Automated Focal Loss for Image based Object Detection

Michael Weber*   Michael Fürst†   J. Marius Zöllner‡

FZI Research Center for Information Technology

Abstract

Current state-of-the-art object detection algorithms still suffer the problem of imbalanced distribution of training data over object classes and background. Recent work introduced a new loss function called focal loss to mitigate this problem, but at the cost of an additional hyperparameter. Manually tuning this hyperparameter for each training task is highly time-consuming.

With automated focal loss we introduce a new loss function which substitutes this hyperparameter by a parameter that is automatically adapted during the training progress and controls the amount of focusing on hard training examples. We show on the COCO benchmark that this leads to an up to 30 % faster training convergence. We further introduced a focal regression loss which on the more challenging task of 3D vehicle detection outperforms other loss functions by up to 1.8 AOS and can be used as a value range independent metric for regression.

1. Introduction

State-of-the-art object detection based on Convolutional Neural Networks (CNNs) currently can be seen as a competition between so called one-stage detectors [20] and two-stage approaches [9]. While the latter achieves better accuracy performances e. g. on one of the currently challenging object detection benchmarks, the COCO benchmark [19], they usually suffer from longer run times. As the number of candidate objects processed in the second stage is not known in advance, the run times furthermore are only roughly predictable in advance.

In contrast, one-stage detectors usually have faster and predictable run times, but are suffering from worse detection accuracy. In [18] the class imbalance has been identified as a source for this performance gap. Two-stage approaches avoid the imbalance problem as they filter most of the background before classification due to their first region

---

*michael.weber@fzi.de

†David_Michael.Fuerst@dfki.de Michael Fürst is now with the German Research Center for Artificial Intelligence

‡zoellner@fzi.de

Figure 1. Focal loss has introduced a method of weighing samples based on their prediction quality. However this weighing is constant during training (blue dashed line). We propose to automatically change the focus dependent on the training progress modeled as the expected probability of a correct prediction $\hat{p}_{correct}$. When the progress is low the focus is only on the worst predictions ($\hat{p}_{correct} = 0.01$, orange). As the progress improves (green) the automated loss shifts the focus to include better predictions only down weighting excellent predictions ($\hat{p}_{correct} = 0.7$, red).

proposal step (*e.g.* RPN [29], Selective Search [37], Edge Boxes [39] or DeepMask [23, 24]). In the second stage, methods like fixed background ratio [8] or online hard example mining (OHEM) [33] are used, to finally balance the training data.

For one-stage detectors, the situation is more challenging: As the detector needs to learn to distinguish between foreground classes and all possible background scenery it has to *see* a lot of possible background data. For this reason, the mentioned two-stage methods are not easily applicable to one-stage detectors. As a lever to increase accuracy the loss functions receive rising spotlight. Early one-stage object detectors made use of static loss functions – meaning that the hyperparameters of the loss were not changed during training of the network. As a first step to mitigate the class imbalance problem, so called α-balancing was added to the loss function to weight the losses for different classes/background according to their relative frequency. This mechanism was introduced to prevent the detectors

Figure 2. We propose two methods to estimate γ dependent on the training progress. The first method controls the relative weight mass assigned to examples that perform below average. Having this at a percentage at 0.7 percent between p and 1.0 performed best in our experiments. The second method uses the Shannon information of a correct prediction. $I(correct)$. The weight distribution they assign to a given training progress is similar. With a low training progress both assign weights only to the worst examples (bottom left) while at a high training progress both only reduce the weights for the best predicted examples (top right).

from always predicting the dominant class which usually is background. A first step towards a dynamic loss function was done by applying α-balancing to each minibatch separately.

With RetinaNet [18], additionally to the class distribution dynamics, a data centric dynamic was added to the loss function they called *focal loss*. For each prediction, its difficulty in each iteration is calculated based on the estimated probability for the correct class. The influence of this information to the loss function is controlled by adding a manually tuned hyperparameter γ.

In this paper, we propose a new loss function based on the focal loss [18] to automatically handle the class imbalance problem. Our *Automated Focal Loss* no longer relies on the manually tuned hyperparameter γ for balancing between easy and hard training examples. We present two methods for automatically adapting the dynamic difficulty term during training process as shown in Figures 1 and 2.

Finally we show that with our automated focal loss training converges to the same AP as the static focal loss [18] on the COCO benchmark, while converging in 30 % less time. Furthermore we tested our automated focal regression loss on the challenging *KITTI 3D Object Detection* dataset [6] demonstrating its effectiveness on less extensive datasets. On this dataset we outperform other losses and show that adding automated focal regression adds a slight edge to our loss over only using automated focal classification.

2. Related Work

CNN based Object Detection: The idea of detecting objects with the help of CNNs goes back to 1998 when LeCun et al. [16] extended the initial CNN classification approach also done by LeCun et al. [15] to the ability of detecting objects within an image. This network transformation which later was named Fully Convolutional Network (FCN) [21] implicitly models a sliding window within the network structure.

This architecture was the foundation of OverFeat [31], one of the first modern CNN based object detectors. Together with SSD [20, 4], YOLO [26, 27, 28] and the most recent RetinaNet [18] OverFeat spans the class of so called one-stage object detectors. They consist of one single convolutional network which extracts features, predicts object positions and classifies the objects.

The other large class of CNN object detectors are the so called two-stage approaches which adopted the two stages of classical object detection approaches into CNN architectures. In this classical chain, detection and classification of objects are separated into two different steps where classification between foreground object and background is applied upon the detection results. Detections in these approaches are usually called region proposals and do have to be thought of as candidate regions within an image where an object could be present. Region-CNN [8] was the first of these approaches while still using a Support Vector Machine (SVM) for the classification task after extracting features with a CNN. The following adaptions of this approach [7, 29, 10, 9, 17] turned the classification part completely into a CNN and enhanced the region proposal generation which dramatically increased the detection speed.

The differences between the one-stage and two-stage approaches have been mainly in execution speed and detection accuracy. While two-stage approaches usually reached a higher detection accuracy but to the cost of a longer execution run time. With the emergence of RetinaNet [18] using a focal loss function with an one-stage approach, they could achieve an comparable accuracy to recent two-stage approaches.

3D Vehicle Detection: Detection is not only limited to the case of two dimensional detection in the image plane. Recent approaches [25, 12] have extended the approach of two dimensional detection to the challenging task of detection in the space of the three dimensional world. The subfield of three dimensional vehicle detection is actively researched due to the development of autonomous vehicles. There are various approaches to estimate the pose of vehicles. They can be broadly divided into two categories. The first category indirectly estimates the pose and dimensions of the vehicle in the three dimensional space. In general these approaches estimate features or constraints onto which a template is fit. Approaches like DeepMANTA [1],

BoxCars [35, 22] and MergeBox [5] are some of the best performing in this category. On the other hand there is the category of networks which directly predict the bounding box in three dimensional space like [2] and the simple 3D detector we propose to test our loss on the challenging task of three dimensional detection. The latter category does not have issues with the approximation due to limitations of the template fitting and is a solution only requiring neural networks and no complex custom made post processing logic.

Loss Functions for Object Detection: The loss function as one crucial part of each one-stage object detection system usually can be divided into three parts: The loss function used for object classification, the analogue function for object position regression as well as the method for combining these two functions into the final loss function. For classification loss usually Softmax cross entropy [31, 27, 4] or Sigmoid cross entropy [18] are used. In [26], a L2 loss is used for the classification part. So called α-balancing is used to balance the influence of different samples based on their class distribution in the training set [20]. In [20] also a dynamic weighting of training samples according to their difficulty is done.

For the position regression task, the L2 loss [31, 26], its variant smooth L2 loss [20] or the similar smooth L1 loss [4] are commonly used functions. The combination of different loss functions is better known from multitask networks like Multinet [36] or [13] but classification and detection can also be seen as tasks within a multitask network. Object detection networks usually make use of a simple sum of losses [26] or a weighted sum of losses [31, 20, 4].

3. Automated Focal Loss

The focal loss as introduced by [18] eliminates the need for hard negative mining [33] and helps with the problem of imbalanced data. However, as stated in the original work, focal loss still needs α-balancing to successfully achieve competitive results on the COCO dataset [19]. By automating the focal loss and dynamically adjusting the focus (γ) to the current training progress, we found that neither hard negative mining nor α-balancing or any other method of simplifying the training is required. Automated focal loss enables the neural network to automatically focus itself onto the most important examples for the current training progress.

To correctly define and apply automated focal loss, a formal definition as done by [18] is required. The first step is to define the probability of the correct class. This means assigning the probability mass of the class that is the correct solution for the task to a variable. In case of a positive sample, this is the probability that the network assigned the class and in the other case it is the probability that the network did not assign to the class. This leads to a definition of $p_{correct}$ and the simplified cross entropy for a single example as follows:

$$p_{correct} = \begin{cases} p & \text{if } y = 1, \\ 1 - p & \text{else.} \end{cases} \quad (1)$$

$$L_{CE} = -\log(p_{correct}) \quad (2)$$

The loss can be weighted with a dynamic factor w which is dependent on the probability of the correct result $p_{correct}$. This factor defines which training samples the network is most focused on at any given point during the training.

$$L_{focused} = w \cdot L, \qquad w = (1 - p_{correct})^{\gamma} \quad (3)$$

So the focused loss as introduced by [18] is dependent on the weight which is computed using $p_{correct}$ and a γ which defines the amount of focus that is intended. Inspecting the derivative for the network variables `net` reveals the weight w remains unchanged in the derivative as a constant.

$$\frac{\partial L_{focused}}{\partial \text{net}} = w \cdot \frac{\partial L}{\partial \text{net}} \quad (4)$$

A high γ (*e.g.* $\gamma = 5$) weights down all samples and gradients that are approximately correct while focusing on the examples which are yielding poor performance. However, as the training progresses the number of poorly classified examples decreases. This leads to down-weighting an increasing number of samples in the loss and therefore in the gradients as well, leading to a significant drop in training speed due to decreasing gradients. On the other hand having a too low γ (*e.g.* $\gamma = 0$) reduces the impact of the weighting factor w. However, on complex tasks such as COCO and 3D detection, this will render the network incapable of learning the problem. Especially when the data is imbalanced the network can easily be overwhelmed by the dominant data.

We propose to adapt γ during the training progress over time overcoming the limitations of choosing a fixed γ. At the beginning of the training γ should be high to achieve a good focus on only the poorly predicted training samples. During training γ needs to shift towards 0 to avoid diminishing gradients due to down weighting of well predicted examples. We decided to model the training progress dependent on the expected probability of the correct prediction $\hat{p}_{correct}$ since this has a direct influence on the expected focal weight (equation 5).

$$E\{\log_{\gamma}(w)\} = E\{1 - p_{correct}\} = 1 - \hat{p}_{correct} \quad (5)$$

The expected probability of the correct prediction $\hat{p}_{correct}$ can be approximated by computing the mean over $p_{correct}$ for a training batch. In case of a small training batch applying a low pass filter like exponential smoothing is recommended. A smoothing via $\hat{p}_{correct} = 0.95 \cdot \text{old} + 0.05 \cdot \text{new}$ worked best in our experiments.

3.1. Choosing γ dependent on $\hat{p}_{correct}$

We generally see two options to define γ subject to the training progress dependent on $\hat{p}_{correct}$. The first option is inspired by the observation that the original focal loss has the issue of the expected weight diminishing as the training progresses. A formal definition to alleviate this issue is to force the integral of weight below the expected probability of the correct class $\hat{p}_{correct}$ to be equal to a fraction k of the total integral of the weight.

$$k = \frac{\int_0^{\hat{p}_{correct}} (1-p)^\gamma dp}{\int_0^1 (1-p)^\gamma dp} \tag{6}$$

This equation can be solved for γ by integrating and re-ordering the equation, leading to a formal definition of γ dependent on the chosen fraction of the weight that should be assigned to samples with a probability less than the expected probability.

$$\gamma = \frac{\log(1-k)}{\log(1-\hat{p}_{correct})} - 1 \tag{7}$$

Having defined γ this leads to the question what fraction k should be assigned to the examples performing worse than the average. The focal loss can only focus on poor predictions, if γ has a positive value. Equation 7 only yields positive and valid values when $\hat{p}_{correct} < k < 1$. Having a k close to $\hat{p}_{correct}$ is undesirable, since this will lead to disabling the focal loss after only a few training epochs as $\hat{p}_{correct}$ increases. Whereas $k = 1$ means disabling the focal loss. Therefore adapting k dependent on the training progress seems reasonable. A simple way is to keep k at a certain point between the lower boundary $\hat{p}_{correct}$ and the upper boundary of 1. This leads to a definition of k using an interpolation parameter h between the lower and the upper boundary.

$$k = (h \cdot \hat{p}_{correct} + (1-h) \cdot 1) \tag{8}$$

Resulting in a final equation to compute γ given h and $\hat{p}_{correct}$.

$$\gamma = \frac{\log(1 - (h \cdot \hat{p}_{correct} + (1-h) \cdot 1))}{\log(1-\hat{p}_{correct})} - 1 \tag{9}$$

While this definition yields a numerically stable computation of γ and fulfills the requirements to model γ in an appropriate way. Namely, for $\hat{p}_{correct} \to 0$ it diverges to $+\infty$ and for $\hat{p}_{correct} \to 1$ it converges to 0 as can be seen in Figure 3. Even though this approach fulfills the requirement of an automatic focus, it trades the hyperparameter γ for the static focus for a hyperparameter h. The hyperparameter gives the flexibility to adapt the loss to the needs of the situation, however if it is not mandatory for success, fewer hyperparameters is the preferred solution.

Figure 3. Gamma is dependent on $\hat{p}_{correct}$. The parameter h defines the point between $\hat{p}_{correct}$ and 1.0 that defines the quantile of weight that is assigned to values less than $\hat{p}_{correct}$ (see Equation 9). A high value like 0.9 leads to very low gamma values across the board leading to a low focus, while low values result in higher gamma values and a stronger focus. We found an interpolation value of $h = 0.7$ to be a good specification.

Defining γ inspired by the information as defined by Shannon [32] can eliminate the need for a hyperparameter. For the loss computation of a neural network, the cross entropy, which has its origin in the information theory by Shannon, is commonly accepted. Therefore picking the information in a correct prediction of the network $I(correct)$ seems reasonable. The information has the property that for an optimal network it will converge towards 0. Since there is no surprise in the outcome, it perfectly matches the ground truth. On the other hand for a poorly trained network, the information increases without any upper limit. Defining γ as the Shannon information of a correct predictions of the network leads to the following equation:

$$\gamma = I(correct) = -log(\hat{p}_{correct}) \tag{10}$$

When checking the definition for sanity the results are as intended. For a low $\hat{p}_{correct}$ representing a low training progress γ is large and therefore the focus on the wrong samples high. As the training progresses and the $\hat{p}_{correct}$ increases the value of γ decreases leading to less focus on the poorly predicted samples (see Figure 4). An ideal network of $\hat{p}_{correct} = 1$ would lead to no focus on poorly predicted examples since there are none.

3.2. Single Target Classification

The application of the automated focal loss to a problem requires defining the loss L that should be focused and defining a computation policy for $\hat{p}_{correct}$. For single target classification the loss L in equation 3 should be normal cross entropy loss L_{CE}. Computation of γ using $\hat{p}_{correct}$ can be done via equation 9 or 10. However, the latter should

Figure 4. Changing γ according to the progress of the network can also be modeled as the information in a correct prediction $I(correct) = -\log(\hat{p}_{correct})$. While the curves for an interpolation value of $h = 0.7$ and the information $I(correct)$ based approach look similar, in our experiments using the information outperformed using a quantile dependent on $\hat{p}_{correct}$. Furthermore the information based approach has no hyperparameters that need tuning.

be preferred, since it yields better results in our experiments and needs no hyperparameter. The required $\hat{p}_{correct}$ can be computed as the average of $p_{correct}$ for the current training batch. To improve the approximation of $\hat{p}_{correct}$ we found that applying an exponential smoothing worked best.

3.3. Multi-Target Classification

In case of multi-target classification, the loss L in equation 3 is the binary cross entropy loss L_{binary_CE}. Computation of γ using $\hat{p}_{correct}$ can be done via equation 9 or 10. Estimating $\hat{p}_{correct}$ is a little more involved than for the case of single target classification. The probability of a correct class $p_{correct}$ is defined as p in case of a class being active and as $1 - p$ when a class is inactive. The best way to estimate $\hat{p}_{correct}$ depends on the problem that is to be solved and is not as easy as simply averaging $p_{correct}$. The average of $p_{correct}$ would be dominated by negative classes since typically in a multi-target classification task only a few classes are active at the same time for a single example.

When positive and negative examples are distributed more or less equally, the solution is defining $\hat{p}_{correct}$ as the mean of the probabilities p assigned to all active classes of the example. Thus avoiding the positive case being overwhelmed by inactive classes. When an example has no positive classes it is called a negative example. For negative examples the probabilities not assigned to the classes $1 - p$ are averaged. However, in the case of a task with a lot of negative examples and only few examples with active classes the computation of $\hat{p}_{correct}$ can be easily overwhelmed by the number of negative examples. Therefore only examples

with at least one active class should be considered for computing the average to estimate $\hat{p}_{correct}$.

3.4. Regression

The original focal loss and the automated focal loss can also be applied to regression. We propose a new method to convert regression predictions into probabilities that can be used for focal loss. Beyond usage in focal loss, these probabilities will also give a representation of the training progress that is independent of value ranges of the problem.

The core idea to apply focal loss to regression problems is to compute the probability that a prediction is better than the labels. The underlying assumption is that labels are not perfectly accurate and have an error. Using the deviation between prediction and ground truth Δx and the deviation between the label and the ground truth Δx_t leads to the formulation of $p_{correct}$ as the following probability that Δx_t is not between $\pm|\Delta x|$.

$$p_{correct} = 1 - p(-|\Delta x| < \Delta x_t < |\Delta x|) \qquad (11)$$

We assume that the labels are distributed around the actual correct ground truth by a Gaussian distribution with a variance of σ^2. With this assumption it is possible to compute the probability $p_{correct}$ using the cumulative distribution function Φ of the Gaussian distribution.

$$p_{correct} = 1 - (\Phi(\frac{|\Delta x|}{\sigma^2}) - \Phi(-\frac{|\Delta x|}{\sigma^2})) \qquad (12)$$

However, to correctly compute the cumulative distribution function the variance σ^2 of the task needs to be estimated. As elaborated in detail in [13] the uncertainty related to a task which is inherent to the task and the labels can be estimated by adding $\log(\sigma^2 + 1)$ to the loss and training the variable σ^2 like a weight of the network. This leads to the final form of the focused loss for regression using $p_{correct}$ as defined in equation 12 with a γ that can be either be constant as in the classical focal loss or even automated as proposed in equations 9 and 10:

$$L_{focused} = w \cdot L + \log(\sigma^2 + 1) \qquad (13)$$

with

$$w = (1 - p_{correct})^{\gamma} \qquad (14)$$

This formulation leads to an implicit trade-off shift during training. At the beginning of the training, the loss can focus on examples with a low probability of being correct, like for the automated focal loss on classification. With progression in the training as the probabilities increase the focus shifts and examples with a low probability are no longer weighted more than examples with a medium probability as seen in Figure 2. This leads to a reduced impact of outliers on the performance of the network.

3D Detection Decoders						
Output	Value Range	layer type	#Filters	Filter Size	stride	Activation
probs	$[0, 1]$	conv	#classes	1x1	1,1	softmax
c_x, c_y	$(-\infty, \infty)$	conv	2	1x1	1,1	-
d	$[0, d_{max}]$	conv	1	1x1	1,1	$d_{max} \cdot \sigma(\cdot)$
$\sin(\theta), \cos(\theta)$	$[-1, 1]$	conv	2	1x1	1,1	$\tanh(\cdot)$
w, h, l	$(-\infty, \infty)$	conv	3	1x1	1,1	-

Table 1. The 3D detector has 5 decoders with differing value ranges, number of filters and activation functions. Each decoder consists of a convolution layer with the parameters given in the table. The bounding box is encoded as the center c_x, c_y in image coordinates relative to the output position. The distance d from the camera, the orientation θ encoded as $\sin(\theta), cos(\theta)$ - to avoid inconsistencies at 0 and 2π - and the dimensions of the bounding box width w, height h and length l.

The beauty of the formulation as a probability is beyond the capability of computing a focal loss on a regression problem. The value of $\hat{p}_{correct}$ is independent of the actual value range of the problem, due to normalizing the absolute distance by the variance of the task σ^2 - assumed σ^2 is estimated correctly. This leads to $\hat{p}_{correct}$ being a new metric on monitoring your training progress.

4. Detection Network

For comparability with the original focal loss we chose RetinaNet as the network for predicting detections on the COCO dataset. Since we propose a new loss the base network remains unchanged from [18].

However, beyond applying the loss on 2D detection on COCO we applied our newly introduced loss on the very challenging task of 3D vehicle detection. To keep the focus on the loss and not the network architecture, we decided to use a simple VGG16 [34] encoder with a decoder specialized on 3D detection as presented in [38]. The VGG16 encoder consists of the VGG16 network weights pre-trained on the ImageNet dataset [3] up to the layer *pool5*. Then two 1x1 convolution layers with 4096 filters similar to the fully connected layers from the original VGG network are appended. On the resulting feature maps a 3D decoder is applied. The 3D decoder consists of 5 parallel streams:

1. A 1x1 convolution softmax prediction for the classification p_{class},

2. a 1x1 convolution layer with 2 filters to predict the center of the object c_x, c_y,

3. a 1x1 convolution layer with 1 filter to predict the distance of the object from the camera,

4. a 1x1 convolution layer with 2 filters and $\tanh$ activation function to predict the orientation θ encoded as $\sin(\theta)$ and $\cos(\theta)$,

5. a 1x1 convolution layer with 3 filters to predict the size of the object encoded as width, height and length (w, h, l) of the object.

All details on the value ranges and the number of filters can be found in Table 1.

5. Experiments

We present experimental results on the bounding box detection task of the COCO benchmark [19] and the challenging KITTI 3D Object Detection dataset [6]. We will not focus on the architecture of the network but rather comparing results achieved with different losses.

On COCO we evaluated our loss by training with the same parameters as the original ResNet50 [11] based RetinaNet in [18]. On our hardware the original focal loss implementation achieved an AP of 30.41 slightly lower than the 30.5 we expected from the paper however this is plausible due to differences in the hardware setup and random initialization. The proposed automated focal loss achieved an AP of 30.38 without α-balancing. Note that the original focal loss can only achieve 30.41 with α-balancing. The automated focal loss with focal regression converged after only 30 hours, whereas the original focal loss with a constant γ converged after 44 hours. This means the automated focal loss with focal regression converged at approximately $\frac{2}{3}$ the time the original focal loss required.

We found that during large parts of the training the value for γ computed as the information in a correct prediction $I(correct)$ stayed at a value of $\gamma = 2.2$ which is very close to the value that [18] found at $\gamma = 2$. Only at the beginning of the training γ started out at a value larger than 6 converging slowly towards 0 once the AP reached a plateau at around 30.0. This demonstrates that automated focal loss is capable of finding the optimal γ for a given problem in only one training run and converging towards a similar result as the focal loss with a γ that was hand tuned with several experiments.

When inspecting the AP_{50} we found automated focal loss with focal regression to have a far superior AP_{50} of 51.18 compared to the $AP_{50} = 46.58$ reproduced with the original focal loss. Using our loss yields an improvement of 4.6 AP_{50} while keeping the AP the same. An overview of the COCO results can be found in Table 2.

COCO benchmark

Approach	AP	AP_{50}	Time to Converge
ResNet 50 - 400 [18]	30.5	47.8	N.A.
Reproduced [18]	**30.41**	46.58	44 h
Ours (automated + focal regression)	30.38	**51.18**	**30 h**

Table 2. The values stated in [18] are slightly higher than those we were able to reproduce. Our automated focal loss using focal regression performs the same as a normal focal loss with α-balancing when comparing the AP. However, the AP_{50} is significantly improved when using the our loss. Furthermore the time required until the network converges is significantly reduced with our loss since it is capable to adapt to the training progress and especially at the beginning makes quicker progress than a focal loss with constant γ.

Beyond evaluating on COCO and comparing to focal loss we tested our approach on the challenging task of 3D detection. We used the VGG16 Encoder with a 3D decoder. The dataset consists of 7481 images posing the challenge to learn a difficult task on few data. We evaluated the simple network trained with a normal loss, α-balanced loss, multi-loss [13], our automated focal classification loss and finally our automated focal classification and regression loss. The input image size was a random crop of size 256 x 256 pixels which contained at least one detection. Each image of the original dataset was cropped in 20 different ways and augmented with geometric augmentation such as horizontal flipping and micro-translations as well as texture augmentation such as intensity, contrast, saturation modification and color jitter (as introduced in [30]).

For training the networks we used the same hyperparameters for all losses. As optimizer we used Adam [14] with default parameters. For the normal loss and the α-balanced loss we chose the weights for all regression losses to be 1.0 and 10.0 for the cross entropy classification loss. The value of α was computed for every batch as $1 - h$ where h is the relative frequency of the class. The learning rate was exponentially decayed starting at 0.0001 and ending at 0.000001 after 160,000 training steps with a batch size of 16. Training was done in 24 hours on a single NVIDIA GTX 1080 Ti with Tensorflow 1.10.

Since we have already shown on COCO that our loss is capable of finding an optimal γ we decided to not test focal loss on 3D detection, since the comparison would be unfair, whereas there is no prior work on what γ is optimal. The automated focal classification loss outperformed the normal loss, α-balanced loss and the multiloss on AOS with 37.0 compared to the next best result at 36.1 achieved by multiloss [13]. Adding automated focal regression loss increased the AOS by 0.3. The full report of all AOS scores and AP scores can be found in Table 3.

6. Conclusion

Focal loss reduces the impact of class imbalance but still relies on α-balancing and picking a convenient focal factor γ for the task. Having a constant γ it is unable to adapt its focus to the current training progress. We presented an approach to overcome this by computing γ dependent on the training progress and introducing a new kind of loss that is capable to shift its focus during training. Our experiments on COCO showed that even though a good constant γ can achieve the same AP as our loss, we outperform regarding AP_{50} and time to convergence in the training process without the need of any hyperparameter tuning to achieve this. We further introduced a novel technique to compute a probability for a regression loss to achieve better performance than the task dependent variance. This enables us to introduce a focal regression loss and a new metric to monitor the training progress of a regression task independent of the value range. We showed that on the KITTI 3D Object Detection dataset our automatic focal loss outperformed other losses.

References

[1] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[2] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009.

[4] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. Berg. DSSD: Deconvolutional Single Shot Detector. *CoRR*, abs/1701.06659, 2017.

[5] N. Gählert, M. Mayer, L. Schneider, U. Franke, and J. Denzler. MB-Net: MergeBoxes for Real-Time 3D Vehicles Detection. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018.

[6] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012.

[7] R. Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*. IEEE, 2015.

KITTI 3D Object Detection

Loss function	top down AP	AOS	prediction FPS
Normal	-	35.5	28.40
α-balance	-	35.9	28.41
Multiloss	20.1	36.1	28.34
Ours (automated focal classification)	24.5	37.0	**28.43**
Ours (automated focal classification + regression)	**25.0**	**37.3**	28.18

Table 3. On the KITTI 3D Object Detection dataset where no optimal γ is known, we applied automated focal loss and automated focal loss with regression. This eliminates the need to empirically determine γ. The results of our loss function are superior to using just a normal loss, α-balancing or multiloss. Our loss achieves these values without the need for α-balancing, hard negative mining or hyperparameter tuning. On the AP from the top down view of the scene we significantly outperform multiloss whilst also outperforming on the average orientation score (AOS). All losses trained the same network designed for simplicity and prediction speed.

[8] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014.

[9] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 37(9):1904–1916, 2015.

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[12] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again. In *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[13] A. Kendall, Y. Gal, and R. Cipolla. Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics. 2018.

[14] D. P. Kingma and J. Ba. ADAM: A Method for Stochastic Optimization. *International Conference on Learning Representations (ICLR)*, 2015.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural computation*, 1(4):541–551, 1989.

[16] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature Pyramid Networks for Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal Loss for Dense Object Detection. *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2018.

[19] 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 *European Conference on Computer Vision (ECCV)*. Springer, 2014.

[20] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

[21] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2015.

[22] A. Mousavian, D. Anguelov, J. lynn, and J. Košecká. 3D Bounding Box Estimation Using Deep Learning and Geometry. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[23] P. Pinheiro, R. Collobert, and P. Dollár. Learning to Segment Object Candidates. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[24] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to Refine Object Segments. In *European Conference on Computer Vision (ECCV)*. Springer, 2016.

[25] P. Poirson, P. Ammirato, C.-Y. Fu, W. Liu, J. Kosecka, and A. C. Berg. Fast Single Shot Detection and Pose Estimation. In *International Conference on 3D Vision (3DV)*, 2016.

[26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[27] J. Redmon and A. Farhadi. YOLO9000: Better, Faster, Stronger. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.

[28] J. Redmon and A. Farhadi. YOLOv3: An Incremental Improvement. *CoRR*, 1804.02767, 2018.

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[30] E. Romera, L. M. Bergasa, J. M. Alvarez, and M. Trivedi. Train Here, Deploy There: Robust Segmentation in Unseen Domains. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018.

[31] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *CoRR*, abs/1312.6229, 2013.

[32] C. E. Shannon. A Mathematical Theory of Communication. *Bell System Technical Journal*, 27(3):379–423, 1948.

[33] A. Shrivastava, A. Gupta, and R. Girshick. Training Region-Based Object Detectors with Online Hard Example Mining. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016.

[34] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICLR)*, 2015.

[35] J. Sochor, J. Špaňhel, and A. Herout. BoxCars: Improving Fine-Grained Recognition of Vehicles Using 3-D Bounding Boxes in Traffic Surveillance. *Transactions on Intelligent Transportation Systems (T-ITS)*, 2018.

[36] M. Teichmann, M. Weber, J. M. Zöllner, R. Cipolla, and R. Urtasun. MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018.

[37] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders. Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[38] M. Weber, M. Fürst, and J. M. Zöllner. Direct 3D Detection of Vehicles in Monocular Images with a CNN based 3D Decoder . In *IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019.

[39] C. L. Zitnick and P. Dollár. Edge Boxes: Locating Object Proposals from Edges. In *European Conference on Computer Vision (ECCV)*. Springer, 2014.