GENERALIZING CROSS ENTROPY LOSS WITH A BETA PROPER COMPOSITE LOSS: AN IMPROVED LOSS FUNCTION FOR OPEN SET RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

Open set recognition involves identifying data instances encountered during test time that do not belong to known classes in the training set. The majority of recent deep learning approaches to open set recognition use a cross entropy loss to train their networks. Surprisingly, other loss functions are seldom used. In our work, we explore generalizing cross entropy with a Beta loss. This Beta loss is a proper composite loss with a Beta weight function. This weight function adds the flexibility of putting more emphasis on different parts of the observation-conditioned class probability (i.e. $P(Y|\boldsymbol{X})$) range during training. We show that the flexibility gained through this is Beta loss function produces consistent improvements over cross entropy loss for open set recognition and produces state of the art results relative to recent methods.

1 INTRODUCTION

Machine learning systems deployed in the real world often encounter data instances that do not belong to any of the classes seen during training. For example, autonomous cars with visual perception systems must handle, to some degree, the unmodeled object classes that could be encountered. Failure to identify unknown classes not only presents a safety risk, but it also hinders performance if the input domain changes significantly. Thus, identifying these novel instances is a critical task for machine learning systems to achieve robustness in the real world, especially in high-stakes settings (Dietterich, 2017).

Open Set Recognition (OSR) (Scheirer et al., 2013; Boult et al., 2019) attempts to addresses this problem of identifying instances belonging to previously unseen classes. Deep learning methods for OSR typically learn a representation such that data instances from novel classes are separable from known classes under this representation. Almost all of the recent deep learning OSR methods use a cross entropy loss combined with a function (e.g. softmax or sigmoid) to map unnormalized logits to probabilities. Surprisingly, other loss functions have not been extensively explored for OSR despite known issues with the use of cross entropy on representation learning for OSR (Goodfellow et al., 2015; Bendale & Boult, 2016).

In order to improve on cross entropy, we explore its properties as a special case of a proper composite loss (Buja et al., 2005; Gneiting & Raftery, 2007; Reid & Williamson, 2010). Proper composite losses have an integral representation that reveals an implicit weight function over the observation-conditioned class probability range (i.e. the range of $P(Y|\boldsymbol{X})$ where Y is the output and $\boldsymbol{X}$ are the features). As we will show, cross-entropy has a U-shaped weight function that places higher weight at the extreme ends of the $P(Y|\boldsymbol{X})$ range. It is unclear if this weight function is the best option for OSR. To our knowledge, there is no prior work that investigates changing this weight function to improve OSR.

In our work, we introduce a loss function specifically designed for OSR based on a proper composite loss with a Beta function used as the weight function. The Beta function is parameterized by α and β parameters which results in a generalization of the weight function used by cross entropy. By changing these two parameters, we change the shape of the weight function, thereby creating a class of loss functions that can be tailored to the specific OSR task.

We call this loss a *Beta loss* and introduce a constrained optimization problem to set its parameters during training. The flexibility gained in tuning the weight function for a given OSR task results in consistent gains in performance over cross entropy. In addition, we demonstrate that these gains result in state of the art performance compared to recent OSR methods.

2 Related Work

Scheirer et al. (2013) provided the first formalization of the OSR problem, including defining open set risk as the risk of misclassifying data that is far away from the training data. Since then, the literature on OSR has been extensive (see (Boult et al., 2019) for a survey) and we provide a brief survey with a focus on recent deep learning methods. One of the first deep learning OSR techniques is OpenMax (Bendale & Boult, 2016), which rescales activation vectors, particularly the extreme ones, through a Weibull distribution to reduce the open set risk. Another category of OSR methods leverage autoencoders for open set detection, including using Extreme Value Theory to model reconstruction error from class conditioned autoencoders (Oza & Patel, 2019), mapping classes into multivariate Gaussians in the latent space using a class-conditional VAE (CVAE) framework Sun et al. (2020) or combining CVAEs with a CapsuleNetwork (Guo et al., 2021). The Classification-Reconstruction learning for Open-Set Recognition (CROSR) method (Yoshihashi et al., 2019) trains a deep hierarchical reconstruction net to perform both reconstruction and supervised classification. Reciprocal Points Learning (RPL) (Chen et al., 2020) introduces the idea of reciprocal points, which are parts of the embedding space that are not from class k and thus more similar to the unknown classes. For each class, the RPL algorithm attempts to separate the space of known instances from the reciprocal points while regularizing this objective with a term that indirectly reduces the open space risk.

A closely related task to OSR is Out Of Distribution (OOD) detection. The task in OOD detection is to detect data instances that are not in the training distribution. In past work, this task often involves detecting data instances that come from a totally different dataset, rather than detecting unseen classes from the same dataset. A simple OOD technique is to detect open set instances by thresholding the maximum softmax probability (Hendrycks & Gimpel, 2017). ODIN builds on Hendrycks & Gimpel (2017) by applying temperature scaling to softmax and adding small perturbations to data instances to improve the max softmax score. Generalized ODIN (G-Odin) (Hsu et al., 2020) further improves on ODIN by removing the need for OOD data to tune hyperparameters. We believe that OOD can be cast as a subproblem under the more general OSR framework, and we refer to the general problem of identifying previously unseen test data as OSR.

The most closely related approach to our work is the focal loss (Lin et al., 2020), which was originally intended to handle imbalanced classes. The focal loss modifies cross entropy to be $(1-p)^{\gamma} log(p)$, where γ is a focusing parameter that focuses training on hard instances that are misclassified. We will show that our Beta loss is different and substantially outperforms focal loss.

The majority of these prior approaches use cross entropy for the loss function. To our knowledge, none of the previous work has investigated changing the weight function of cross entropy to improve OSR. Thus, our work investigates a previously unexplored direction for OSR. Finally, we mention other strategies from OSR and OOD that are orthogonal to our approach and could be combined with our work to improve performance even further. These strategies used include incorporating an auxiliary dataset of outliers (Hendrycks et al., 2019; Liu et al., 2020), using generative models to create informative synthetic examples that augment the training data (Ge et al., 2017; Neal et al., 2018; Yue et al., 2021) and using self-supervision to guide the latent representation learning (Perera et al., 2020; Tack et al., 2020).

3 Background

We first introduce the notation that we will use. Vectors and matrices are denoted with boldface font while scalars are not. Let the training set $\mathscr{D} = \{(\boldsymbol{X_1}, \boldsymbol{Y}_1,), \ldots, (\boldsymbol{X_N}, \boldsymbol{Y}_N)\}$ be made up of N training examples. The ith training instance $(\boldsymbol{X_i}, \boldsymbol{Y}_i)$ consists of a D-dimensional feature vector $\boldsymbol{X_i} = (X_i^1, \ldots, X_i^D)$ and a class label vector $\boldsymbol{Y_i} = (Y_i^1, \ldots, Y_i^K)$ that is a one-hot encoding vector of size K, corresponding to the K known classes during training. In this vector, if the ith instance belongs to class k, then $Y_i^k = 1$ and all other components have value 0. If we refer to features in

general, we use the notation $\boldsymbol{X}$ (without the subscript index). Similarly, we use $\boldsymbol{Y}$ to refer to class labels when we do not refer to a specific instance. Finally, the notation $-k$ indicates "not from class k".

During testing, there are an additional U classes not seen during training that data instances can be drawn from. The OSR task is to classify whether a data instance belongs to one of the K known classes or whether the instances come from an unseen class.

3.1 Proper Composite Losses

We briefly provide an explanation of proper composite losses, following the presentation by Reid & Williamson (2010) with some minor notational modifications. For more details, we refer the reader to (Reid & Williamson, 2010; Buja et al., 2005). To simplify the explanation of proper composite losses, for the entirety of Section 3.1, Y is a binary output variable taking value 0 or 1 (rather than dealing with the K-component class label vector $\boldsymbol{Y}$).

Proper losses Let $\eta(\boldsymbol{X},\boldsymbol{\theta}) = P(Y = 1|\boldsymbol{X},\boldsymbol{\theta})$ be the class probability function. Here, $\boldsymbol{\theta}$ are the parameters of the model for the class probability function. We will drop $\boldsymbol{\theta}$ from the notation when it is not important for the discussion and simply denote the class probability function as $\eta(\boldsymbol{X})$. We make η an explicit function of the features $\boldsymbol{X}$ to emphasize that it takes features as input.

We want an estimate of the class probability $\eta(\boldsymbol{X})$ from training data by fitting a model $q(\boldsymbol{X})$ using a loss function $\ell(Y, q(\boldsymbol{X}))$, where $Y = \{0,1\}$ is a binary response variable. The loss function $\ell(Y,V)$ can be defined in terms of two partial loss functions $\ell_1(V) = \ell(1,V)$ and $\ell_0(V) = \ell(0,V)$ for when the ground truth label is 1 and 0 respectively. Specifically $\ell(Y,V) = [\![y = 1]\!]\ell_1(V) + [\![y = 0]\!]\ell_0(V)$. For example, the log loss used in binary cross entropy is defined as $\ell(Y,\eta(\boldsymbol{X})) = -Y\,log(\eta(\boldsymbol{X}) - (1-Y)\,log(1-\eta(\boldsymbol{X}))$ with partial losses $\ell_1(\eta(\boldsymbol{X})) = -log(\eta(\boldsymbol{X}))$ and $\ell_0(\eta(\boldsymbol{X})) = -log(1-\eta(\boldsymbol{X}))$.

Given a loss function, we can define the *point-wise risk* as:

$$L(\eta(\boldsymbol{X}),V) = E_{Y\sim\eta(\boldsymbol{X})}[\ell(Y,V)] = \eta(\boldsymbol{X})\ell_1(V) + (1-\eta(\boldsymbol{X}))\ell_0(V)$$

The point-wise risk is the expectation of the loss with the labels Y being drawn from a Bernoulli distribution with parameter $\eta(\boldsymbol{X})$. Let $\eta(\boldsymbol{X})$ be the true value of the class probability conditioned on $\boldsymbol{X}$. A *proper loss* (also known as a proper scoring rule in Statistics) is a loss $\ell(Y,V)$ with the property that the point-wise risk $L(\eta(\boldsymbol{X}),\hat{\eta}(\boldsymbol{X}))$ is minimized by $\hat{\eta}(\boldsymbol{X}) = \eta(\boldsymbol{X})$ and is said to be Fisher consistent (Gneiting & Raftery, 2007).

Proper losses have an integral representation that reveals the use of a weight function (Shuford et al., 1966; Savage, 1971; Schervish, 1989). In order to describe this integral, we first define the cost-weighted misclassification loss, with cost parameter $c \in (0,1)$ as:

$$\ell_c(0,\hat{\eta}(\boldsymbol{X})) = c[\![\hat{\eta}(\boldsymbol{X}) \geq c]\!] \tag{1}$$

$$\ell_c(1,\hat{\eta}(\boldsymbol{X})) = (1-c)[\![\hat{\eta}(\boldsymbol{X}) < c]\!] \tag{2}$$

Equation 1 is the cost of a false positive (i.e. c) while Equation 2 is the cost of a false negative (i.e. $(1-c)$).

With this definition, a proper loss $\ell(Y,\hat{\eta}(\boldsymbol{X}))$ can be represented as a weighted integral of cost-weighted misclassification losses (Theorem 6 of Reid & Williamson (2010)):

$$\ell(Y,\hat{\eta}(\boldsymbol{X})) = \int_0^1 \ell_c(Y,\hat{\eta}(\boldsymbol{X}))\,w(c)\,dc \tag{3}$$

In Equation 3, $w(c)$ is a weight function $w : (0,1) \mapsto [0,\infty)$ that satisfies $\int_\epsilon^{1-\epsilon} w(c)\,dc < \infty, \forall \epsilon > 0$. A more intuitive representation of this property can be found in the corollary to Theorem 6 in Reid & Williamson (2010), which illustrates the effect of the weight function on the partial losses:

$$\ell_1(\hat{\eta}(\boldsymbol{X})) = \int_{\hat{\eta}(\boldsymbol{X})}^1 (1-c)\,w(c)\,dc \tag{4}$$

$$\ell_0(\hat{\eta}(\boldsymbol{X})) = \int_0^{\hat{\eta}(\boldsymbol{X})} c\,w(c)\,dc \tag{5}$$

Thus, if certain values of c produce high weight according to $w(c)$, they will have more influence on the partial losses, thereby emphasizing the importance of these values. Since the range of c is the same as that of $P(Y|\boldsymbol{X})$, we can view $w(c)$ as focusing in different parts of this probability range. A large variety of commonly used loss functions can be produced by changing the weight function (see Table 1 of Reid & Williamson (2010)). The key to our approach will be specializing this weight function to the specific task of OSR.

Proper Composite Losses An inverse link function $\Psi^{-1}(q(\boldsymbol{X}))$ is often needed to map the real-valued output of a model (i.e. $q(\boldsymbol{X})$) to the range $[0,1]$. For example, in logistic regression $q(\boldsymbol{X}) = \boldsymbol{\beta}^T\boldsymbol{X}$ where the inverse link function for log loss is the sigmoid function $\Psi^{-1}(p) = \frac{1}{1+e^{-p}}$, with forward link being the logistic function $\Psi(p) = log(\frac{p}{1-p})$. A loss function that uses an inverse link function is called a *composite loss* because it composes the inverse link function with $q(\boldsymbol{X})$ i.e. $\eta(\boldsymbol{X}) = \Psi^{-1}(q(\boldsymbol{X}))$. We denote a composite loss as $\ell^{\Psi}(Y,V) = \ell(Y,\Psi^{-1}(V))$, along with partial composite losses $\ell_1^{\Psi}(V)$ and $\ell_0^{\Psi}(V)$.

Making a composite loss proper requires accounting for the effects of the link function on the partial losses. If Ψ is a strictly monotone link function and $\ell_1^{\Psi}(V)$ and $\ell_0^{\Psi}(V)$ are differentiable partial losses, then Theorem 10 of Reid & Williamson (2010) states that ℓ^{Ψ} is a *proper* composite loss if and only if:

$$\frac{-\ell_1^{\Psi'}(\hat{\eta}(\boldsymbol{X}))}{1-\hat{\eta}(\boldsymbol{X})} = \frac{\ell_0^{\Psi'}(\hat{\eta}(\boldsymbol{X}))}{\hat{\eta}(\boldsymbol{X})} = \frac{w(\hat{\eta}(\boldsymbol{X}))}{\Psi'(\hat{\eta}(\boldsymbol{X}))} \tag{6}$$

for all $\hat{\eta}(\boldsymbol{X}) \in (0,1)$. Equation 6 shows that once the weight function and the link function are specified, then the partial loss functions are also specified (except for additive constants). More importantly, the weight and link function pairing has implications on the convexity of the loss function (see Sections 15 and 16 of Buja et al. (2005) and Section 6 of Reid & Williamson (2010)).

3.2 The Beta family of proper composite losses

Buja et al. (2005) show how to create a family of proper losses by using a Beta function as the weight function. The beta function is parameterized by α and β and has the following form:

$$w(c) = c^{\alpha-1}(1-c)^{\beta-1} \tag{7}$$

Using Equations 4 and 5, the corresponding partial loss functions are:

$$\ell_1(q) = \int_q^1 c^{\alpha-1}(1-c)^{\beta}dc \tag{8}$$

$$\ell_0(q) = \int_0^q c^{\alpha}(1-c)^{\beta-1}dc \tag{9}$$

Setting α and β to specific values produces many commonly used losses such as boosting loss, cross entropy, squared error loss and misclassification loss. For instance, setting $\alpha = 0$ and $\beta = 0$ produces $\ell_0(q) = -log(1-q)$ and $\ell_1(1-q) = -log(q)$, which results in binary cross entropy. When α and β are integer multiples of $\frac{1}{2}$, the integrals in Equations 9 and 8 have closed form expressions.

Section 15 of Buja et al. (2005) describes conditions for convex loss functions involving the Beta family for weight functions and scaled logistic links $\Psi^{-1}(V) = \frac{1}{1+e^{-V/\sigma}}$ where σ is a scaling parameter. We restate an important corollary from that section as a theorem below.

Theorem 1 *Proper losses in the Beta family and scaled logistic links result in convex composite losses iff* $\alpha, \beta \in [-1,0]$ *and* $\sigma > 0$.

4 Methodology

Our aim is to validate the Beta family of proper composite losses for the task of Open Set Recognition. We apply the Beta family to high dimensional datasets, of images, under the framework of deep neural networks for classification.

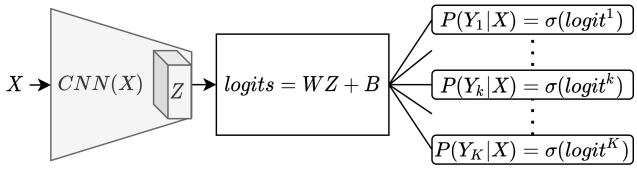

Figure 1: A diagram of the architecture used for Beta loss with a σ =sigmoid inverse link function.

4.1 The Neural Network Architecture

Figure 1 illustrates the neural network architecture that we use in our experiments. The architecture uses a CNN as its backbone and has K heads, corresponding to the K known classes in our training set, connected to a common latent embedding layer. These K heads correspond to K one-vs-all predictions, with the kth output predicting class-k membership of the input and a given output's prediction is unaffected by the value of any other output $logit^{-k}$. The output of the kth head is computed by applying the logistic inverse link function (i.e. $\sigma(X) = (1 + exp(-X))^{-1}$) to the kth logit, resulting in the output $P(Y^k = 1|X) = \sigma(logit^k)$.

This architecture is much simpler than some of the other deep architectures used for OSR. As we will show in Section 5, a simpler architecture with our novel loss function outperforms OSR methods with more sophisticated architectures.

4.2 Beta Loss

During training, each output contributes a one-vs-all loss term to the overall loss. This one-vs-all loss term is calculated using the Beta weight function (Equation 7) combined with a scaled logistic link function with $\sigma = 1$, which results in a sigmoid function. The partial losses $\ell_0(q)$ and $\ell_1(q)$ are shown in Equations 9 and 8. If we keep $\alpha, \beta \in [0,1]$, then according to Theorem 1, this weight-link function combination results in a convex loss for a given output k.

The overall Beta loss (Equation 10) sums up the one-vs-all losses over all the K outputs. Since the sum of convex functions is also convex, the overall Beta loss is also convex.

$$L_{beta}(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{N}\sum_{i=1}^{N}\sum_{k=1}^{K} Y_i^k \ell_1(1 - \hat{\eta}^k(\boldsymbol{X}, \boldsymbol{\theta})) + (1 - Y_i^k)\ell_0(\hat{\eta}^k(\boldsymbol{X}, \boldsymbol{\theta})) \tag{10}$$

In Equation 10, $\hat{\eta}^k(\boldsymbol{X}, \boldsymbol{\theta})$ is the prediction of $P(Y_i^k = 1|\boldsymbol{X}, \boldsymbol{\theta})$ from the output of the neural network. The Beta loss could involve values of α and β that are not multiples of $\frac{1}{2}$ and thus do not have close form expressions for ℓ_1 and ℓ_0. Fortunately, the integrals are one-dimensional and approximating these integrals to an arbitrary precision can be done by applying a simple numerical integration technique like the trapezoidal rule. However, in some cases, a large number of trapezoids may be needed to achieve the desired precision due to the rapid changes at the endpoints of the partial losses for a large number of classes K. To mitigate the problem of high memory usage, we have experimented with training a multi-layer perceptron to approximate the integral; the approximation is promising and we will explore it more fully in future work.

Selecting α and β Selecting the values of α and β before seeing the data can be challenging. Naively optimizing α, β and the neural network parameters $\boldsymbol{\theta}$ simultaneously will produce a meaningless loss function as the optimization will minimize the loss by flattening the weight function towards zero in as many places as possible; in effect, this will cause α and β to approach ∞.

In order to understand the effects of α and β, we used CIFAR10 as a development dataset and apply a posthoc grid search over the two parameters to see their effect on AUC on an OOD task with CIFAR10 data as inliers (see Section 4.6). Table 1 provides a brief overview of our grid search for

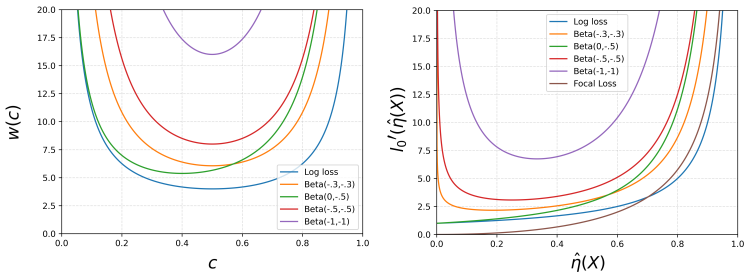

Figure 2: **(Left)**: The Beta weight function for a few values of α and β. **(Right)**: The derivative of partial loss ℓ_0 (the plot for the derivative of ℓ_1 is similar but reflected). Beta losses with lower values of α and β place much more emphasis on instances that are close to 0 (i.e. close to correct predictions) than Focal and Log loss.

the cases where $\alpha = \beta$. In general, our experiments showed that values near $\alpha, \beta = -0.4$ produce the highest AUC values on CIFAR10.

Figure 2 (left) shows a plot of the weight function for a few select parameters of α and β. A plot of the derivative of the partial loss function ℓ_0 (Figure 2 (right)) is also instructive as it shows how Beta functions with parameters close to -0.5 skew the curve shape to be higher towards 0, which effectively increases the area under the curve towards the left for the integral. This shape effectively places more weight on examples that are close to correct i.e. having $\hat{\eta}(\boldsymbol{X}) = P(Y_i^k = 1|\boldsymbol{X})$ be close to 0 for an instance with true label 0. In effect, this weight function emphasizes "core" instances for that class.

We use this information to create a constrained optimization problem that learns the neural network parameters $\boldsymbol{\theta}$ as well as the values for α and β. First, we constrain the values of $\alpha, \beta \in [-1, 0]$ to preserve the convexity requirements from Theorem 1. Second, we add an L1 norm regularization term that penalizes α and β from deviating too much from -1. Without this regularization term, the optimization will always return $\alpha = 0$ and $\beta = 0$ since these values minimize the objective. The λ parameter handles the tension between tethering the α and β parameters to -1 and minimizing the objective, thereby resulting in α and β parameters in-between the endpoints.

$$\begin{aligned} min_{\alpha,\beta,\boldsymbol{\theta}} \quad & L_{beta}(\boldsymbol{Y}, \boldsymbol{X}, \boldsymbol{\theta}) + \lambda * (|\alpha + 1| + |\beta + 1|) \\ \text{s.t.} \quad & -1 \leq \alpha \leq 0, -1 \leq \beta \leq 0 \end{aligned} \tag{11}$$

To set the value of λ, we again perform posthoc tuning on CIFAR10 for the same OOD task to get the value of λ that yields the best AUC for the Beta loss algorithm under investigation (see Table 2). We then apply this value of λ (tuned over CIFAR10) to the other datasets in our evaluation.

4.3 Baseline methods and Detection functions

We compare our OSR results with a large variety of methods, including: Sigmoid (binary cross entropy with a sigmoid link), Softmax (cross entropy followed applied over softmax), Focal Loss (Lin et al., 2020), Openmax (Bendale & Boult, 2016), G-Openmax (Ge et al., 2017), OSRCI (Neal et al., 2018), CROSR (Yoshihashi et al., 2019), C2AE (Oza & Patel, 2019), GFROR (Perera et al., 2020), CGDL (Sun et al., 2020), RPL (Chen et al., 2020), CVCap (Guo et al., 2021), G-Odin (Hsu et al., 2020), and CSI (Tack et al., 2020). For Sigmoid, Softmax, Focal loss, G-Odin and CSI, we ran these algorithms ourselves and report the AUC averaged over 5 random splits. For the other algorithms, we present results as reported in Table 1 from Guo et al. (2021).

$\alpha = \beta$	0.0	-0.1	-0.2	-0.3	-0.4	-0.5	-0.6	-0.7	-0.8	-0.9	-1.0
AUC	0.952	0.955	0.990	0.979	0.990	0.989	0.988	0.981	0.980	0.986	0.978

Table 1: The AUC for different values of α and β on the AUC of CIFAR-10, averaged over the OOD datasets. Due to space constraints, we only include results for $\alpha = \beta$.

λ	0.01	0.05	0.1	0.5	1	5	10
Sigmoid β	0.953	0.967	0.982	0.988	**0.992**	0.975	0.974
Softmax β	0.987	0.989	0.990	0.990	**0.991**	0.979	0.970
G-Odin β	**0.993**	0.979	0.985	0.990	0.987	0.981	0.971
CSI β	0.966	0.966	**0.967**	0.966	0.962	0.964	0.960

Table 2: A posthoc sensitivity analysis of the effect of Beta loss λ on various algorithms' AUC for CIFAR-10, averaged over the five OOD datasets.

Previous methods (Neal et al., 2018; Hendrycks & Gimpel, 2017) that use softmax simply use the probability of the max predicted class to detect open set instances, but we find using L_2 norm of the latent representation z, $S(x) = ||CNN(x)||_2$ improves performance. Unless otherwise noted, our experiments use the L_2 metric as the detecting function for all methods.

For both Generalized ODIN and CSI, we use their proposed functions for OSR as we found these metrics were consistently better than L_2. Generalized ODIN uses the largest $f(x)$ i.e. the largest logit of the K classes before the learned temperature is applied. CSI simply uses the max softmax probability, $S(x) = \underset{y}{argmax}\ P(y|x)$, averaged over four rotations of an input image.

4.4 Replacing loss functions in existing methods with Beta loss

We also investigate replacing the cross entropy loss in existing OSR methods with Beta loss. Although this swap of loss functions may violate the theoretical assumptions of the Beta loss, it gives us an idea of the effect of the Beta loss while keep the rest of the algorithm in tact. In addition to cross entropy plus softmax, we replace cross entropy with Beta loss in two state of the art OSR techniques: G-Odin (Hsu et al., 2020) and CSI (Tack et al., 2020).

G-Odin is an OOD method that normalizes the final prediction layer of a CNN from $f(x) = Wx + b$ to $f(x) = \frac{Wx+b}{||W||||x||}$ for each class, which is normalized again by a single predictive logit $g(x) = sigmoid(W_g x + b_g)$, and finally followed by a softmax activation. Although they also propose an image preprocessing strategy to improve OSR, we found that preprocessing produced consistently worse results than simply using $f(x)$ as an outlier score.

CSI is a self-supervised method for learning a rich embedded representation of a dataset by contrasting a given sample with distributionally-shifted augmentations of itself. While their method generalizes to unsupervised learning, we use their supervised implementation, which fine tunes on the labels of a given dataset.

Other OSR methods from Section 4.3 do not have a standard classification loss that can be easily swapped with our Beta loss. For example, CVCap (Guo et al., 2021) used class labels by minimizing the KL-divergence between a class-conditional representation and a target fixed representation.

To determine the value of λ for the Beta loss versions of Softmax, G-Odin, and CSI, we perform a posthoc sensitivity analysis on λ over the CIFAR10 dataset. As with Sigmoid, we then select the λ value that produces the best AUC on CIFAR10 and apply it to the other datasets in our evaluation. Table 2 shows that the value of lambda that maximizes AUC for CIFAR10 is not always the same for different methods.

4.5 Experimental Setup

For all baseline methods, we use identical hyper-parameter settings and models to the original papers. Experiments described in Section 4.4 follow identical training procedures to the original papers for CSI and G-Odin and only replaced the cross entropy loss with Beta loss. For all of our other experiments, we use a DenseNet (Huang et al., 2017) and follow all the training procedures described by the original work. We use a batch size 64 for 300 epochs with a weight decay of 0.0001 for all weights except on the logits. Models are trained with Stochastic gradient desecent, using a mometum of 0.9, learning rate 0.1, with a learning rate schedule at 50% and 75% epochs. Finally, all weights are initialized with He-initialization (He et al., 2015).

For our high resolution mars datasets, we use transfer learned models, which have been shown to elicit strong performance even on small datasets Huh et al. (2016). We use baseline publicly downloadable pre-trained models: a Resnet-50 (Targ et al., 2016) trained on ImageNet. This model is trained similarly to the DenseNet, but only needed 30 epochs to converge.

4.6 DATASETS

OSR For our small-sized experiments, we use multiple common benchmark datasets, all of which are 32×32 RGB images. For some experiments we use different splits of the training set for alternate analysis, following the setup described in (Neal et al., 2018). CIFAR6 randomly selects 6 classes as inliers to be trained on, with the other 4 considered unknown. CIFAR10+ and CIFAR50+ select the 4 non-animal classes as inlier, then uses the more diverse CIFAR100 to select outliers 10 and 50 animal classes, respectively, are randomly selected. We also use SVHN and TinyImageNet datasets, where 6 and 20 classes are selected from the 10 and 200 possible classes.

OOD We evaluate models trained on the entirety of both cifar10 and cifar100 datasets, and we compare these models against a variety of outlier datasets such as: cropped Imagenet, resized Imagenet, cropped LSUN, resized LSUN, iSUN and SVHN. While we report the averaged AUCs, full results, as well as results on the fixed versions of the resized datasets, are available in the appendix.

Mars We also use a recently released large-sized labeled dataset of Mars Surface images, from the Mars Science Laboratory (MSL) (Grotzinger et al., 2012) and Mars satellite images collected by the High Resolution Imaging Experiment (HiRISE) instrument onboard the Mars Reconnaissance Orbiter (McEwen et al., 2007). Both datasets (Wagstaff et al., 2021) are high-resolution at 227×227, with RGB and greyscale images respectively. The Rover dataset contains $2,900$ images taken on the surface of the planet Mars, split into 19 classes. The Mars satellite dataset contains $10,815$ images sorted 8 classes. We consulted a domain expert from NASA to split the dataset into meaningful openset tasks. For the Rover dataset, we set the "artifact" class (images of low quality or contain missing data) to be unknown and the rest to be inlier. We also split the dataset by "solday", where only classes occurring during the first 100 martian days of data collection are used as inliers ($K = 14$ classes), and the rest are used as outliers. Finally the orbital dataset sets "impact ejecta" and "spider" are used as outliers due to their low occurrence in the full dataset.

5 RESULTS AND DISCUSSION

OSR has a variety of metrics used for determining the effectiveness of different methods. The most popular being area under the receiver operating characteristic curve (AUC) (Boult et al., 2019). AUC provides a calibration-free measure and characterizes the performance for a given score by varying the discrimination threshold between inlier data and openset data. Other metrics exists such as true negative rate at 95% true positive rate (TNR@TPR95), which forces inlier recall to be directly at 95%. While having a high TNR is more challenging than a high AUC, we find the results to be highly correlated. Another metric is F-score, which combines outliers as a $K+1$ class and measures the accuracy. We do not find this metric to be meaningful as it requires a threshhold for choosing outliers, and conflates the difficulties of the OSR task with strict classification. Therefore, we report only AUCs in our main text. TNRs and accuracy are reported in the supplementary material.

Table 3 shows the main results. All experiments are averaged over five random splits of known and unknown classes for both OSR (Neal et al., 2018) and OOD detection (Hendrycks & Gimpel, 2017). We draw attention to three elements. First, adding Beta Loss to each to a given method results in a very consistent improvement in performance for almost all tasks. Second, we achieve start-of-the-art results by using CSI with Beta loss for most OSR tasks, and we achieve start-of-the-art results by using Generalized Odin with Beta Loss on the standard OOD datasets. Lastly, using Beta Loss with the inverse logistic link function, in a one-vs-all classification setting, results in significant improvements over using standard binary cross entropy (Sigmoid β vs Sigmoid).

Table 4 shows the results of applying Beta Loss to the mars datasets. Here we find these datasets to be much more challenging. While no one method achieves the best results for all three mars-OSR tasks, adding Beta loss to a given method generally results in a performance increase, including

	Svhn	Cifar6	Cifar10+	Cifar50+	Tiny Imagenet	Cifar10 avg	Cifar100 avg
Softmax	0.886	0.677	0.816	0.805	0.577	-	-
Openmax	0.894	0.695	0.817	0.796	0.576	-	-
G-Openmax	0.896	0.675	0.827	0.819	0.58	-	-
OSRCI	0.91	0.699	0.838	0.827	0.586	-	-
CROSR	0.899	-	-	-	0.589	-	-
C2AE	0.892	0.711	0.81	0.803	0.581	-	-
GFROR	0.955	0.831	-	-	**0.657**	-	-
CGDL	0.896	0.681	0.794	0.794	0.653	-	-
RPL	0.931	0.784	0.885	0.881	0.624*	-	-
CVCap	0.956	0.835	0.888	0.889	0.558*	-	-
Focal*	0.915	0.830	0.891	0.883	0.636	0.949	0.877
Softmax*	0.931	0.838	0.924	0.917	0.646	0.971	0.890
Sigmoid*	0.890	0.806	0.857	0.855	0.612	0.934	0.730
G-Odin*	0.932	0.859	0.931	0.933	0.641	0.990	0.970
CSI*	**0.958**	0.857	0.967	0.963	0.623	0.977	0.896
Sigmoid β	0.927	0.843	0.939	0.933	0.625	0.989	0.751
Softmax β	0.935	0.837	0.942	0.940	0.646	0.991	0.971
G-Odin β	0.937	0.853	0.935	0.937	0.646	**0.992**	**0.973**
CSI β	**0.958**	**0.874**	**0.975**	**0.967**	0.628	0.988	0.923

Table 3: Full results of AUC averaged over 5 random splits. A Beta loss (indicated with a β) shows a consistent improvement over log loss for nearly every dataset with pure classification loss. These results are mainly significant (see the supplement for full standard deviations, TNR at TPR95, and OOD results). The top third of the table is as reported in (Guo et al., 2021), including the empty cells. * denotes our own runs of the experiment.

a substantial increase of about 0.1 AUC in the case of Softmax β on Artifact. Note that the λ parameter was not tuned on the mars datasets and we may experience an even larger improvement with tuning of λ.

	Artifact	HiRISE	Solday
Sigmoid	0.604	0.627	0.529
Softmax	0.655	0.654	0.583
G-Odin	0.843	0.608	0.523
Sigmoid β	0.615	0.634	**0.596**
Softmax β	0.759	0.640	0.588
G-Odin β	**0.850**	**0.655**	0.532

Table 4: AUC of Mars tasks averaged over 5 trials. Beta loss generally provides a consistent improvement over the baseline loss functions.

6 Conclusion

We have introduced a new loss function for OSR called the Beta loss, which generalizes cross entropy loss. The Beta loss is a proper composite loss with an associated weight function over the $P(Y|\boldsymbol{X})$ range that can be tailored to a specific OSR task. This additional flexibility produces consistent increases in AUC over cross entropy and these increases yield state of the art results relative to recent OSR algorithms.

For future work, we will explore other link functions aside from the logistic link function that still preserve the convexity of the proper composite loss function. It remains an open question if there are other weight and link function combinations that are more effective for OSR than Beta loss.

REFERENCES

Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1563–1572, 2016.

T. E. Boult, S. Cruz, A. Dhamija, M. Gunther, J. Henrydoss, and W. Scheirer. Learning and the unknown: Surveying steps toward open world recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9801–9807, 2019.

A. Buja, W. Stuetzle, and Y. Shen. Loss functions for binary class probability estimation and classification: Structure and applications. Technical report, University of Pennsylvania, Nov 2005.

G. Chen, L. Qiao, Y. Shi, P. Peng, J. Li, T. Huang, S. Pu, and Y. Tian. Learning open set network with discriminative reciprocal points. In *Proc. of European Conference in Computer Vision (ECCV)*, pp. 507–522, 2020.

T. G. Dietterich. Steps toward robust artificial intelligence. *AI Magazine*, 38(3):3–24, 2017.

Z. Ge, S. Demyanov, and R. Garnavi. Generative openmax for multi-class open set classification. In *Proc. of British Machine Vision Conference (BMVC)*, 2017.

T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, March 2007.

I. Goodfellow, J. Shelns, and C. Szegedy. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*, 2015.

John P Grotzinger, Joy Crisp, Ashwin R Vasavada, Robert C Anderson, Charles J Baker, Robert Barry, David F Blake, Pamela Conrad, Kenneth S Edgett, Bobak Ferdowski, et al. Mars science laboratory mission and science investigation. *Space science reviews*, 170(1):5–56, 2012.

Yunrui Guo, Guglielmo Camporese, Wenjing Yang, Alessandro Sperduti, and Lamberto Ballan. Conditional variational capsule network for open set recognition. *CoRR*, abs/2104.09159, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=HyxCxhRcY7`.

Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning, PMLR 97*, pp. 3519–3529, 2019.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(2):318–327, 2020. doi: 10.1109/TPAMI.2018.2858826.

W. Liu, X. Wang, J. Owens, and Y. Li. Energy-based out-of-distribution detection. In *Advances in Neural Information Processing Systems 33*, 2020.

Alfred S McEwen, Eric M Eliason, James W Bergstrom, Nathan T Bridges, Candice J Hansen, W Alan Delamere, John A Grant, Virginia C Gulick, Kenneth E Herkenhoff, Laszlo Keszthelyi, et al. Mars reconnaissance orbiter's high resolution imaging science experiment (hirise). *Journal of Geophysical Research: Planets*, 112(E5), 2007.

Lawrence Neal, Matthew Olson, Xiaoli Fern, Weng-Keen Wong, and Fuxin Li. Open set learning with counterfactual images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 613–628, 2018.

P. Oza and V. M. Patel. C2ae: Class conditioned autoencoder for open-set recognition. In *Proc. of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2302–2311, 2019.

Pramuditha Perera, Vlad I. Morariu, Rajiv Jain, Varun Manjunatha, Curtis Wigington, Vicente Ordonez, and Vishal M. Patel. Generative-discriminative feature representations for open-set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11814–11823, June 2020. doi: 10.1109/CVPR42600.2020.01183.

M. D. Reid and R. C. Williamson. Composite binary losses. *Journal of Machine Learning Research*, 11:2387–2422, Dec 2010.

L. J. Savage. Elicitation of personal probabilities and expectations. *J. of the American Statistical Association*, 66(336):783–801, 1971.

W. J. Scheirer, A. Rocha, A. Sapkota, and T. E. Boult. Towards open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(7):1757–1772, 2013.

M. J. Schervish. A general method for comparing probability assessors. *The Annals of Statistics*, 17 (4):1856–1879, 1989.

E. Shuford, A. Albert, and H. E. Massengill. Admissible probability measurement procedures. *Psychometrika*, 31(2):125–145, June 1966.

X. Sun, Z. Yang, C. Zhang, K.-V. Ling, and G. Peng. Conditional gaussian distribution learning for open set recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13480–13489, 2020.

J. Tack, S. Mo, J. Jeong, and J. Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems 33*, 2020.

Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

Kiri Wagstaff, Steven Lu, Emily Dunkel, Kevin Grimes, Brandon Zhao, Jesse Cai, Shoshanna B Cole, Gary Doran, Raymond Francis, Jake Lee, et al. Mars image content classification: Three years of nasa deployment and recent advances. *arXiv preprint arXiv:2102.05011*, 2021.

Ryota Yoshihashi, Wen Shao, Rei Kawakami, Shaodi You, Makoto Iida, and Takeshi Naemura. Classification-reconstruction learning for open-set recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4016–4025, 2019.

Zhongqi Yue, Tan Wang, Qianru Sun, Xian-Sheng Hua, and Hanwang Zhang. Counterfactual zero-shot and open-set visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15404–15414, 2021.

A FULL EXPERIMENTAL RESULTS

The complete results of all over experiments, averaged over the 5 splits can be found in Tables 6, 7, 8, 9 and 10. Tables 6 and 7 contain all the AUCs and TNR metrics, respectively, for OOD experiments, both for CIFAR10 and CIFAR100. Tables 8 and 9 contain all the AUCs and TNR metrics, respectively, for OSR experiments. Finally, all the average accuracies for both OSR experiments, CIFAR10 and CIFAR100 can be found in Table 10.

A.1 LATENT SPACE COMPARISONS

	linear CKA	rbf kernel CKA
test	0.716	0.764
train	0.758	0.806

Table 5: Comparing the latent representation of a baseline sigmoid model with a beta-weighted sigmoid model $\lambda = 1$ on the cifar10 dataset.

We were also curious as to how much the latent space changes when the loss function changes from binary cross entropy to the Beta loss, with both techniques using a sigmoid function as the inverse link function. To answer this question, we applied the Centered Kernel Alignment (CKA) similarity index (Kornblith et al., 2019) to the latent representations obtained from the two techniques of baseline sigmoid and sigmoid with Beta loss. For each, we got the feature representation of all $50,000$ data instances chosen from the training set and another $10,000$ instances from the test set. Table 5 shows the results of CKA applied between both trainings and test datasets. Linear CKA produced similarity indices of 0.716 (test) and 0.758 (training) while RBF kernel CKA produced similarity indices of 0.764 (test) and 0.806 (training). These results indicate that while most of the latent representations are similar, the relatively smaller amount of latent space changes caused by training with Beta loss do have an impact on OSR performance.

indata	method	Imagenet	Imagenet resize	Imagenet resize fixed	LSUN	LSUN resize	LSUN resize fixed	iSUN	svhn
cifarall	focal	0.949±.013	0.945±.015	0.907±.004	0.948±.014	0.971±.007	0.898±.012	0.960±.012	0.922±.016
cifarall	sigmoid	0.936±.014	0.936±.018	0.875±.006	0.932±.007	0.962±.014	0.868±.010	0.950±.016	0.887±.041
cifarall	softmax	0.976±.007	0.973±.010	0.930±.002	0.979±.006	0.981±.008	0.911±.009	0.979±.010	0.937±.052
cifarall	godin	0.985±.002	0.985±.003	0.944±.002	0.989±.001	0.991±.002	0.930±.007	0.990±.002	0.998±.001
cifarall	CSI	0.983±.013	0.969±.030	0.939±.039	0.981±.008	0.981±.014	0.936±.030	0.978±.005	0.973±.025
cifarall	sigmoid β	0.988±.002	0.986±.003	0.923±.004	0.991±.001	0.990±.001	0.915±.010	0.990±.001	0.990±.007
cifarall	godin β	0.989±.004	0.989±.004	0.944±.001	0.990±.001	0.993±.001	0.936±.001	0.993±.001	0.999±.000
cifarall	softmax β	0.982±.003	0.978±.005	0.926±.001	0.989±.001	0.981±.004	0.902±.002	0.981±.004	0.998±.001
cifarall	CSI β	0.993±.002	0.982±.009	0.942±.036	0.996±.002	0.986±.005	0.943±.028	0.986±.005	0.984±.014
cifar100	focal	0.876±.020	0.856±.033	0.763±.004	0.857±.011	0.888±.028	0.702±.003	0.871±.027	0.916±.021
cifar100	sigmoid	0.738±.049	0.698±.066	0.725±.004	0.751±.015	0.733±.062	0.671±.006	0.706±.064	0.756±.040
cifar100	softmax	0.874±.027	0.840±.054	0.777±.007	0.939±.005	0.873±.035	0.713±.006	0.852±.044	0.963±.013
cifar100	godin	0.968±.003	0.972±.004	0.795±.005	0.955±.007	0.969±.006	0.714±.010	0.969±.005	0.987±.001
cifar100	CSI	0.953±.004	0.818±.044	0.802±.001	0.962±.004	0.859±.037	0.746±.011	0.873±.031	0.912±.014
cifar100	sigmoid β	0.673±.082	0.658±.060	0.667±.006	0.870±.029	0.699±.060	0.605±.004	0.680±.062	0.928±.027
cifar100	softmax β	0.858±.017	0.798±.025	0.773±.004	0.941±.005	0.839±.024	0.699±.007	0.810±.030	0.971±.007
cifar100	godin β	0.974±.006	0.977±.007	0.795±.002	0.948±.007	0.979±.011	0.700±.013	0.977±.009	0.986±.002
cifar100	CSI β	0.968±.004	0.865±.028	0.811±.001	0.987±.003	0.885±.025	0.753±.011	0.881±.029	0.951±.008

Table 6: Full AUC results on cifar10 and cifar100 OOD detection datasets averaged over 5 splits with standard deviations.

indata	method	Imagenet	Imagenet resize	Imagenet resize fixed	LSUN	LSUN resize	LSUN resize fixed	iSUN	svhn
cifarall	focal	0.704±.099	0.680±.099	0.564±.017	0.726±.072	0.834±.048	0.487±.070	0.776±.066	0.511±.118
cifarall	sigmoid	0.735±.102	0.733±.118	0.441±.008	0.735±.089	0.830±.105	0.425±.017	0.789±.114	0.431±.086
cifarall	softmax	0.876±.039	0.856±.062	0.612±.018	0.891±.039	0.898±.051	0.510±.045	0.884±.064	0.712±.217
cifarall	godin	0.922±.016	0.925±.020	0.697±.012	0.946±.009	0.958±.012	0.616±.037	0.952±.014	0.994±.002
cifarall	CSI	0.915±.083	0.855±.123	0.676±.116	0.872±.075	0.901±.070	0.666±.092	0.927±.249	0.849±.144
cifarall	sigmoid β	0.940±.010	0.929±.016	0.590±.024	0.950±.006	0.951±.006	0.530±.050	0.951±.007	0.950±.039
cifarall	softmax β	0.901±.021	0.876±.033	0.591±.009	0.943±.008	0.898±.026	0.476±.017	0.894±.026	0.990±.008
cifarall	godin β	0.952±.024	0.954±.027	0.701±.005	0.952±.006	0.977±.005	0.650±.005	0.973±.005	0.997±.001
cifarall	CSI β	0.974±.009	0.914±.047	0.693±.136	0.992±.004	0.936±.025	0.681±.086	0.935±.022	0.918±.082
cifar100	focal	0.399±.052	0.339±.079	0.208±.010	0.336±.026	0.407±.099	0.116±.011	0.350±.083	0.564±.151
cifar100	sigmoid	0.295±.075	0.240±.082	0.194±.009	0.292±.050	0.264±.089	0.105±.006	0.224±.084	0.272±.096
cifar100	softmax	0.436±.088	0.356±.128	0.231±.010	0.715±.021	0.407±.082	0.118±.004	0.370±.083	0.805±.058
cifar100	godin	0.812±.018	0.839±.022	0.265±.011	0.756±.035	0.818±.039	0.102±.012	0.821±.032	0.934±.009
cifar100	CSI	0.713±.020	0.285±.076	0.298±.007	0.768±.026	0.397±.080	0.142±.020	0.389±.072	0.521±.066
cifar100	sigmoid β	0.172±.075	0.113±.046	0.126±.013	0.562±.052	0.150±.031	0.068±.006	0.117±.024	0.691±.092
cifar100	softmax β	0.368±.072	0.216±.057	0.227±.008	0.714±.020	0.273±.053	0.105±.004	0.227±.053	0.848±.041
cifar100	godin β	0.851±.035	0.872±.042	0.265±.005	0.732±.032	0.886±.065	0.092±.014	0.881±.049	0.927±.013
cifar100	CSI β	0.822±.030	0.320±.084	0.304±.007	0.935±.017	0.412±.074	0.167±.019	0.409±.075	0.687±.058

Table 7: Full TNR at TPR 95% results on cifar10 and cifar100 OOD detection datasets averaged over 5 splits with standard deviations.

method	svhn	cifar6	cifar10+	cifar50+	TinyImagenet
focal	0.915±.019	0.830±.028	0.891±.012	0.883±.006	0.636±.018
sigmoid	0.890±.012	0.806±.030	0.857±.028	0.855±.022	0.612±.021
softmax	0.941±.009	0.848±.018	0.924±.006	0.917±.008	0.646±.024
godin	0.932±.009	0.859±.027	0.931±.015	0.933±.003	0.651±.016
CSI	0.958±.006	0.857±.036	0.967±.007	0.963±.001	0.623±.018
sigmoid β	0.927±.006	0.843±.024	0.939±.016	0.933±.007	0.625±.022
softmax β	0.935±.008	0.837±.020	0.942±.018	0.940±.006	0.646±.025
godin β	0.928±.010	0.859±.022	0.936±.009	0.936±.006	0.646±.020
CSI β	0.958±.008	0.874±.021	0.975±.007	0.967±.001	0.628±.019

Table 8: Full AUC results on OSR tasks averaged over 5 splits with standard deviations.

method	svhn	cifar6	cifar10+	cifar50+	tinyimagenet
focal	0.630±.067	0.327±.047	0.477±.047	0.452±.046	0.089±.010
sigmoid	0.635±.020	0.322±.048	0.455±.044	0.446±.036	0.074±.021
softmax	0.706±.029	0.371±.055	0.576±.026	0.558±.027	0.099±.022
godin	0.700±.023	0.366±.071	0.636±.073	0.655±.016	0.102±.012
CSI	0.760±.048	0.423±.064	0.822±.040	0.802±.008	0.086±.008
sigmoid β	0.645±.032	0.348±.037	0.698±.054	0.667±.031	0.088±.006
softmax β	0.672±.034	0.328±.056	0.699±.065	0.690±.022	0.086±.011
godin β	0.704±.026	0.379±.046	0.646±.055	0.656±.028	0.096±.015
CSI β	0.741±.067	0.466±.048	0.868±.039	0.844±.006	0.083±.008

Table 9: Full TNR at TPR 95% results on OSR tasks averaged over 5 splits with standard deviations.

method	svhn	cifar6	cifar10+ cifar50+	Tiny Imagenet	cifar10 all-classes	cifar100
focal	0.972±.003	0.959±.010	0.968±.001	0.507±.043	0.949±.002	0.763±.003
sigmoid	0.974±.003	0.964±.009	0.971±.003	0.473±.038	0.952±.001	0.743±.003
softmax	0.972±.002	0.962±.009	0.969±.001	0.535±.046	0.950±.001	0.766±.002
godin	0.971±.004	0.962±.009	0.968±.001	0.516±.040	0.950±.001	0.758±.002
CSI	0.973±.005	0.958±.004	0.975±.001	0.500±.002	0.936±.008	0.787±.002
sigmoid β	0.969±.004	0.953±.010	0.962±.001	0.512±.040	0.940±.001	0.556±.010
softmax β	0.969±.003	0.952±.010	0.964±.001	0.516±.037	0.940±.001	0.755±.004
godin β	0.972±.004	0.960±.011	0.969±.001	0.529±.039	0.953±.000	0.761±.000
CSI β	0.970±.003	0.957±.006	0.976±.001	0.498±.001	0.932±.002	0.780±.002

Table 10: Full accuracy results of our experiments on all OSR datasets the with standard deviations.