameter as described in Section 4.3 of the Supplementary Material. Apart from zero F_1-Score for optimizing Accuracy with soft sets on the Synthetic 5% dataset, the results are similar within each metric regardless of dataset, batch size, and Heaviside approximation. Performance on the Synthetic 5% dataset, optimizing F_1^l with soft sets for all batch sizes is comparable, but better than F_1^s.

6.2 Balancing between Precision and Recall

Experimental results for balancing between precision and recall as described in Section 5.4 of the main paper are reported in Table 3 for the remaining three tabular datasets. These supplementary results are similar to those presented in the main paper.

Table 1: Losses (rows): F_1, Accuracy, and AUROC via the proposed method (*) using the sigmoid approximation; F_1-Score† via adversarial approach [2] and AUROC‡ via WMW statistic [14].

		CocktailParty ($\mu \pm \sigma$)			**Adult** ($\mu \pm \sigma$)		
	Loss	F_1-Score	Accuracy	AUROC	F_1-Score	Accuracy	AUROC
(1)	F_1*	0.73 ± 0.02	0.84 ± 0.01	0.81 ± 0.01	0.61 ± 0.04	0.74 ± 0.07	0.77 ± 0.03
(2)	F_1†	0.30 ± 0.06	0.76 ± 0.01	0.60 ± 0.02	0.16 ± 0.02	0.78 ± 0.00	0.55 ± 0.01
(3)	Accuracy*	0.71 ± 0.02	0.85 ± 0.01	0.78 ± 0.01	0.36 ± 0.03	0.81 ± 0.00	0.61 ± 0.01
(4)	AUROC*	0.55 ± 0.02	0.48 ± 0.01	0.60 ± 0.00	0.46 ± 0.01	0.42 ± 0.00	0.59 ± 0.01
(5)	AUROC‡	0.01 ± 0.03	0.70 ± 0.03	0.50 ± 0.00	0.00 ± 0.00	0.76 ± 0.00	0.50 ± 0.00
(6)	BCE	0.70 ± 0.02	0.85 ± 0.01	0.78 ± 0.01	0.26 ± 0.06	0.80 ± 0.01	0.58 ± 0.02
		Mammography ($\mu \pm \sigma$)			**Kaggle** ($\mu \pm \sigma$)		
	Loss	F_1-Score	Accuracy	AUROC	F_1-Score	Accuracy	AUROC
(1)	F_1*	0.50 ± 0.29	0.89 ± 0.16	0.75 ± 0.14	0.81 ± 0.02	1.00 ± 0.00	0.89 ± 0.02
(2)	F_1†	0.46 ± 0.08	0.98 ± 0.00	0.66 ± 0.04	0.76 ± 0.06	1.00 ± 0.00	0.83 ± 0.04
(3)	Accuracy*	0.00 ± 0.00	0.98 ± 0.00	0.50 ± 0.00	0.72 ± 0.25	1.00 ± 0.00	0.84 ± 0.12
(4)	AUROC*	0.22 ± 0.01	0.34 ± 0.00	0.63 ± 0.01	0.25 ± 0.01	0.33 ± 0.00	0.64 ± 0.00
(5)	AUROC‡	0.00 ± 0.01	0.88 ± 0.12	0.50 ± 0.00	0.00 ± 0.00	0.93 ± 0.15	0.50 ± 0.00
(6)	BCE	0.56 ± 0.11	0.99 ± 0.00	0.71 ± 0.06	0.50 ± 0.33	1.00 ± 0.00	0.73 ± 0.16

7 Additional Comparisons with Other Methods of Binary Classification

7.1 Other Baselines

We compare our method with other baselines on the tabular datasets, shown in Table 4. Included in these supplementary results are neural network classifiers trained on Dice [8], and SVM$^{\text{perf}}$ [5] to which [10] compares. SVM$^{\text{perf}}$ is trained on Errorrate loss (E), F_1 loss (F_1), and AUROC loss (ROC). Our method outperforms the Dice loss in all cases. Our method is comparable or better than SVM$^{\text{perf}}$.

7.2 Weighted Loss Results

A common method of dealing with sample imbalance is weighting. In this section, we show that our method can also be used with weighting. During training, weighted the loss by the amount of class imbalance in each dataset and compared our method with binary cross entropy. More specifically, we computed sample weights W for each dataset between negative (n) and positive (p) samples where positive samples always correspond to the minority class:

$$W_n = \frac{1}{|n|}\frac{|n|+|p|}{2.0} \qquad W_p = \frac{1}{|p|}\frac{|n|+|p|}{2.0}$$

The weights calculated for our datasets are shown in Table 5.

Table 6 shows losses (rows): F_1 and Accuracy (Acc) trained with the linear (l) and sigmoid (s) approximations compared with the traditional binary cross-entropy. Compared to Table 1 in the main paper, in Table 6 the F_1 loss (line 1) decreased in performance and Accuracy loss (line 2) increased in performance. In all cases, our method without weighting performs similarly or better than the BCE baseline with weighting.

Another method of dealing with class imbalance is oversampling. Oversampling is when samples are repeatedly sampled from the minority class until class balance is reached. We applied oversampling to the training split of each dataset. Using this technique we achieved a positive versus negative sample split in each dataset nearer 50/50, detailed in Table 7.

The performance of our method versus the BCE baseline, both with oversampling, is shown in Table 8. In general our method performs better without oversampling. The BCE baseline shows improvement in some cases compared to BCE with weighting. In all cases, our method without oversampling performs similarly or better than the BCE baseline with oversampling.

Table 2: Accuracy and F_1 loss (rows) via the proposed method (*) with the sigmoid (s) and linear (l) approximations by batch sizes $\{128, 1024, 2048, 4096\}$. The Synthetic 5% dataset F_1-Score is zero when trained with Accuracy over soft sets loss. Other results are similar within each metric. See text for details.

Synthetic 5% Dataset					
		Accuracy ($\mu \pm \sigma$)			
	Loss	$B = 128$	1024	2048	4096
(1)	F_1^l*	0.95 ± 0.00	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
(2)	F_1^s*	0.84 ± 0.16	0.78 ± 0.18	0.77 ± 0.18	0.74 ± 0.17
(3)	Accuracy$^l*$	0.95 ± 0.00	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.01
(4)	Accuracy$^s*$	0.95 ± 0.01	0.95 ± 0.00	0.95 ± 0.01	0.96 ± 0.01
(5)	BCE	0.96 ± 0.01	0.95 ± 0.01	0.95 ± 0.01	0.95 ± 0.00
		F_1-Score ($\mu \pm \sigma$)			
	Loss	$B = 128$	1024	2048	4096
(1)	F_1^l*	0.52 ± 0.04	0.50 ± 0.06	0.52 ± 0.05	0.48 ± 0.04
(2)	F_1^s*	0.32 ± 0.18	0.27 ± 0.22	0.24 ± 0.20	0.21 ± 0.17
(3)	Accuracy$^l*$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
(4)	Accuracy$^s*$	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
(5)	BCE	0.22 ± 0.10	0.21 ± 0.11	0.12 ± 0.08	0.09 ± 0.06
Synthetic 33% Dataset					
		Accuracy ($\mu \pm \sigma$)			
	Loss	$B = 128$	1024	2048	4096
(1)	F_1^l*	0.84 ± 0.01	0.84 ± 0.01	0.84 ± 0.00	0.85 ± 0.01
(2)	F_1^s*	0.84 ± 0.01	0.83 ± 0.01	0.83 ± 0.02	0.83 ± 0.01
(3)	Accuracy$^l*$	0.85 ± 0.01	0.85 ± 0.01	0.86 ± 0.01	0.85 ± 0.01
(4)	Accuracy$^s*$	0.85 ± 0.00	0.85 ± 0.01	0.85 ± 0.01	0.85 ± 0.01
(5)	BCE	0.84 ± 0.01	0.85 ± 0.01	0.84 ± 0.01	0.85 ± 0.01
		F_1-Score ($\mu \pm \sigma$)			
	Loss	$B = 128$	1024	2048	4096
(1)	F_1^l*	0.77 ± 0.01	0.78 ± 0.01	0.78 ± 0.01	0.78 ± 0.01
(2)	F_1^s*	0.77 ± 0.01	0.76 ± 0.01	0.77 ± 0.02	0.77 ± 0.01
(3)	Accuracy$^l*$	0.77 ± 0.01	0.77 ± 0.01	0.78 ± 0.01	0.78 ± 0.01
(4)	Accuracy$^s*$	0.77 ± 0.01	0.77 ± 0.01	0.77 ± 0.01	0.77 ± 0.01
(5)	BCE	0.74 ± 0.01	0.74 ± 0.02	0.74 ± 0.01	0.74 ± 0.02

7.3 Image Data

We also compared the performance of our proposed method to neural network classifiers trained with Dice as the loss function [8]. Results from Dice (logit only) and Dice^s (sigmoid output) are in Table 9. Our method greatly outperformed Dice and DICE^s, likely due to the fact that Dice is designed for image segmentation, rather than binary classification.

Table 3: Optimizing F_β-Scores ($\beta = \{1, 2, 3\}$) via our method to balance between precision and recall while maximizing F_1-Score.

			Adult ($\mu \pm \sigma$)			
	Loss	F_1-Score	F_2-Score	F_3-Score	Precision	Recall
(1)	F_1*	0.64 ± 0.01	0.72 ± 0.03	0.75 ± 0.05	0.54 ± 0.04	0.78 ± 0.07
(2)	F_2*	0.43 ± 0.02	0.65 ± 0.02	0.79 ± 0.01	0.27 ± 0.02	1.00 ± 0.00
(3)	F_3*	0.39 ± 0.01	0.61 ± 0.01	0.76 ± 0.01	0.24 ± 0.01	1.00 ± 0.00
			Kaggle ($\mu \pm \sigma$)			
	Loss	F_1-Score	F_2-Score	F_3-Score	Precision	Recall
(1)	F_1*	0.82 ± 0.04	0.80 ± 0.04	0.80 ± 0.04	0.84 ± 0.04	0.79 ± 0.04
(2)	F_2*	0.81 ± 0.02	0.80 ± 0.03	0.80 ± 0.03	0.83 ± 0.02	0.79 ± 0.03
(3)	F_3*	0.82 ± 0.03	0.81 ± 0.04	0.81 ± 0.04	0.82 ± 0.04	0.81 ± 0.04
			CocktailParty ($\mu \pm \sigma$)			
	Loss	F_1-Score	F_2-Score	F_3-Score	Precision	Recall
(1)	F_1*	0.74 ± 0.01	0.75 ± 0.02	0.75 ± 0.02	0.73 ± 0.03	0.75 ± 0.02
(2)	F_2*	0.68 ± 0.02	0.80 ± 0.01	0.85 ± 0.01	0.54 ± 0.03	0.91 ± 0.01
(3)	F_3*	0.60 ± 0.02	0.77 ± 0.01	0.85 ± 0.01	0.43 ± 0.02	0.95 ± 0.02

Table 4: Other baselines on tabular data. See text for details.

		CocktailParty ($\mu \pm \sigma$)			**Adult** ($\mu \pm \sigma$)		
	Loss	Accuracy	F_1-Score	AUROC	Accuracy	F_1-Score	AUROC
(1)	Dice	0.68 ± 0.04	0.49 ± 0.02	0.56 ± 0.02	0.50 ± 0.15	0.40 ± 0.02	0.55 ± 0.02
(2)	SVM_E^{perf}	0.82	0.66	0.75	0.81	0.60	0.75
(3)	SVM_{F1}^{perf}	0.78	0.69	0.78	0.51	0.48	0.66
(4)	SVM_{ROC}^{perf}	0.76	0.67	0.77	0.32	0.41	0.56
		Mammography ($\mu \pm \sigma$)			**Kaggle** ($\mu \pm \sigma$)		
	Loss	Accuracy	F_1-Score	AUROC	Accuracy	F_1-Score	AUROC
(1)	Dice	0.85 ± 0.17	0.19 ± 0.09	0.65 ± 0.06	0.67 ± 0.15	0.10 ± 0.06	0.76 ± 0.06
(2)	SVM_E^{perf}	0.98	0.51	0.68	1.00	0.80	0.89
(3)	SVM_{F1}^{perf}	0.98	0.59	0.82	1.00	0.80	0.90
(4)	SVM_{ROC}^{perf}	0.61	0.11	0.77	0.67	0.01	0.83

Table 5: Dataset sample weights. See text for details.

Dataset	Negative	Positive
CocktailParty	0.72	1.65
Adult	0.66	2.07
Mammography	0.51	21.55
Kaggle	0.50	290.25

Table 6: Weighted loss results. See text for details.

		CocktailParty ($\mu \pm \sigma$)		**Adult** ($\mu \pm \sigma$)	
	Loss	F_1-Score	Accuracy	F_1-Score	Accuracy
(1)	F_1^l	0.72 ± 0.01	0.80 ± 0.01	0.45 ± 0.03	0.41 ± 0.06
(2)	F_1^s	0.71 ± 0.03	0.77 ± 0.06	0.42 ± 0.02	0.36 ± 0.05
(3)	Accuracy^l	0.75 ± 0.02	0.83 ± 0.01	0.57 ± 0.04	0.66 ± 0.06
(4)	Accuracy^s	0.72 ± 0.02	0.82 ± 0.02	0.54 ± 0.03	0.61 ± 0.06
(5)	BCE	0.75 ± 0.02	0.85 ± 0.01	0.56 ± 0.05	0.80 ± 0.02
		Mammography ($\mu \pm \sigma$)		**Kaggle** ($\mu \pm \sigma$)	
	Loss	F_1-Score	Accuracy	F_1-Score	Accuracy
(1)	F_1^l	0.31 ± 0.04	0.90 ± 0.01	0.13 ± 0.02	0.98 ± 0.01
(2)	F_1^s	0.28 ± 0.05	0.88 ± 0.04	0.41 ± 0.17	0.99 ± 0.01
(3)	Accuracy^l	0.34 ± 0.04	0.92 ± 0.01	0.42 ± 0.08	1.00 ± 0.00
(4)	Accuracy^s	0.35 ± 0.04	0.92 ± 0.01	0.38 ± 0.11	0.99 ± 0.00
(5)	BCE	0.43 ± 0.06	0.95 ± 0.01	0.21 ± 0.05	0.99 ± 0.00

Table 7: Data balancing via Oversampling. See text for details.

	Entire Dataset			Oversampled Train Split		
Dataset	Total	Positive		Total	Positive	
CocktailParty	4800	1454	30.29%	4284	2142	50%
Adult	48842	11687	23.93%	39564	19782	50%
Mammography	11183	260	2.32%	13970	6985	50%
Kaggle	284807	492	0.17%	363894	181947	50%

Table 8: Oversampling Results. See text for details.

		CocktailParty ($\mu \pm \sigma$)		**Adult** ($\mu \pm \sigma$)	
	Loss	F_1-Score	Accuracy	F_1-Score	Accuracy
(1)	F_1^l	0.72 ± 0.02	0.80 ± 0.01	0.45 ± 0.02	0.42 ± 0.05
(2)	F_1^s	0.71 ± 0.02	0.79 ± 0.01	0.43 ± 0.02	0.39 ± 0.04
(3)	Accuracy^l	0.74 ± 0.02	0.83 ± 0.02	0.55 ± 0.06	0.62 ± 0.10
(4)	Accuracy^s	0.73 ± 0.03	0.82 ± 0.02	0.53 ± 0.04	0.59 ± 0.07
(5)	BCE	0.75 ± 0.02	0.85 ± 0.01	0.59 ± 0.03	0.81 ± 0.02
		Mammography ($\mu \pm \sigma$)		**Kaggle** ($\mu \pm \sigma$)	
	Loss	F_1-Score	Accuracy	F_1-Score	Accuracy
(1)	F_1^l	0.38 ± 0.07	0.93 ± 0.02	0.73 ± 0.06	1.00 ± 0.00
(2)	F_1^s	0.06 ± 0.03	0.48 ± 0.04	0.69 ± 0.07	1.00 ± 0.00
(3)	Accuracy^l	0.49 ± 0.07	0.96 ± 0.01	0.75 ± 0.04	1.00 ± 0.00
(4)	Accuracy^s	0.49 ± 0.06	0.96 ± 0.01	0.74 ± 0.04	1.00 ± 0.00
(5)	BCE	0.53 ± 0.04	0.97 ± 0.00	0.57 ± 0.06	1.00 ± 0.00

Table 9: Losses (rows): DICE and DICE^s optimize the DICE coefficient. DICE is logit-only, and DICE^s uses a sigmoid output layer.

CIFAR-10-Transportation Results ($\mu \pm \sigma$)			
	Loss	Accuracy	F_1-Score
(6)	*DICE*	0.521 ± 0.02	0.625 ± 0.01
(7)	*DICE*^{s}	0.546 ± 0.04	0.638 ± 0.02

CIFAR-10-Frog Results ($\mu \pm \sigma$)			
	Loss	Accuracy	F_1-Score
(6)	*DICE*	0.230 ± 0.02	0.205 ± 0.00
(7)	*DICE*^{s}	0.277 ± 0.04	0.216 ± 0.01

References

[1] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[2] Rizal Fathony and Zico Kolter. Ap-perf: Incorporating generic performance metrics in differentiable learning. In *AISTATS*, pages 4130–4140, 2020.

[3] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *TKDE*, 21(9):1263–1284, 2009.

[4] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, 2012.

[5] Thorsten Joachims. Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226, 2006.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *ICLR*, 2015.

[7] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[8] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, 2016.

[9] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814, 2010.

[10] Harikrishna Narasimhan, Rohit Vaish, and Shivani Agarwal. On the statistical consistency of plug-in classifiers for non-decomposable performance measures. In *NeurIPS*, pages 1493–1501, 2014.

[11] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[12] Machine Learning Group UBL. Credit card fraud detection. `https://www.kaggle.com/mlg-ulb/creditcardfraud`. Accessed: 2022.

[13] Kevin S Woods, Jeffrey L Solka, Carey E Priebe, Chris C Doss, Kevin W Bowyer, and Laurence P Clarke. Comparative evaluation of pattern recognition techniques for detection of microcalcifications. In *Biomedical Image Processing and Biomedical Visualization*, 1993.

[14] Lian Yan, Robert H Dodier, Michael Mozer, and Richard H Wolniewicz. Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *ICML*, 2003.

[15] Gloria Zen, Bruno Lepri, Elisa Ricci, and Oswald Lanz. Space speaks: towards socially and personality aware visual surveillance. In *Proceedings of the 1st ACM international workshop on Multimodal pervasive video analysis*, pages 37–42, 2010.