Focal Loss for Dense Object Detection

- [[📚 1708.02002.pdf]]
- **Basic Idea**:
  - Since single-stage detectors do not use region proposals (they do a dense prediction i.e. at each spatial location predict boxes of various shapes, sizes and aspect ratio etc.), there is a high class imbalance between positive (with object) and negative class. Can we alleviate this class imbalance problem by dynamically weighing hard samples (both positive and negative)?
  - Due to this class imbalance the training initially is unstable. Can we use better initialisation technique to overcome this class imbalance?

- **NOTE:** There are two concepts overlapped here. Dominant (Negative) vs Rare (Positive) class. Hard vs Easy examples. The assumption is that since negative class is dominant, hence can be easily classified. Therefore, as a proxy for solving class imbalance problem, the authors discount the easily classified examples. Therefore, it can happen that even hard-negatives are discounted less. However, in practice this rarely happens as validated by the following experiment.

  🟡 **P7**

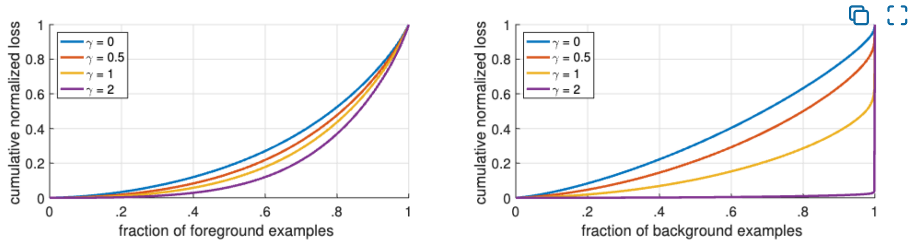

Figure 4. Cumulative distribution functions of the normalized loss for positive and negative samples for different values of γ for a *converged* model. The effect of changing γ on the distribution of the loss for positive examples is minor. For negatives, however, increasing γ heavily concentrates the loss on hard examples, focusing nearly all attention away from easy negatives.

- **Why class imbalance is a problem?**
  - 🟡 **P3** training is inefficient as most locations are easy negatives that contribute no useful learning signal;
  - 🟡 **P3** en masse, the easy negatives can overwhelm training and lead to degenerate models Degenerate model as in which always predicts negative class.

- **Earlier attempts to combat class-imbalance**
  - In two-stage detectors, region proposal significantly filters regions with background. Thereafter, while selecting batches sampling heuristics are applied such as 🟡 **P2** fixed foreground-to-background ratio (1:3), or online hard example mining (OHEM) [31], are performed to maintain a manageable balance between foreground and background.
  - 🟡 **P2** This inefficiency is a classic problem in object detection that is typically addressed via techniques such as bootstrapping [33, 29] or hard example mining [37, 8, 31].
  - *Balanced cross-entropy loss*: Used in YOLO.

- **Focal Loss**
  -

🟡 **P1**

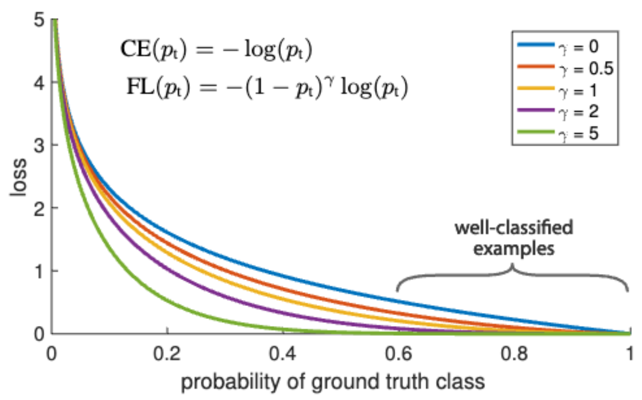

Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

- **Cross-Entropy(CE)** loss for binary classification:

  $$CE(p,y) = \begin{cases} -log(p) & \text{if } y = 1 \\ -log(1-p) & \text{otherwise} \end{cases}$$

  where $y \in \{\pm 1\}$ is the ground-truth label and p is the predicted probability for class with label $y = 1$. Define p_t as

  $$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise} \end{cases}$$

  Therefore, p_t represents the predicted probability for the ground-truth label class i.e. the correct class.
  🟡 **P3** One notable property of this loss, which can be easily seen in its plot, is that even examples that are easily classified (pt .5) incur a loss with non-trivial magnitude. When summed over a large number of easy examples, these small loss values can overwhelm the rare class.
- **Balanced Cross-Entropy** (Used in YOLO):

  $$CE(p_t) = -\alpha_t log(p_t)$$

  where α_t is a weighing term defined similar to p_t. The objective of this is to weigh rare class examples higher than dominant class (has the similar effect of over-sampling rare class or maintaining foreground to background ratio in the samples).
  🟡 **P3** Whileα balances the importance of positive/negative examples, it does not differentiate between easy/hard examples.
- **Focal Loss**

  $$FL(p_t) = -(1-p_t)^\gamma log(p_t)$$

  where $\gamma \geq 0$ is a tunable focussing parameter.
- Properties of Focal loss:
  - 🟡 **P3** When an example is misclassified and pt is small, the modulating factor is near 1 and the loss is unaffected. As pt → 1, the factor goes to 0 and the loss for well-classified examples is down-weighted.

An example can only be misclassified when p_t is small.

  - 🟡 **P3** The focusing parameter γ smoothly adjusts the rate at which easy examples are downweighted. When γ = 0, FL is equivalent to CE, and as γ is increased the effect of the modulating factor is likewise increased (we found γ = 2 to work best in our experiments).
  - 🟡 **P3** Intuitively, the modulating factor reduces the loss contribution from easy examples and extends the range in which an example receives low loss. For instance, with γ = 2, an example classified with pt = 0.9 would have 100× lower loss compared with CE and with pt ≈ 0.968 it would have1000× lower loss. This in turn increases the importance of correcting misclassified examples (whose loss is scaled down by at most 4× for pt ≤ .5 and γ = 2).
    The last one is true because $\gamma \leq \left(1 - \frac{1}{2}\right)^2 = \frac{1}{4}$.
  - Note that for both hard as well as easy examples, the loss is getting down weighed. It's just that for hard examples it is less and for easy it is more.
  - As γ is increased the effect of down-weighing is more pronounced especially for the misclassified samples.
  - 🟡 **P5** We emphasize that when training RetinaNet, the focal loss is applied to all ~100k anchors in each sampled image. This stands in contrast to common practice of using heuristic sampling (RPN) or hard example mining (OHEM, SSD) to select a small set of anchors (e.g., 256) for each minibatch.
  - 🟡 **P5** The total focal loss of an image is computed as the sum of the focal loss over all ~100k anchors, normalized by the number of anchors assigned to a ground-truth box. We perform the normalization by the number of assigned anchors, not total anchors, since the vast majority of anchors are easy negatives and receive negligible loss values under the focal loss.

- α-balanced version of focal loss works better in practice:

$$FL(p_t) = -\alpha_t(1-p_t)^{\gamma}log(p_t)$$

- 🟡 **P3** Finally, we note that the implementation of the loss layer combines the sigmoid operation for computing p with the loss computation, resulting in greater numerical stability
- 🟡 **P5** n general α should be decreased slightly as γ is increased (for γ = 2, α = 0.25 works best) 🟡 **P7** We observe that lower α's are selected for higher γ's (as easy negatives are downweighted, less emphasis needs to be placed on the positives).
  α gives more emphasis to the foreground (dominant/hard class). So, when γ is increased we are already giving more emphasis to hard examples (dominated by foreground class), hence if we keep high values of α then training may only focus on hard examples which is also not something we would want.

- **Class Imbalance and Model Initialization**
  - 🟡 **P4** Binary classification models are by default initialized to have equal probability of outputting either y = −1 or 1. Under such an initialization, in the presence of class imbalance, the loss due to the frequent class can dominate total loss and cause instability in early training.
  - 🟡 **P4** To counter this, we introduce the concept of a 'prior' for the value of p estimated by the model for the rare class (foreground) at the start of training. We denote the prior by π and set it so that the model's estimated p for examples of the rare class is low, e.g. 0.01.
  - This helps the model to initially predict the frequent class.
  - 🟡 **P5** For the final conv layer of the classification subnet, we set the bias initialization to b = − log((1 − π)/π), where π specifies that at 🟡 **P6** the start of training every anchor should be labeled as foreground with confidence of ~π. We

use π = .01 in all experiments, although results are robust to the exact value. As explained in §3.3, this initialization prevents the large number of background anchors from generating a large, destabilizing loss value in the first iteration of training.
Intuition:

$$\frac{1}{1+e^{-z}} = \pi \implies z \quad = -log\left(\frac{1-\pi}{\pi}\right)$$

Now, since π represents the probability of the rare class, which we want to be low. Therefore, we want $z = w^T x + b$ to be low. Assume, $x \sim \mathcal{N}(0, \sigma)$ and $\mathbb{E}[w^T x] = 0$. Therefore, b can be set as $log\left(\frac{1-\pi}{\pi}\right)$.

- 🟡 **P4** this to improve training stability for both the cross entropy and focal loss in the case of heavy class imbalance

- **Tricks of the trade**:
  - 🟡 **P5** We note that unlike most recent work, we use a class-agnostic bounding box regressor which uses fewer parameters and we found to be equally effective This is similar to YOLO and YOLOv2. Class-specific bounding box regressor would be for each class you have separate bounding box regressor. Note that this is different from YOLOv2, which had separate classification head for each anchor box (there is no class-specific bounding box regressor).
  - 🟡 **P6** The training loss is the sum the focal loss and the standard smooth L1 loss used for box regression [10]

$$smooth_{L1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

    The above smooth L1 loss is less sensitive to outliers. This is same as the Huber Loss.

- **Results**
  - 🟡 **P7** Analysis of the Focal Loss: The conclusion of this experiment is that for the positive examples the loss is mostly dominated by the top 20% of the hard examples, whereas for the negative examples, the loss is mostly 0 for almost all of the examples except for extremely hard negative examples (indicated by the sharp peak in the graph at 1). Moreover, CDF for negative examples show that for $\gamma = 0$ negative examples get assigned high loss values and therefore indicates the efficacy of focal loss.
  - 🟡 **P7** Like the focal loss, OHEM puts more emphasis on misclassified examples, but unlike FL, OHEM completely discards easy examples. We also implement a variant of OHEM used in SSD [22]: after applying nms to all examples, the minibatch is constructed to enforce a 1:3 ratio between positives and negatives to help ensure each minibatch has enough positives.

- **Relation to other losses**
  - *Robust Estimation*: Losses such as Huber loss 🟡 **P3** reduce the contribution of outliers by down-weighting the loss of examples with large errors (hard examples). In contrast, rather than addressing outliers, our focal loss is designed to address class imbalance by down-weighting inliers (easy examples) such that their contribution to the total loss is small even if their number is large. In other words, the focal loss performs the opposite role of a robust loss: it focuses training on a sparse set of hard examples

- *Hinge Loss*: 🟡 **P7** Finally, in early experiments, we attempted to train with the hinge loss [13] on pt, which sets loss to 0 above a certain value of pt. However, this was unstable and we did not manage to obtain meaningful results.
  Look at this to understand how hinge loss helps in focussing training on hard examples. The reason why I think focal loss works better than hinge loss is that hinge completely ignores easy examples just like OHEM whereas focal loss still takes them into account.

Unlinked References