Small Object Detection in Satellite Imagery

Claire Huang
chuang20@stanford.edu

Derek Huang
huangda@stanford.edu

Edward Lee
edlee1@stanford.edu

Abstract

Small objects ($<1024px^2$) remain challenging for state-of-the-art object detection models. We examine this problem in the context of satellite image object detection, which suffers from severe class and bounding box area imbalance. In particular, we explore anchor box and loss function modifications in order to focus standard object detection models like YOLOv3 and Faster-RCNN on small objects without increasing training or detection overhead. We observe particular success with area-based weights on the classification loss term, which is not otherwise linked to bounding box area.

1. Introduction

Recent algorithmic advancements in the field of computer vision have contributed heavily towards complete image understanding. Detecting objects with high accuracy has extended the field of computer vision to a variety of real-world applications, from medical image diagnoses to self-driving cars. In all these scenarios, detecting small objects is crucial for accurate and successful performance. For instance, early stage diagnosis often involves the detection of abnormalities that occupy only a few pixels on an image. Autonomous driving, on the other hand, necessitates the classification of small and distant objects, like pedestrians. While current object detection methods have made significant general progress on datasets like PASCAL VOC [8] and MS COCO [19], they still often under-perform on small objects [14], with average precision of small object detection being as low as half that of larger objects.

1.1. Problem Statement

Our task is a standard object detection task that uses high resolution satellite imagery from the 2018 xView challenge dataset [15]. The xView dataset is currently the largest publicly available satellite imagery dataset, thereby providing significant access to small object data. The input to our algorithm is an image, where we then use various convolutional models to predict both the bounding boxes and classes of all objects contained in the image.

2. Related Work

2.1. Small object detection

State-of-the-art object detection models struggle on small objects for a few primary reasons: there are usually fewer examples of small objects in training data, and each example is low resolution, yielding noisy representations in the models [16] [14]. The simplest approach to addressing small object data imbalance is data re-sampling techniques, such as undersampling [5] and oversampling. Undersampling generally shows poor performance in CNNs, as it discards training data [2]. More advanced oversampling techniques avoid overfitting by training on interpolated synthetic examples [3]; however these examples are computationally intensive to generate in the deep representation space of CNNs [1]. [14] further augments oversampled images by artificially pasting multiple copies of the small objects per image; however, this necessitates access to segmentation masks. In general, oversampling techniques increase the time required to train object detection models.

Another approach to addressing small object data imbalance is new loss functions. Focal loss [18] reshapes cross-entropy loss to downweight easily classified examples (large objects, for example) and allow hard examples from minority classes (small objects) to contribute more to weight updates. The winners of the xView Challenge propose a modified focal loss that softens the response to hard examples with a piecewise function [22]. Class-balanced loss [4] is another extension of focal loss that defines the α hyperparameter to be the effective number of samples per class. We find loss function modifications a promising area for exploration because they are less likely to adversely affect training time or cause overfitting.

2.2. Satellite images

A number of satellite image detection papers focus on class-specific detection. For example, [13] achieves high performance on vehicle detection by using prior knowledge about road network locations and vehicle appearance, both of which are highly specific and resource intensive preprocessing steps.

For general techniques, [23] proposes a super-resolution

preprocessing step, which appears to improve small object performance on SSD but not for other object detection models. A few papers propose architecture modifications to improve small object detection in satellite images. YOLT [7] is a modification to YOLO, with a denser detection grid and separately trained classifiers for large and small objects. Although the additional classifier negligibly affects detection runtime, it does increase training overhead. [6] finds that YOLT outperforms vanilla object detection models like YOLO and Faster-RCNN.

3. Methods

There are two major classes of object detection methods: one-shot detectors and two-stage detectors. One-shot detectors divide the input image into grid cells and detect and classify bounding boxes per cell in one pass, while two-stage detectors first run the input image through a region proposal network before proceeding to detection and classification. One-shot detectors are generally faster, while two-stage detectors are more accurate. We use one of each type of detector for our experiments: YOLOv3 [20] for one-shot detection and Faster R-CNN [21] for two-stage detection.

3.1. YOLOv3

YOLOv3 is a one-shot detector with an underlying convolutional neural network as its feature extractor. YOLOv3 utilizes uses a backbone architecture with 53 convolutional layers known as Darknet-53. Darknet-53, like any other convolutional feature extractor, progressively downsamples the original input. In order to boost object detection at various scales, YOLOv3 employs a Feature Pyramid Network-like [17] structure. The FPN takes the input map at various depths of the network and adds it element-wise it with the 2x upsampled featured map of two layers prior. In doing so, the feature map is able to handle both high and low level information information. YOLOv3 predicts at 3 different scales, where each different scale determines which layer of Darknet-53 to draw the initial feature map from. The anchors are split across the scales. Detection is done by using a **1x1x(Bx(4+1+C))** kernel onto each feature map. The output is therefore equivalent to a 1D tensor, where for each of the **B** present bounding box predictions, there are 4 location predictions (t_x, t_y, t_w, t_h) which stand for the center of the bounding box and the width and height offsets, 1 confidence score prediction indicating whether there is an object in the box, and **C** class predictions. Note that YOLOv3 uses multiclass prediction rather than a softmax layer.

To improve small object detection over the baseline, we focused on modifying anchor box sizes and adjusting the loss function to favor small objects.

3.1.1 Anchor boxes

Instead of predicting bounding boxes from scratch, YOLOv3 predicts offsets from predetermined anchors. During training, only one anchor gets assigned to each ground truth. As mentioned earlier, the model outputs 4 coordinates for each bounding box (t_x, t_y, t_w, t_h). If the cell is offset from the top left corner of the image by (c_x, c_y) and the anchor has predefined width and height p_w, p_h, then the offset predictions are computed as follows:

$$
\begin{aligned}
b_x &= \sigma(t_x) + c_x \\
b_y &= \sigma(t_y) + c_y \\
b_w &= p_w e^{t_w} \\
b_h &= p_h e^{t_h}
\end{aligned}
$$

As we can see, the anchor dimensions p_w, p_h define the size of the predicted bounding box.

Anchor box sizes are usually computed across the entire dataset using k-means clustering. However, we propose modifying the anchor generation to favor smaller anchor boxes, thus making the model more sensitive to small objects.

3.1.2 Baseline loss function

YOLOv3's baseline loss function can be modeled as follows, where the first two terms represent region proposal (localization and area) loss and the last two terms represent classification (confidence and class) loss.

$$
\begin{aligned}
& c_{\text{pos}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \left((t_x - \hat{t}_x)^2 + (t_y - \hat{t}_y)^2 \right) \\
& + c_{\text{area}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \left((\sqrt{t_w} - \sqrt{\hat{t}_w})^2 + (\sqrt{t_h} - \sqrt{\hat{t}_h})^2 \right) \\
& + c_{\text{conf}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{obj} \left[-\log(\sigma(c)) + \sum_{k=1}^{C} CE(\hat{y}_k, \sigma(s_k)) \right] \\
& + c_{\text{cls}} \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{i,j}^{noobj} \left[-\log(1 - \sigma(c)) \right]
\end{aligned}
$$

In our experiments, we tried tuning each of four coefficients to better understand how each term contributes to the model's performance.

3.1.3 Area Weights

Although both focal loss [18] and reduced focal loss [22] have shown success improving object detection performance in general, the loss functions don't focus specifically on small objects.

As such, we attempt to introduce weights that can be added to loss functions that will explicitly push the model to focus on small objects.

First, we propose a naive discretized constant weighting of the classification term, increasing the coefficient for small object bounding boxes with area less than threshold $t_a \text{px}^2$, where $t_a = 1024$ by COCO's definition of small objects.

$$c_{\text{cls}} = \text{AW-d}(c, A) = \begin{cases} 1 & A \leq t_a \\ \frac{1}{c} & A > t_a \end{cases}$$

Next, we extend this piecewise idea with a partially continuous function, inspired by reduced focal loss [22]:

$$c_{\text{cls}} = \text{AW-p}(c, A) = \begin{cases} 1 & A \leq t_a \\ \left(\frac{t_a}{A}\right)^c & A > t_a \end{cases}$$

Finally, we propose removing the piecewise component for a fully continuous function that steeply penalizes very small objects:

$$c_{\text{cls}} = \text{AW-c}(c, A) = \left(\frac{t_a}{A}\right)^c$$

3.2. Faster R-CNN

Like YOLOv3, Faster R-CNN [21] depends on an underlying convolutional neural network as a feature extractor. In this case, ResNet [12] and its variants are often the backbone of choice. An FPN [17] can also be attached to the backbone to improve accuracy by extracting features from different scales within the backbone.

Unlike YOLOv3, which predicts boxes in a single stage, Faster R-CNN relies on two stages. The first stage is called the Region Proposal Network (RPN), and it focus on detecting candidate bounding boxes, also known as Regions of Interest (RoIs), for classification in the next stage. The second stage is a Fast R-CNN [10] which classifies each candidate bounding box and adjusts the bounding boxes with a regression. The two stages are typically connected with RoIPool [10] or RoIAlign [11], which extract feature maps from each RoI proposed by the RPN, which can then be input into the Fast R-CNN.

To improve small object detection over the baseline, we focused primarily on adjusting the loss function, motivated by the improvements made by [18], [4], and [22]. We list the different modifications we made to the loss function.

3.2.1 Multi-task weights

Similar to YOLOv3 [20], Faster R-CNN [21] takes into account multiple loss functions, including a classification loss and a bounding box regression loss. As mentioned in [14], we experiment with adjusting weights of each part of the loss function to increase how much small objects contribute to the loss function. Since each small object often pairs much more rarely with anchors compared to large objects, this can mean that bounding box regression loss is contributed to more heavily by larger objects. On the other hand, classification loss is typically size-independent. Thus, one way of increasing contribution of small objects to the loss function may be to decrease the weight of bounding box regression loss and increase the weight of classification loss.

3.2.2 Focal Loss

Focal loss [18] is based on principle that we may be able to improve performance, especially in object detection tasks, by giving more weight to regions or samples that are harder to classify.

Given the probability of the ground-truth p_t, and hyperparameters α and γ, focal loss can be described as follows:

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

This can help significantly in object detection due to a major part of the task being to differentiate foreground and background, with background objects often representing a much larger portion of an image, and many being very easy to classify as background.

Experiments are also done using either softmax or sigmoid to determine how allowing multi-class prediction might affect results.

3.2.3 Reduced Focal Loss

Focal loss [18] has been shown to improve object detection accuracy on the COCO dataset, but a significant difference between COCO and the dataset we're working with, xView, is quality. As mentioned before, xView [15] has mislabelled data. If we use the vanilla focal loss implementation, we may run the risk of focusing training on these mislabelled boxes and reduce the performance of the detector.

As such, [22] constructs a variant called reduced focal loss. Given probability of ground-truth p_t, focal loss hyperparameters α and γ, and a threshold t_r, reduced focal loss can be formulated as:

$$\text{RFL}(p_t) = -\alpha \log(p_t) \times \begin{cases} 1 & p_t < t_r \\ \left(\frac{1-p_t}{1-t_r}\right)^\gamma & p_t \geq t_r \end{cases}$$

By weighting low probabilities equally, we can reduce the weight we place on incorrect labels and allow the model to keep focus on correct samples as well. [22] finds that using reduced focal loss can improve performance of Faster R-CNN on xView, whereas focal loss actually harms it.

3.2.4 Area Weights

Similar to YOLOv3, we develop loss functions that introduce area weights to encourage the model to fucs on small objects. To do this, we use the AW-p weights for a given area A, hyperparameter β, and area threshold t_a, as:

$$\text{AW}(A) = \begin{cases} 1 & A \leq t_a \\ \left(\frac{t_a}{A}\right)^{\beta} & A > t_a \end{cases}$$

We multiply the loss for each example by their respective weight. For example, this makes area-adjusted cross-entropy into:

$$\text{ACE}(p_t, A) = -\text{AW}(A)\log(p_t)$$

Note that when $\beta = 0$, $\text{AW}(A) = 1$, meaning cross-entropy is just area-adjusted cross entropy where $\beta = 0$. By choosing t_a to focus on those objects considered small in the dataset (e.g. 32^2 in COCO), we are able to improve small object detection while still taking into account and learning on larger images as well.

4. Data

Our satellite imagery dataset comes from the DIUx xView 2018 Detection Challenge [15]. The original xView dataset consists of 846 high resolution labeled training images, 281 validation images and an unknown number of test images, with each image covering approximately 1 km^2 in approximately 3000^2 pixels. The dataset contains over 1 million objects across 60 classes, somewhat comparable to the MS COCO dataset commonly used in object detection. While the original xView dataset had default splits for training, validation, and testing, labels for the validation and test set were never released to the general public and our requests to obtain the validation labels were denied by the xView organizers. To obtain our own train/val/test split, we performed an 80/10/10 split on the training dataset to produce 676 photos for training, 85 photos for validation, and 85 photos for testing. As such, our metrics are evaluated on a smaller dataset and thus may not be consistent with the performance of other published results.

4.1. Comparison to MS COCO

The data does differ from MS COCO in several major ways. Perhaps the most severe is that objects and bounding boxes can often be mislabelled, something regularly discussed in articles written about xView. Other differences not related to dataset quality exist as well. Due to the nature of the dataset, objects of a given class are often not seen at widely varying scales. For example, all objects in the "small car" category are around 100 pixels large, and oil tankers are on the order of 10^4 pixels. Additionally, distribution of objects in an image is often very clustered, with areas of next-to-no objects and other areas with many. When we chipped the dataset to chips with 500 pixels in width and height, we observe that 47% of chips do not have any bounding boxes. And for those chips with bounding boxes, the median number of objects per chip is 18, which is much higher than popular object detection datasets like MS COCO [19].

4.2. Data manipulation

We standardized our data preprocessing and augmentation process to use 700x700 chips, inspired by the 1st place winner of the xView Challenge [22]. Previous small object detection papers have benefited from data augmentation. For our preprocessing, we apply the following augmentation techniques during train-time only: (1) **Rotation**: Satellite image object detectors need to be particularly invariant to object rotation. As such, we augment the entire dataset by rotating each chip by 10, 90, 180, and 270. (2) **Flip**: Similar to the rotations, we randomly flip chips during training (25% chance horizontal flip, 25% chance vertical flip). (3) **Scale**: To address variations in scale, we randomly rescale every chip to between 500x500 and 900x900 during training. (4) **Color**: xView objects come from a variety of lighting and background color contexts. We add random hue, saturation, and exposure distortion to chips during training to address lighting and color variations.

4.3. Implementation Differences

Due to differences in the codebases used for YOLOv3 and Faster R-CNN ([24] and [9], respectively), we were unable to standardize the data each model saw while training completely. For example, [24] trains by randomly extracting chips from a given train image, while [9] does not have any chipping functionality, and thus the dataset was instead modified by chipping all images as needed, with 80 pixels of overlap between chips. Additionally, evaluation code in both codebases, though both based on PASCAL VOC's [8] method of evaluating performance, may be different. Similarly, methods used for data augmentation came from different libraries, and thus cannot be guaranteed to be the same. In the interest of getting more results in the limited amount of time, we decided it best to ignore these differences. As such, performance between YOLOv3 and Faster R-CNN cannot be directly compared without additional investigation.

5. Experiments

5.1. Evaluation

The xView Challenge evaluated its participants using interpolated mean average precision (mAP) metric, which measures overlap between ground truth and predicted bounding boxes. They defined the IoU threshold for true

Figure 1: **Small false positives.** Comparing results on the same stretch of road for baseline YOLOv3 (left) and YOLOv3 + AW-c (right). Red is false positives, blue is false negatives.

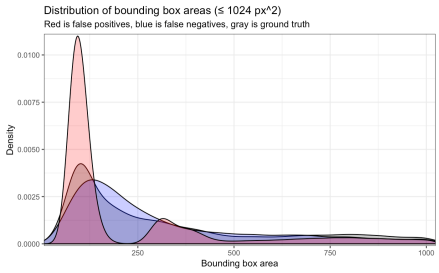


Figure 2: Bounding box area distributions for baseline YOLOv3

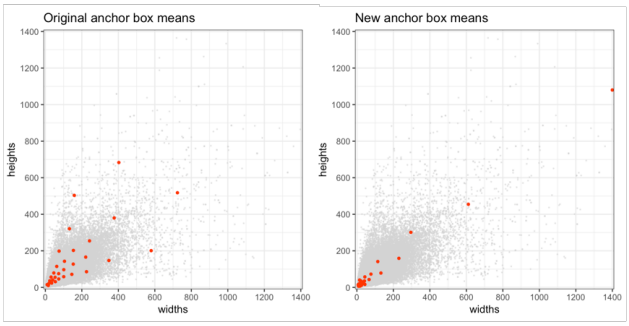


Figure 3: **Original vs new anchor box dimensions.** The red points are the anchor boxes; the gray points are all bounding boxes in the training data. Note the higher concentration of small anchor boxes in the plot on the right.

Model	AP	$AP_{C,S}$	$AP_{A,S}$
YOLOv3	0.0118	0.0065	0.00255
small anchors	0.0396	0.0283	0.0111
AW-d	0.0397	0.0134	0.0099
AW-p	0.0663	**0.0324**	0.0108
AW-c	**0.0689**	0.0298	**0.0112**
small anchors + AW-c	0.0514	0.0130	0.0080

Table 1: YOLOv3 Results

positives to be 0.5. They also included an $AP_{C,S}$ metric for the set of classes it deemed as small objects.

Because we are focusing on small object detection, we add an $AP_{A,S}$ metric which is calculated as the mAP over all test and validation bounding boxes that are considered "small" under MS COCO's definition: 32^2 pixels or fewer.

5.2. YOLOv3

5.2.1 Baseline

We preserved many of the training hyperparameters from [24], as they were able to produce reasonable baseline performance on the xView dataset: batch size of 8, and learning rate of 0.0001 with weight decay of 0.0005. As previously mentioned, [24] randomly samples 8 chips from each image per epoch; we did not modify this behavior. Although [24] ran their final model for 300 epochs over 3 days, we elected to run our models for 30 epochs (7 hours) on a T4 GPU in the interest of leaving more time for experimentation.

After analyzing the evaluation results of the baseline after 30 epochs, we noticed that the model suffered from widespread false positives: only two of the 60 classes had fewer detected false positives than ground truth bounding boxes. Unsurprisingly, most of these false positives fell under our definition of small objects ($<1024\text{px}^2$); in fact the distribution of false positive bounding box areas was heavily skewed to $< 250\text{px}^2$ (see Fig 2). See the left side of Fig 1 for an example of many small false positives.

5.2.2 Anchor boxes

To improve small object detection, we first changed our computation of the anchors. The baseline YOLOv3 computed 30 anchors across the entire dataset using k-means clustering. We instead computed the first 20 anchors only across objects with bounding box area less than 1024px^2 and the next 10 anchors across objects with bounding box area greater than 1024px^2. Refer to Fig 3 for a visualization of the newly generated anchors.

The results of YOLOv3 + small anchors are in Table 1; the smaller anchors did indeed appear to improve overall and small-object-specific mAP.

5.2.3 Loss term coefficients

We first naively tweaked the coefficients on YOLOv3's four term loss function (position, area, confidence, classification) to understand how each term affects overall and small object performance. The default coefficients were $c_{\text{pos}} = 2$, $c_{\text{area}} = 4$, $c_{\text{conf}} = 1.5$, and $c_{\text{cls}} = 1$.

As proposed by Kisantal et al [14], we first reduced the coefficients of the region proposal loss (c_{pos} and c_{area}). Since

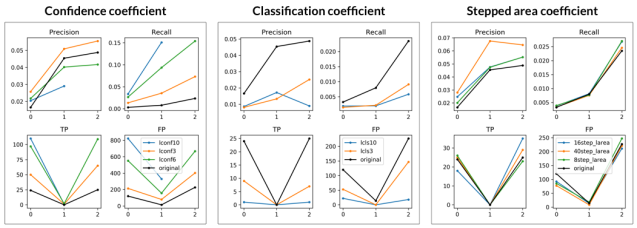


Figure 4: **YOLOv3 loss term coefficient experiments.** Experiments conducted on three epochs, plotting the validation recall, precision, number of true positives, and number of false positives. All experiments are compared against the original coefficients (in black).

each ground truth is paired only with one bounding box with the highest IoU, larger objects that span more grid cells can be matched with multiple anchor boxes, whereas smaller objects are often paired with just one anchor with a low IoU. This means small objects contribute less to region proposal loss whereas medium and larger objects contribute more.

Second, we tried increasing the confidence term coefficient in an attempt to penalize low confidence false positives of the baseline. Within the first few epochs, we observed that a greater coefficient leads to more true positives, but at the cost of more false positives, thus increasing recall and decreasing precision (see Fig 4, left). We reason that a higher weighting of the confidence term penalizes very confident incorrect predictions and very unconfident correct predictions, thus encouraging the network to produce many moderately confident predictions and achieving the opposite of our originally desired effect.

Third, we tried increasing the classification term coefficient in an attempt to reduce baseline false positives that were incorrectly classified. We observed that a greater coefficient leads to a more conservative network that outputs fewer positive identifications, both true and false (see Fig 4, mid).

Modifying loss term coefficients as constants appeared to affect the model's global reporting of identifications, making it challenging to specifically target the baseline's false negatives.

5.2.4 Adding area weights to loss functions

Since the distribution of the baseline's false negatives was highly uneven across bounding box areas, we next tried incorporating area-based loss term weighting instead of using naive constants.

First, we tried our naive discretized constant weighting (AW-d) of the classification term. Higher values of the weighting c more heavily emphasized objects under the $t_a = 1024$ threshold and were indeed able to slightly target the false positive rate, thus slightly increasing the precision

Figure 5: **YOLOv3 error mode.** An example of poor YOLOv3 + small anchors + ACE-c performance on dense small vehicles in a parking lot. Red is false positives, blue is false negatives, green is true positives.

while maintaining very similar recall (see Fig 4, right).

Next, we tried our partially continuous weighting (AW-p) with $c = .25$, which yielded the highest mAP from experimentation. The added continuous function for $A > t_a$ further down-weights large bounding boxes, thus improving overall performance. However, after observing that AW-p still produced significant false positives with area $<250\text{px}^2$, we removed the piecewise component for a fully continuous function (AW-c), which extended the scaled penalties to objects under t_a. As Table 1 shows, AW-c resulted in the highest overall mAP and small object mAP, while AW-p resulted in the highest small class mAP. Empirically, as shown in Fig 1, AW-c helped reduce false positives that dominated the baseline's output.

5.2.5 Combining anchor boxes and area weights

Interestingly, when we combined small anchor boxes and area weights, overall and small object performance both dropped quite precipitously. While we don't fully understand why, evaluation examples like Fig 5 illustrate the danger of small anchor boxes and increased false positives. Notice in the bottom of the image that each car has multiple (mostly red) bounding boxes around it, likely due to the increased number of anchor boxes proposing car-sized objects for a given detection grid area. Thus, successful use of small anchors requires stronger false positive rejection, for which the methods we explored were only moderately successful.

5.3. Faster R-CNN

5.3.1 Baseline

Due to all the different hyperparameters available to tune in object detection models, especially in Faster R-CNN due to the multiple stages, we began by adjusting various hyperparameters, inspired mainly by choices made by the challenge's top-performing model [22]. These choices didn't

RPN	RoI	AP	$AP_{C,S}$	$AP_{A,S}$
CE	CE	**0.2065**	0.0874	0.0716
FL	CE	0.2036	**0.0890**	0.0671
FL	FL(sig)	0.1285	0.0573	0.0359
RFL	CE	0.2064	0.0889	**0.0744**
RFL	RFL(sig)	0.1364	0.0656	0.0407
RFL	RFL(sft)	0.1533	0.0764	0.0485

Table 2: Focal Loss Results

seem to make a huge difference in scores, and eventually, we settled upon a baseline using ResNet-50 with an FPN as a backbone with anchors of size (16, 32, 64, 128, 256), an RPN batch size of 256, an RPN positive fraction of 0.5, an RoI batch size of 512, and an RoI positive fraction of 0.25.

Models were trained over 100,000 iterations on a 2-image batch size using SGD + momentum with a base learning rate of 0.0025, decreasing learning rate by an order of magnitude at 70,000 and 90,000 iterations. Training typically took around 12 hours on a Tesla T4. Detection per image was limited to 200 at test-time, though it doesn't seem like bumping that up helps much.

Experiments were done in stages, with the best model from one stage being used in later experiments. This helped reduce time and compute needed to get results.

5.3.2 Focal loss variants

We began by exploring how focal loss affected performance. We ran experiments with focal loss and reduced focal loss using the parameters given in [18], namely $\alpha = 0.25$ and $\gamma = 2$. Since [22] noted that focal loss did not help when added to the R-CNN, for both focal loss and reduced focal loss we ran one experiment with the adjusted loss function only on the RPN. We also tried adding focal loss and reduced focal loss to the R-CNN to corroborate [22]'s results, trying out both sigmoid for a more multi-class task as well as softmax for the single-class task at hand.

As can be seen in table 2, both focal loss and reduced focal loss only seem to help when added to the RPN layer, with softmax causing less of a drop than sigmoid. It's difficult to say why, but it may be due to the low α causing reduced weight on classification loss. Additionally, it seems that although focal loss and reduced focal loss do seem to help (reduced focal loss moreso than focal loss), it isn't by much, only increasing $mAP_{A,S}$ by around .3%. Since sigmoid seemed to perform worse in general, we decided to only use softmax for the remainder of the experiments. Also, since reduced focal loss seemed to perform better than focal loss, we decided to stick with reduced focal loss moving forward as well.

RPN	RoI	β	AP	$AP_{C,S}$	$AP_{A,S}$
CE	CE	0.5	**0.2065**	0.0874	0.0716
ACE	CE	0.5	0.2039	0.0941	0.0718
ACE	ACE	0.5	0.1856	**0.0970**	**0.0786**
ARFL	ACE	0.5	0.1827	0.0817	0.0693
ARFL	ARFL	0.5	0.1255	0.0683	0.0453

Table 3: Area Weight Results

RPN	RoI	β	AP	$AP_{C,S}$	$AP_{A,S}$
ACE	ACE	0.0	0.2065	0.0874	0.0716
ACE	ACE	0.125	**0.2091**	0.0946	0.0806
ACE	ACE	0.25	0.1993	0.0931	**0.0849**
ACE	ACE	0.5	0.1856	**0.0970**	0.0786
ACE	ACE	1.0	0.1574	0.0879	0.0708

Table 4: β-selection Results

5.3.3 Adding area weights to loss functions

Seeing that focal loss didn't seem to help a significant amount, we tried adding area-based weights to see if that would change anything. As can be seen in table 3, area-based weights did seem to help with small object detection specifically, though there were decreases in overall AP due to it. Area-adjusted cross-entropy seems to perform the best, with a 1% improvement on $AP_{C,S}$ when added to both RPN and R-CNN loss. It also looks like reduced focal loss just doesn't help compared to vanilla cross-entropy with area weights. In the remaining experiments, we thus focused on area-based cross-entropy applied to both RPN and R-CNN.

5.3.4 β selection

With area-based weights seeming to help, we wanted to try figure out the best choice of β. Examining table 4, we can see β between 0.125 and 0.5 have pretty similar results, with differing results across $AP_{C,S}$ and $AP_{A,S}$. We cared more about $AP_{A,S}$ since the metric focuses more on detection of all small objects rather than just objects in a given class. Thus, we chose $\beta = 0.25$ for our remaining experiment.

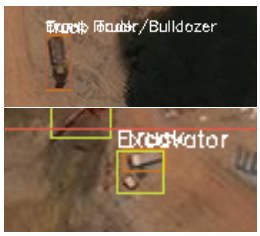
5.3.5 Multi-task weights

Due to lack of time, we were only able to train one last model. For this experiment, we doubled classification loss and halved bounding box regression loss to hypothetically increase the influence small objects had on the loss function. It seems to have improved $AP_{C,S}$ and $AP_{A,S}$ compared to not adding it, indicating that there may have been some effect. However this change is a bit small. Still, by adjusting

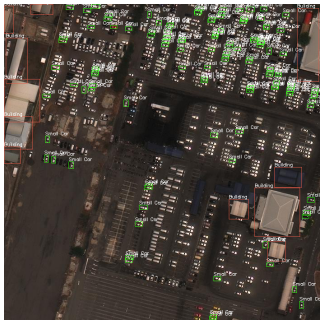
model	AP	$AP_{C,S}$	$AP_{A,S}$
baseline	0.2065	0.0874	0.0716
+ RPN ACE	0.2039	0.0941	0.0718
+ RoI ACE	0.1856	0.0970	0.0786
+ $\beta = 0.25$	0.1993	0.0931	0.0849
+ multi-task weights	0.2023	0.0966	0.0869

Table 5: Ablation Results

(a) Ground-truth (top) vs. predictions (bottom). Predictions miss a few dump trucks and misclassify all of them as trucks.



(b) Instances where predictions for multiple classes overlap the same object.



(c) Too few detections being made

Figure 6: **A few common error modes for final model.** Errors modes chosen were those primarily occurring to small objects.

weights of training examples losses and weights of losses in the multitask objective, we were able to improve small object detection by 0.92% based on $AP_{C,S}$, and 1.5% based on $AP_{A,S}$. Table 5 provides an ablation from baseline to our final model.

5.3.6 Error Analysis

A few common ways the model fails are provided in figure 6. The biggest problem, which we tried to address in this report but were unable to solve completely, seems to be classification. In figure 6a, we can see that the model is able to detect many of the objects in the image. However, it is unable to identify them as dump trucks rather than just plain trucks. This may be because trucks and dump trucks are visually really similar, and distinguishing them when they take up so little of the image may be difficult. And similarly, in figure 6b, we see that the model can, again, detect the objects in the image, but is unable to correctly classify the objects as multiple classes rather than just 1. Due to the similar nature of some of the object classes, it may have been a good idea to make the problem a multi-class prediction problem rather than have a single ground-truth class. However, since the xView challenge [15] does evaluate on single-class ground-truth, that is what we stuck to.

The model also seems to struggle with detecting objects in chips with a lot of objects in them. In figure 6c, we can see that there are likely hundreds of cars in the chip, but only a relatively small portion are predicted to be cars. This may explain the unexpected low performance of the model on detecting small cars, which make up a large portion of the dataset, compared to other classes with fewer instances in the dataset. One hypothesis for why so few of the cars in the chip are detected is due to the manually set detections per image threshold. Since the code was originally used for COCO, which has significantly fewer instances per image, this detection threshold may be severely holding back detection performance on xView. However, bumping detections per image at test time from 200 to 400 does not seem to change performance on small cars, so this might not actually be the issue.

6. Conclusion

We have been able to improve performance in small object detection on the xView dataset by adjusting hyperparameters and adding area-based weights to loss functions in both YOLOv3 and Faster R-CNN. These changes can improve performance on small objects by over a percent compared to our baseline, though it can come at a cost to performance in other categories. However, it is hard to know how these results may change given more time or more compute resources. With Faster R-CNN, for example, results from [22] do not entirely match up with results found, and part of this may be that [22] may have been able to train models to completion and tune learning rate schedules to each model while we set a single schedule and did not adjust with the model. As such, reduced focal loss may do better given more time to train, but it's benefits may not be as evident given the training time. Future research might thus simply be adding more compute resources, or it may be exploring larger architectural changes, like incorporating GANs as in [16]. It might also mean identifying the salient characteristics of area-based weights by trying out different ways of calculating these weights, similar to how [18] explores alternative formulations of focal loss and demonstrates relatively equal performance.

7. Contributions and Acknowledgements

Edward Lee worked on the Faster R-CNN improvements, Derek Huang worked on small anchor boxes and loss functions for YOLOv3, and Claire Huang worked on loss functions and visualizations for YOLOv3.

We received assistance from Burak Uzkent, who introduced us to the xView Challenge dataset and gave us an overview of single-shot and two-stage detector architectures.

Credits to Facebook Research's Faster R-CNN [9] implementation in Pytorch, as well as Ultralytics' YOLOv3[24] implementation in Pytorch for their entry into the xView Challenge.

References

[1] S. Ando and C.-Y. Huang. Deep over-sampling framework for classifying imbalanced data, 2017.

[2] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. 2017.

[3] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. 2011.

[4] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie. Class-balanced loss based on effective number of samples, 2019.

[5] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling. pages 1–8, 2003.

[6] A. V. Etten. Satellite imagery multiscale rapid detection with windowed networks, 2018.

[7] A. V. Etten. You only look twice: Rapid multi-scale object detection in satellite imagery, 2018.

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[9] Facebook Research. Mask-RCNN Baseline. https://github.com/facebookresearch/maskrcnn-benchmark, 2019.

[10] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015.

[11] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

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

[13] X. Jin and C. Davis. Vehicle detection from high-resolution satellite imagery using morphological shared-weight neural networks. *Image and Vision Computing*, 25:1422–1431, 09 2007.

[14] M. Kisantal, Z. Wojna, J. Murawski, J. Naruniec, and K. Cho. Augmentation for small object detection. *CoRR*, abs/1902.07296, 2019.

[15] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord. xview: Objects in context in overhead imagery. *CoRR*, abs/1802.07856, 2018.

[16] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual generative adversarial networks for small object detection, 2017.

[17] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.

[18] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. *CoRR*, abs/1708.02002, 2017.

[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. pages 740–755, 2014.

[20] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[21] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[22] N. Sergievskiy and A. Ponamarev. Reduced focal loss: 1st place solution to xview object detection in satellite imagery. *CoRR*, abs/1903.01347, 2019.

[23] J. Shermeyer and A. V. Etten. The effects of super-resolution on object detection performance in satellite imagery, 2018.

[24] Ultralytics. xView YOLOv3. https://github.com/ultralytics/xview-yolov3, 2019.