Detecting Plains and Grevy's Zebras in the Real World

Jason Parham
Rensselaer Polytechnic Institute
parhaj@rpi.edu

Charles Stewart
Rensselaer Polytechnic Institute
stewart@cs.rpi.edu

Abstract

Photographic censusing can be partly automated by leveraging the power of computer vision detection algorithms. Detecting zebras in the real world can be challenging due to varying viewpoints of the animal, natural and artificial occlusions, and overlapping animals. To address these challenges, we evaluate three detection algorithms: Hough Forests by [8], the YOLO network by [20], and Faster R-CNN [21]. We train the detectors on a soon-to-be-released dataset of 2,500 images containing 3,541 bounding boxes of plains zebras (Equus quagga) and 2,672 bounding boxes of Grevy's zebras (Equus grevyi). The detection errors are analyzed by species, viewpoint, and density (the number of bounding boxes per image). The best detector in our evaluation reports a detection mAP of 55.6% for plains and 56.6% for Grevy's.

1. Introduction

A population census is an essential first measure for judging the general health of a species. A population count is integral to ecology, where the endangered status of a species is dictated by the number of remaining animals. However, a census can be difficult to obtain manually due to time and resource limitations. A photographic census that leverages computer vision techniques could help with this daunting, yet important, task. To focus our discussion, we will review the process of performing a photographic census on images of plains zebras (*Equus quagga*) and Grevy's zebras (*Equus grevyi*), as seen in Figures 1 and 2.

Once images have been collected, the next step in producing a photographic census is generating reliable bounding boxes and species labels for each relevant animal. This detection process is required to filter irrelevant images (e.g. pictures with no zebras), to focus further processing on only relevant image portions (i.e. coarse background elimination), and to provide the rough number of animal sightings. The removal of surrounding background has the benefit of eliminating distracting information that could otherwise confuse an identification algorithm. Therefore, the detections and their quality play an important role in the *overall* quality of a photographic census by filtering input images and providing pre-processing for identification. This is the problem we focus on in this paper.

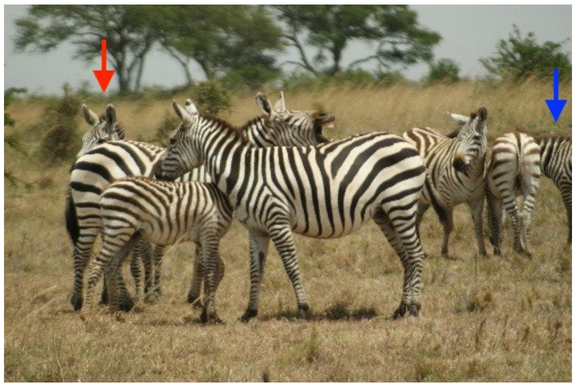

Figure 1: Challenges of the detection problem for zebras include varying viewpoints, natural and artificial occlusions, and overlapping animals. The image above shows 7 individual zebras with 5 differing viewpoints and 5 occlusions of differing severity. The highlighted animals are almost completely occluded, but still clearly discernible.

The detection problem as applied to photographs of zebras has several real-world challenges: varying viewpoints, natural and artificial occlusions, overlapping animals, non-rigid body structures, and large changes in visual appearance (e.g. genetic variations, dust, scarring). While these are not unique to this particular application, they are quite pronounced. Looking at Figure 1, the head of the highlighted zebra (red) is clearly visible, but the rest of the animal is almost completely occluded save a few of its legs. The cut-off animal on the far right (blue) only has a small section of neck visible whereas its neighbor to the left is facing completely away from the camera. Needless to say, zebras in the real world can be frustratingly uncooperative with respect to our task of trying to detect them. These enhanced challenges elevate the problem to a degree of difficulty not often seen in standard computer vision benchmarking competitions like PASCAL VOC [5] and ILSVRC [23].

To address these challenges, we apply two different detection approaches: Hough Forests and Convolutional Neural Networks (CNNs). Random forest algorithms are easy to train due to having fewer hyper-parameters and built-in regularization. The Hough Forests variant, in particular, is somewhat resilient to partial and occluded objects due to its voting scheme [7, 30]. CNNs have shown to perform well on classification [14, 24, 26, 27], localization [4, 9, 24, 28], and detection [11, 17, 20, 21] problems due to their ability to learn complex representations of supervised training data. For these reasons, we evaluate a random forest-based detector and two CNN-based detectors on images of plains and Grevy's zebras.

This paper has three main contributions: 1) to compare three detection algorithms on real-world images of zebras, 2) to enumerate unsolved detection challenges in this context, and 3) to provide implementation details for the detector currently used by IBEIS – the Image Based Ecological Information System[1] (see also [19]). We will soon be releasing the first public version of the dataset used during this evaluation. We also publish our Python bindings for the Hough Forests implementation by Gall et al. [8][2] and the YOLO implementation by Redmon et al. [20][3]. The remainder of this paper will begin by reviewing related work in Section 2. A brief introduction to the different detector approaches will follow in Section 3. The experimental evaluations are presented in Section 4 and conclusions are drawn in Section 5. A comparative table of detections from each of the three algorithms can be seen in Figure 6.

2. Related Work

The use of Hough Forests (i.e. Hough-transform random forests) for object detection was demonstrated by Gall et al. in [7]. The authors showed that random forests have advantageous training properties and extend naturally to patch-based image textures. They argue that the leaf nodes of a random forest tree can be considered a "discriminative codebook", which they use to generate classification probabilities. Furthermore, by training to optimize for both classification and regression within the same tree, they are able to learn a spatial relationship of where a classified image patch is likely located in relation to an object center. Their idea is extended by Barinova et al. [1] to address occluding objects. A more comprehensive analysis of Hough Forests is presented in [8]. Others [3, 6, 31] have applied random forests to face, pose, and action recognition. We evaluate a customized version of the implementation introduced in [8].

The literature for CNN-based detectors is vast and rapidly evolving. For this paper, we focus on approaches that integrate object localization directly into the detection pipeline via some form of neural network. A localization CNN is used by He et al. [11], the recent winners of the ILSVRC 2015 object detection challenge, who used their novel *deep residual learning* to train an extremely deep CNN detector. Their approach was based on their previous Faster R-CNN detector [21], which used a novel Region Proposal Network (RPN) for object localization in conjunction with their existing classification network from [9]. The Faster R-CNN detector was also the winner of several tasks in ILSVRC 2015. We analyze the detection performance of off-the-shelf Faster R-CNN.

The usage of a localization network, however, is not unique to [11] nor their RPN precursor. Other CNN-powered salient object detectors have been introduced, including Google's DeepMultiBox [4], the "YOLO" (You Only Look Once) network [20], and others [12, 13, 16, 32, 33]. The key insight to all of these approaches is that a saliency localizer can perform well while being class agnostic. This suggests that the saliency of an object can be generalized within the neural network by training on *classless* bounding boxes. To generate final detections, the intermediate saliency bounding boxes are classified with an additional, dedicated classification network. This saliency generalization, as shown in [20], has benefits of extending to unknown classes and abstract representations (i.e. paintings) of known classes.

The insight behind YOLO – and what sets it apart from the networks mentioned above – is that the bounding box proposal network can be combined with the classification network into a single network architecture. This integration, however, does come with downsides: a complex loss function, additional hyper-parameters, an unpredictable error gradient, and the loss of fine-grained detection performance. However, the ability to train YOLO as a unified pipeline makes it advantageous for real-world applications due to its efficiency and lack of additional machinery. Due to their network consolidation, the authors of the YOLO network were able to report real-time performance using GPUs. For this evaluation, we also report the performance of YOLO as a comparison against Faster R-CNN.

3. Approach

An overview of each algorithm we evaluate, along with significant implementations details, is given below.

3.1. Hough Forests

Hough Forests are an ensemble of random binary trees. Each tree attempts to optimize the performance of classification and regression by performing a series of binary pixel tests. The authors of [7, 15] demonstrate that training a ran-

[1] ibeis.org (accessed March 6, 2016)
[2] github.com/bluemellophone/pyrf
[3] github.com/bluemellophone/pydarknet

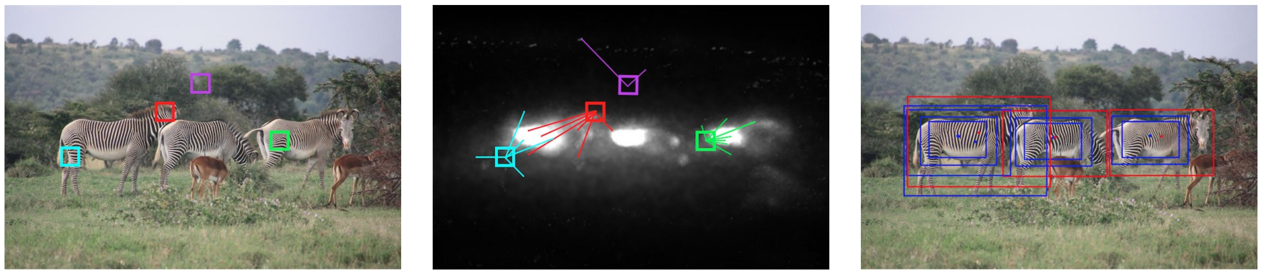
(a) (b) (c)

Figure 2: Hough Forests test patches are extracted densely over a test image (a) and are classified using an ensemble of random binary trees into a collection of leaves. Each leaf has a set of positive and negative patches given to it during training, which are used to make weighted probabilistic Hough votes into an aggregate Hough image (b). The high-probability object center peaks (white) are used to generate bounding box proposals (blue). The proposals are filtered with non-maximum suppression to create the final detections (red). Compare the votes of the red and purple test patches in (a, b); the purple votes are sporadic and do not accumulate whereas the red votes contribute to an object center. The blurring on peaks is due to voting confusion.

dom forest tree in this combined fashion benefits both generalization and accuracy. The first pre-processing step of training extracts a collection of small image patches (32×32 pixels) from the ground-truth in order to compose a large set (60,000) of positive and negative training patches. Importantly, each positive patch records its relative offset to the center of its corresponding object in the ground-truth.

During training, each tree is given the same set of patches. Our implementation uses an ensemble of 10 trees. Each tree generates tests that split the set at each non-leaf node, which performs a random binary pixel test on each image patch P. The test, as formulated in [8], is

$$test_{\alpha,p,q,r,s,\tau}(P) = \begin{cases} 0, & \text{if } P^{\alpha}(p,q) < P^{\alpha}(r,s) + \tau \\ 1, & \text{otherwise} \end{cases} \tag{1}$$

where α is the channel of the image patch, (p, q) is a random location in the patch, (r, s) is a different random location in the patch, and τ is a threshold offset. Every node is allowed to pick its own randomized test, randomly seeded to promote diversity.

Every internal node samples randomly over the parameters α, p, q, r, s, τ to find the best binary test that minimizes either the positive-negative classification error or the positive-patch regression error. The node will minimize (with $p = 0.5$) either the binary cross-entropy classification error or the regression offset sum-squared difference error (negative patches are ignored since, by definition, they have no center offset to an object center). The binary tree builds itself recursively until either a node is too deep in the tree or has too few patches. During our training, each tree is trained to a depth of 16 layers or creates a leaf when a node has fewer than 20 patches.

During test time, each leaf node holds a collection of positive and negative patches. A leaf's positive class probability is computed as the percentage of positive patches out of all the patches it received during training. Each positive patch in a leaf makes a weighted probabilistic vote to where it thinks the center of the object is in the test image. As shown in Figure 2, these Hough-transform votes (a) are computed densely across the entire test image and aggregated over multiple scales to generate a combined Hough image (b). The bright white spots in the Hough image indicates the probable locations of object centers. Thresholded peaks are selected as candidate center proposals, object bounding boxes are derived from the locations of patches that voted for the particular peaks (c, blue), and non-maximum suppression is applied to produce the final detection regions (c, red).

The Hough Forests detector has some distinct advantages, chiefly that 1) it is easy to parallelize across multiple CPUs for efficient parallel processing and 2) the voting scheme will aggregate probabilities originating from *any* location on an animal. For example, if only the face and neck of a zebra are visible in an image, the face and neck tree leaves will still make probabilistic votes for where it thinks the center of a zebra should be. This voting scheme makes Hough Forests resilient to occlusions and makes it an attractive solution to the challenges present in our dataset. However, the image patches are too small to learn precise localization information. This results in a distinct blur of voting confusion surrounding an object's center in the Hough image. Moreover, our formulation of Hough Forests is trained as a *binary* classifier (one-vs-all) and therefore cannot natively represent multiple classes within the same tree. This poses a problem with representing multiple viewpoints of the same species, as viewpoints have conflicting spatial representations for an object's "center". This conflict causes confusion during training and detection, which hurts accuracy.

The evaluated version of Hough Forests improves on the efficiency and accuracy reported in [8] by adding OpenMP multi-CPU parallelization, adding new image channels, drastically increasing the number of binary tests performed at each node during training, and making more intelligent bounding box regression decisions with the coordinates of voting patches. The details explained in this section are meant to augment the summary in our previous work [19], which used Hough Forests to detect plains zebras and Masai giraffes from photographs taken at the Nairobi National Park in Nairobi, Kenya.

3.2. Faster R-CNN

The Faster R-CNN network by Ren et al. [21] is the third iteration of the R-CNN approach introduced by Girshick et al. [9, 10]. Each iteration of the detector sports a simplified pipeline and speed improvements. The latter has transformed R-CNN from a research tool into a viable, near-real-time detector. For our evaluation, we use the off-the-shelf Python bindings of [21].

The motivation behind Faster R-CNN is that the Selective Search [29] candidate proposal phase used by its precursors is a major speed bottleneck. To address this speed problem, the authors reimplement the bounding box candidate proposal as a neural network and brand it as a Region Proposal Network (RPN). The key insight to training Faster R-CNN is that the RPN is a "separate" network from the classification network inherited from [9], but to reduce processing the two networks share most of their convolutional filters. During training, the RPN and classifier are given the same fixed proposals and the two networks alternate to reach convergence. Both networks apply their updates in turn to the shared convolutional filters.

During test time, these shared convolutional features are only computed once. The RPN adds additional convolutional layers on top of the shared filters to generate localization predictions. The benefit of this architecture is that the training is *mostly* unified, which dramatically increases speed performance during both training and testing. Furthermore, replacing Selective Search with the RPN also improves accuracy. However, the branching top of the network involves additional complexity during training and the network still does not quite achieve real-time performance. Our Faster R-CNN network runs at about 6 fps on high-end GPUs. That being said, Faster R-CNN still boasts state-of-the-art performance for detection [23]. The implementation of Faster R-CNN seen in this evaluation is unmodified other than training for 30,000 iterations (with the newer and faster "end-to-end" scheme) on different classes. The training diverged several times before a stable model was produced because the RPN did not generate valid bounding boxes.

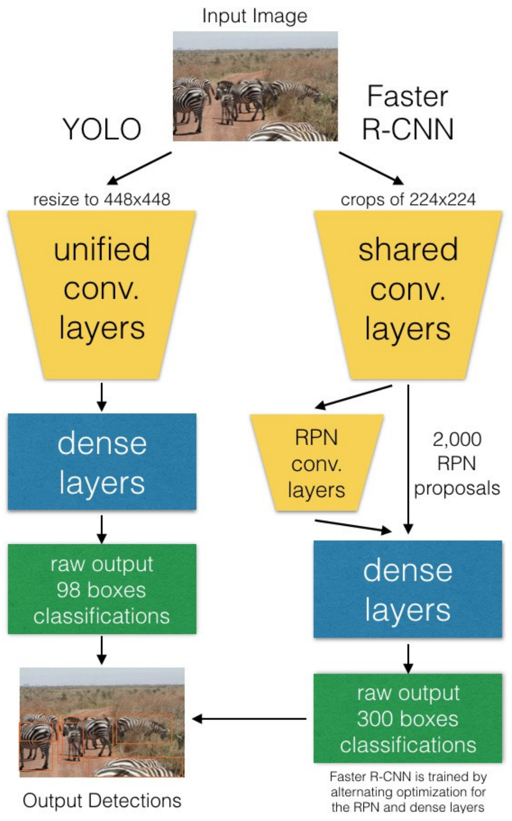

Figure 3: The YOLO network is a unified architecture that is trained top-to-bottom to minimize bounding box regression and classification error. In contrast, Faster R-CNN has a separate Region Proposal Network (RPN) that proposes salient object bounding box proposals, which are then classified to produce class probabilities. Faster R-CNN is trained by alternating the training between the RPN and the classification "networks" till convergence, each applying their gradient to the shared convolutional layers.

3.3. YOLO

In contrast to Faster R-CNN, the YOLO network, introduced by Redmon et al. [20], implements a truly unified network architecture. The network produces multi-class bounding box candidates *directly* from a single forward inference on an image. See Figure 3 for a high-level comparison between Faster R-CNN and YOLO. The architecture of YOLO is somewhat unusual as it takes a very large input image (448×448 pixels) and produces 98 detection regions from a grid of 7×7 classification cells.

The YOLO network is trained to optimize a complex, multi-part loss function. The mathematical definition of the loss function is presented in [20]. In summary, it has

5 components: a regression-weighted sum-squared difference loss (SSD) for each cell's bounding box center x and y pixel, a regression-weighted SSD for the square-root of each bounding box width and height, a conditional SSD for the saliency probability of whether an object exists in a bounding box, a classification-weighted conditional SSD for the saliency probability of whether an object does *not* exist in a bounding box, and a conditional SSD for the class probabilities of each cell. To combat model instability, the authors use the regression and classification weights in the loss function to appropriately balance the error gradient and utilize a learning rate schedule that starts intentionally low, increases, then decreases again around iteration 600. While these heuristics help, the network diverged several times during our own training before a stable random initialization was chosen and the model converged. We let our network train for 24,000 iterations.

The training of YOLO is done entirely as a unified network. This is in contrast to [21] and [4], which require additional machinery to train, are not completely unified, or do not directly produce classified bounding boxes. The benefit of this integration is most notably speed. The resized training images are given to the network in batches of 64 and the error gradient for each of the 98 network detections is computed directly using the ground-truth bounding boxes, which are mapped onto the range $[0, 1]$. During test time, the entire image is given to the network, which outputs a vector encoding of the 98 bounding box coordinates, a saliency probability for each bounding box, and class probabilities for each of the 49 (7×7) classification cells. Each 64×64-pixel classification cell contributes two bounding box proposals. The bounding box saliency probabilities are combined with the corresponding cell's classification probabilities to create the final class probabilities assigned to each bounding box. The class with the highest probability becomes the bounding box label and detections are generated by thresholding the low scoring probabilities.

3.4. Tradeoffs

The YOLO network has distinct advantages over Hough Forests: 1) it has significantly more parameters to model the training data, 2) has a larger approximate receptive field for better regression performance, and 3) is inherently multi-class. In comparison to Faster R-CNN, YOLO achieves real-time performance and, as mentioned previously, simplifies the entire detection pipeline down to a single forward inference. However, YOLO is more difficult to train with its poorly-behaving error gradient and has several network-specific hyper-parameters. Like Faster R-CNN, the YOLO network (realistically) requires a GPU to train, and both take significantly longer to train over Hough Forests. Both CNN networks were trained for about 24 hours using two Titan X GPUs whereas the ensemble of 10 Hough Forests trees was trained in just under 3 hours on a quad-core CPU. That being said, the testing speeds of both CNNs are at least two orders of magnitude faster compared to our Hough Forests implementation, which runs at roughly 15 seconds per image across 9 scales.

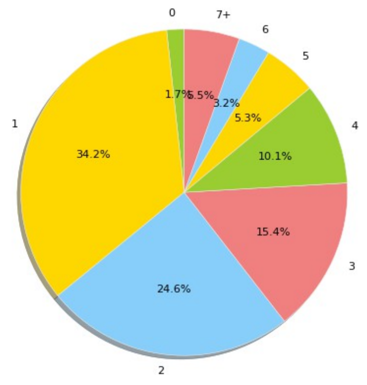

Figure 4: The distribution of densities (bounding boxes per image) in the dataset. A density of 0 indicates that the image was taken containing no animals. The maximum density of any image in the dataset is 23.

On top of training speed advantages, our Hough Forests implementation does not require nearly as much training data. Furthermore, both CNNs initialize their convolutional filters with pre-trained weights [18] before fine-tuning on our dataset. Between the CNN-powered detectors, the YOLO network was able to utilize empty images (images with no ground-truth bounding boxes of any species) during training whereas Faster R-CNN was not. This allowed YOLO to see slightly more images during training, which represents 1.7% of the dataset (Figure 4, density 0).

4. Experiments

Our experiments are designed to illustrate the overall performance of the algorithms as applied to our dataset and to provide a breakdown of errors into three categories: species, viewpoint, and bounding box *density*. The bounding box density represents the number of boxes within an image, which is a surrogate for the amount of overlap (and therefore occlusion) between animals. A more comprehensive analysis of detection errors due to occlusion will be presented in future work.

4.1. Dataset

Our dataset was constructed from 2,500 images taken by ecologists, field technicians, computer vision researchers, and volunteer citizen scientists [2, 25] working in Kenya. Images were taken primarily of two sub-species of zebras: plains zebras (`zebra_plains`) and Grevy's zebras (`zebra_grevys`). Bounding boxes and species labels

Viewpoint	Plains	Grevy's	Unspec.	Total
`left`	**1,965**	565	120	2,650
`front-left`	226	116	29	371
`front`	83	69	32	184
`front-right`	104	137	17	258
`right`	424	**1,029**	147	1,600
`back-right`	168	326	36	530
`back`	190	244	36	470
`back-left`	381	186	25	592
Total	**3,541**	**2,672**	**442**	**6,655**

Table 1: The distribution of viewpoints in the dataset. The unbalanced distribution of viewpoints is in part due to the behavioral characteristic of the photographed species and in part due to preferences of field scientists for mark-recapture studies.

were then annotated by hand. In addition to zebras, bounding boxes were generated for other animals present in the images and assigned to the `unspecified` species tag. Finally, viewpoint information was annotated for each bounding box by assigning it to one of 8 views of the animal's body: `left`, `front-left`, `front`, `front-right`, `right`, `back-right`, `back`, and `back-left`. The entire dataset has 6,655 ground-truthed bounding boxes with 3,541 plains, 2,672 Grevy's and 442 unspecified labels. The breakdown of viewpoints by species is shown in Table 1. A challenge to photographing real-world zebras is that capturing a balanced number of viewpoints can be difficult, with `front` being the least photographed in our dataset. The strong bias in plain zebras toward left side viewpoints and Grevy's zebras toward right side viewpoints is for historical reasons in the way animals were identified for mark-recapture studies [19, 22].

The dataset was exported from our research software IBEIS into the PASCAL VOC format. All images were resized to a maximum linear dimension of 500 pixels while maintaining aspect ratio. The data were split into subsets including 60% training, 20% validation, and 20% testing. For all training in this evaluation, the training and validation sets were combined and the trained models were evaluated against only the test set. The splittings of the sets, while random, were balanced to respect both the distribution of species *and* the distribution of densities. For example, approximately 80% of the images containing plains zebras were used for training, but also approximately 80% of images with only 1 ground-truth bounding box, and 80% of images with 2 bounding boxes, and so on. A complete breakdown of the bounding box densities can be seen in Figure 4. For the densities that had fewer than 5 images, an even split was used (when possible). Viewpoint was not considered for this balancing procedure because many of the images contain multiple photographed animals photographed from differing viewpoints. A total of 501 images (with 1,343 ground-truth detections) comprise the test set.

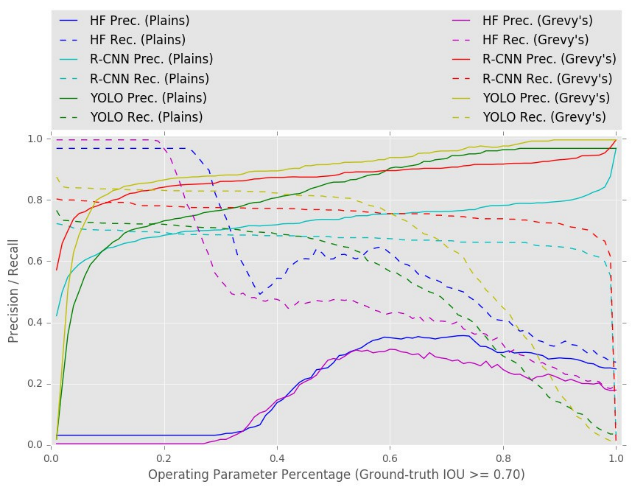

Figure 5: The localization-only precision and recall curves plotted against a chosen operating parameter. The solid lines represent precision and dashed represent recall. For the CNN-powered detectors this operating parameter is the bounding box confidence probabilities and for Hough Forests it represents the (unbounded) weighting on the Hough votes. As we can see, both CNN-powered detectors have much better precision than Hough Forests. The drop in Hough Forests precision above roughly 0.6 is due to the bounding box regression scheme.

4.2. Results

The three different detectors were evaluated by calculating the IOU (intersection over union) percentage between the detections and the ground-truth. A detection was considered correct if 1) the bounding box IOU $\geq$ 0.7 and 2) the species classification was correct. A classification error means the IOU threshold was met, but the classification was incorrect. Otherwise, the detection was registered as a localization error. Looking at Table 2, we can see that Hough Forests makes by far the most classification and location errors. The YOLO network achieves the highest number of correct detections, but makes far more classification errors compared to Faster R-CNN. This is due to the fact that Faster R-CNN makes more localization errors, but almost never makes an incorrect classification. Hough Forests makes the fewest classification errors, but this can be deceiving since it has almost double the number of localization errors as YOLO.

The precision and recall curves for each algorithm are graphed together in Figure 5. Precision is graphed using solid lines and recall with dashed. The x-axis represents the varying of a chosen operating parameter for each algorithm.

Algorithm	Correct	Class	Location
Hough Forests	540	2	801
Faster R-CNN	695	6	642
YOLO	752	175	416

Table 2: The correct and failed detections of each algorithm, combined for both species. The YOLO network gets the most detections correct, but has significantly more classification errors than Faster R-CNN. Faster R-CNN, while it makes more localization errors almost never guesses the incorrect species. There are a total of 1,343 test ground-truth detections (714 of plains, 536 of Grevy's, and 93 unspecified).

For the CNNs, this operating parameter is the sigmoid detection probability. For Hough Forests, this parameter is an unbounded weight on the Hough votes, which was given an artificial maximum for testing. The x-axis is the percentage along which these parameters were varied.

The precisions of the CNNs (red, yellow, green, cyan) dwarf the precisions of the Hough Forests detector (blue, purple). The YOLO detector is able to perform slightly better than Faster R-CNN, but this should not be a surprise considering YOLO's large input size. Faster R-CNN does perform multiple crops of the image during test time, but it does seem a little advantageous that YOLO gets a higher-resolution input image (albeit tested once). The precision curve for our Hough Forests implementation – due to its Hough voting scheme – has non-intuitive, non-monotonic behavior. As the voting weight is increased, there exists a point (roughly 0.6) above which the detections become *worse*. This is because the algorithm begins to over-estimate the bounding box coordinates and starts to lose detections. The artificial maximum operating point constrained on Hough Forests was chosen to illustrate this trend. Eventually, its recall *and* precision both go to, or almost to, zero.

4.3. Errors

Since it is quantitatively our best detector, the breakdowns of errors by species, viewpoint, and density are calculated from only the YOLO detections. For each test image, we use the detections that correspond to the highest precision. In this way, the best possible bounding boxes *for each image* are used; we analyze the error of the detections that failed either localization or classification. We make these decisions because it allows for a transparent view of the algorithm's failure cases.

4.3.1 Species Error

Species errors mostly occur when `unspecified` species are labeled as one of the two zebra species. We report (in decreasing order) the percentages of error within the instances found in the test set: `unspecified` - 83/93 (89.3%), `zebra_grevys` - 227/536 (42.4%), `zebra_plains` - 281/714 (39.4%). Note that these percentages are not meant to sum to 100.0%, as they are the individual rates of error within each subset.

4.3.2 Viewpoint Error

Front and back viewpoints cause the most errors. This is most likely due to having fewer overall training examples and being less visually distinct. The full breakdown of errors by viewpoint: `front` - 26/37 (70.3%), `back` - 65/97 (67.0%), `front-right` - 29/52 (55.8%), `back-right` - 51/100 (51.0%), `right` - 154/319 (48.3%), `back-left` - 52/120 (43.3%), `front-left` - 29/76 (38.2%), `left` - 185/542 (34.1%).

4.3.3 Density Error

Unsurprisingly, the presence of more animals in an image leads to more errors, though not uniformly. This suggests that occlusion is a major cause of errors. The error rates for the different numbers of ground truth detections per image are 7+ - 20/28 (71.4%), 6 - 7/16 (43.8%), 5 - 9/27 (33.3%), 4 - 10/50 (20.0%), 3 - 18/77 (23.4%), 2 - 6/123 (4.9%), 1 - 44/171 (25.7%), 0 - 0/9 (0.0%). Note that the descending order of the failure rates do not align with the descending densities (densities of 1, 2, and 3 are outliers). This, again, is most likely due to having unbalanced training data – which these outliers comprise over 75% – and the detectors being unable to handle occluded animals.

5. Conclusion

We have shown that the YOLO detector outperforms both Faster R-CNN and Hough Forests on our dataset. The YOLO detector achieves a detection mAP of 55.6% for plains and 56.6% for Grevy's. The most significant source of error was density (i.e. occlusions). The detection of occluded and overlapping animals seems to be a challenging case for modern detectors and is a prime subject for future research. Finally, the density metric is not perfect for measuring occlusion. Our ongoing evaluations will soon provide a more comprehensive analysis of the errors caused by different forms and severity of occlusion.

References

[1] O. Barinova, V. Lempitsky, and P. Kholi. On detection of multiple object instances using hough transforms. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 34(9):1773–1784, Sep. 2012. 2

[2] J. P. Cohn. Citizen science: Can volunteers do real research? *BioScience*, 58(3):192–197, Mar. 2008. 5

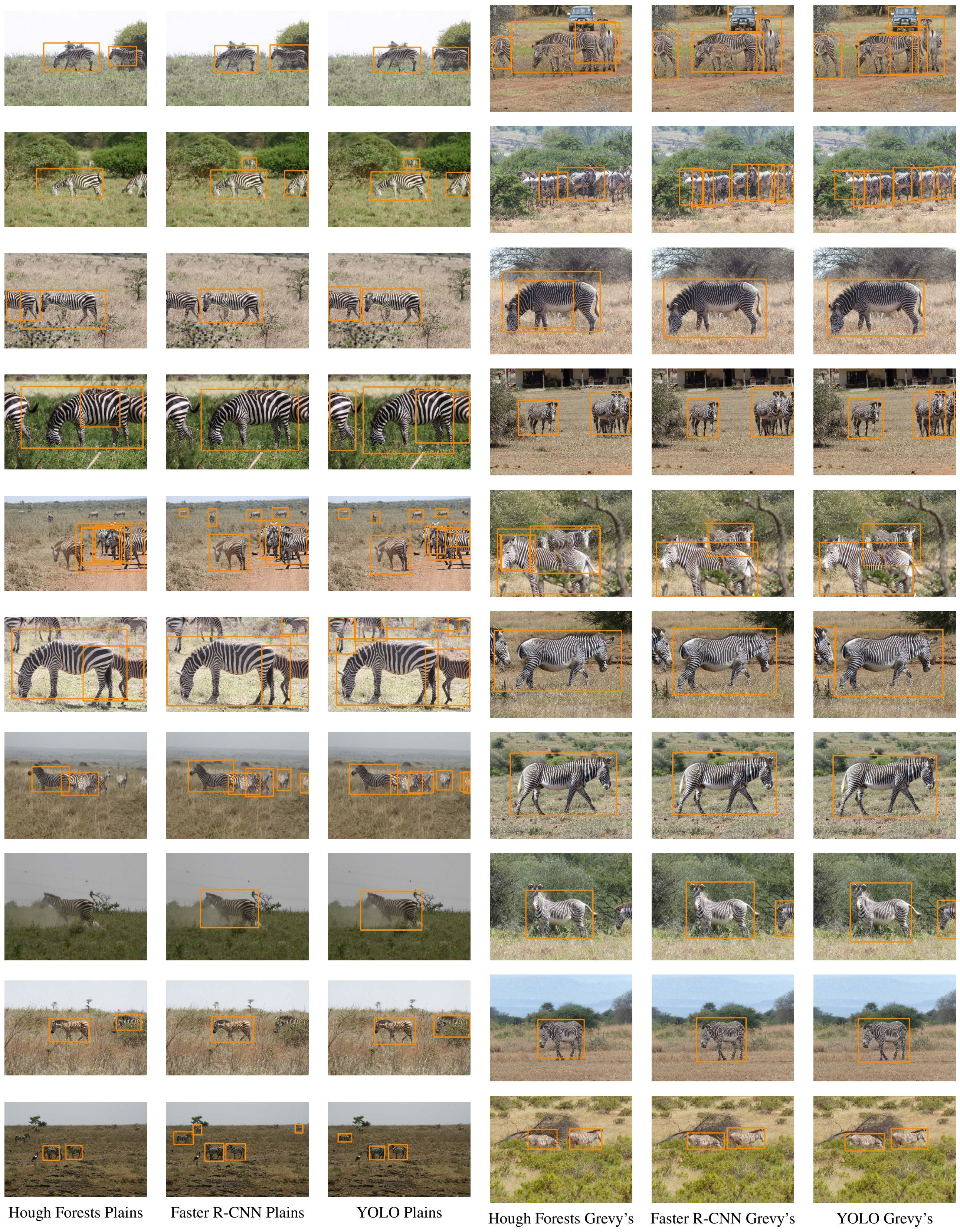

Figure 6: Detections on a set of 20 images. The operating point was set to 0.8 for the CNNs and 0.6 for Hough Forests.

[3] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool. Real-time facial feature detection using conditional regression forests. In *Proc. 2012 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 2578–2585, Providence, RI, Jun. 2012. 2

[4] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In *Proc. 2014 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 2147–2154, Columbus, OH, Jun. 2014. 2, 5

[5] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.*, 88(2):303–338, Jun. 2010. 2

[6] G. Fanelli, J. Gall, and L. Van Gool. Real time head pose estimation with random regression forests. In *Proc. 2011 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 617–624, Colorado Springs, CO, Jun. 2011. 2

[7] J. Gall and V. Lempitsky. Class-specific Hough forests for object detection. In *Proc. 2009 IEEE Conf. Comput. Vis. and Pattern Recog.*, CVPR '09, pages 1022–1029, Miami, FL, Jun. 2009. 2

[8] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE Trans. Pattern Anal. and Mach. Intell.*, 33(11):2188–2202, Sep. 2011. 1, 2, 3, 4

[9] R. Girshick. Fast R-CNN. In *Proc. 2015 IEEE Int. Conf. Comput. Vis.*, pages 1440–1448, Santiago, Chile, Dec. 2015. 2, 4

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. 2014 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 580–587, Columbus, OH, Jun. 2014. 4

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385:1–12, Dec. 2015. 2

[12] S. He, R. W. Lau, W. Liu, Z. Huang, and Q. Yang. Supercnn: A superpixelwise convolutional neural network for salient object detection. *Int. J. Comput. Vis.*, 115(3):330–344, Apr. 2015. 2

[13] J. Hosang, R. Benenson, P. Dollr, and B. Schiele. What makes for effective detection proposals? *CoRR*, abs/1502.05082:1–16, Feb. 2015. 2

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Adv. in Neural Inform. Process. Syst. 25*, NIPS '12, pages 1097–1105, Lake Taho, NV, Dec. 2012. 2

[15] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *Int. J. Comput. Vis.*, 77(1):259–289, May 2008. 2

[16] H. Li, H. Lu, Z. Lin, X. Shen, and B. Price. LCNN: Low-level feature embedded CNN for salient object detection. *CoRR*, abs/1508.03928:1–9, Aug. 2015. 2

[17] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proc. 2015 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 3431–3440, Boston, MA, Jun. 2015. 2

[18] M. Oquab, L. Bottou, I. Laptev, and J. Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proc. 2014 IEEE Conf. Comput. Vis. and Pattern Recog.*, CVPR '14, pages 1717–1724, Columbus, OH, Jun. 2014. 5

[19] J. R. Parham. *Photographic censusing of zebra and giraffe in the Nairobi National Park*. M.s. thesis, Dept. Comput. Sci., Rensselaer Polytechnic Inst., Troy, NY, Nov. 2015. 2, 4, 6

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640:1–10, Jun. 2015. 1, 2, 4

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Adv. in Neural Inform. Process. Syst. 28*, pages 91–99, Montréal, Québec, Canada, Dec. 2015. 1, 2, 4, 5

[22] D. I. Rubenstein. Ecology, Social Behavior, and Conservation in Zebras. In R. Macedo, editor, *Behavioral ecology of tropical animals*, volume 42 of *Advances in the Study of Behavior*, chapter 7, pages 231 – 258. Elsevier, Burlington, MA, 2010. 6

[23] O. Russakovsky and J. Deng *et al.* ImageNet Large Scale Visual Recognition Challenge. *Int. J. Comput. Vis.*, 112(2):1–42, Apr. 2015. 2, 4

[24] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. OverFeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229:1–16, Dec. 2013. 2

[25] J. Silvertown. A new dawn for citizen science. *Trends in Ecol. & Evolution*, 24(9):467–471, Sep. 2009. 5

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556:1–14, Sep. 2014. 2

[27] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller. Striving for simplicity: The all convolutional net. *CoRR*, abs/1412.6806:1–14, Dec. 2014. 2

[28] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842:1–12, Sep. 2014. 2

[29] K. E. A. van de Sande, J. R. R. Uijlings, T. Gevers, and A. W. M. Smeulders. Segmentation as selective search for object recognition. In *Proc. 2011 IEEE Int. Conf. Comput. Vis.*, pages 1879–1886, Barcelona, Spain, Nov. 2011. 4

[30] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proc. 2006 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 37–44, New York, NY, Jun. 2006. 2

[31] A. Yao, J. Gall, and L. Van Gool. A hough transform-based voting framework for action recognition. In *Proc. 2010 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 2061–2068, San Francisco, CA, Jun. 2010. 2

[32] M. D. Zeiler and R. Fergus. Visualizing and Understanding Convolutional Networks. *CoRR*, abs/1311.2901, Nov. 2013. 2

[33] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *Proc. 2015 IEEE Conf. Comput. Vis. and Pattern Recog.*, pages 1265–1274, Boston, MA, Jun. 2015. 2