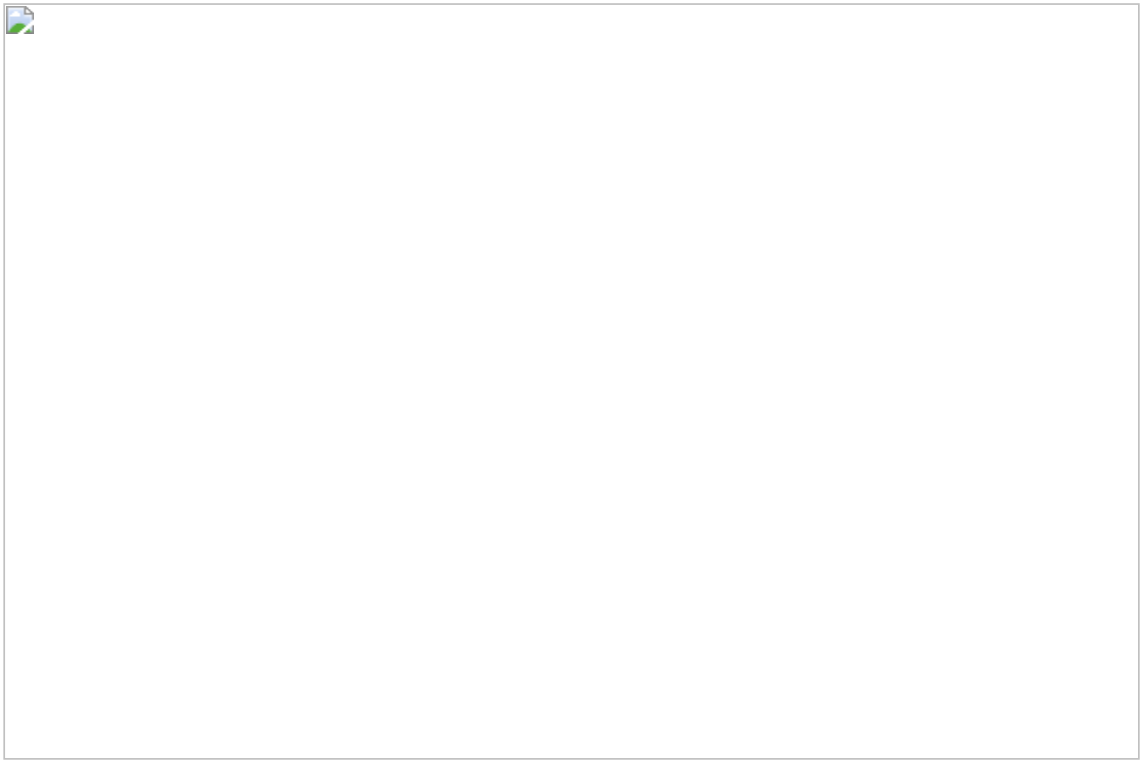
TensorFlow Object Detection API

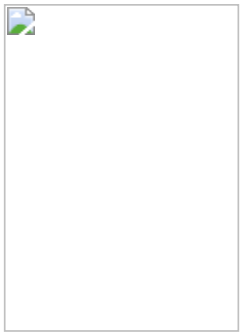
Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. At Google we've certainly found this codebase to be useful for our computer vision needs, and we hope that you will as well.



Contributions to the codebase are welcome and we would love to hear back from you if you find this API useful. Finally if you use the TensorFlow Object Detection API for a research publication, please consider citing:

```
"Speed/accuracy trade-offs for modern convolutional object detectors."
Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z,
Song Y, Guadarrama S, Murphy K, CVPR 2017
```

[link][bibtex]



Support for TensorFlow 2 and 1

The TensorFlow Object Detection API supports both TensorFlow 2 (TF2) and TensorFlow 1 (TF1). A majority of the modules in the library are both TF1 and TF2 compatible. In cases where they are not, we provide two versions.

Although we will continue to maintain the TF1 models and provide support, we encourage users to try the Object Detection API with TF2 for the following reasons:

- We provide new architectures supported in TF2 only and we will continue to develop in TF2 going forward.
- The popular models we ported from TF1 to TF2 achieve the same performance.
- A single training and evaluation binary now supports both GPU and TPU distribution strategies making it possible to train models with synchronous SGD by default.
- Eager execution with new binaries makes debugging easy!

Finally, if are an existing user of the Object Detection API we have retained the same config language you are familiar with and ensured that the TF2 training/eval binary takes the same arguments as our TF1 binaries.

Note: The models we provide in TF2 Zoo and TF1 Zoo are specific to the TensorFlow major version and are not interoperable.

Please select one of the links below for TensorFlow version-specific documentation of the Object Detection API:

Tensorflow 2.x

- Object Detection API TensorFlow 2
- TensorFlow 2 Model Zoo

Tensorflow 1.x

- Object Detection API TensorFlow 1
- TensorFlow 1 Model Zoo

Whats New

SpaghettiNet for Edge TPU

We have released SpaghettiNet models optimized for the Edge TPU in the Google Tensor SoC.

SpaghettiNet models are derived from a TuNAS search space that incorporates group convolution based Inverted Bottleneck blocks. The backbone and detection head are connected through MnasFPN-style feature map merging and searched jointly.

When compared to MobileDet-EdgeTPU, SpaghettiNet models achieve +2.2% mAP (absolute) on COCO17 at the same latency. They also consume <70% of the energy used by MobileDet-EdgeTPU to achieve the same accuracy.

Sample config available here.

Thanks to contributors: Marie White, Hao Xu, Hanxiao Liu and Suyog Gupta.

DeepMAC architecture

We have released our new architecture, **DeepMAC**, designed for partially supervised instance segmentation. DeepMAC stands for Deep Mask-heads Above CenterNet, and is based on our CenterNet implementation. In our paper we show that DeepMAC achieves state-of-the-art results for the partially supervised instance segmentation task without using any specialty modules or losses; just better mask-head architectures. The findings from our paper are not specific to CenterNet and can also be applied to Mask R-CNN or without any detector at all. Please see links below for more details

- DeepMAC documentation.
- Mask RCNN code in TF Model garden code base.
- DeepMAC Colab that lets you run a pre-trained DeepMAC model on user-specified boxes. Note that you are not restricted to COCO classes!
- Project website - git.io/deepmac

Thanks to contributors: Vighnesh Birodkar, Zhichao Lu, Siyang Li, Vivek Rathod, Jonathan Huang

Mobile Inference for TF2 models

TF2 OD API models can now be converted to TensorFlow Lite! Only SSD models currently supported. See documentation.

Thanks to contributors: Sachin Joglekar

TensorFlow 2 Support

We are happy to announce that the TF OD API officially supports TF2! Our release includes:

- New binaries for train/eval/export that are designed to run in eager mode.
- A suite of TF2 compatible (Keras-based) models; this includes migrations of our most popular TF1.x models (e.g., SSD with MobileNet, RetinaNet, Faster R-CNN, Mask R-CNN), as well as a few new architectures for which we will only maintain TF2 implementations:
    1. CenterNet - a simple and effective anchor-free architecture based on the recent Objects as Points paper by Zhou et al.
    2. EfficientDet - a recent family of SOTA models discovered with the help of Neural Architecture Search.
- COCO pre-trained weights for all of the models provided as TF2 style object-based checkpoints.
- Access to Distribution Strategies for distributed training --- our model are designed to be trainable using sync multi-GPU and TPU platforms.
- Colabs demo'ing eager mode training and inference.

See our release blogpost here. If you are an existing user of the TF OD API using TF 1.x, don't worry, we've got you covered.

Thanks to contributors: Akhil Chinnakotla, Allen Lavoie, Anirudh Vegesana, Anjali Sridhar, Austin Myers, Dan Kondratyuk, David Ross, Derek Chow, Jaeyoun Kim, Jing Li, Jonathan Huang, Jordi Pont-Tuset, Karmel Allison, Kathy Ruan, Kaushik Shivakumar, Lu He, Mingxing Tan, Pengchong Jin, Ronny Votel, Sara Beery, Sergi Caelles Prat, Shan Yang, Sudheendra Vijayanarasimhan, Tina Tian, Tomer Kaftan, Vighnesh Birodkar, Vishnu Banna, Vivek Rathod, Yanhui Liang, Yiming Shi, Yixin Shi, Yu-hui Chen, Zhichao Lu.

MobileDet GPU

We have released SSDLite with MobileDet GPU backbone, which achieves 17% mAP higher than the MobileNetV2 SSDLite (27.5 mAP vs 23.5 mAP) on a NVIDIA Jetson Xavier at comparable latency (3.2ms vs 3.3ms).

Along with the model definition, we are also releasing model checkpoints trained on the COCO dataset.

Thanks to contributors: Yongzhe Wang, Bo Chen, Hanxiao Liu, Le An (NVIDIA), Yu-Te Cheng (NVIDIA), Oliver Knieps (NVIDIA), and Josh Park (NVIDIA).

Context R-CNN

We have released Context R-CNN, a model that uses attention to incorporate contextual information images (e.g. from temporally nearby frames taken by a static camera) in order to improve accuracy. Importantly, these contextual images need not be labeled.

- When applied to a challenging wildlife detection dataset (Snapshot Serengeti), Context R-CNN with context from up to a month of images outperforms a single-frame baseline by 17.9% mAP, and outperforms S3D (a 3d convolution based baseline) by 11.2% mAP.
- Context R-CNN leverages temporal context from the unlabeled frames of a novel camera deployment to improve performance at that camera, boosting model generalizeability.

Read about Context R-CNN on the Google AI blog here.

We have provided code for generating data with associated context here, and a sample config for a Context R-CNN model here.

Snapshot Serengeti-trained Faster R-CNN and Context R-CNN models can be found in the model zoo.

A colab demonstrating Context R-CNN is provided here.

Thanks to contributors: Sara Beery, Jonathan Huang, Guanhang Wu, Vivek Rathod, Ronny Votel, Zhichao Lu, David Ross, Pietro Perona, Tanya Birch, and the Wildlife Insights AI Team.

Release Notes

See notes for all past releases.

Getting Help

To get help with issues you may encounter using the TensorFlow Object Detection API, create a new question on StackOverflow with the tags "tensorflow" and "object-detection".

Please report bugs (actually broken code, not usage questions) to the tensorflow/models GitHub issue tracker, prefixing the issue name with "object_detection".

Please check the FAQ for frequently asked questions before reporting an issue.

Maintainers

- Jonathan Huang (@GitHub jch1)
- Vivek Rathod (@GitHub tombstone)
- Vighnesh Birodkar (@GitHub vighneshbirodkar)
- Austin Myers (@GitHub austin-myers)
- Zhichao Lu (@GitHub pkulzc)
- Ronny Votel (@GitHub ronnyvotel)
- Yu-hui Chen (@GitHub yuhuichen1015)
- Derek Chow (@GitHub derekjchow)