Context R-CNN: Long Term Temporal Context for Per-Camera Object Detection Supplementary Material

Sara Beery*†, Guanghang Wu†, Vivek Rathod†, Ronny Votel†, Jonathan Huang†
California Institite of Technology* Google†

1. Implementation Details

We implemented our attention modules within the Tensorflow Object Detection API open-source Faster-RCNN architecture with Resnet 101 backbone [2]. Faster-RCNN optimization and model parameters are not changed between the single-frame baseline and our experiments, and we ensure robust single-frame baselines via hyperparameter sweeps. We train on Google TPUs (v3) [3] using MomentumSGD with weight decay 0.0004 and momentum 0.9. We construct each batch using 32 clips, drawing four frames for each clip spaced 1 frame apart and resizing to 640×640. Batches are placed on 8 TPU cores, colocating frames from the same clip. We augment with random flipping, ensuring that the memory banks are flipped to match the current frames to preserve spatial consistency. All our experiments use a softmax temperature of $T = .01$ for the attention mechanism, which we found in early experiments to outperform .1 and 1.

2. Dataset Statistics and Per-Class Performance

Each of the real-world datasets (Snapshot Serengeti, Caltech Camera Traps, and CityCam) has a long-tailed distribution of classes, which can be seen in Figure 3. Dealing with imbalanced data is a known challenge across machine learning disciplines [1, 5], with rare classes (classes not well-represented during training) frequently proving difficult to recognize.

In Figure 5 in the main text, we demonstrate that the per-class performance universally improves for Snapshot Serengeti (SS). In Figure 1, we show the per-class performance for Caltech Camera Traps (CCT). and CityCam (CC). Performance on CCT improves for all classes from the single frame model. We see that for one class in CC, "Middle Truck", our method performs slightly worse; However, this class is relatively ambiguous, as the concept of "middle" size is not well-defined.

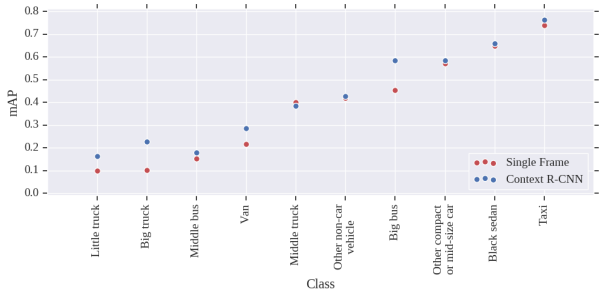
3. Spatiotemptoral Encodings

We normalize the spatial and temporal information for each object we include in the contextual memory bank. In order to do so, we choose to use a single float between 0 and 1 to represent each of: year, month, day, hour, minute, x center coordinate, y center coordinate, object width, and object height.

We normalize each element as follows:

- **Year**: We select a reasonable window of possible years covered by our data, 1990-2030. We normalize the year within that window, representing the year in question as $\frac{year-1990}{2030-1990}$.
- **Month**: We normalize the month of the year by 12 months, *i.e.* $\frac{month}{12}$.
- **Day**: We normalize the day of the month by 31 days for simplicity, regardless of how many days there are in the month in question, *i.e.* $\frac{day}{31}$.

(a) Caltech Camera Traps.



(b) CityCam.

Figure 1: **Performance per class.** Performance comparison from single-frame to our memory-based model. Note this reports mAP for each class averaged across IoU thresholds, as popularized by the COCO challenge [4].

(a) Before.

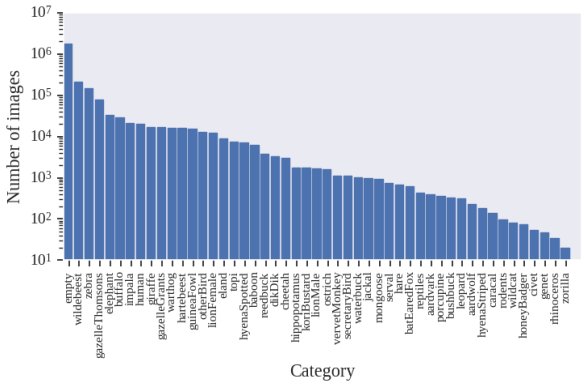
(b) After.

Figure 2: **Our system is robust to a static camera being accidentally shifted.** Before and after example of a camera that had been bumped by an animal. The images are from the same camera. The first image was taken August 8th 2010, the next August 9th 2010. We find that the system can still utilize contextual information across a camera shift.

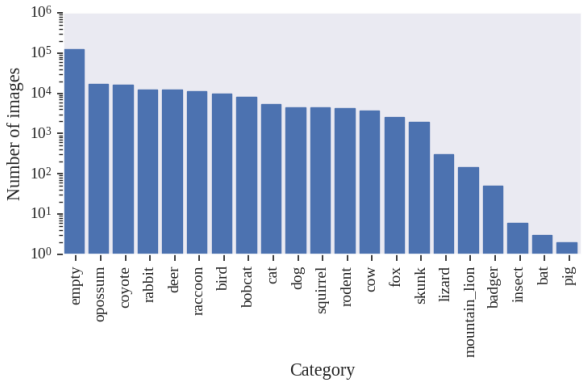
- **Hour**: We normalize the hour of the day by 24 hours, *i.e.* $\frac{hour}{24}$.
- **Minute**: We normalize the minute of the hour by 60 minutes, *i.e.* $\frac{minute}{60}$.
- **X Center Coordinate**: We normalize the x coordinate pixel location by the width of the image in pixels, *i.e.* $\frac{x\_center\_location\ (pixels)}{image\_width\ (pixels)}$
- **Y Center Coordinate**: We normalize the y coordinate pixel location by the height of the image in pixels, *i.e.* $\frac{y\_center\_location\ (pixels)}{image\_height\ (pixels)}$
- **Width of Object**: We normalize the object width in pixels by the width of the image in pixels, *i.e.* $\frac{object\_width\ (pixels)}{image\_width\ (pixels)}$
- **Height of Object**: We normalize the object height in pixels by the height of the image in pixels, *i.e.* $\frac{object\_height\ (pixels)}{image\_height\ (pixels)}$

4. Camera Movement

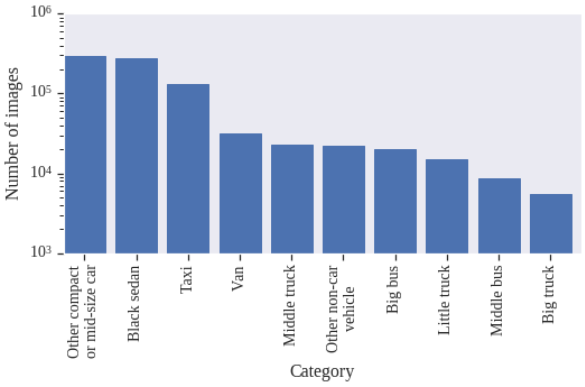
Our system has no hard requirements about the camera being static, instead we leverage the fact that it is static



(a) Snapshot Serengeti.



(b) Caltech Camera Traps.



(c) CityCam.

Figure 3: **Imbalanced class distributions.** Images per category for each of the three datasets. Note the y-axis is in log scale.

implicitly through our memory bank to provide appropriate and relevant context. We find that our system is robust to static cameras that get moved, unlike traditional background modeling approaches. In Snapshot Serengeti in particular, the animals have a tendency to rub against the camera posts and cause camera shifts over time. Figure 2 shows a "before and after" example of a camera being bumped or

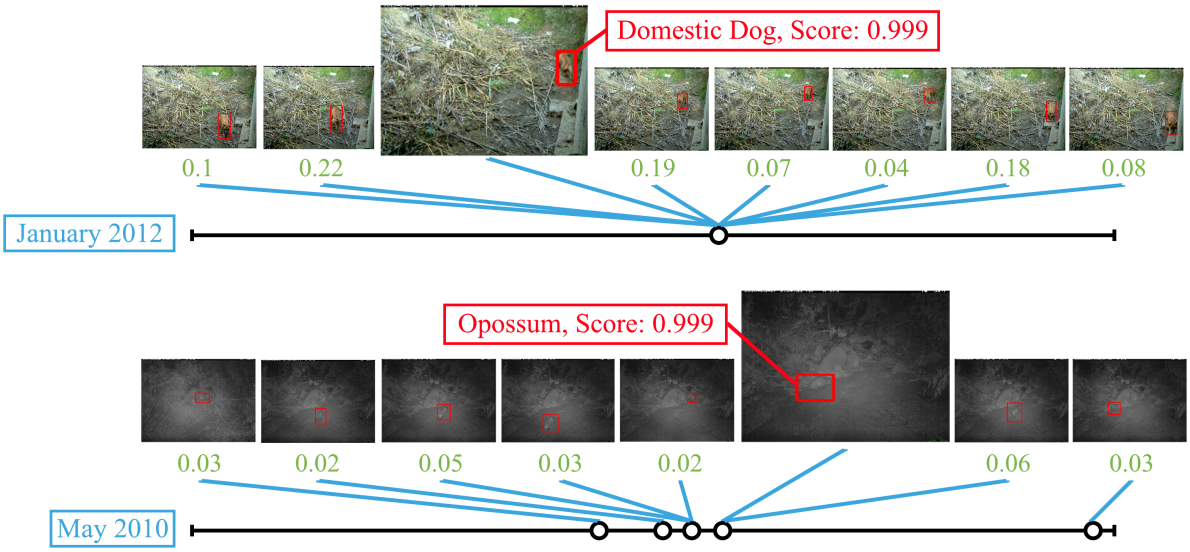


Figure 4: **Visualizing attention.** In each example, the keyframe is shown at a larger scale, with Context R-CNN's detection, class, and score shown in red. We consider a time horizon of one month, and show the images and boxes with highest attention weights (shown in green). The model pays attention to objects of the same class, and the distribution of attention across time can be seen in the timelines below each example.

moved.

5. Attention Visualization

In Figure 4 in the main text, we visualize attention over time for two examples from Snapshot Serengeti. In Figure 4 we show examples from Caltech Camera Traps. Similarly to the visualizations of attention on SS, we see that attention is adaptive to the most relevant information, paying attention across time as needed. The model consistently learns to attend to objects of the same class.

In Figure 5, we visualize how Context R-CNN learns to learn and attend to unlabeled background classes, namely rocks and bushes. Remember that these exact camera locations were never seen during training, so the model has learned to use temporal context to determine when to ignore these salient background classes. It learns to cluster background objects of a certain type, for example bushes, across the frames at a given location. Note that these attended background objects are not always the same instance of the class, which makes sense as background classes may maintain visual similarity within a scene even if they aren't the exact same instance of that type. Species of plants or types of rock are often geographically clustered.

References

[1] Sara Beery, Yang Liu, Dan Morris, Jim Piavis, Ashish Kapoor, Markus Meister, and Pietro Perona. Synthetic examples improve generalization for rare classes. *arXiv preprint arXiv:1904.05916*, 2019. 1

[2] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7310–7311, 2017. 1

[3] Sameer Kumar, Victor Bitorff, Dehao Chen, Chiachen Chou, Blake Hechtman, HyoukJoong Lee, Naveen Kumar, Peter Mattson, Shibo Wang, Tao Wang, et al. Scale mlperf-0.6 models on google tpu-v3 pods. *arXiv preprint arXiv:1909.09756*, 2019. 1

[4] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 1

[5] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 1

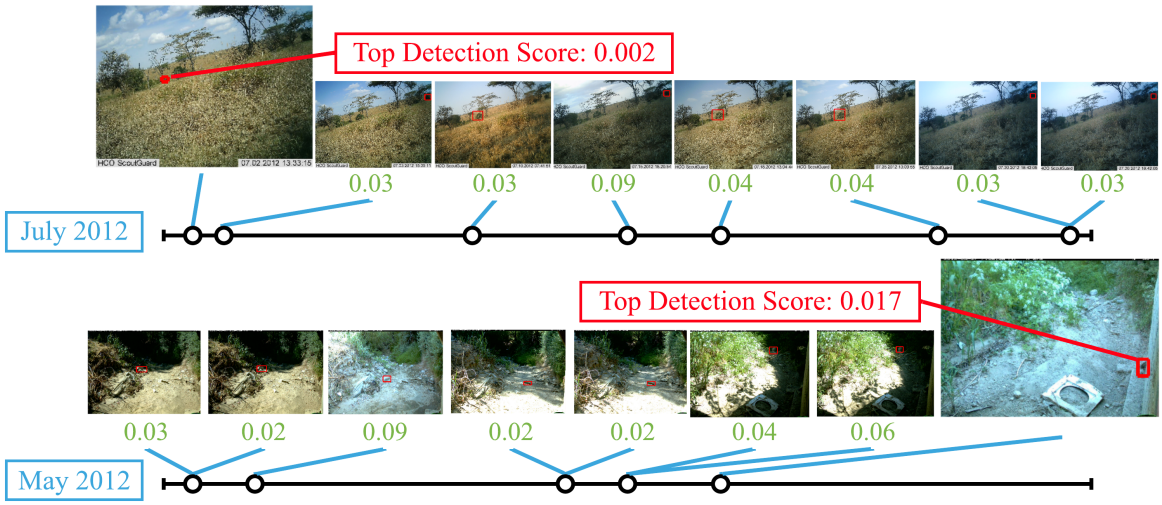


Figure 5: **Visualizing attention on background classes.** In each example, the keyframe is shown at a larger scale, with Context R-CNN's detection, class, and score shown in red. We consider a time horizon of one month, and show the images and boxes with highest attention weights (shown in green). The first example is from SS, it shows a detected bush (an unlabeled, background class), and shows that Context R-CNN attends to the same bush over time, as well as *different* bushes in the frame. In the second example, from CCT, we see a similar situation with the background class "rock."