Loss Scheduling for Class-Imbalanced Image Segmentation Problems

Taubert, Oskar
oskar.taubert@kit.edu
Steinbuch Centre for Computing
Karlsruhe Institute of Technology

Götz, Markus
markus.goetz@kit.edu
Steinbuch Centre for Computing
Karlsruhe Institute of Technology

Schug, Alexander
a.schug@fz-juelich.de
Jülich Supercomputing Centre
Forschungszentrum Jülich

Streit, Achim
achim-streit@kit.edu
Steinbuch Centre for Computing
Karlsruhe Institute of Technology

March 5, 2021

Abstract

When training a classifier the choice of loss function heavily influences the characteristics of the resulting model. The most commonly used loss function for classification is cross entropy. In image segmentation problems where each pixel is assigned to a particular class, overlap-based losses have recently been shown to improve classifier performance especially

for datasets with an imbalanced class distribution. This is particularly relevant to segmentation because class imbalance mitigation strategies used in regular classification are often not applicable. Overlap-based losses, however, have different drawbacks. We are aiming at combining the upsides of different losses with a simple scheduling scheme during training while minimizing their downsides. Gradually transitioning from an overlap-based dice loss to cross entropy allows to reliably select a distinct minimum in the optimization landscape as a valuable alternative to results obtained from traditional unscheduled loss functions. We demonstrate the efficacy of our approach on different combinations of loss functions, datasets, and models.

1 Introduction

Deep Learning is the de facto standard when it comes to classification in general and semantic image segmentation in particular [1]. One degree of freedom in designing classification models is the choice of loss function applied during training. Depending on their specific characteristics, different loss functions lead to different optimization landscapes and accordingly trained models. Since neural networks are trained with different flavors of gradient-descent optimization, a rough landscape with many local minima is difficult to optimize and smoother landscapes are preferred. Hence, the main concern in the selection of a loss is usually its convergence behavior. Other criteria include discriminative performance on a range of metrics that inform model selection but are not used in the training directly. Popular performance metrics are accuracy, the F_1 score, or an expected misclassification cost. For some applications of probabilistic regression, a measure of model calibration or reliability is required as a special case of classification. In this context, if the considered classifier is well calibrated, its output can be interpreted as an approximation of the probability of the sample belonging to a particular class instead of being an opaque score. For image segmentation problems, where the model assigns each pixel a categorical label to generate a dense feature map, overlap-based losses and metrics such as the Jaccard index and dice loss are often used [2]. Compared to the standard cross entropy, dice loss has been found to perform better on class-imbalanced problems. Its drawback is that, unlike cross entropy, dice loss is not a proper scoring function and thus produces models whose outputs have to be viewed as opaque scores rather than as probabilities. In this light, it is desirable to combine features of different loss functions for some applications. In biomedical image segmentation [3], for example, it would be advantageous to have both good performance on class-imbalanced datasets and an interpretable score. As our contribution to class imbalanced classification problems, we propose a loss scheduling scheme in order to combine the characteristics of different loss functions. We apply our proposed method to two different segmentation models on three standard class-imbalanced datasets. We find that we can consistently drive the model to a minimum that increases class-averaged recall and is hard to find using static losses.

2 Class Imbalance Mitigation

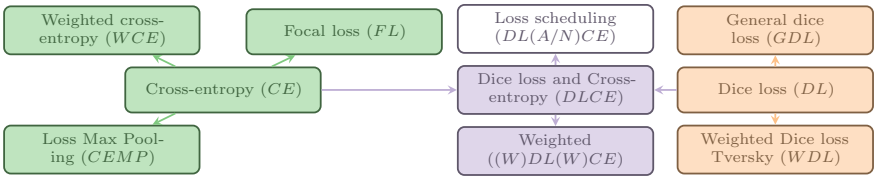


Figure 1: Taxonomy of class-imbalance mitigating loss functions for image segmentation problems. Arrows indicate usage.

Some recent publications have focused on class imbalance in general and class imbalance for image segmentation in particular [4–6]. Many real-world classification datasets do not have an even distribution of classes: some classes, e.g., "road", occur considerably more often in a dataset of street level photographs than others like "person" [7–9]. The same applies to segmentation datasets where two types of class imbalances can be distinguished, i.e. class imbalances at the sample level or at the pixel level. Sample-level class imbalance leads to a concentration of the underrepresented class to few batches and hinders training convergence. Similarly to classification tasks, this type of imbalance can be addressed during data collection by including class representatives uniformly [10]. Pixel-level class imbalance where only few pixels of a sample containing a particular class are harder to address at the data collection stage. In this work, we will focus on pixel-level class imbalance. One class of approaches involves adapting the loss function to focus on the underrepresented class or "hard" examples. The simplest of these applies a by-class weight to the used loss, where each weight usually is proportional to the inverse class frequency. A second group of methods aims to reduce the chances of many small per-pixel contributions to the loss from a single image or batch overwhelming few larger ones by reweighting individual pixel losses based on their magnitude. Both focal loss [6] and loss max-pooling [4] fall in this category. Another loss often employed in image segmentation is based on the dice overlap [2] of prediction and ground truth. Overlap-based losses are reported to perform better on imbalanced datasets but often do not converge as well as traditional classification losses. This issue can be addressed by using a weighted sum of two losses from each category to fit a model [11]. One approach that does not involve modifying the loss function is stochastic sampling, which can also be used for sample-level class imbalance. For segmentation training, ground-truth pixels are masked out with a probability proportional to their class label frequency. This approach is not further explored in this work for two reasons: First, not training on large parts of the training data while still performing all computations makes the already expensive training process even less efficient both in terms of computational expense and data efficiency. Second, it has been reported [5] that stochastic oversampling on classification tasks causes a model to overfit the underrepresented class.

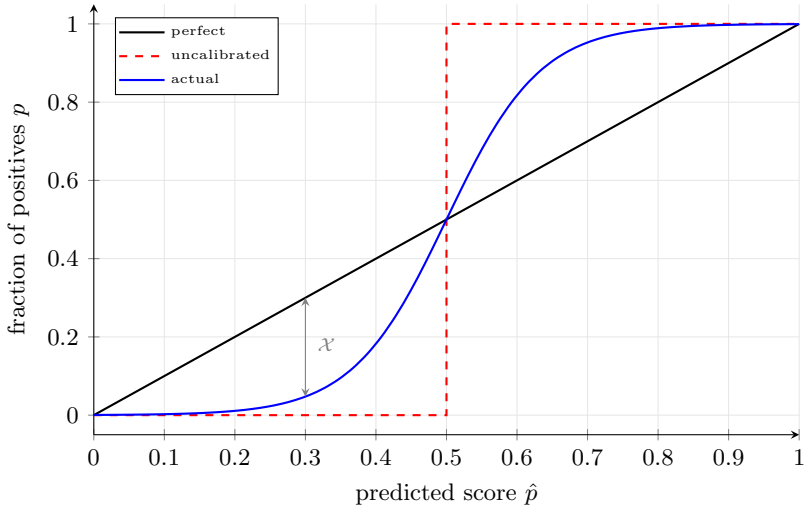


Figure 2: Idealized schematic depictions of a reliability diagram to compare the confidence of the model output to the ground-truth accuracy for each probability.

3 Preliminaries

The loss functions considered in this work were selected based on their relevance for imbalanced multi-class semantic segmentation. For class-weighted losses, we use inverse class frequency scaled such that the weights sum up to the number of classes. Furthermore, Y denotes the set of ground-truth labels with pixel values y_x and $\hat{Y}$ the set of predicted probability values $\hat{y_x}$ for a given data item x. Fig. 1 gives an overview of the different loss functions. The green group contains distribution-based, the red group overlap-based and the purple group composite losses, respectively.

For many classification problems it is important that the classification model is *well calibrated*. Considering a set of instances predicted to belong to a particular class with probability p by a calibrated classifier, a fraction equal to p should truly belong to that class. Calibrated confidence improves the interpretability of classification models and allows them to be directly incorporated into other probabilistic models. Still, perfect calibration on a realistic training dataset is not achievable. Fig. 2 shows an empirical way to illustrate a model's calibration by plotting the approximated real probability over the confidence of the model. Guo et al. [12] highlight a range of methods to calibrate a trained model. Since these methods require either sufficient numbers of predictions in as many reliability histogram bins as possible or training another calibration model on the validation set, it is desirable to produce as well calibrated models as possible from the outset.

3.1 Cross Entropy

Cross entropy is one of the most widely used losses for arbitrary classification tasks [13]. It measures the dissimilarity between the true class distribution Y and the predicted probabilities $\hat{Y}$. The class-weighted variant can be expressed as

$$WCE_x = -\sum_{c}^{C} w_c \cdot y_{cx} \log\left(\hat{y}_{cx}\right) . \tag{1}$$

In the unweighted variant CE of the weighted cross entropy (WCE), all weights w_c are set to one. While cross entropy is a proper scoring function and theoretically promises a well calibrated model, in practice cross-entropy trained models are often not perfectly calibrated due to practical concessions making the training feasible in the first place [12].

3.2 Dice Loss

(a) Schematic characteristics of optimization landscapes for cross entropy loss.

(b) Schematic characteristics of optimization landscapes for dice loss.

Figure 3: Schematic representations of different optimization landscapes. We expect cross entropy to be smoother in general but its global optimum to be hard to hit.

Dice loss has been defined first by Milletari et al. [2]. It is a differentiable measure of the overlap of ground-truth labels with the predicted segmented area. A generalized version of the dice loss (GDL) has been introduced by Sudre et al. [2], which includes a smoothing value ε to make the computation numerically more stable. In this work, we employ a weighted hybrid between these flavors, including the smoothing term as well as the quadratic denominator terms from the initial version. It is defined as:

$$WDL_x = 1 - 2\frac{\sum_{c=1}^{C} w_c \sum_x y_{cx}\hat{y}_{cx} + s}{\sum_{c=1}^{C} w_c \sum_n y_{cx}^2 + \hat{y}_{cx}^2 + s + \varepsilon} \tag{2}$$

Analogous to cross entropy, an unweighted dice loss DL can be obtained by setting all class weights w_c to one. The weighted dice loss as presented here can be seen as a special version of another class-imbalance mitigating loss—the Tversky loss [14]. Instead of a singular weight w_c, the Tversky loss defines two separate factors α and β for the WDL denominator components. Accordingly, the Tversky loss for the case $\alpha = \beta = w_c$ is covered in this work.

3.3 Focal Loss

Another approach to solving the class imbalance problem is pixel loss reweighting. Instead of emphasizing entire underrepresented classes, it enhances the loss of individual difficult-to-learn training samples. One such reweighting scheme is focal loss [6]

$$FL_x = -(1-\hat{y}_x)^\gamma y_x \log(\hat{y}_x) = (1-\hat{y}_x)^\gamma CE_x \,, \tag{3}$$

where γ is an adjustable focusing parameter. Focal loss assumes that the easier-to-predict over-represented classes in the training sample, even if the per-pixel loss is small, contribute significantly more to the overall loss due to sheer volume. Hence, they need to be down-weighted in order to be on the same scale as the underrepresented classes. The factor $(1-\hat{y}_x)^\gamma$ reduces a training pixels x's loss contribution when its prediction $\hat{y}_x$ is close to $y_x = 1$ and focuses on the badly classified pixels with larger individual losses.

3.4 Loss Max-Pooling

Similarly to focal loss, loss max-pooling [4] uses a modulating factor to enhance larger contributions to the overall loss while suppressing smaller ones. In short, the max-pooled cross entropy loss is then written as:

$$CEMP_x = w_x \cdot y_x \log(\hat{y}_x) = w(CE_x) \cdot CE_x \tag{4}$$

The weight $w(CE_x)$ reweights the largest pixel losses and ignores the smallest ones. For details on its computation we refer the interested reader to the original publication. Note that we apply the loss max-pooling over an entire batch rather than a single image.

3.5 Compound Loss

Compound loss uses a weighted sum of two different loss functions to combine their strengths or mitigate their drawbacks. For example, dice loss is often combined with cross entropy in order to aid training with a smoothened optimization landscape while still directly optimizing for an overlap measure.

4 Loss Scheduling

Our contribution—loss scheduling—addresses the key issue of class imbalance by dynamically distorting the optimization landscape. Consider a model architecture that potentially produces a perfect, well calibrated classifier if optimized under cross entropy. Then, the global optimum of the optimization surface is typically hard to find using a straightforward gradient-descent based optimization algorithm. For, e.g., a binary classification problem with a single overrepresented class, gradient-descent based algorithms tend to preferably locate local optima where the underrepresented class is neglected. This is because such local minima

are shallower but also wider than the global optimum, i.e., easier to find. Fig. 3a shows a sketch of this optimization landscape. To circumvent this issue, we propose to transform the optimization landscape during the actual optimization process in order to funnel the model into a different optimum performing better on the underrepresented class. Since overlap-based measures have been shown to improve performance in class-imbalanced image segmentation [2], we add a dice-loss component to the overall cross-entropy loss function (see Section 3.2). As the model is to be optimized on cross entropy to produce a well calibrated model, the dice-loss contribution is reduced over time while increasing the cross-entropy contribution. We call this gradual transition from one loss function to another over the course of the training process *loss scheduling*. If the minimum obtained from the overlap-based loss is close to the new cross-entropy minimum as sketched in Fig. 3b and the transition is gradual, the model converges to this new minimum. We explore two different schedules here (cf. Fig. 4): The "Naïve" schedule is a simple linear ramp from dice loss to cross entropy. In the "Alternating" schedule, we ramp linearly from cross entropy to dice loss before applying a sinoidal decay back to cross entropy. We finish with some more iterations using only cross-entropy loss. Mathematically, the exact formulation is

$$w_{\mathrm{CE}}(i) = \max\left(0, \min\left(1, \frac{i-s}{e-s}\right)\right) \tag{5}$$

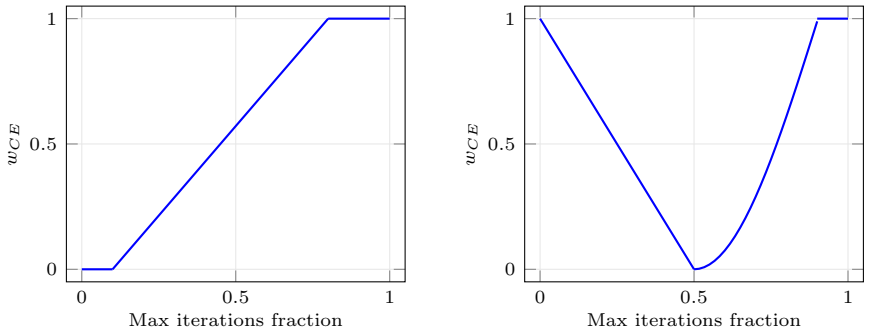
for the naïve schedule where i is the iteration count and s and e are beginning and end of the linear transition, respectively, and

$$\begin{aligned} w_{\mathrm{CE}}(i) = \min\left(1, \max\left(0, 1 - \frac{i}{s}\right)\right) \\ +1 - \cos\left(\min\left(\frac{\pi}{2}, \max\left(0, \frac{\pi\,(i-s)}{2\,(e-s)}\right)\right)\right) \end{aligned} \tag{6}$$

for the alternating schedule where s and e are beginning and end of the sinoid. In general the loss schedule has to be adapted to the learning rate schedule.

5 Empirical Evaluation

We evaluated the different loss functions presented previously and our *loss scheduling* approach on two network architectures: A fully convolutional network (FCN) [15] and DeepLabV3 [16] with a pre-trained ResNet-101 [17] backbone. We used three datasets: 1. the PASCAL Visual Object Classes 2012 (VOC) [7] dataset with a total of 11 540 images and 20 different classes, 2. the Semantic Boundaries Dataset (SBD) [9] with a total of 11 355 images and 20 different classes, and 3. the Cityscapes [18] dataset with the utilized total of 5000 fine-annotated images and 30 classes. Fig. 5 illustrates the imbalance in class distribution for these datasets. In order to model a best-case scenario, we also used models pre-trained on MS-COCO [8] restricted to the same classes as VOC and SBD. MS-COCO is a considerably larger dataset with over 200 000 labeled images.



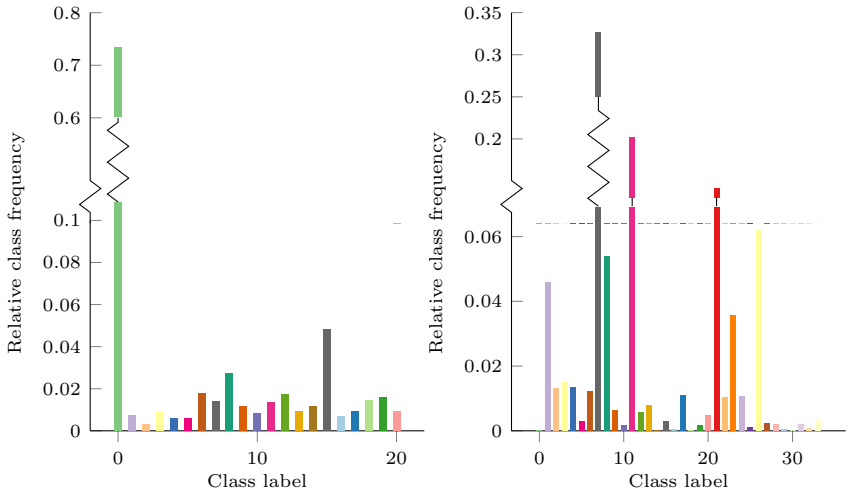
(a) Naïve (N). (b) Alternating (A).

Figure 4: Different loss scheduling schemes. The cross entropy loss contribution, i.e., weight w_{CE}, is depicted against the fraction of total optimization epochs.

Table 1: Results of the empirical evaluation of the FCN and DeepLabV3 neural network architectures using all loss functions on the VOC, SBD, and Cityscapes datasets. Losses prefixed with W are class-weighted. Losses suffixed with MP are max-pooled. Two losses separated by N or A are naïvely or alternatingly scheduled respectively. CE and DL denote cross entropy and dice loss, respectively.

		VOC						SBD						Cityscapes					
		acc	F_1	rec	cal	S_{acc}	S_{rec}	acc	F_1	rec	cal	S_{acc}	S_{rec}	acc	F_1	rec	cal	S_{acc}	S_{rec}
FCN	*prefine*	0.95	0.85	0.85	0.91	0.93	*0.87*	0.92	0.82	0.81	0.93	0.93	*0.87*	-	-	-	-	-	-
	WCE	0.89	0.68	0.64	0.95	0.92	*0.77*	0.87	0.68	0.64	0.93	0.90	*0.76*	0.84	nan	0.43	0.95	0.89	*0.59*
	WDL	0.89	0.65	0.62	0.93	0.91	*0.74*	0.87	0.67	0.64	0.89	0.88	*0.75*	0.83	nan	0.34	0.84	0.83	*0.49*
	FL	0.88	0.67	0.64	0.62	0.73	*0.63*	0.87	0.67	0.64	0.60	0.71	*0.62*	0.83	nan	0.43	0.69	0.75	*0.53*
	CEMP	0.88	0.66	0.63	0.45	0.60	*0.53*	0.86	0.66	0.63	0.35	0.50	*0.45*	0.83	nan	0.44	0.46	0.59	*0.45*
	DLCE	0.89	0.68	0.64	0.95	0.92	*0.77*	0.87	0.68	0.64	0.95	0.91	*0.77*	0.84	nan	0.43	0.95	0.89	*0.59*
	WDLCE	0.89	0.68	0.65	0.96	0.92	*0.78*	0.88	0.68	0.64	0.96	0.92	*0.77*	0.84	nan	0.45	0.94	0.88	*0.61*
	DLWCE	0.81	0.60	0.75	0.97	0.88	*0.84*	0.79	0.59	0.76	0.96	0.87	***0.85***	0.76	0.39	0.57	0.91	0.83	*0.70*
	WDLWCE	0.79	0.59	0.75	0.96	0.87	*0.84*	0.77	0.58	0.76	0.96	0.85	***0.85***	0.75	nan	0.55	0.89	0.81	*0.68*
	DLNCE	0.89	0.67	0.64	0.92	0.90	*0.76*	0.87	0.67	0.64	0.92	0.90	*0.75*	0.83	nan	0.41	0.96	0.89	*0.58*
	DLACE	0.89	0.68	0.64	0.93	0.91	*0.76*	0.87	0.67	0.64	0.93	0.90	*0.76*	0.84	nan	0.42	0.96	0.89	*0.59*
	WDLNCE	0.89	0.68	0.65	0.93	0.91	*0.76*	0.87	0.68	0.64	0.92	0.90	*0.76*	0.83	nan	0.43	0.96	0.89	*0.60*
	WDLACE	0.89	0.68	0.65	0.94	0.91	*0.77*	0.87	0.68	0.64	0.93	0.90	*0.76*	0.84	nan	0.44	0.95	0.89	*0.60*
	DLNWCE	0.68	0.51	0.76	0.93	0.78	*0.83*	0.64	0.50	0.77	0.87	0.74	*0.82*	0.70	0.37	0.58	0.90	0.79	*0.70*
	DLAWCE	0.67	0.51	0.76	0.89	0.77	*0.82*	0.64	0.50	0.77	0.87	0.74	*0.82*	0.69	0.37	0.58	0.91	0.79	***0.71***
	WDLNWCE	0.69	0.52	0.76	0.96	0.80	***0.85***	0.64	0.50	0.77	0.90	0.75	*0.83*	0.70	0.37	0.58	0.88	0.78	*0.70*
	WDLAWCE	0.68	0.52	0.76	0.93	0.79	*0.84*	0.65	0.50	0.77	0.90	0.75	*0.83*	0.69	0.37	0.58	0.89	0.78	*0.70*
DeepLabV3	*prefine*	0.96	0.89	0.91	0.95	0.96	*0.93*	0.93	0.86	0.88	0.96	0.95	*0.92*	-	-	-	-	-	-
	WCE	0.92	0.80	0.83	0.89	0.90	*0.86*	0.91	0.80	0.83	0.91	0.91	*0.87*	0.85	nan	0.47	0.89	0.87	*0.61*
	WDL	0.88	nan	0.62	0.89	0.89	*0.73*	0.90	nan	0.81	0.85	0.87	*0.83*	0.81	nan	0.23	0.80	0.80	*0.36*
	FL	0.90	0.77	0.84	0.71	0.80	*0.77*	0.90	0.79	0.83	0.65	0.75	*0.73*	0.84	nan	0.47	0.71	0.77	*0.56*
	CEMP	0.92	0.79	0.82	0.69	0.79	*0.75*	0.90	0.79	0.81	0.54	0.68	*0.65*	0.84	nan	0.47	0.48	0.61	*0.47*
	DLCE	0.91	0.79	0.83	0.89	0.90	*0.86*	0.91	0.80	0.84	0.91	0.91	*0.87*	0.84	nan	0.46	0.88	0.86	*0.61*
	WDLCE	0.92	0.80	0.83	0.88	0.90	*0.86*	0.91	0.81	0.84	0.90	0.90	*0.87*	0.84	nan	0.48	0.86	0.85	*0.62*
	DLWCE	0.86	0.71	0.89	0.82	0.84	*0.85*	0.84	0.72	0.90	0.81	0.82	*0.85*	0.78	0.43	0.61	0.93	0.85	***0.74***
	WDLWCE	0.86	0.72	0.88	0.85	0.85	***0.87***	0.84	0.73	0.90	0.83	0.83	*0.86*	0.77	nan	0.57	0.93	0.85	*0.71*
	DLNCE	0.91	0.77	0.84	0.90	0.90	***0.87***	0.90	0.80	0.85	0.90	0.90	*0.87*	0.84	nan	0.43	0.91	0.87	*0.58*
	DLACE	0.91	0.79	0.83	0.90	0.90	*0.86*	0.91	0.80	0.84	0.91	0.91	*0.87*	0.84	nan	0.46	0.90	0.87	*0.61*
	WDLNCE	0.92	0.79	0.83	0.91	0.91	***0.87***	0.90	0.80	0.84	0.92	0.91	***0.88***	0.84	nan	0.46	0.89	0.87	*0.61*
	WDLACE	0.92	0.79	0.83	0.89	0.90	*0.86*	0.91	0.80	0.84	0.92	0.91	***0.88***	0.84	nan	0.47	0.88	0.86	*0.61*
	DLNWCE	0.74	0.60	0.90	0.72	0.73	*0.80*	0.74	0.63	0.91	0.65	0.69	*0.76*	0.73	0.39	0.61	0.93	0.82	***0.74***
	DLAWCE	0.76	0.62	0.90	0.68	0.72	*0.78*	0.74	0.63	0.91	0.67	0.70	*0.77*	0.73	0.40	0.61	0.94	0.82	***0.74***
	WDLNWCE	0.77	0.63	0.90	0.79	0.78	*0.84*	0.75	0.64	0.91	0.70	0.72	*0.79*	0.73	0.40	0.61	0.93	0.82	***0.74***
	WDLAWCE	0.76	0.63	0.9	0.72	0.74	*0.80*	0.75	0.64	0.91	0.71	0.73	*0.80*	0.73	0.41	0.6	0.93	0.82	*0.73*

For training we used 30 epochs, training only the classifier head with a batch size of 12 and an initial learning rate of 10^{-3}. The weight decay regularization is set to 10^{-4}. Over the course of the training, the learning rate is linearly reduced to 0. For compound losses we experimented with different ratios. As the choice only had a minor impact on the result, compound losses presented here use a 1:1 ratio. With the naïve loss scheduler, the transition begins after 30% of iterations and finishes after 90%. For the alternating loss scheduler, the sinoid phase starts



(a) VOC.

(b) Cityscapes.

Figure 5: Relative class frequencies of evaluated datasets. The class distribution of VOC is similar to the one of SBD

after 50% of iterations and finishes after 90%. We trained with normalization, random resize, flip, and crop as data augmentation. For the larger Cityscapes images, doubled resize and crop sizes were used.

5.1 Metrics

We employ several metrics to quantify and compare the quality of the trained classifiers. Based on the confusion matrix over the validation set, we use the well-known metrics [19] per-pixel accuracy, and class averaged $F-1$-score and recall. Class-averaged metrics put more emphasis on the underrepresented classes since an individual pixel in a class with many pixels overall contributes less to the metric. Furthermore, we introduce a score to quantify the calibration of a model. As depicted in Fig. 2, a well calibrated classifier would produce class-membership probabilities rather than an opaque confidence score. The calibration is estimated based on calibration curves: We first bin the model output for each pixel and class. Then we plot the fraction of positive labels for each bin over the mean output. As a measure of a model's reliability, we define a scalar calibration score, which is computed by a metric similar to the expected calibration error (ECE) introduced in [20]:

$$\mathrm{cal} = 1 - \frac{4}{N} \cdot \sum_i |p_i - \hat{p}_i| = 1 - \frac{4}{N} \cdot \sum_i |\mathcal{X}_i| \tag{7}$$

N is the number of bins, p_i the fraction of positives in bin i, and $\hat{p}_i$ the mean predicted score. This calibration score is equal to 1 if all bins land on the identity, and we call it N-approximately perfectly calibrated. The calibration score is equal to 0 if the model is calibrated according to the dashed red line in Fig. 2. Note that the cal score is in $[-1, 1]$ but all models we trained produce a calibration score greater than 0. Unless specified otherwise, we set $N = 10$ to ensure a sufficient number of samples in each bin. We use this score rather than ECE since we need a score that is defined for multi-class problems and fits into the following two definitions. In order to simultaneously evaluate calibration and classification performance, we introduce two more metrics: S_{acc} and S_{rec}; the harmonic mean of calibration score, and accuracy and recall, respectively.

5.2 Experimentation Environment

All experiments have been performed on computational nodes with an Intel Xeon Gold 6126 CPU @ 2.60GHz as host processor and an NVidia Tesla V100 GPGPU as AI accelerator. A Redhat Enterprise Linux with kernel version 3.10.0 has served as operating system. The GPU driver, CUDA runtime, and PyTorch [21] were in versions 418.87.00, 10.1, and 1.4.0, respectively. The source code may be obtained from the authors' repository [22].

5.3 Results

We compare the different loss functions over three datasets and two models. Table 1 shows metrics computed over the validation set with a converged model averaged over four training runs as defined in Section 3 and Section 4. Most of the losses seem to be fairly robust against initialization with the exception of pure dice loss. A compound loss includes both loss names, in the case of a scheduled loss separated by an indicator of the schedule used. The $prefine$ line shows results using a model pre-trained on MS-COCO. Model hyperparameters can be found in [23] and in [22]. The F_1 score of models that ignore at least one class is not defined and marked as nan. The first notable observation is that the $prefine$ model always performs best. The second best S_{rec} for each architecture after the $prefine$ model is highlighted in bold. Focal loss and max pooled cross entropy produce performance metrics comparable with the other traditional losses but have the worst calibration scores. Models using a traditional loss or a compound loss with unweighted cross entropy have similar performance and calibration metrics. Compound or scheduled losses with class weighted cross entropy sacrifice global accuracy for class-averaged recall. On VOC and SBD, DeepLabV3 tends to have better overall accuracy, F_1, and recall but worse calibration compared to FCN with the exception of dice loss. On Cityscapes only losses scheduled towards weighted cross entropy and DLWCE compound loss specifically produce models that do not ignore at least one minority class. The different schedules do not display a consistent trend over the range of models. We also tested loss scheduling from focal loss to cross entropy with similar results as for the dice loss version. Examining individual examples shows that

the DeepLabV3 models are more reliable in picking classes consistent with the overall image.

6 Conclusion

We introduced a novel approach to addressing the class imbalance problem which is of particular relevance to image segmentation problems. In the examples explored in this work loss scheduling was consistently able to drive a model towards a local minimum in the optimization landscape, that put more emphasis on underrepresented classes. Even though a compound loss was also able to find this new optimum, our loss seems to be better able to find underrepresented classes, while not being too sensitive to changes in scheduling parameters. Especially the models trained on Cityscapes illustrate this. More concretely, high recall models might be useful for problems where false negatives in the underrepresented class are expensive, e.g. biomedical pathology detection. Losses based on reweighting cross entropy to focus on hard examples may produce better performing models than found here, especially if more effort is expended on tuning their hyperparameters. However the reweighting appears to have adverse impact on the calibration of the trained model. While loss scheduling was not able to reproduce the performance of models pre-trained on a larger dataset, we expect this to be achievable with different choices of loss functions or more elaborate (e.g. oscillating) loss schedules at least for some datasets. The general idea of dynamically distorting the optimization landscape to drive the training in a particular direction before converging on the actually desired loss function is applicable in a wider context. It is also extendable to using more than two component losses or arbitrarily elaborate schedules. Loss scheduling may also improve performance on problems like graph edge labeling or dense feature map prediction.

Acknowledgments

This work is supported by the Helmholtz Association Initiative and Networking Fund under project number ZT-I-0003 and the Helmholtz AI platform grant.

References

[1] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.

[2] C. Sudre, W. Li, T. Vercauteren, S. Ourselin, and J. Cardoso, "Generalised Dice Overlap as a Deep Learning Loss Function for Highly Unbalanced Segmentations," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2017, pp. 240–248.

[3] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Nelson, G. Corrado, J. Hipp, L. Peng, and M. Stumpe, "Detecting Cancer Metastases on Gigapixel Pathology Images," 2017.

[4] S. R. Bulo, G. Neuhold, and P. Kontschieder, "Loss Max-Pooling for Semantic Image Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, July 2017, pp. 7082–7091.

[5] N. Chawla, K. Bowyer, L. Hall, and P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, Jun 2002.

[6] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE, Oct 2017, pp. 2980–2988.

[7] M. Everingham, L. Van Gool, C. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge," *International Journal of Computer Vision*, vol. 88, pp. 303–338, 2010.

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 740–755.

[9] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik, "Semantic Contours from Inverse Detectors," in *2011 International Conference on Computer Vision (ICCV)*. IEEE, Nov 2011, pp. 991–998.

[10] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2009, pp. 248–255.

[11] F. Isensee, J. Petersen, A. Klein, D. Zimmerer, P. Jaeger, S. Kohl, J. Wasserthal, G. Koehler, T. Norajitra, S. Wirkert, and K. Maier-Hein, "nnU-Net: Self-adapting Framework for U-Net-Based Medical Image Segmentation," in *Bildverarbeitung für die Medizin 2019*. Springer Fachmedien Wiesbaden, 2019, pp. 22–22.

[12] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On Calibration of Modern Neural Networks," in *Proceedings of the 34th International Conference on Machine Learning*. JMLR, 2017, pp. 1321–1330.

[13] K. Murphy, *Machine Learning: a Probabilistic Perspective*. MIT Press, 2012.

[14] S. S. M. Salehi, D. Erdogmus, and A. Gholipour, "Tversky Loss Function for Image Segmentation Using 3D Fully Convolutional Deep Networks," in *International Workshop on Machine Learning in Medical Imaging*. Springer, 2017, pp. 379–387.

[15] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015, pp. 3431–3440.

[16] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," 2017, [accessed at 2020-02-26]. [Online]. Available: https://arxiv.org/abs/1706.05587

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016, pp. 770–778.

[18] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The Cityscapes Dataset for Semantic Urban Scene Understanding," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2016, pp. 3213–3223.

[19] T. Fawcett, "An Introduction to ROC Analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[20] M. P. Naeini, G. f. Cooper, and M. Hausknecht, "Obtaining well calibrated probabilities using bayesian binning," 2015. [Online]. Available: https://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9667/9958

[21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[22] Taubert, Oskar and Götz, Markus and Schug, Alexander and Streit, Achim, "Loss Scheduling Code Repository," 2020, [online, accessed at 2020-10-08]. [Online]. Available: https://github.com/KIT-MBS/loss-scheduling

[23] Torch Contributors, "torchvision," 2019, [online, accessed at 2020-02-04]. [Online]. Available: https://pytorch.org/docs/stable/torchvision/index.html