Deep Neural Networks with Intersection over Union Loss for Binary Image Segmentation

Floris van Beers, Arvid Lindström, Emmanuel Okafor and Marco A. Wiering
Bernoulli Institute, Department of Artificial Intelligence, University of Groningen, Nijenborgh 9, Groningen, Netherlands
f.van.beers@student.rug.nl, arvid.lindstrom@gmail.com, {e.okafor, m.a.wiering}@rug.nl

Keywords: Deep Learning, Image Segmentation, Loss Function, Intersection Over Union, Jaccard Index

Abstract: In semantic segmentation tasks the Jaccard Index, or Intersection over Union (IoU), is often used as a measure of success. While this measure is more representative than per-pixel accuracy, state-of-the-art deep neural networks are still trained on accuracy by using Binary Cross Entropy loss. In this research, an alternative is used where deep neural networks are trained for a segmentation task of human faces by optimizing directly an approximation of IoU. When using this approximation, IoU becomes differentiable and can be used as a loss function. The comparison between IoU loss and Binary Cross Entropy loss is made by testing two deep neural network models on multiple datasets and data splits. The results show that training directly on IoU significantly increases performance for both models compared to training on conventional Binary Cross Entropy loss.

1 INTRODUCTION

Semantic segmentation aims to map each pixel in an image to its associated label such as car, building and pedestrian. In the field of image segmentation with deep neural networks, increasingly complex systems are created to improve on the semantic segmentation task. All these systems compete in complexity and state-of-the-art performance. A commonality between early works described in (Shelhamer et al., 2017), more recent works such as SegNet (Badrinarayanan et al., 2017) and comparative studies (Siam et al., 2018) is that they use Intersection over Union (IoU), also known as the Jaccard Index, as a measure of success on test images. IoU is much more indicative of success for segmentation tasks, compared to pixel-wise accuracy, especially when the input data is significantly sparse. When labels used for training consist of 80-90% background, and only a small percentage of positive labels, a naive measure such as accuracy can score up to 80-90% by labeling everything as background. Because IoU does not concern itself with true negatives, this naive solution will never occur if IoU is used as the loss function, even with highly sparse data. While this research is focused on binary segmentation, the naive solution problem amplifies itself when applied to segmentation over multiple classes as the ratio of background to a specific class becomes worse with more classes.

With the assumption that IoU as a measure of success is helpful, this research attempts to use a loss function based on IoU to train a segmentation model directly. While research has been done extensively on which loss functions are best for which task (Janocha and Czarnecki, 2017; Zhao et al., 2017), the default used for image segmentation is still usually Cross Entropy (Shelhamer et al., 2017; Badrinarayanan et al., 2017; Noh et al., 2015). Cross Entropy is a loss function that is, mathematically, much more closely related to accuracy than IoU, even though the final performance of semantic segmentation models is measured using IoU. By defining a loss function more closely related to IoU the training process could be improved. As such the question that needs to be answered is as follows: Can a model trained on an IoU loss function perform better than a model trained on Binary Cross Entropy (BCE) loss? Previous research on optimizing IoU (Rahman and Wang, 2016) has proposed a loss function based directly on IoU. This loss function is tested on a single model, which is trained on the separate classes of several semantic segmentation datasets. Significant improvements are found by applying this loss function to a single model. In this research, since binary segmentation is considered, the comparison will be between this loss function based on IoU, detailed in section 2, and BCE. Optimization of the IoU has also been used in other work on semantic segmentation, as shown in work with the Conditional Random Field (Ahmed et al., 2015) and work on probabilistic models (Nowozin, 2014). In another work a different loss function is proposed, which is also based on IoU (Yuan et al., 2017). This implementation strays further from IoU by applying a quadratic component. As such, in this work, the loss function without this quadratic component is used as an implementation of IoU loss. Another loss function proposed to optimize IoU is the Lovsz-Softmax loss (Berman and Blaschko, 2017). By using similar argumentation, this loss function was not considered in this work due to it straying further from a purely IoU based approach.

This research extends previous work on the topic (Rahman and Wang, 2016), and evaluates the performance differences between loss functions on multiple models. In addition, to focus solely on the difference in performance based on the loss functions, the models will use the same structure and parameters. A base model has been created that functions when changing the loss function, keeping all other parameters constant. To further emphasize the difference in performance based purely on the use of a new loss function a sufficiently dense dataset has been chosen. This avoids skewing the results in favor of IoU loss, which theoretically performs better with a sparser dataset. In previous research (Rahman and Wang, 2016), the data is much sparser due to the use of separate classes in multi-class segmentation datasets, such as PASCAL VOC2011 (Everingham et al., 2011).

In section 2, the models, datasets, and the loss functions will be elaborated upon. Some mathematical adaptation of IoU will be used in order to make the calculation differentiable. In section 3 the experimental setup will be detailed, both in terms of hyperparameters and dataset splits. In section 4 the outcome of the experiments will be presented. These results will be put into context in section 5 and evaluated for statistical relevance. Furthermore, this section shows some example output. Final conclusions and suggestions for future work will be conveyed in section 6.

2 METHODS

2.1 Models

In this research, two different encoders are extended into fully convolutional networks (FCNs) for their use as a semantic segmentation model. These encoders are the convolutional parts of VGG-16 (Simonyan and Zisserman, 2014) and ResNet-50 (He et al., 2015).

We have trained our own custom neural network systems using pretrained weights from the earlier mentioned neural network systems. These were previously trained on the ImageNet dataset (Deng et al., 2009). The training of the custom systems employs face images from two separate face datasets used as input to the proposed methods. All experiments were carried out with the aid of the Keras deep learning framework (Chollet et al., 2015), because it contains rich libraries for computer vision tasks.

2.1.1 VGG-16/BFCN-32s

The VGG-16 network (Simonyan and Zisserman, 2014) is a well-established deep neural network (DNN) used for classification. It is a convolutional network with 16 convolutional layers after which there are several fully connected layers. Finally, a softmax layer determines the classification outcome. It has been trained extensively on the ImageNet dataset (Deng et al., 2009). VGG-16 has previously been extended to a fully convolutional network (Shelhamer et al., 2017). For this research the same extension from VGG-16 to FCN-32s was used as described in previous work (Shelhamer et al., 2017). A notable difference is the number of output classes. Where the original FCN-32s is modelled to segment 21 classes, i.e. the 20 classes of the VOC-2011 dataset (Everingham et al., 2011) and background, the adaptation used here considers only 2 classes. These classes are face and background.

The main steps in adapting VGG-16 to the segmentation task remain the same as described in previous work (Shelhamer et al., 2017). The fully connected layers of VGG-16 are replaced with fully connected convolutional layers. The first of these convolutional layers has 4096 feature maps, a kernel size of 7×7 and a stride of 1. This means that they are essentially fully connected feature maps. The second fully connected layer is replaced by a similar layer, but with a kernel size of 1×1. These layers use a rectified linear unit activation. Finally, the fully connected layer that is the size of the label space is replaced with a fully convolutional layer with the same purpose. This layer has 21 feature maps, one for each class, in the original creation of FCN-32s (Shelhamer et al., 2017) and has 1 feature map in our adaptation, named BFCN-32s (Binary FCN-32s). This final layer goes from feature space to label space and uses a linear activation to create the feature map.

To make the model fully convolutional, the resulting 7×7 feature maps have been upsampled by a trainable deconvolution layer. This layer uses a stride of 32 to regain the original image size and counteracts the size decreases performed by the max-pooling layers done in each convolutional block of the encoder. Since the number of classes is reduced to 2, a softmax layer as used in the original model (Shelhamer et al., 2017) is no longer necessary. Instead, the up-sampling layer uses a sigmoid activation to map the output pixels to values between 0 and 1. These steps create a model where the input image is the same size as the output image, allowing the model to be trained pixel-wise end-to-end.

As with the implementation of FCN-32s (Shelhamer et al., 2017), all layers of the VGG encoder were frozen before training on any new data, with the exception of the last convolutional layers positioned in convolutional block 5. This is done to speed up the training process. In addition, the size of the datasets used would not influence earlier layers which are already sufficiently trained on ImageNet (Deng et al., 2009).

2.1.2 ResNet/FCResNet

Another well-performing classification network is ResNet-50 (He et al., 2015). This model uses residual learning in a 50-layer DNN to classify images. It has been trained on the ImageNet dataset (Deng et al., 2009). The same steps as described in section 2.1.1 were taken to make a fully convolutional version from ResNet, named FCResNet in this research. While ResNet does not have fully-connected layers, but only a softmax layer the size of the label space, a fully connected convolutional layer was added. After this a convolutional layer was used to replace the softmax layer and perform the same conversion from feature space to label space as mentioned in section 2.1.1. The fully connected convolutional layer, as described in section 2.1.1, is used to facilitate the transition between the feature extraction part of the model and the reduction to label space. Without this layer, the reduction to label space is too aggressive and therefore loses a lot of information. This fully connected convolutional layer is similar to the one used in BFCN-32s and uses 4096 feature maps, a kernel size of 7 $\times$ 7 and a stride of 1. The convolution layer replacing the softmax layer reduces the feature maps from 4096 to 1 through linear activations. Finally, a similar up-sampling layer was added to obtain an output image with the same size as the input image, resulting in a pixel-wise end-to-end trainable version of ResNet. This up-sampling layer uses a sigmoid activation, as explained in section 2.1.1.

As with BFCN-32s, some layers of FCResNet were frozen. Using the same argumentation as in section 2.1.1, all layers of the encoder were frozen except the last convolutional block, named res5.

2.2 Datasets

To explore the differences in performance from these models two pixel-wise labeled datasets have been used. These are the Labeled Faces in the Wild: Part Labels dataset (LFW) (Kae et al., 2013) and the HELEN dataset (Le et al., 2012). Both sets have been chosen for their relatively dense appearance of positive labels. The objective is to use densely distributed data instead of sparse, because the latter might skew the results in favor of an IoU approach, as explained in section 1.

2.2.1 LFW

Labeled Faces in the Wild: Part Labels (Kae et al., 2013) is a dataset containing 2927 images of faces, an example of which is shown in figure 1. These images are labeled in two ways, but for this research, only the pixel-wise labeling is considered, as seen in figure 2. This labeling is in three classes, namely face, hair and background. Since the task considered is binary segmentation, some preprocessing had to be performed. For different experimental settings, we consider the hair to be either part of the face, as shown in figure 3 or not, as shown in figure 4. This results in a binary labeling. The data was split by using 292 images (10%) for testing. The amount of training and validation images are dependent on the experimental settings as detailed in section 3.

Figure 1: Example Input Image LFW

2.2.2 HELEN

The HELEN dataset (Le et al., 2012) is also a dataset of faces, containing 2330 images. It is labeled in multiple classes for separate parts of the face, such as

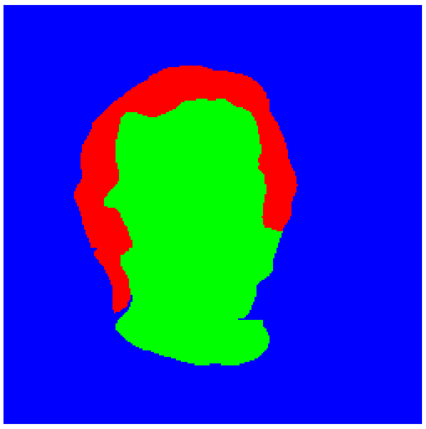
Figure 2: Example 3-class Label LFW

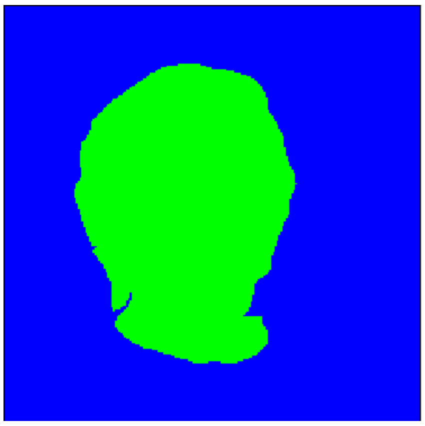
Figure 3: Example LFW label with hair

mouth and hair. As with LFW this dataset has to be preprocessed in such a way that it can be used for binary image segmentation. As such we construct new labels out of the provided labels that either label all parts of the face as face and the rest as background, or label the hair as background as well. This results in the same data structure as with the LFW dataset. With this dataset, 233 images (10%) are used for testing. The amount of images for training and validation are dependent on the experimental setup.

2.3 Loss Functions

In assessing the effectiveness of an IoU loss function, a baseline has to be established. Binary Cross Entropy (BCE) loss, or log loss, is such a baseline in that it is used by default in a wide range of recent classification and segmentation works (Shelhamer et al., 2017; Badrinarayanan et al., 2017; Siam et al., 2018; Noh et al., 2015).

The formula for binary cross entropy loss can be seen in equation 1. In this equation T refers to the true label image, T_x refers to a single element of that label,

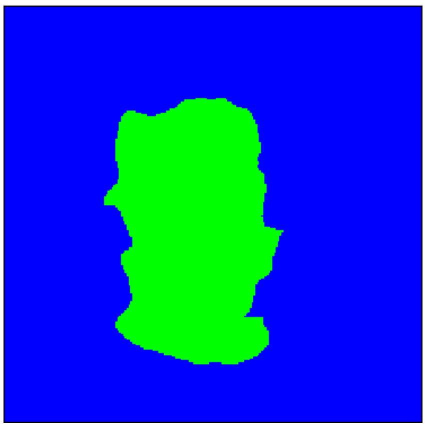
Figure 4: Example LFW label without hair

P refers to the prediction of the output image and P_x to a single element of that prediction.

$$L_{BCE} = \sum_x -(T_x \log P_x + (1 - T_x)\log(1 - P_x)) \quad (1)$$

The cases of $P_x = 1$ and $P_x = 0$ would lead to log(0), which is undefined. To prevent this, the values from P are clipped in the range $[\varepsilon, 1-\varepsilon]$. This is done by the Keras framework, where ε is set to 1×10^{-7}. In equation 1, it can be seen that BCE, while incorporating an element of probability, smoothed out by the log component, awards both true positives and true negatives, while penalizing false positives and false negatives. Referring back to the problem described in section 1, this can lead to simplistic solutions to segmentation when the data is significantly sparse, by labeling all output as background.

The other loss function we use is one that directly incorporates the value for Intersection over Union. This loss function is proposed in previous work (Rahman and Wang, 2016), and this previous work describes the mathematical aspects, which will be elaborated upon here as well. The original equation for IoU can be given as:

$$IoU = \frac{|T \cap P|}{|T \cup P|} \quad (2)$$

As before, in equation 2, T stands for the true label image, P for the prediction of the output image and the symbols are taken from set theory. This IoU is then taken as the average over the entire set of pixels producing an IoU value between 0 and 1. These set symbols are, however, not differentiable. To apply these set symbols in their true form, the numbers in T and P need to be absolute 1's and 0's. However, while the label image T contains these values, the output P contains values between 1 and 0 due to the sigmoid activation in the final up-sampling layer of the network. To solve this, an approximation of IoU can be made using probabilities. This gives the equation for this approximation IoU':

$$IoU' = \frac{|T * P|}{|T + P - (T * P)|} = \frac{I}{U} \tag{3}$$

Here T and P remain the same, but $T * P$ is the element-wise multiplication of T and P. In the numerator, this gives an approximation of Intersection by giving the probability of P_x when T_x is 1 and giving 0 otherwise. As such, the intersection is highest when P_x is 1 wherever T_x is 1, exactly as is to be expected. The denominator is an addition of T and P with a deduction of the Intersection, just as in a regular calculation for the union, to mitigate the effect of counting the intersection area twice.

After reducing IoU from set operations to arithmetic operations, producing IoU', the formula is differentiable. As a loss function, the error needs to approach 0 when results become better. To achieve this the loss function is defined in terms of IoU' as such:

$$L_{IoU} = 1 - IoU' \tag{4}$$

This loss L_{IoU} is applied to each element in a batch and added producing a value between 0 and batch-size, when IoU for each sample approaches 1 or 0, respectively. This is a loss function that can, again, be minimized. To achieve this it needs to be differentiated, which is done in the following way:

$$\frac{\partial L_{IoU}}{\partial P_x} = \frac{-U * \frac{\partial I}{\partial P_x} + I * \frac{\partial U}{\partial P_x}}{U^2} \tag{5}$$

$$= \frac{-U * T_x + I * (1 - T_x)}{U^2} \tag{6}$$

These derivatives, and the backpropagation that follows, are computed by the Keras framework during training when the equation used to calculate IoU is replaced with the differentiable approximation.

3 EXPERIMENTS

Training the models using the datasets and loss functions described in section 2 is done according to certain design choices, such as hyper-parameters of the models and dataset usage. Table 1 shows the different splits of the datasets such that 8 distinct experimental setups are created. Applying these 8 data splits to each combination of BFCN32-s and FCResNet, either with BCE or IoU loss, results in 32 distinct setups, the results of which are presented in section 4.

To ensure no other factors would influence the outcome of these experiments, any other hyper-parameters have been kept constant. The relevant hyper-parameters can be seen in table 2. While most parameters are found through a parameter sweep based on the results from previous work (Shelhamer et al., 2017), the patience parameter is less self-explanatory. It determines the number of epochs without improvements after which the training stops. This is our point of convergence. As such this value is more important for total learning time than the number of epochs, the maximum of which was never reached.

Table 1: Different data uses: Number of images used for training and validation is given by # training and # validation respectively. The Hair? column determines whether hair is included in the face (yes) or in the background (no).

Dataset	# Training	# Validation	Hair?
LFW	2342	292	yes
LFW	1000	100	yes
LFW	2342	292	no
LFW	1000	100	no
HELEN	1864	233	yes
HELEN	1000	100	yes
HELEN	1864	233	no
HELEN	1000	100	no

Table 2: Experimental parameters

Parameter	Value
Epochs	1000
Batch-size	100
Patience	20
Learning Rate	0.0001
Optimizer	RMSprop

Using these data splits, models and parameters, each of the models is trained until convergence as determined by the patience parameter. After training the model was tested on previously unseen data of the dataset it was trained on. For both datasets this was the last 10%, i.e. 292 images for LFW and 233 images for HELEN, as mentioned in section 2.2.

4 RESULTS

The results of the experiments described in section 3 can be seen in tables 3, 4, 5 and 6. These tables are each structured similarly. Each table displays the results of a different combination of model and dataset and presents each of the 4 data splits within that dataset. Results are evaluated on three metrics: original binary accuracy, Intersection over Union and epochs before convergence. For each metric, the results from the models trained on IoU loss are in the L_{IoU} column and the results from binary cross entropy

Table 3: Results from BFCN-32s trained and tested on LFW: The name describes whether that setting used all of the training and validation images (big) or only 1000 training images and 100 validation images (small) and whether hair is included as part of the face (hair) or not (no hair). Accuracy refers to the pixel-wise accuracy score for that setting. Time needed for convergence is measured in epochs. Columns marked L_{IoU} show results for IoU loss. Columns marked L_{BCE} show results for binary cross entropy loss.

Name	Accuracy		IoU-Score		Convergence	
	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}
big, hair	0.974	**0.978**	**0.921**	0.906	**91**	146
small, hair	0.962	**0.963**	**0.887**	0.857	116	**98**
big, no hair	**0.977**	0.976	**0.896**	0.852	89	**83**
small, no hair	**0.970**	0.967	**0.872**	0.837	155	**117**

Table 4: Results from BFCN-32s trained and tested on HELEN: see table 3 for clarification.

Name	Accuracy		IoU-Score		Convergence	
	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}
big, hair	0.961	0.961	**0.877**	0.838	**125**	130
small, hair	**0.943**	0.940	**0.820**	0.779	**118**	131
big, no hair	0.983	**0.984**	**0.905**	0.877	**83**	125
small, no hair	**0.978**	0.977	**0.871**	0.837	**93**	129

Table 5: Results from FCResNet trained and tested on LFW: see table 3 for clarification.

Name	Accuracy		IoU-Score		Convergence	
	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}
big, hair	0.942	**0.947**	**0.829**	0.826	75	**64**
small, hair	0.922	**0.942**	0.777	**0.818**	**44**	113
big, no hair	**0.959**	0.943	**0.821**	0.730	78	**59**
small, no hair	**0.951**	0.949	**0.798**	0.757	72	**67**

Table 6: Results from FCResNet trained and tested on HELEN: see table 3 for clarification.

Name	Accuracy		IoU-Score		Convergence	
	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}	L_{IoU}	L_{BCE}
big, hair	**0.926**	0.905	**0.768**	0.673	62	**50**
small, hair	0.916	**0.917**	**0.747**	0.723	**62**	97
big, no hair	**0.967**	0.962	**0.816**	0.775	**68**	72
small, no hair	**0.963**	0.960	**0.796**	0.766	88	**80**

loss are in the L_{BCE} column. Better performances are marked in bold.

These tables clearly show that while accuracy and convergence favor both IoU and BCE seemingly at random, the IoU-score is distinctly higher for the new IoU loss function in almost all experimental settings. The exact nature and significance of this improvement will be detailed in section 5.

5 DISCUSSION

The experiments performed and the results reported in section 4 show the patterns for the three metrics used to compare the two loss functions. Each of these patterns will be discussed briefly. While binary accuracy has been established as being less relevant as a measure of success for a segmentation task, it is noteworthy nonetheless to show that it does not decrease with the use of the IoU loss function.

Binary Accuracy. As can be observed in tables 3, 4, 5 and 6, IoU-loss and BCE-loss score higher on accuracy in 56.25% and 37.5% of the cases, respectively. However, whether IoU-loss or BCE-loss results in a higher binary accuracy, the results are within 2.5% in every comparison. A paired t-test on these values remains inconclusive, with a p-value of 0.52, implying no significant difference between the mean performance of these loss functions based on binary accuracy.

Intersection Over Union. Contrary to binary accuracy, the results for Intersection over Union show consistent and significant improvements in almost all experimental settings when using IoU-loss. A paired t-test shows that IoU-loss scores better with a p-value of 4.7×10^{-4}. This is undoubtedly significant. This p-value is not surprising as IoU performs better in 15 out of 16 test cases. With the exception of one test case, improvements of multiple percentage points can be seen across the board. On average over all cases the error margin is reduced by 17.5%.

Convergence While the focus of this research was on a comparison of performance of the two loss functions, an improvement in training time would also be relevant. From the results, however, it can be seen that these training times vary wildly. A paired t-test shows that the average training time for IoU is lower with a p-value of 0.256. This shows that no informed conclusion can be drawn about either loss function converging faster.

Models. When we compare the results of BFCN-32s and the results of FCResNet, we observe that the BFCN-32s model performs better. Although the accuracies obtained with the different models do not differ very much, the IoU scores of BFCN-32s are in general much higher. While the encoder part of FCResNet is more developed than the encoder of BFCN-32s, the performance of the decoder is worse. This is due to the choice of only applying a single fully convolutional layer in FCResNet compared to two fully convolutional layers in BFCN-32s. This choice was based on the amount of trainable parameters that resulted in adding these layers. The output of FCResNet's encoder has considerably more parameters than the output of BFCN-32s and, as such, connecting this to a fully convolutional layer amplifies this disparity. A second convolutional layer in FCResNet was, therefore, not feasible and thus the results for this model are worse than the results for BFCN-32s.

Example Output Images. In figures 5 and 6 example outputs are shown. These images are from the experimental setting using the BFCN-32s model and the LFW dataset. They correspond to the example input and label shown in figure 1 and figure 4, respectively. While neither segmentation is perfect, the output for IoU in figure 5 is clearly showing the contours of the hairless face better than the output for BCE in figure 6. Unfortunately, for both images, some upsampling artifacts remain. This is most likely due to the coarse deconvolution layer, which upsamples by a factor of 32. These up-sampling artifacts are much more pronounced upon initialization and are never fully removed during training.

Dataset Size. In section 3 multiple dataset splits were described. The distinction between large and small was made to evaluate the effect of the size of the dataset on the effectiveness of either loss function. With the results presented in section 4 it can be seen that whenever a smaller dataset is used, performance suffers slightly, as is to be expected. However, whether IoU or BCE performs better does not change based on the size of the dataset.

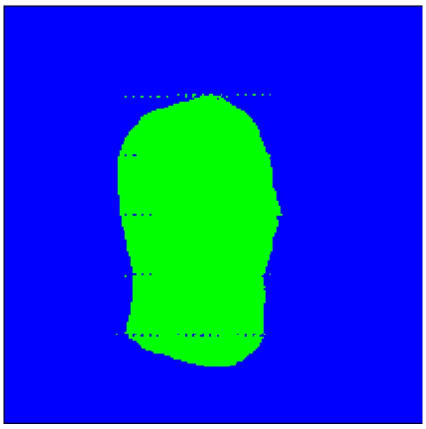
Figure 5: Example Output: BFCN-32s with IoU loss on LFW without hair

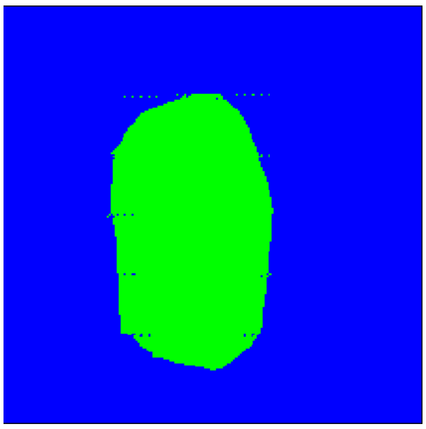
Figure 6: Example Output: BFCN-32s with BCE loss on LFW without hair

6 CONCLUSION

Taking into account both the statistical analysis of the results and the output images presented, it is clear that training a segmentation model directly on the Intersection over Union objective can lead to significant improvements. The statistical analysis shows a significant improvement over all categories when using IoU loss over BCE loss when the common measure of success IoU is used. As such IoU loss is definitely an option worth considering when attempting to improve a segmentation model for state-of-the-art performance. This improvement on segmentation is in accordance with the results from previous work (Rahman and Wang, 2016). This research shows that these improvements are independent of the sparsity of data.

What is also shown by this research is that in an ever-continuing attempt to improve state-of-the-art Deep Neural Nets, the area of loss functions has not been fully explored. This is in agreement with conclusions from other work (Janocha and Czarnecki, 2017), which state that while cross entropy has been an unquestionable favourite, adopting one of the various other losses can be equally, if not more, effective. These conclusions, together with the conclusion from this and other research on the effectiveness of IoU loss show the same thing. More and more research is being done towards architectures, creating deeper or different convolutional networks, while a significant improvement can already be made by choosing a different loss function.

While this research shows that performance improves significantly for the models that were used, this can not be claimed for every model. As such research on IoU loss with other models, such as the well-established SegNet (Badrinarayanan et al., 2017), may support our hypothesis that IoU loss performs better in general for semantic segmentation tasks.

In section 1 an explanation is given why the loss function by (Rahman and Wang, 2016) is preferred to the loss functions proposed by (Yuan et al., 2017; Berman and Blaschko, 2017). In future research, it is also interesting to compare these loss functions based on IoU directly.

Finally, the claim has been made that the benefit from training on IoU directly will only magnify when a model is presented with sparse data. This has not been evaluated in this research and can be done by expanding the models presented here to perform a segmentation task on multiple classes. This would significantly reduce the amount of positive samples in a dataset and thus be a way to explore the hypothesis that an IoU loss function outperforms binary cross entropy on sparser data. In previous work (Rahman and Wang, 2016) sparse data is already used. However, here the sparsity of the data is taken as it is and not isolated to determine its effect on the performance of the IoU loss function. As such future work could focus specifically on certain datasets, comparing performance on sparse and dense data.

REFERENCES

Ahmed, F., Tarlow, D., and Batra, D. (2015). Optimizing expected intersection-over-union with candidate-constrained CRFs. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1850–1858.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495.

Berman, M. and Blaschko, M. B. (2017). Optimization of the jaccard index for image segmentation with the Lovász Hinge. *CoRR*, abs/1705.08790.

Chollet, F. et al. (2015). Keras. https://keras.io.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database.

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

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

Janocha, K. and Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *CoRR*, abs/1702.05659.

Kae, A., Sohn, K., Lee, H., and Learned-Miller, E. (2013). Augmenting CRFs with Boltzmann machine shape priors for image labeling. In *the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Le, V., Brandt, J., Lin, Z., Bourdev, L., and Huang, T. S. (2012). Interactive facial feature localization. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part III*, ECCV'12, pages 679–692, Berlin, Heidelberg. Springer-Verlag.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

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

Rahman, M. A. and Wang, Y. (2016). Optimizing intersection-over-union in deep neural networks for image segmentation. In *International Symposium on Visual Computing*.

Shelhamer, E., Long, J., and Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651.

Siam, M., Gamal, M., Abdel-Razek, M., Yogamani, S., and Jägersand, M. (2018). RTSeg: Real-time semantic segmentation comparative study. *CoRR*, abs/1803.02758.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556.

Yuan, Y., Chao, M., and Lo, Y. C. (2017). Automatic skin lesion segmentation using deep fully convolutional networks with Jaccard distance. *IEEE Transactions on Medical Imaging*, 36(9):1876–1886.

Zhao, H., Gallo, O., Frosio, I., and Kautz, J. (2017). Loss functions for image restoration with neural networks. *IEEE Transactions On Computational Imaging*, 3(1):47–57.