NeuroIoU: Learning a Surrogate Loss for Semantic Segmentation

G Nagendar[1]
nagendar.g@research.iiit.ac.in
Digvijay Singh[1]
digvijay.singh@research.iiit.ac.in
V. Balasubramanian[2]
vineethnb@iith.ac.in
C.V Jawahar[1]
jawahar@iiit.ac.in

[1] Center for Visual Information Technology, IIIT Hyderabad, India
[2] Indian Institute of Technology Hyderabad, India

Abstract

Semantic segmentation is a popular task in computer vision today, and deep neural network models have emerged as the popular solution to this problem in recent times. The typical loss function used to train neural networks for this task is the cross-entropy loss. However, the success of the learned models is measured using Intersection-Over-Union (IoU), which is inherently non-differentiable. This gap between performance measure and loss function results in a fall in performance, which has also been studied by few recent efforts. In this work, we propose a novel method to automatically learn a surrogate loss function that approximates the IoU loss and is better suited for good IoU performance. To the best of our knowledge, this is the first such work that attempts to learn a loss function for this purpose. The proposed loss can be directly applied over any network. We validated our method over different networks (FCN, SegNet, UNet) on the PASCAL VOC and Cityscapes datasets. Our results on this work show consistent improvement over baseline methods.

1 Introduction

Among vision tasks, semantic segmentation has recently attracted a lot of attention, where the objective is to classify each pixel into its corresponding semantic class. It has many applications like autonomous driver navigation, object detection, and video surveillance. For the last few years, Convolutional Neural Networks (CNN)s have shown to be very effective for the semantic segmentation task [2, 10, 16, 20]. CNNs for semantic segmentation typically use the cross-entropy loss for training the network, given as: $L_{CE} = -\frac{1}{N}\sum_{i=1}^{c} y_i log(y_i^{'})$, where y_i is the ground truth, $y_i^{'}$ is the output class score predicted from the network for the pixel i, c is the total number of classes, and N is the number of pixels in the image. In general, a softmax layer is used to map class scores to class probabilities. While the vision community use pixel-wise cross-entropy loss to train the networks, the standard performance measure that is used for the semantic segmentation problem is the intersection-over-union (IoU). For

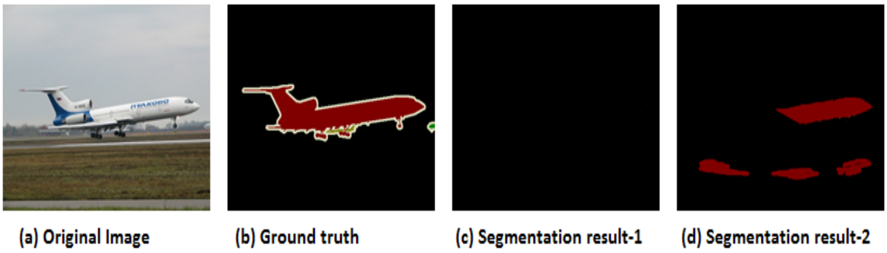

Figure 1: Illustration of cross entropy loss for semantic segmentation: (a) Original image (b) Ground truth segmentation (c) All pixels predicted as background (d) Few pixels correctly predicted as foreground.

an object present in a given image, the *IoU* measure gives the overlap between the predicted region and the ground truth region of the object. The *IoU* measure is popularly used for semantic segmentation due to its ability to handle class imbalances.

Consider the example in Figure 1. For the given image, two segmentation results are given in Figures 1 (c) and 1 (d). The segmentation result in Figure 1 (c) classifies all the pixels as background, and the segmentation result in Figure 1 (d) correctly classifies few object pixels and also classifies few background pixels as object pixels. These two results have same cross-entropy loss. However, the *IoU* measure for the segmentation result in Figure 1 (c) is zero, clearly indicating a gap between the loss function used in training the network and the metric used for measuring the performance of the network.

The key limitation for using *IoU* directly as loss in semantic segmentation is due to its non-differentiability [15]. This restricts its use as a loss function in deep networks. While there have been a few recent attempts to address this issue (described in Section 2), they have their own limitations and are handcrafted approximations. In this paper, we propose a novel method to automatically learn a surrogate loss function that is better suited to attain good *IoU* performance. To the best of our knowledge, this is the first such work that attempts to learn a loss function for this purpose. We call the proposed loss function as the *NeuroIoU* loss, which can be integrated with any deep semantic segmentation CNN. We validate our method by integrating the *NeuroIoU* loss in the FCN, SegNet and UNet models, and evaluate their performance on the PASCAL VOC and Cityscapes datasets. Our results show promise, and a consistent increase in the performance across all class labels for all models on both datasets.

The remainder of this paper is organized as follows. Section 2 briefly reviews earlier related work. The proposed *NeuroIoU* loss, as well as the methodology to train CNNs using the *NeuroIoU* loss, are presented in Section 3. Experiments and results are discussed in Section 4, followed by concluding remarks in Section 5.

2 Related Work

Current state-of-the-art models for semantic segmentation use fully convolutional networks [2, 10, 16, 20]. In these networks, for a given input, we obtain class scores for each pixel, and the class scores are mapped to probabilities using a softmax layer. These probabilities also capture the likelihood of the pixels being the foreground (object/class) or background. In all these models, the cross-entropy loss function [8] is applied over this probability map pixel-wise. The cross-entropy loss is closely tied to the overall classification accuracy. If the number of examples for foreground and background are balanced, then the cross-entropy loss works well. However, in a typical object category segmentation, these may not be balanced, thus raising questions on why cross-entropy loss may not the best choice for loss functions in the object category segmentation task.

As already mentioned, the IoU loss cannot be used in deep networks due to its non-differentiability [15]. In recent years, a few approximations to the IoU measure have been proposed for various applications. These methods [1, 11, 12, 13, 14, 18] attempt different techniques to optimize the IoU measure in the given problem. In [11], the IoU measure is optimized specifically for the object detection and localization problem using joint kernel maps in a SVM context. They are only able to optimize over bounding boxes of the object, and not over full pixel-wise segmentation. In [18], structured Markov Random Field models are used for the same purpose. In [12], special purpose message passing algorithms are used for the optimization. [13] provides a Bayesian framework for optimizing the IoU, where the authors propose an approximate algorithm using parametric linear programming. In [14], the authors provide a framework for optimizing Expected IoU (EIoU). In [1], instead of directly optimizing the model with respect to IoU, the authors select a few candidate segmentations for optimization. However, in contrast to the present work, all these methods do not provide a differentiable approximation to the IoU measure, which can be used for training contemporary deep learning models.

The work closest to ours is [15], which provides a differentiable approximation of the IoU measure, and hence can be used in training deep learning networks. Rahman and Wang presented a handcrafted approximation of the IoU measure in [15]. For a given image I, let $V = \{1, \dots, m\}$ be the set of all its pixels and X be its probability map obtained from the network. Let $Y = \{0,1\}^V$ be the groundtruth of I, where 0 represents background and 1 represents foreground (object). Then, the approximated IoU measure defined in [15] is given as $IoU - Appx = \frac{\sum_v X_v * Y_v}{\sum_v X_v + Y_v - X_v * Y_v}$. The authors showed that this a differentiable function. For the given image, the numerator in $IoU - Appx$ is the sum of true positive pixel probabilities. The value in the denominator can be simplified as the sum of number of object pixels in the ground truth and false positive pixel probabilities. We can observe that this expression approximates the IoU measure well if the probabilities of background pixels are near zero and probabilities of object pixels are near 1, which is not always practical. In this paper, we instead propose a method to directly learn the loss function corresponding to good IoU performance for the given problem. The proposed loss can be integrated with any existing deep semantic segmentation network. We now present our method.

3 Learning a Surrogate IoU Loss for Semantic Segmentation

For a given image, the widely used IoU measure gives the similarity between the predicted region and actual region (ground truth) of the object present in the image, and is given by: $IoU = \frac{TP}{TP+FP+FN}$, where TP, FP and FN denote the counts of true positives, false positives and false negatives respectively. If the output of the semantic segmentation model exactly matches with the ground truth, then its IoU becomes 1, which is desired. So, we ideally want the IoU measure to be near to 1.Owing to the discrete nature of the counts, the IoU measure is inherently non-differentiable [15]. Instead of replacing the non-differentiable IoU measure with proxies such as cross-entropy or other recently proposed measures [15], we use a neural network as a function approximator to automatically learn the surrogate IoU loss to train models for semantic segmentation. Note that to incorporate the IoU measure directly as a loss function to train models, we need to minimize $1 - IoU$. The IoU loss can hence be defined as:

$$L_{IoU} = 1 - IoU = 1 - \frac{TP}{TP + FP + FN} = \frac{FP + FN}{TP + FP + FN} \tag{1}$$

In order to directly optimize the IoU loss in deep networks, we required the gradient of IoU loss with respect to the outputs (or parameters) of a semantic segmentation network. However, since IoU is not differentiable, the IoU loss L_{IoU} is also non-differentiable. We hence cannot directly compute its gradients, which restricts us from using the IoU loss function to train deep networks. We instead approximate the IoU loss using a surrogate neural network, and obtain its derivative across this new network to get the required gradients (using the backpropagation-to-image trick). We now describe our methodology to automatically learn a differentiable approximation of the IoU loss.

3.1 Defining the Surrogate IoU-Loss Network

Neural networks have established themselves as very good function approximators over the years [6, 7, 9, 17], especially when given sufficient amounts of input-output data pairs from the function. In particular, neural networks can also be to approximate discontinuous and non-differentiable functions (please see sample results of our empirical studies in Section 4.2). If we need to approximate a function $f : \mathbb{R}^N \rightarrow \mathbb{R}$ using neural networks, we need the input and output data for the given function. i.e. we need $D = \{(\mathbf{x}_i, y_i) : i = 1, \cdots, n\}$, where $y_i = f(\mathbf{x}_i)$ and n is the number of data points.

We propose our methodology in a manner which allows us to use this framework with any existing CNN used for semantic segmentation. In semantic segmentation, the output of the CNN gives a probability score map, i.e. for each pixel, it gives the probability of being part of each of the classes considered. To approximate the IoU loss using neural networks, we need to define the input data over the continuous domain. To this end, we define the outputs of the semantic segmentation CNN in terms of probability counts TP_{pr}, FP_{pr} and FN_{pr} as below:

$$\begin{gathered} TP_{pr} = \sum_{x_i} P(x_i), \text{ where } x_i \in TP, \quad FP_{pr} = \sum_{x_i} P(x_i), \text{ where } x_i \in FP \\ FN_{pr} = \sum_{x_i} P(x_i), \text{ where } x_i \in FN \end{gathered} \tag{2}$$

TP_{pr}, FP_{pr} and FN_{pr} are the sums of probabilities of pixels in TP, FP and FN respectively (which provide us a continuous-domain equivalent of the corresponding TP, FP and FN counts). Given a CNN model, we can calculate TP_{pr}, FP_{pr} and FN_{pr} using the available ground truth. Our IoU loss approximator network is hence defined by the input-output data pairs: $D = \{((TP_{pr_i}, FP_{pr_i}, FN_{pr_i}), L_{IoU_i}) : i = 1, \cdots, n\}$, where $L_{IoU_i} = 1 - \frac{TP_i}{TP_i + FP_{i+} FN_i}$ is the actual IoU loss computed from the pixel counts. We generate the dataset D to train this IoU-loss approximator network as follows:

- For the given training data, we first train a CNN, Θ_{CE}, using cross-entropy loss.
- After training, for each training image, we store the output probability map obtained from Θ_{CE}.

- For each image, we compute its TP_{pr}, FP_{pr}, FN_{pr} using Equation 2.

We now have the dataset D to train the approximator network. We call this surrogate neural network as *NeuroIoU*, and the loss as *NeuroIoU* loss. We use the Mean-Squared Error loss function, as defined below, to train this network using this data:

$$E(w) = \sum_i [NeuroIoU(TP_{pr_i}, FP_{pr_i}, FN_{pr_i}) - L_{IoU_i}]^2 \tag{3}$$

In semantic segmentation, it is possible that two different segmentation results may have the same IoU score, i.e. two different sets of TP, FP and FN counts (or TP_{pr}, FP_{pr}, FN_{pr}) may have the same IoU value. This affects the approximation performance. To make the approximation more robust to such issues, apart from TP_{pr}, FP_{pr}, FN_{pr}, we also include TP, FP and FN counts as inputs to the *NeuroIOU* network. In particular, we add $\frac{TP_{pr}}{|TP|}$, $\frac{FP_{pr}}{|FP|}$ and $\frac{FN_{pr}}{|FN|}$ as inputs to the network, where, $|TP|, |FP|, |FN|$ are the counts of TP, FP and FN. Our empirical studies (detailed in Section 4) showed that with more inputs, the network was more robust in the approximation.

3.2 Training Semantic Segmentation Models with *NeuroIoU* Loss

In this section, we present a novel formulation for training deep semantic segmentation networks by minimizing the *NeuroIoU* loss.

The *NeuroIoU* loss takes TP_{pr}, FP_{pr} and FN_{pr} as inputs and approximates the corresponding IoU loss. Once we train the network, we use the backprop-to-image trick [19] to compute the derivative of the *NeuroIoU* loss with respect to the outputs of the original CNN. These derivatives are computed as follows. For simplicity, we consider a single hidden layer in the surrogate network (this can be easily extended for multiple hidden layers). If S is the activation function used in the network, then for the given input $x_i = (TP_{pr_i}, FP_{pr_i}, FN_{pr_i})$, $NeuroIoU(TP_{pr_i}, FP_{pr_i}, FN_{pr_i})$ is computed as:

$$NeuroIoU(TP_{pr_i}, FP_{pr_i}, FN_{pr_i}) = \sum_{j=1}^{k} w_j^{op} S(x_i^T w_j^h) \tag{4}$$

where k is the number of neurons in the hidden layer, $w^h \in \mathbb{R}^{k\times 3}$ are weights in the hidden layer and $w^{op} \in \mathbb{R}^k$ are the weights in the output layer. The derivatives of the *NeuroIoU* loss w.r.t. the CNN outputs are given as:

$$\begin{gathered} \frac{\partial Neuro_{IoU}}{\partial TP_{pr}} = \sum_{j=1}^{k} w_j^{op} w_{j1}^h S^{'}(x^T w_j^h), \frac{\partial Neuro_{IoU}}{\partial FP_{pr}} = \sum_{j=1}^{k} w_j^{op} w_{j2}^h S^{'}(x^T w_j^h) \\ \frac{\partial Neuro_{IoU}}{\partial FN_{pr}} = \sum_{j=1}^{k} w_j^{op} w_{j3}^h S^{'}(x^T w_j^h) \end{gathered} \tag{5}$$

where $S^{'}$ is the derivative of the activation function S.

We now have the gradients of *NeuroIoU* with respect to TP_{pr_i}, FP_{pr_i} and FN_{pr_i}, and can retrain the initial CNN, Θ_{CE}, using these gradients. The training procedure is summarized in Algorithm 1. We consider the semantic segmentation network Θ_{CE} initially trained using cross-entropy loss. We then finetune this network using *NeuroIoU* loss, by using the

Algorithm 1 Training a Semantic Segmentation Network using *NeuroIoU* Loss

```
Input: (i) Set of training images $I$ and their ground truth labels $G$; (ii) $NeuroIoU$ loss for each training image (obtained from the surrogate-IoU neural network); (iii) Initial trained CNN model $\Theta_{CE}$ trained on standard cross entropy loss
Output: Updated semantic segmentation CNN model $\Theta^{'}$
Let $\Theta^{'} = \Theta_{CE}$
Repeat until convergence
for $\mathbf{x} \in I$ do
    (1). Compute $\Theta^{'}(\mathbf{x})$, the probability score map for $\mathbf{x}$
    (2). Compute the $TP_{pr}$, $FP_{pr}$ and $FN_{pr}$ for $\mathbf{x}$ using Equations 2
    (3). Using Equations 5, compute $\frac{\partial NeuroIoU}{\partial TP_{pr}}$, $\frac{\partial NeuroIoU}{\partial FP_{pr}}$ and $\frac{\partial NeuroIoU}{\partial FN_{pr}}$
    (4). Using the gradients obtained from step (3), backpropagate through the $\Theta^{'}$ network and update the network weights
end for
```

gradients of the *NeuroIoU* loss w.r.t. the outputs of Θ_{CE} in Equations 5. To this end, we remove the softmax layer from the Θ_{CE} network. For a given training image $\mathbf{x}$, we compute its probability score map $\Theta_{CE}(\mathbf{x})$ from the network. We then update the network weights using backpropagation, by using the gradients of *NeuroIoU* loss with respect to probability score maps. These steps are continued till convergence. In each iteration, the *NeuroIoU* loss increases the pixel probabilities for true positive and false negative pixels, as well as decreases the pixel probabilities for false positive pixels. This increases the true positive count and reduces the false positive and false negative counts for each image, thus increasing overall *IoU* performance.

3.3 Approximation using Prediction Loss

Instead of approximating the *IoU* loss using probability counts as discussed in Section 3.1, we can also approximate using the loss values obtained for each pixel during the prediction obtained from the original model. We use hinge loss, and these values are obtained from the classification layer. Hinge loss values have been previously used for approximating the *IoU* loss [3]. In this new setting, input to the *NeuroIoU* loss are TP_H, FP_H and FN_H, which are computed as $TP_H = \sum_{x_i} h(x_i)$, where $x_i \in TP$, $FP_H = \sum_{x_i} h(x_i)$, where $x_i \in FP$ and $FN_H = \sum_{x_i} h(x_i)$, where $x_i \in FN$. $h(x_i)$ is the hinge loss for the pixel x_i.

4 Experiments

In this section, we validate the performance of the proposed *NeuroIoU* loss to train state-of-the-art deep semantic segmentation models for object category segmentation.

4.1 Datasets and Experimental Settings

We conducted all our experiments on two popular datasets for semantic segmentation, PASCAL VOC 2011 [5] and Cityscapes [4]. For both datasets, we train the network on the provided benchmark training set and test on the respective validation set, because the ground-truth for the test set is not publicly available and our focus is on comparison to the baseline method.

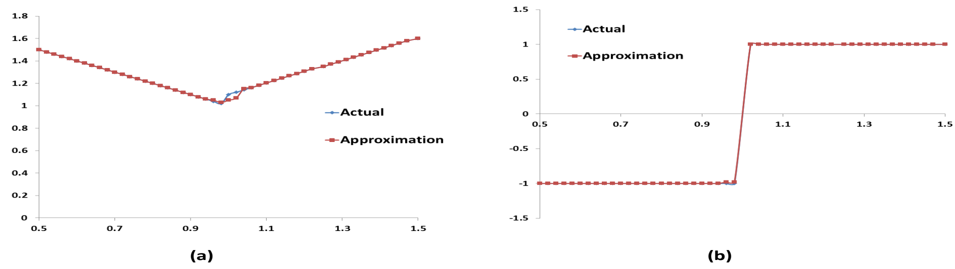

Figure 2: Approximation of a function (non-differentiable at $x = 1$) and its derivatives using neural networks: (a) Approximation of the function; (b) Approximation of its derivatives.

For the PASCAL VOC dataset, we resized the training images to 375×500, and for Cityscapes, we resized the images to 512×1024. For all our experiments, we used a fixed learning rate of 10^{-5}, momentum of 0.99 and weight decay parameter of 0.0005. We integrated the proposed *NeuroIoU* loss into 3 contemporary semantic segmentation networks, FCN [10], Segnet [2] and UNet [16]. We take the original FCN, Segnet and UNet with cross-entropy loss as the baseline for studying the performance our proposed *NeuroIoU* loss, as well as compare our results to [15] on models that were validated in their work (FCN). In the experiments, we refer to the proposed method as *NeuroIoU*, the approximation in [15] as $IoU - Appx$ and the deep networks trained with standard cross-entropy loss as CE. In our surrogate neural network used to approximate the IoU, we use 4 hidden layers and 100 nodes in each layer.

4.2 Function Approximation using Neural Networks

In order to understand the usefulness of the Surrogate-IoU network for approximating the IoU measure, we conducted simple experiments on the capability of neural networks to approximate a discontinuous function and its derivatives. Consider $f(x) = \begin{cases} 2-x & \text{if } x < 1 \\ x+0.1 & \text{if } x \geq 1 \end{cases}$ and its derivative $f^{'}(x) = \begin{cases} -1 & \text{if } x < 1 \\ 1 & \text{if } x > 1 \end{cases}$. We can check that $f(x)$ is not continuous and hence not differentiable at $x = 1$. Results for this approximation using a simple one hidden-layer neural network are given in Figure 2. Figure 2 (a) show the approximation of the function and Figure 2 (b) shows the approximation of its derivatives. We can observe that the approximations obtained almost exactly matches the original function. At $x = 1$, there is a sudden change in the derivative due to the discontinuity, but the neural network is able to approximate this sudden peak with a smooth curve. Since $f(x)$ is not differentiable at $x = 1$, we did not plot its derivative value at $x = 1$. The approximation of the derivative at $x = 1$ lies between -1 and 1.

4.3 Results on Object Category Segmentation

The results for FCN, Segnet and UNet on PASCAL VOC are shown in Figure 3. It shows the comparison of the proposed *NeuroIoU* loss and the baseline cross-entropy loss. We can observe that the proposed *NeuroIoU* loss consistently outperforms the cross-entropy loss in all the categories, with a performance gain for all the 3 networks. This corroborates our claim that this approach can be integrated into any existing deep learning network for improving its performance in semantic segmentation. From the figure, we also notice that the performance

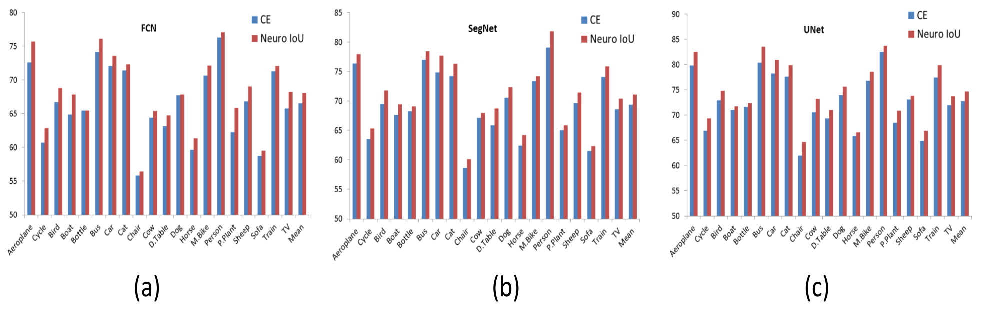

Figure 3: Results on PASCAL VOC: (a) FCN (b) SegNet (c) UNet. Here, CE is the network trained using cross-entropy loss, and *NeuroIoU* is the network trained using proposed *NeuroIoU* loss.

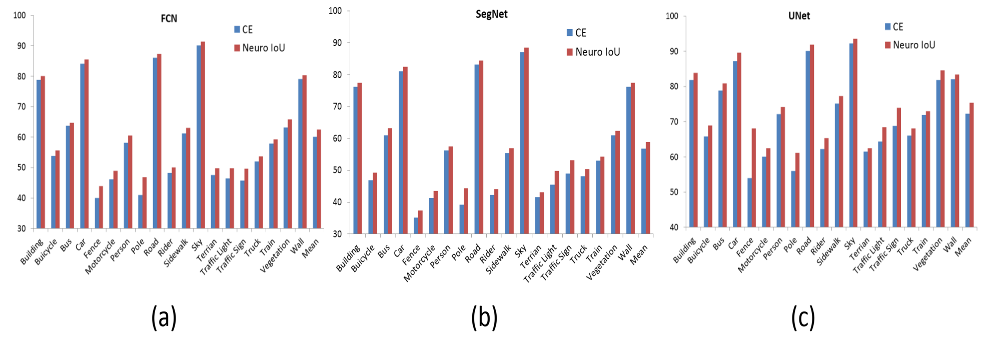

Figure 4: Results on CityScapes: (a) FCN (b) SegNet (c) UNet. Here, CE is the network trained using cross-entropy loss, and *NeuroIoU* is the network trained using proposed *NeuroIoU* loss.

improvements are more significant for some classes, where the foreground to background pixel ratio is very small.

Similar results for FCN, SegNet and UNet on the Cityscapes dataset are presented in Figure 4. Once again, we observe that the proposed *NeuroIoU* loss outperformed cross-entropy loss over all the classes for all the 3 networks. Note again that the maximum performance improvement is obtained for the classes where the ratio of foreground to background pixels is very small. We also tested our method against using weighted cross-entropy loss on a subset of classes with this foreground-background imbalance from the Cityscapes dataset. One could argue that weighting cross-entropy loss suitably can account for class imbalance in the training set. Using weighted cross-entropy, we obtained an *IoU* of 58.4, whereas *NeuroIoU* gave an *IoU* of 60.2 for the same subset. The proposed *NeuroIoU* method does not need such explicit reweighting, since it implicitly learns what is necessary for overall IoU performance.

We also show some qualitative results in Figure 5. From the figure, we can observe that, in comparison to the cross entropy loss, the proposed $Neuro_{IoU}$ loss tends to fill the gaps in the segmentation. It means it tends to recover some of the false negative errors made by the cross entropy loss. In the 4th image (Row-4), we can observe that *NeuroIoU* is able to localize small objects (pole), which the cross entropy-trained model could not. In the 5th image (Row-5), we show a failure case for the *NeuroIoU* loss. Here, *NeuroIoU* is not able

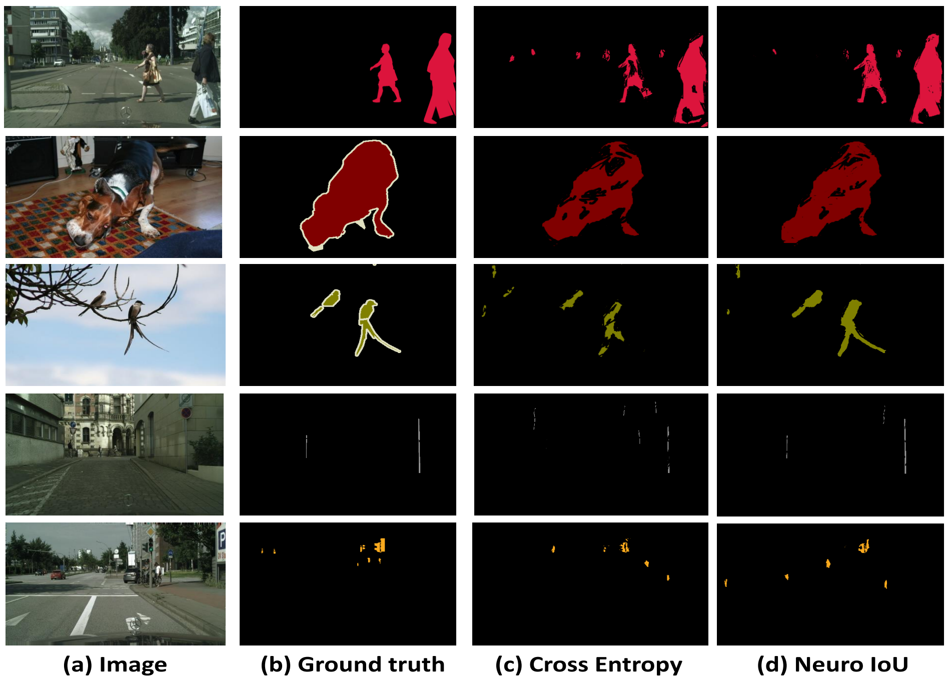

Figure 5: Few qualitative results. (a) Original image (b) Ground truth segmentation (c) Segmentation obtained from cross entropy loss (d) Segmentation obtained using *NeuroIoU*.

to find the small traffic signals present in the image. Note, however that, these objects are also not detected by the model trained using cross-entropy loss.

Comparison to previous work: Rahman and Wang proposed a approximation to the *IoU* measure in [15] (as stated in Section 2). The results of comparing the proposed method to this approximation are given in Table 1, where CE represents networks trained using cross-entropy loss. The results are given for FCN over PASCAL VOC dataset (as in [15]). The results for [15] are obtained from our implementation of their work. The results given for training the model using cross entropy loss (CE) are obtained for a fixed learning rate. Note that these performance numbers can be improved by experimenting with different learning rates, weight decay and other parameters. Our ob-

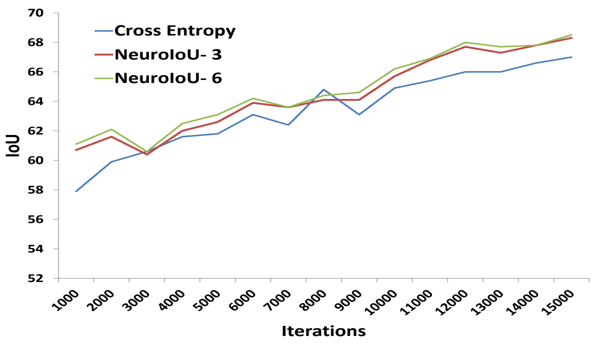

Figure 6: Comparison of performance of NeuroIoU with 3 inputs vs NeuroIoU with 6 inputs using FCN on PASCAL VOC. While both performed better than CE, NeuroIoU-6 was more consistent.

	Aeroplane	Cycle	Bird	Boat	Bottle	Bus	Car	Cat	Chair	Cow
CE	70.58	59.71	67.71	64.84	63.48	75.17	72.05	70.40	53.81	64.37
IoU-Appx [15]	72.72	61.78	68.61	67.29	64.31	76.57	73.03	70.82	54.18	64.06
NeuroIoU	73.68	61.84	69.80	67.83	64.49	77.73	73.57	71.28	54.82	65.38
$NeuroIoU_{No-Int}$	73.65	61.80	69.80	67.82	64.38	77.73	73.57	71.27	54.77	65.37
$NeuroIoU_{Hinge}$	**75.14**	**63.35**	**70.71**	**69.75**	**65.84**	**79.80**	**74.97**	**73.03**	**55.47**	**66.16**

	D.Table	Dog	Horse	M.Bike	Person	P.Plant	Sheep	Sofa	Train	TV	Mean
CE	64.12	65.71	58.64	70.61	77.28	62.27	66.85	56.73	72.27	64.79	66.06
IoU-Appx [15]	65.63	64.17	58.70	71.20	77.39	64.14	67.72	58.38	72.20	66.07	66.94
NeuroIoU	65.98	65.82	60.37	72.10	78.04	66.62	69.02	59.52	73.06	67.18	67.90
$NeuroIoU_{No-Int}$	65.97	65.82	60.37	72.04	78.03	66.62	68.98	59.52	73.03	67.18	67.87
$NeuroIoU_{Hinge}$	**67.60**	**66.74**	**62.73**	**73.68**	**79.89**	**68.03**	**71.62**	**60.85**	**74.72**	**69.27**	**69.46**

Table 1: Comparison of *NeuroIoU* with cross-entropy loss and [15] on PASCAL VOC.

jective is to show the relative performance gain for different methods over CE loss under the same training conditions. The relative improvement in performance of [15] over cross entropy loss (CE) is what we intend to show, rather than the absolute *IoU* value. Here, the *NeuroIoU* loss is used to retrain the network obtained from training using the CE loss. It is evident that the *NeuroIoU* loss outperforms this surrogate loss [15] over all the classes. Compared to the cross-entropy loss, the surrogate loss in [15] is under performing for some classes. However, the *NeuroIoU* loss consistently outperforms cross-entropy loss over all the classes.

We compare 2 other variants of *NeuroIoU* called $NeuroIoU_{No-Int}$ and $NeuroIoU_{Hinge}$ with *NeuroIoU* in Table 1. In $NeuroIoU_{No-Int}$, we train the network without any initialization, i.e., the model trained using cross-entropy loss is not used to initialize the network. In $NeuroIoU_{Hinge}$, *IoU* is approximated using hinge loss values instead of probabilities from the final layer (as in Section 3.3). From Table 1, we can observe that the network trained without initialization from a pre-trained model ($NeuroIoU_{No-Int}$) is able to obtain similar performance as the network trained using the initialization (*NeuroIoU*) obtained using cross-entropy loss. We can also observe that $NeuroIoU_{Hinge}$ outperformed all other methods.

We also studied the performance of the Surrogate-IoU neural network over different inputs in Figure 6. In $NeuroIoU-6$, the neural network is trained over 6 inputs $TP_{pr}, FP_{pr}, FN_{pr}$, $\frac{TP_{pr}}{|TP|}, \frac{FP_{pr}}{|FP|} and \frac{FN_{pr}}{|FN|}$, while the neural network is trained only on 3 inputs $TP_{pr}, FP_{pr}, FN_{pr}$ in $NeuroIoU-3$. The results are given for 15000 iterations. From the figure, we notice that $NeuroIoU-6$ is better than $NeuroIoU-3$ (we observed the same trend for other datasets too), while both outperform cross-entropy loss.

5 Conclusions

In this work, we have demonstrated a new approach for learning the loss function for semantic segmentation using deep neural network models. Our experimental results demonstrate that the proposed loss function can be flexibly applied to any network. We demonstrated the effectiveness of the proposed method on state-of-the-art deep semantic segmentation network models on two popular datasets, and the results showed consistent improvement on all datasets across the models. Qualitatively, we obtained improved segmentation quality, in particular on small objects. While validated on semantic segmentation, the proposed approach can be used for other tasks that are evaluated using non-differentiable measures.

References

[1] Faruk Ahmed, Daniel Tarlow, and Dhruv Batra. Optimizing expected intersection-over-union with candidate-constrained crfs. In *ICCV*, 2015.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. In *PAMI*, 2017.

[3] Maxim Berman and Matthew B. Blaschko. Optimization of the jaccard index for image segmentation with the lovász hinge. In *CVPR*, 2018.

[4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CoRR*, 2016.

[5] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results.

[6] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural Netw.*, 1990.

[7] Kurt Hornik, Maxwell Stinchcombe, Halbert White, and Peter Auer. Degree of approximation results for feedforward networks approximating unknown mappings and their derivatives. *Neural Computation*, 1994.

[8] Y. Bengio I. Goodfellow and A. Courville. Deep learning. In *MIT Press*, 2016.

[9] Yoshifusa Ito. Approximation capability of layered neural networks with sigmoid units on two layers. *Neural Computation*, 1994.

[10] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[11] Yang Wang-Steven N. Robinovitch Ze-Nian Li Mani Ranjbar, Tian Lan and Greg Mori. Optimizing nondecomposable loss functions in structured prediction. In *PAMI*, pages 911–924, 2012.

[12] S. Nowozin. Optimal decisions from probabilistic models: The intersection-over-union case. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 548–555, 2014.

[13] Sebastian Nowozin. Optimal decisions from probabilistic models: the intersection-over-union case. In *CVPR*, 2014.

[14] Batra D Premachandran V, Tarlow D. Empirical minimum bayes risk prediction: How to extract an extra few% performance from vision models with just three more parameters. In *CVPR*, 2014.

[15] Md. Atiqur Rahman and Yang Wang. Optimizing intersection-over-union in deep neural networks for image segmentation. In *ISVC*, 2016.

[16] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI (3)*, pages 234–241, 2015.

[17] T. Tran-Cong T. Nguyen-Thien a. Approximation of functions and their derivatives: A neural network implementation with applications. *Applied Mathematical Modelling*, 1999.

[18] Daniel Tarlow and Richard Zemel. Structured output learning with high order loss functions. In *AISTATS*, pages 1212–1220, 2012.

[19] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[20] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2016.