G-RMI Object Detection

2nd ImageNet and COCO Visual Recognition Challenges Joint Workshop
ECCV 2016, Amsterdam

Jonathan Huang (jonathanhuang@google.com)

with Alireza Fathi, Ian Fischer, Sergio Guadarrama, Anoop Korattikara, Kevin Murphy,
Vivek Rathod, Yang Song, Chen Sun, Zbigniew Wojna, Menglong Zhu
October 9, 2016

Google Research and Machine Intelligence

Team Roster


Alireza Fathi


Ian Fischer


Sergio Guadarrama


Jonathan Huang

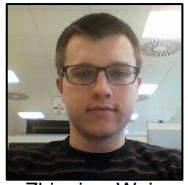

Anoop Korattikara

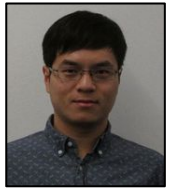

Kevin Murphy


Vivek Rathod


Yang Song


Chen Sun

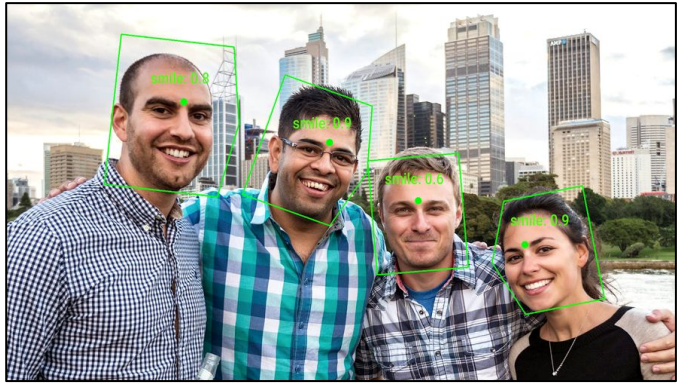

Zbigniew Wojna

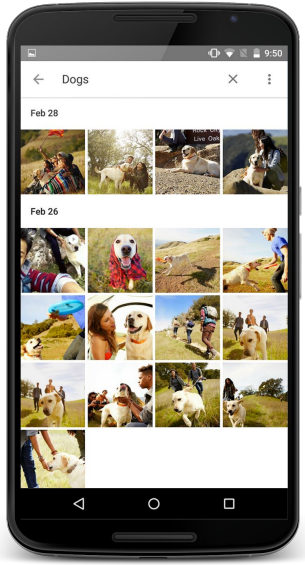

Menglong Zhu

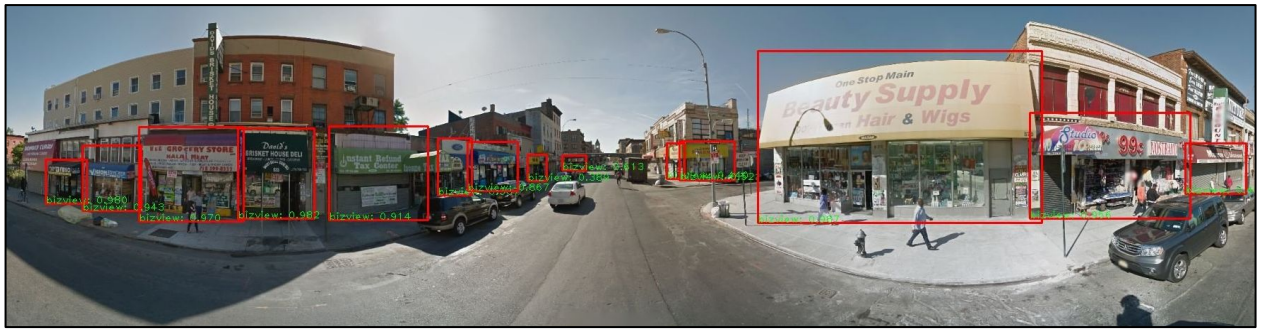
G-RMI Results (test-challenge)

	AP	AP50	AP75	APS	APM	APL	AR1	AR10	AR100	ARS	ARM	ARL	date
G-RMI	**0.415**	**0.624**	**0.453**	**0.239**	**0.439**	**0.548**	**0.343**	**0.552**	**0.606**	**0.428**	**0.646**	**0.746**	**9/18/2016**
MSRA_2015	0.373	0.589	0.399	0.183	0.419	0.524	0.321	0.477	0.491	0.273	0.556	0.679	11/26/2015
Trimps-Soushen	0.363	0.583	0.386	0.166	0.417	0.506	0.317	0.482	0.5	0.274	0.564	0.68	9/16/2016
Imagine Lab	0.352	0.533	0.388	0.153	0.38	0.52	0.318	0.501	0.528	0.304	0.587	0.722	9/17/2016
FAIRCNN	0.335	0.526	0.366	0.139	0.378	0.477	0.302	0.462	0.485	0.241	0.561	0.664	11/26/2015
CMU_A2_VGG16	0.324	0.532	0.34	0.151	0.357	0.451	0.296	0.463	0.472	0.251	0.523	0.651	9/19/2016
ION	0.31	0.533	0.318	0.123	0.332	0.447	0.279	0.431	0.457	0.238	0.504	0.628	11/26/2015
ToConcoctPellucid	0.286	0.5	0.295	0.105	0.334	0.423	0.277	0.396	0.404	0.173	0.471	0.595	9/16/2016
Wall	0.284	0.49	0.29	0.06	0.316	0.476	0.268	0.408	0.433	0.185	0.485	0.65	9/17/2016
hust-mclab	0.278	0.485	0.289	0.109	0.308	0.398	0.26	0.371	0.377	0.159	0.425	0.549	9/18/2016
CMU_A2	0.257	0.46	0.261	0.059	0.287	0.417	0.248	0.355	0.365	0.105	0.43	0.582	11/27/2015
UofA	0.255	0.437	0.268	0.08	0.273	0.391	0.251	0.354	0.359	0.147	0.389	0.56	11/27/2015
Decode	0.224	0.414	0.222	0.05	0.239	0.369	0.229	0.33	0.338	0.101	0.388	0.54	11/27/2015
Wall_2015	0.205	0.364	0.21	0.043	0.199	0.339	0.218	0.307	0.318	0.109	0.33	0.497	11/27/2015
SinicaChen	0.19	0.363	0.181	0.042	0.199	0.31	0.209	0.301	0.309	0.095	0.335	0.499	11/19/2015
UCSD	0.188	0.369	0.176	0.035	0.188	0.315	0.206	0.303	0.313	0.09	0.342	0.519	11/27/2015
"1026"	0.179	0.32	0.177	0.026	0.18	0.303	0.177	0.248	0.254	0.051	0.283	0.412	11/27/2015

Object Detection @ Google







Object Detection in TensorFlow

32,986 stars, 14,327 forks on Github
(as of Sept 29, 2016)

Your laptop

Datacenters

Mobile

Raspberry Pi

Tensor Processing Unit

- Deploy models anywhere
- Scalable
- For research **and** production
- State-of-the-art performance
- Support leading methods

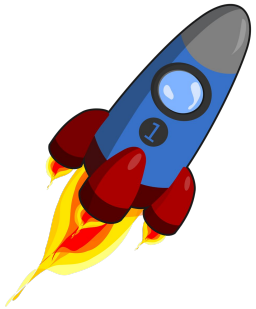
  Multibox/SSD, Fast/Faster

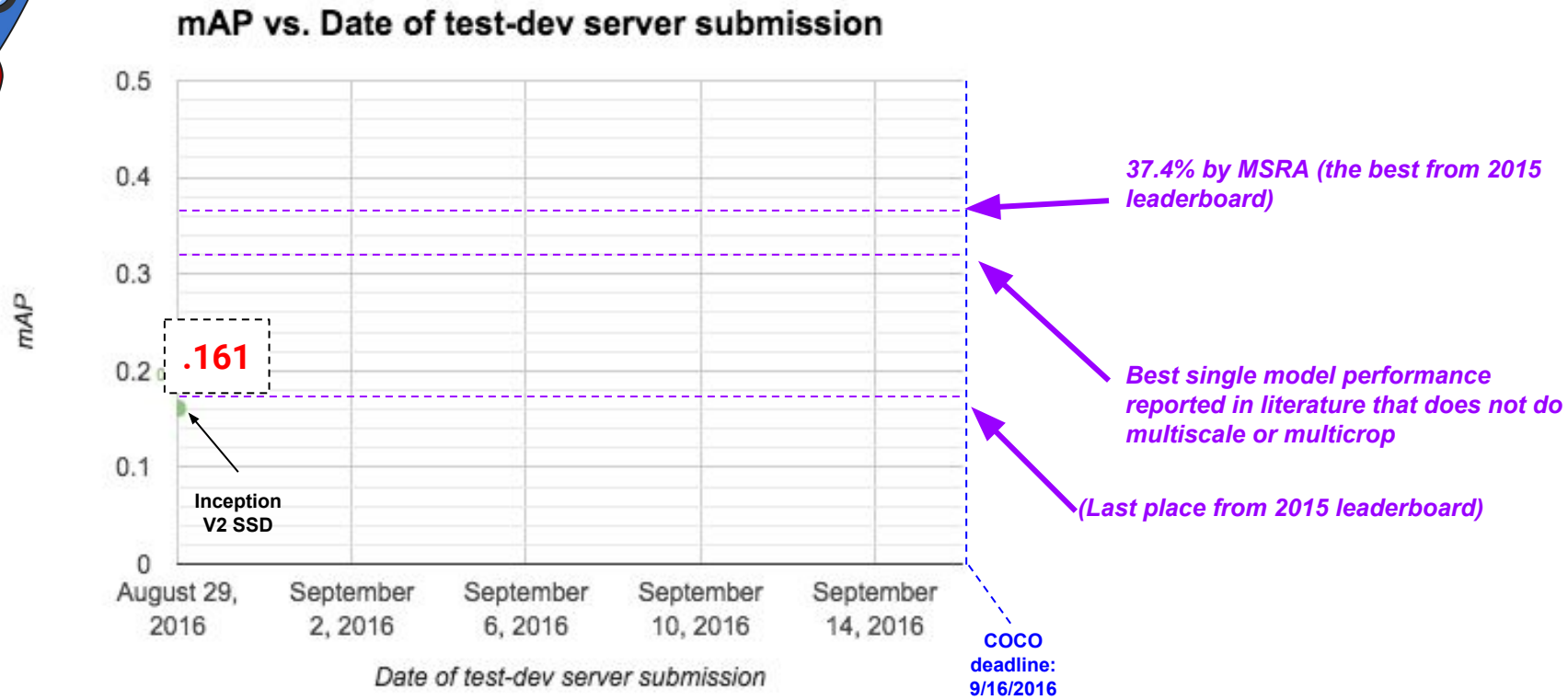
  RCNN, etc...

Race to the Top

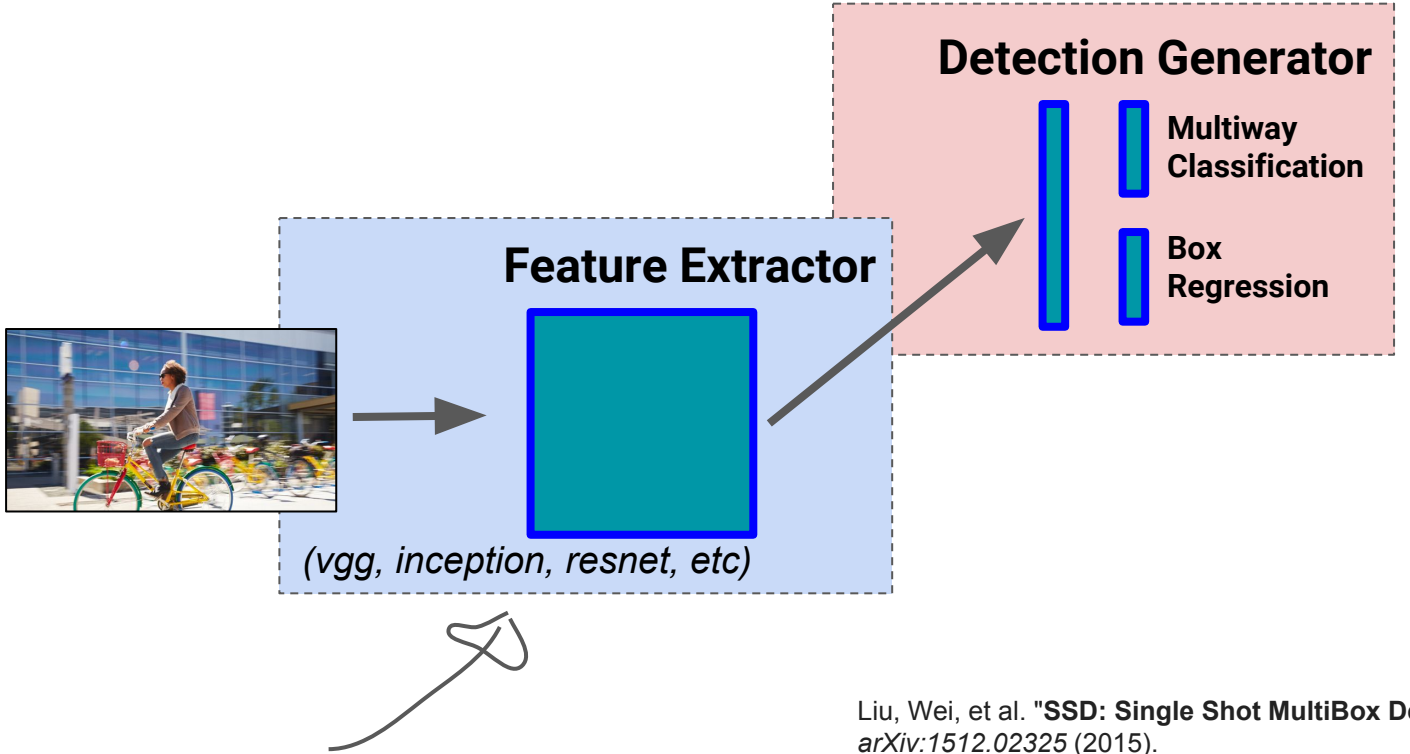
First submission to test-dev: Tensorflow implementation of SSD(ish) model with 224x224 input images

Liu, Wei, et al. "**SSD: Single Shot MultiBox Detector**." *arXiv preprint arXiv:1512.02325* (2015).





Single Shot or Class-wise RPN models

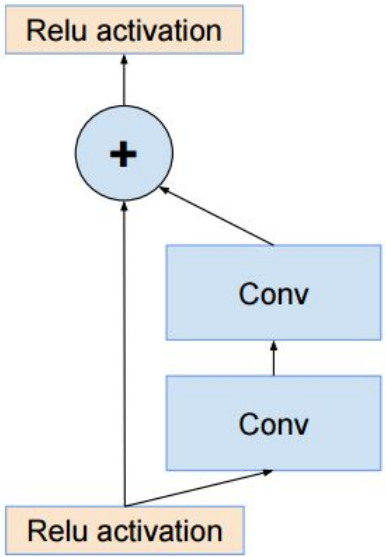


Use TF-*slim* model zoo to swap in multiple feature extractor architectures

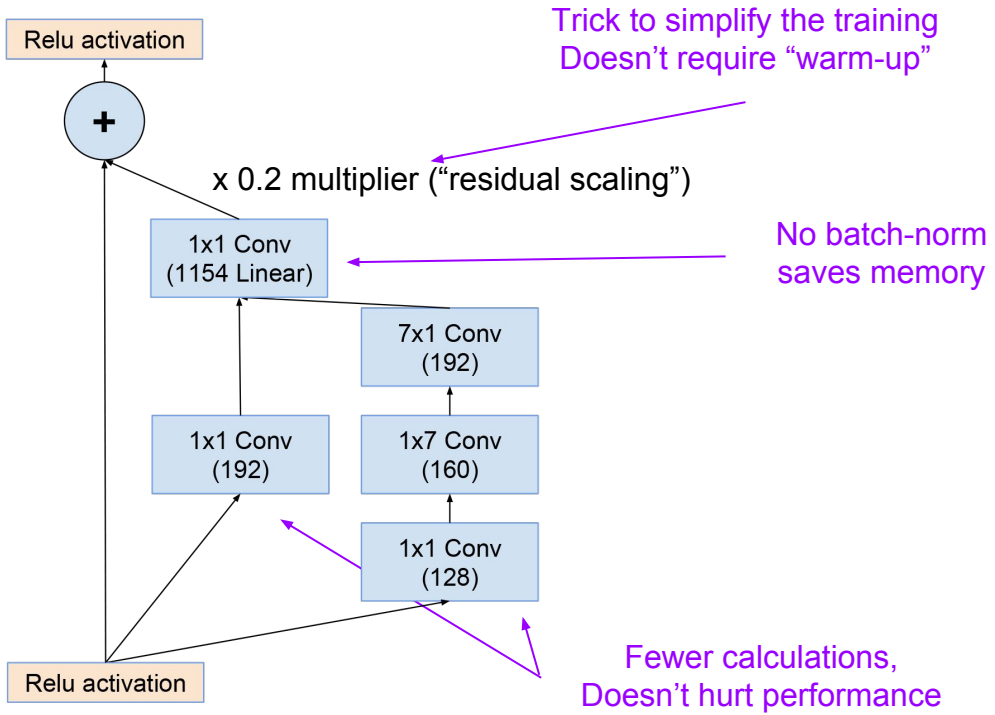
Liu, Wei, et al. "**SSD: Single Shot MultiBox Detector**." *arXiv preprint arXiv:1512.02325* (2015).

Dai, Jifeng, et al. "**R-FCN: Object Detection via Region-based Fully Convolutional Networks**." arXiv preprint arXiv:1605.06409 (2016).

Residual Blocks vs. Inception Resnet Blocks



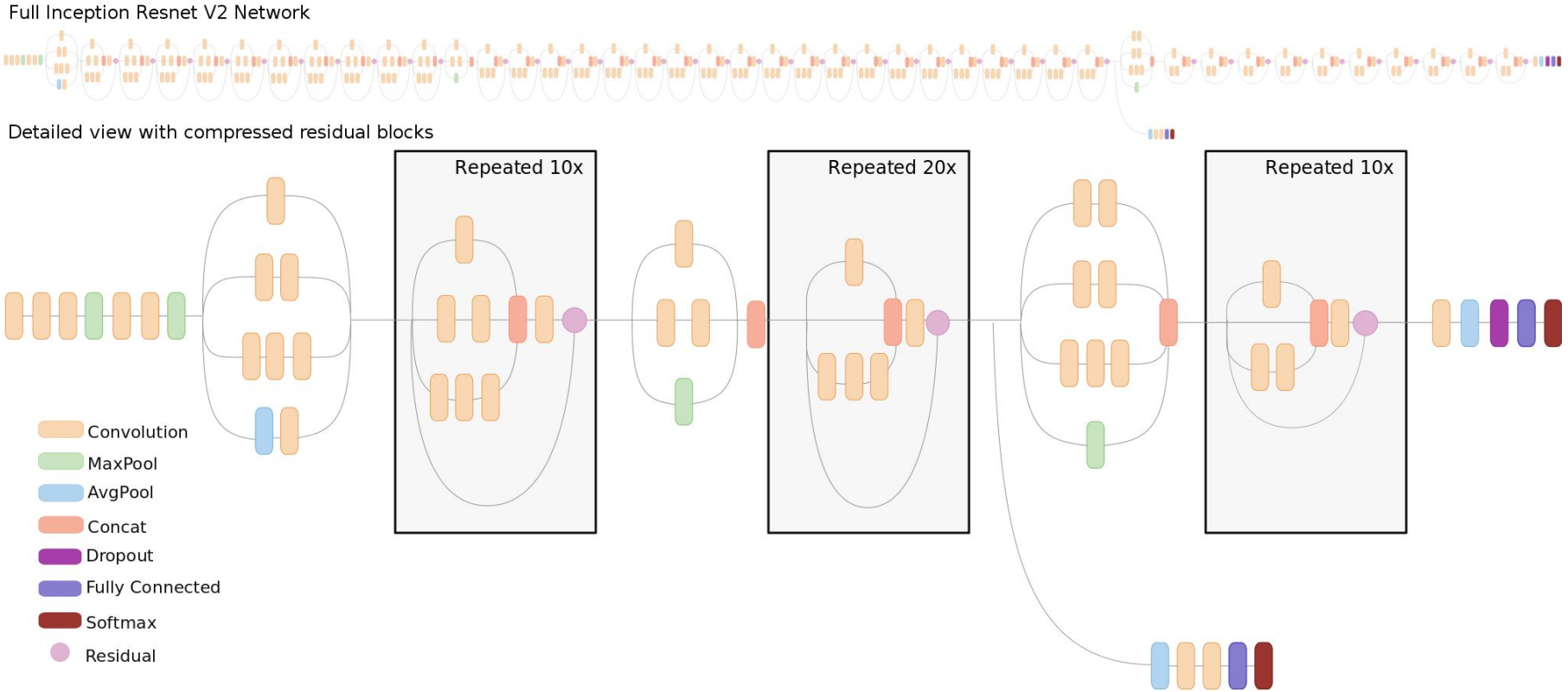
Residual Block



Inception Resnet Block

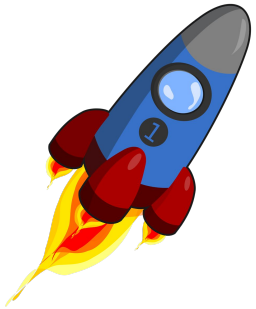
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning by Szegedy et al.

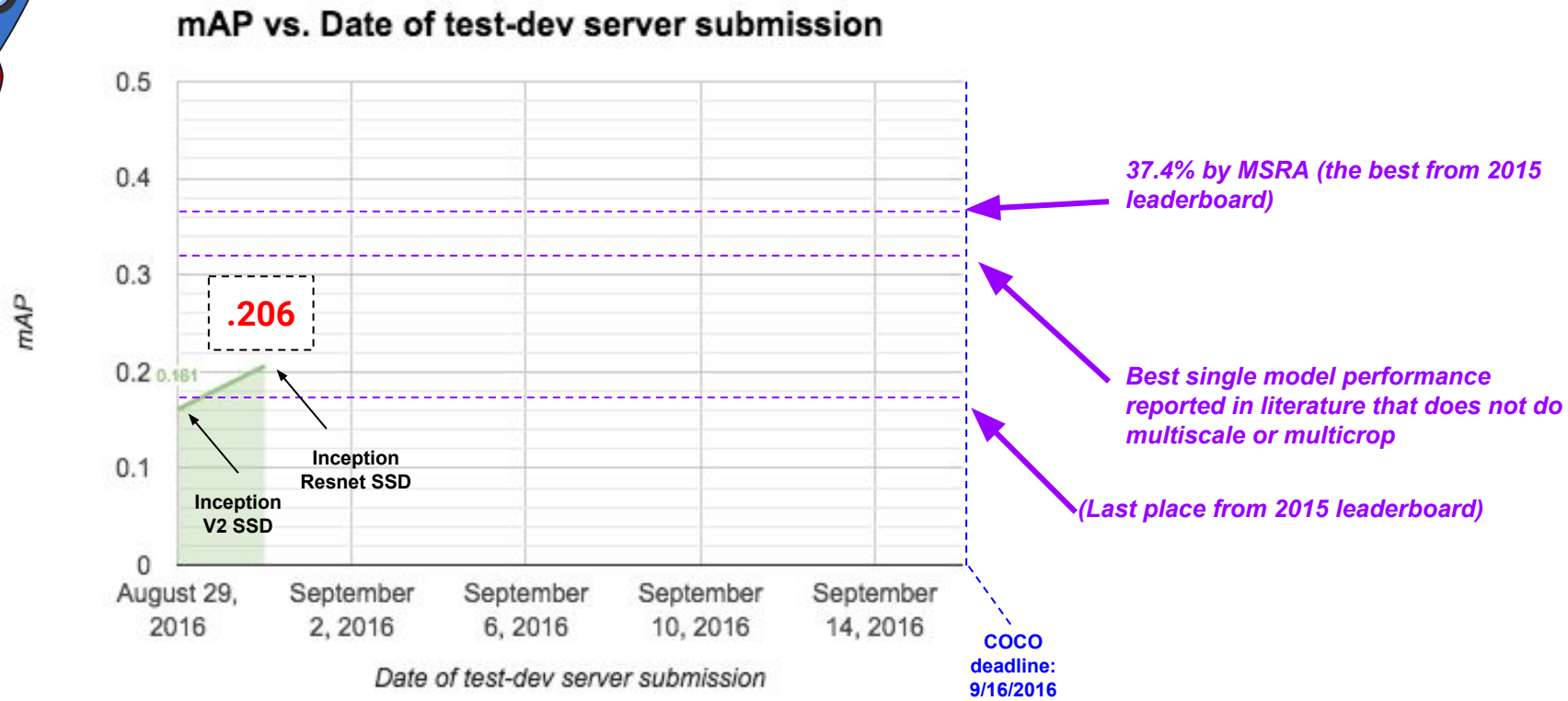
Inception Resnet (v2) Feature Extractor



Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning by Szegedy et al.

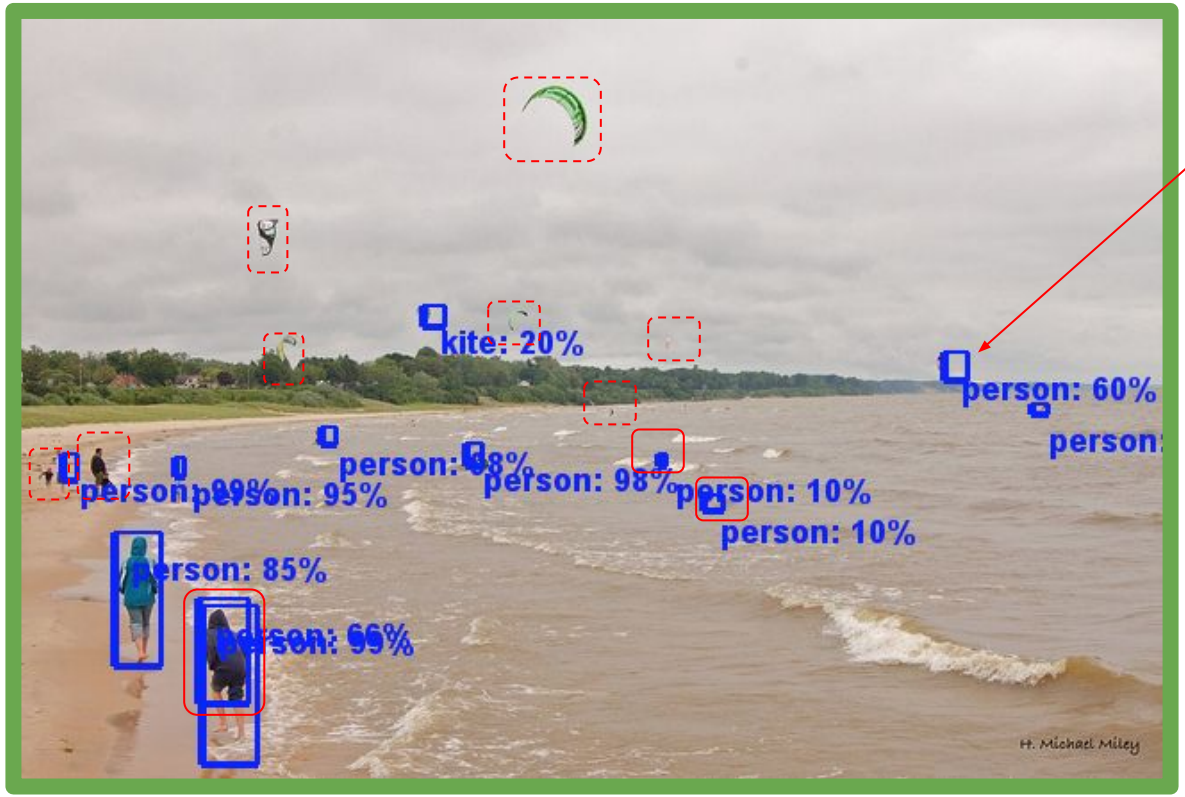
Race to the Top






H. Michael Miley

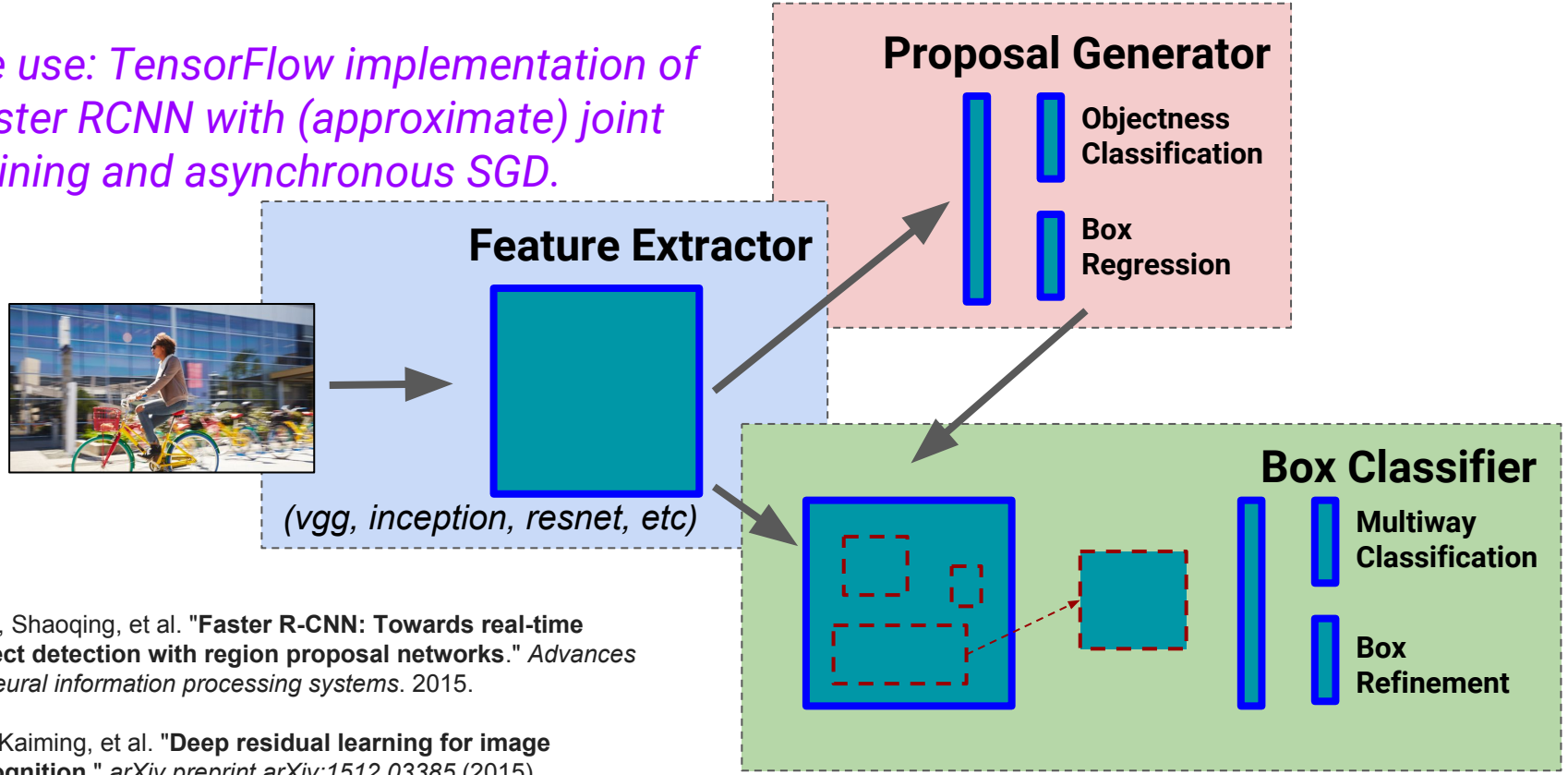
Inception Resnet SSD



inaccurate localization
for small objects

Faster RCNN based models

We use: TensorFlow implementation of Faster RCNN with (approximate) joint training and asynchronous SGD.



Ren, Shaoqing, et al. "**Faster R-CNN: Towards real-time object detection with region proposal networks**." *Advances in neural information processing systems*. 2015.

He, Kaiming, et al. "**Deep residual learning for image recognition**." *arXiv preprint arXiv:1512.03385* (2015).

Key TF-ops

(now available in open source tensorflow release)

`tf.image.non_max_suppression(boxes, scores, max_output_size, iou_threshold=None, name=None)`

Greedily selects a subset of bounding boxes in descending order of score,

pruning away boxes that have high intersection-over-union (IOU) overlap with previously selected boxes. Bounding boxes are supplied as [y1, x1, y2, x2], where (y1, x1) and (y2, x2) are the coordinates of any diagonal pair of box corners and the coordinates can be provided as normalized (i.e., lying in the interval [0, 1]) or absolute. Note that this algorithm is agnostic to where the origin is in the coordinate system. Note that this algorithm is invariant to orthogonal transformations and translations of the coordinate system; thus translating or reflections of the coordinate system result in the same boxes being selected by the algorithm.

The output of this operation is a set of integers indexing into the input collection of bounding boxes representing the selected boxes. The bounding box coordinates corresponding to the selected indices can then be obtained using the tf.gather operation. For example:

selected_indices = tf.image.non_max_suppression(boxes, scores, max_output_size, iou_threshold) selected_boxes = tf.gather(boxes, selected_indices)

Args:

- `boxes`: A `Tensor` of type `float32`. A 2-D float tensor of shape `[num_boxes, 4]`.
- `scores`: A `Tensor` of type `float32`. A 1-D float tensor of shape `[num_boxes]` representing a single score corresponding to each box (each row of boxes).
- `max_output_size`: A `Tensor` of type `int32`. A scalar integer tensor representing the maximum number of boxes to be selected by non max suppression.
- `iou_threshold`: An optional `float`. Defaults to `0.5`. A float representing the threshold for deciding whether boxes overlap too much with respect to IOU.
- `name`: A name for the operation (optional).

`tf.image.crop_and_resize(image, boxes, box_ind, crop_size, method=None, extrapolation_value=None, name=None)`

Extracts crops from the input image tensor and bilinearly resizes them (possibly

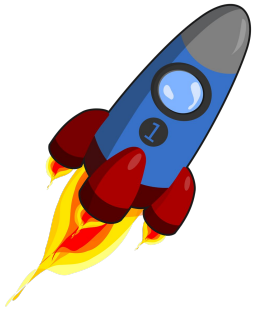
with aspect ratio change) to a common output size specified by `crop_size`. This is more general than the `crop_to_bounding_box` op which extracts a fixed size slice from the input image and does not allow resizing or aspect ratio change.

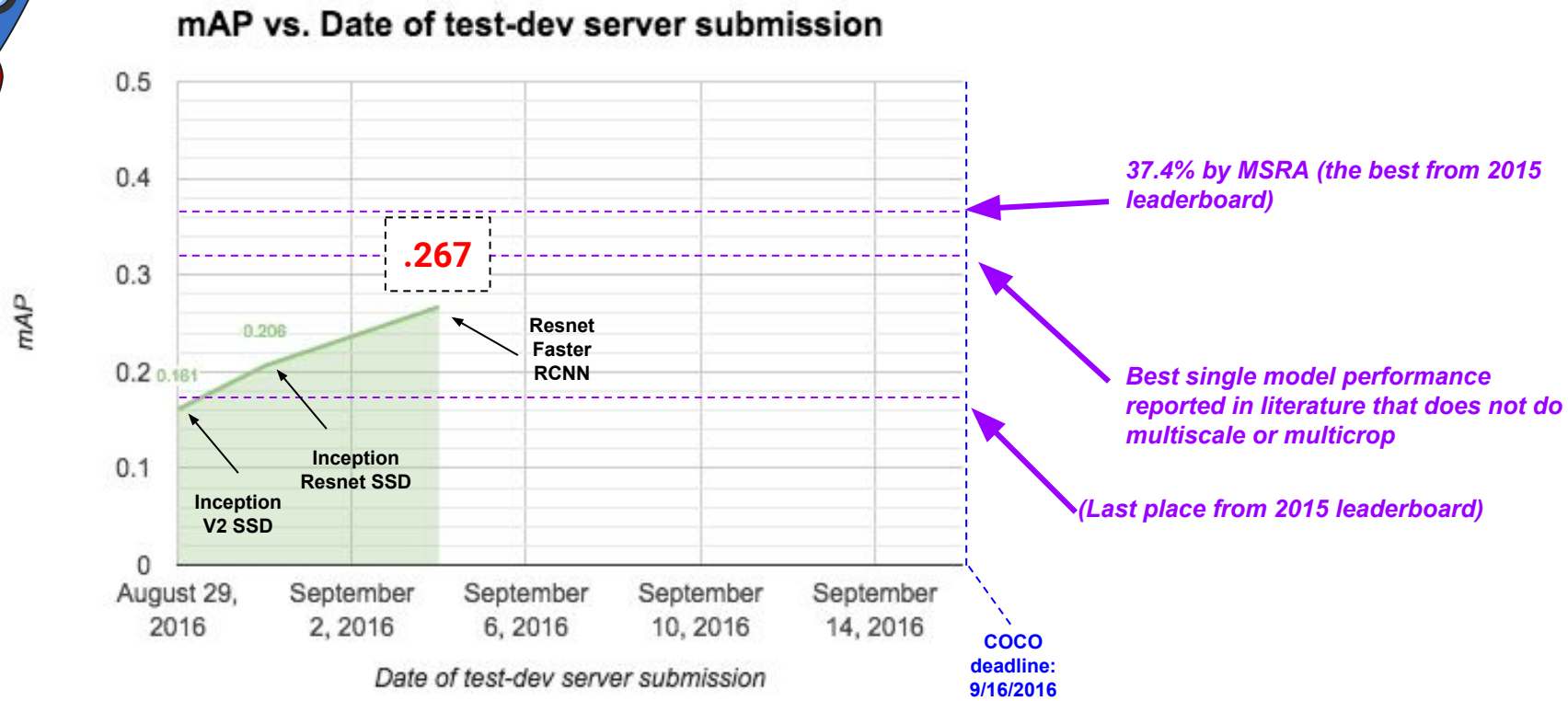
Returns a tensor with `crops` from the input `image` at positions defined at the bounding box locations in `boxes`. The cropped boxes are all resized (with bilinear interpolation) to a fixed `size = [crop_height, crop_width]`. The result is a 4-D tensor `[num_boxes, crop_height, crop_width, depth]`.

Args:

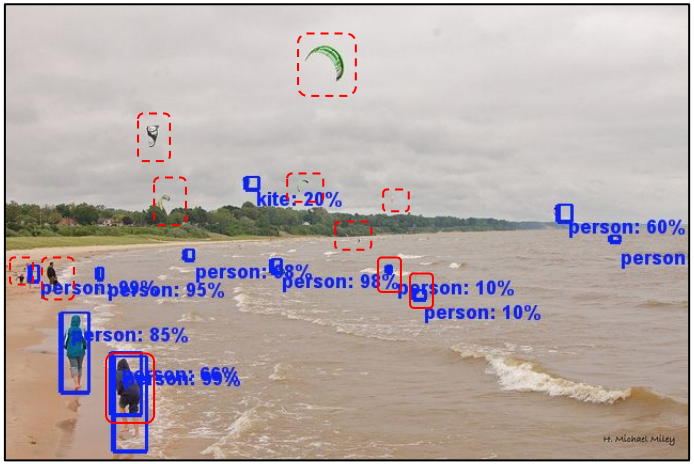
- `image`: A `Tensor`. Must be one of the following types: `uint8`, `int8`, `int16`, `int32`, `int64`, `half`, `float32`, `float64`. A 4-D tensor of shape `[batch, image_height, image_width, depth]`. Both `image_height` and `image_width` need to be positive.
- `boxes`: A `Tensor` of type `float32`. A 2-D tensor of shape `[num_boxes, 4]`. The `i`-th row of the tensor specifies the coordinates of a box in the `box_ind[i]` image and is specified in normalized coordinates `[y1, x1, y2, x2]`. A normalized coordinate value of `y` is mapped to the image coordinate at `y * (image_height - 1)`, so as the `[0, 1]` interval of normalized image height is mapped to `[0, image_height - 1] in image height coordinates. We do allow y1 > y2, in which case the sampled crop is an up-down flipped version of the original image. The width dimension is treated similarly. Normalized coordinates outside the`[0, 1]`range are allowed, in which case we use`extrapolation_value` to extrapolate the input image values.
- `box_ind`: A `Tensor` of type `int32`. A 1-D tensor of shape `[num_boxes]` with int32 values in `[0, batch)`. The value of `box_ind[i]` specifies the image that the `i`-th box refers to.
- `crop_size`: A `Tensor` of type `int32`. A 1-D tensor of 2 elements, `size = [crop_height, crop_width]`. All cropped image patches are resized to this size. The aspect ratio of the image content is not preserved. Both `crop_height` and `crop_width` need to be positive.
- `method`: An optional `string` from: `"bilinear"`. Defaults to `"bilinear"`. A string specifying the interpolation method. Only 'bilinear' is supported for now.
- `extrapolation_value`: An optional `float`. Defaults to `0`. Value used for extrapolation, when applicable.
- `name`: A name for the operation (optional).

Race to the Top

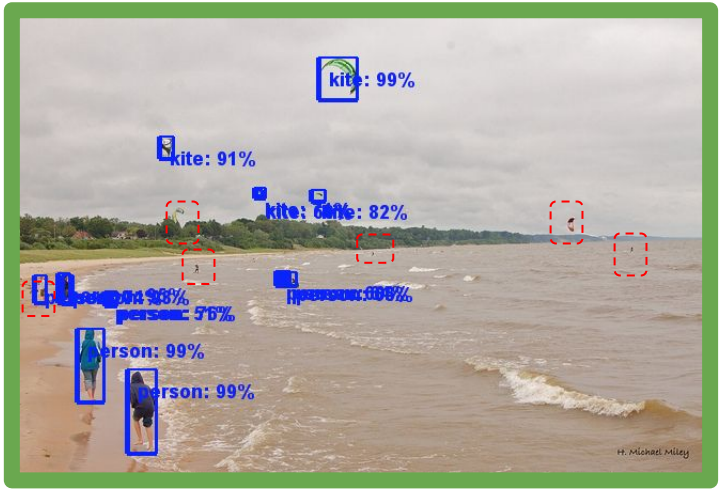




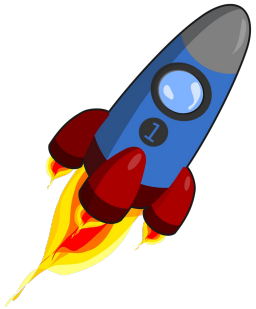
Inception Resnet SSD

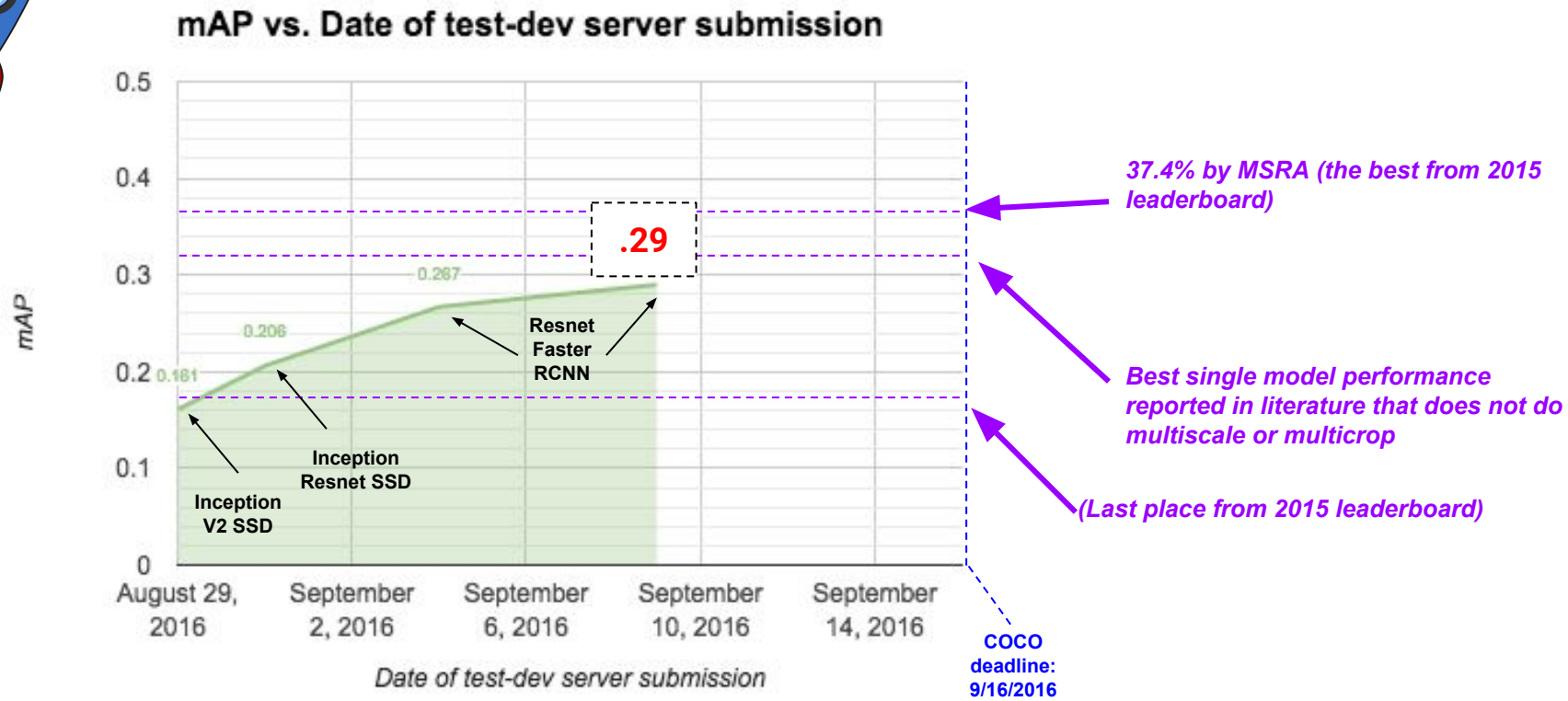


Resnet Faster RCNN

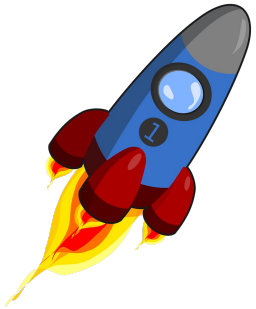


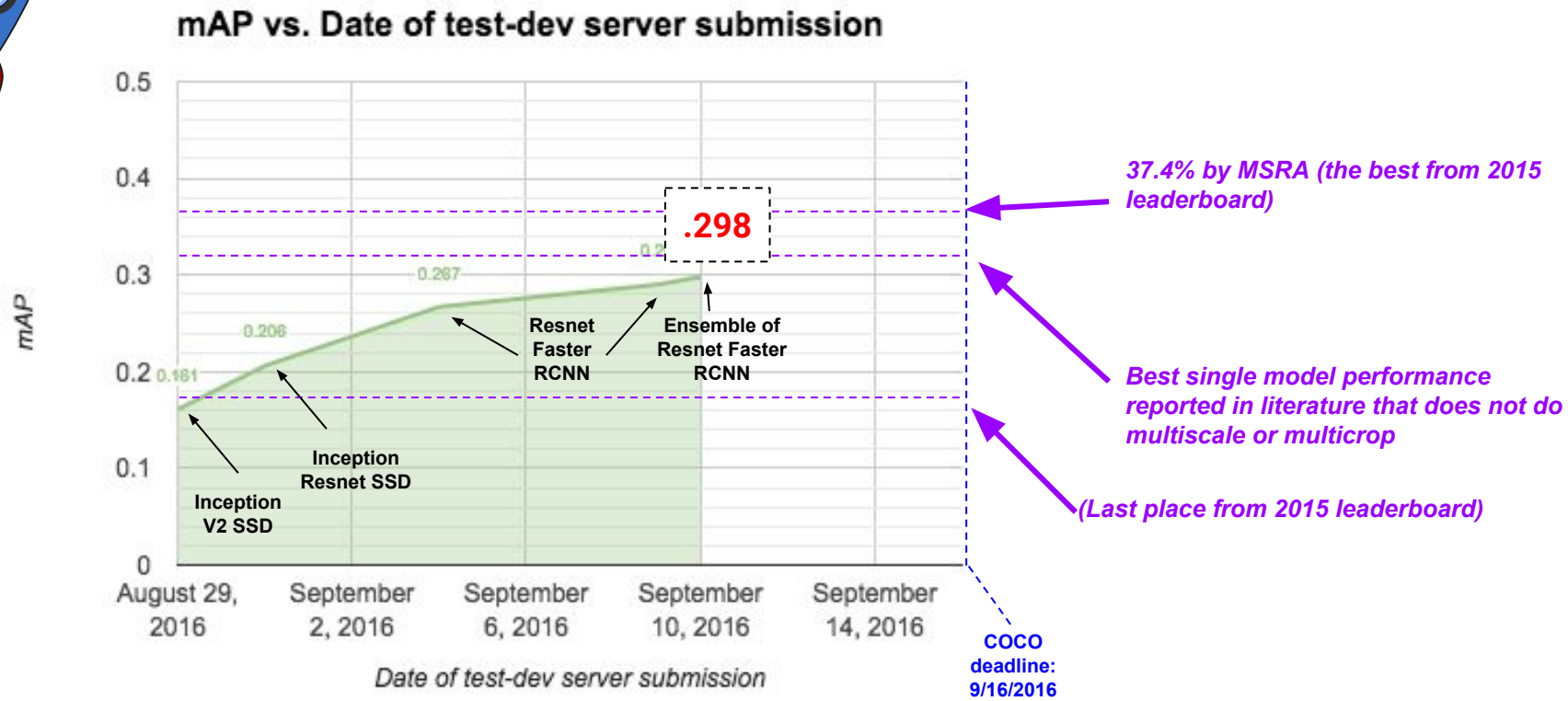
Race to the Top



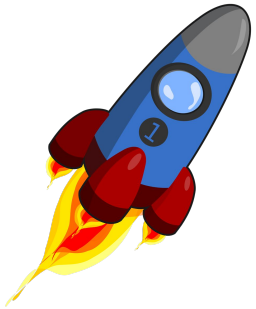


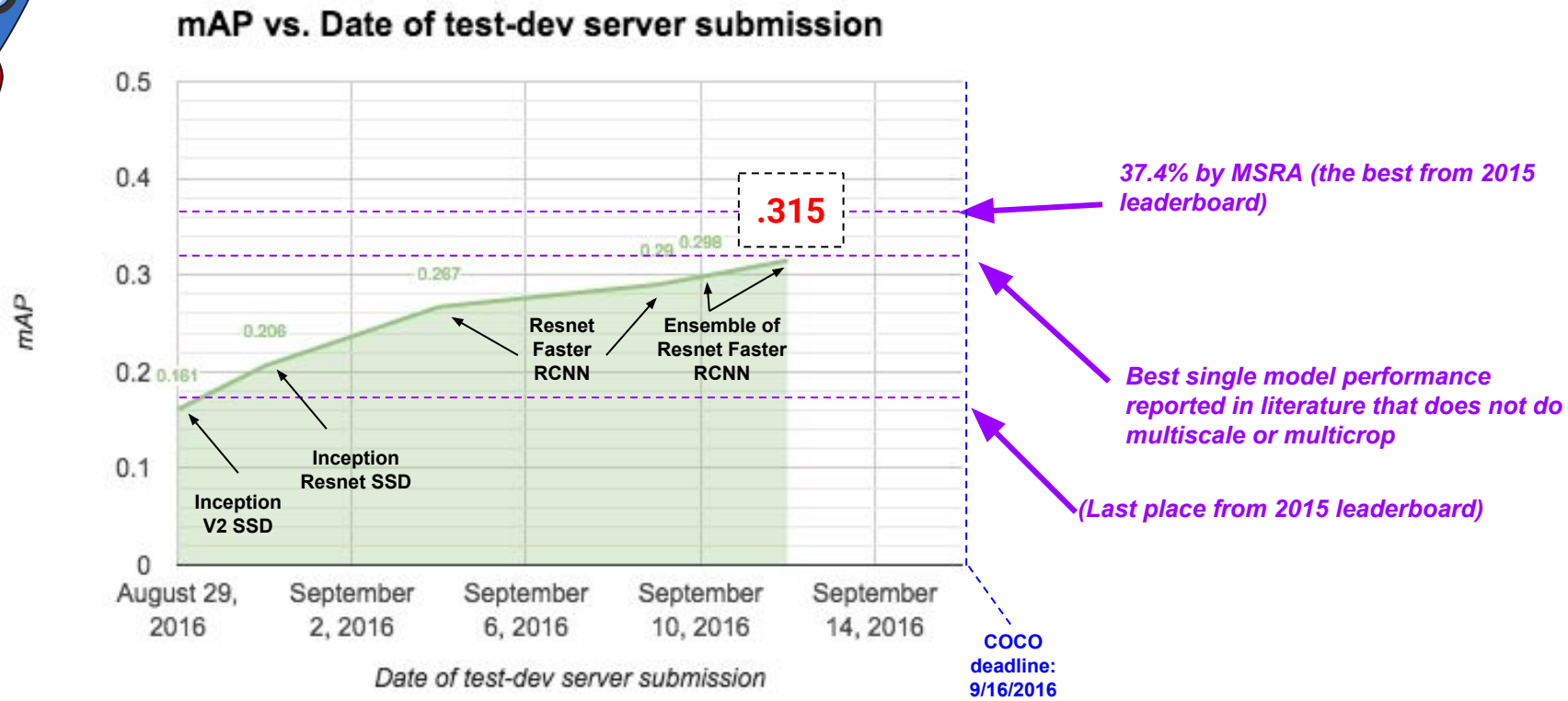
Race to the Top



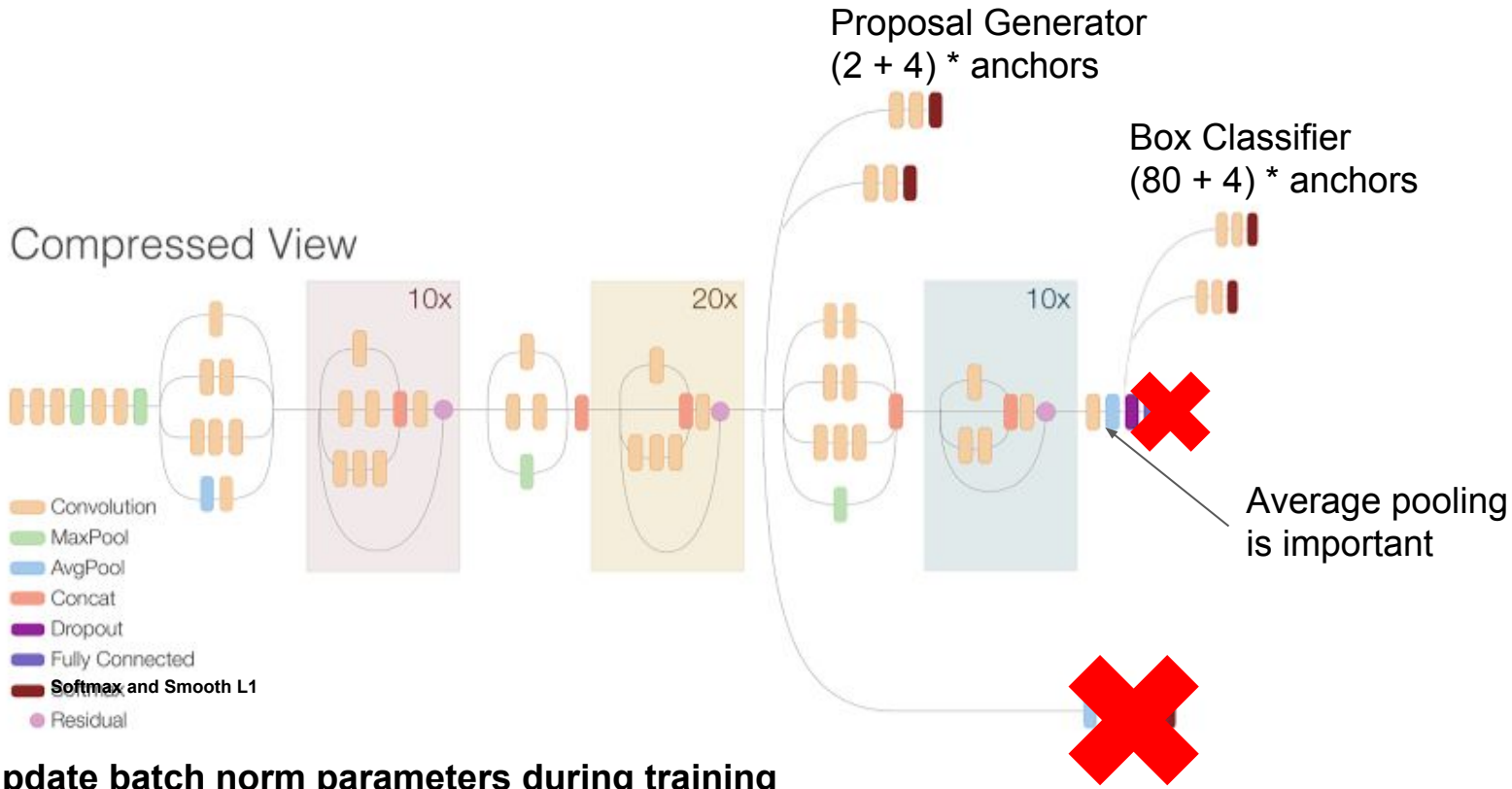


Race to the Top



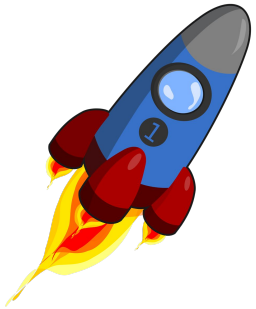


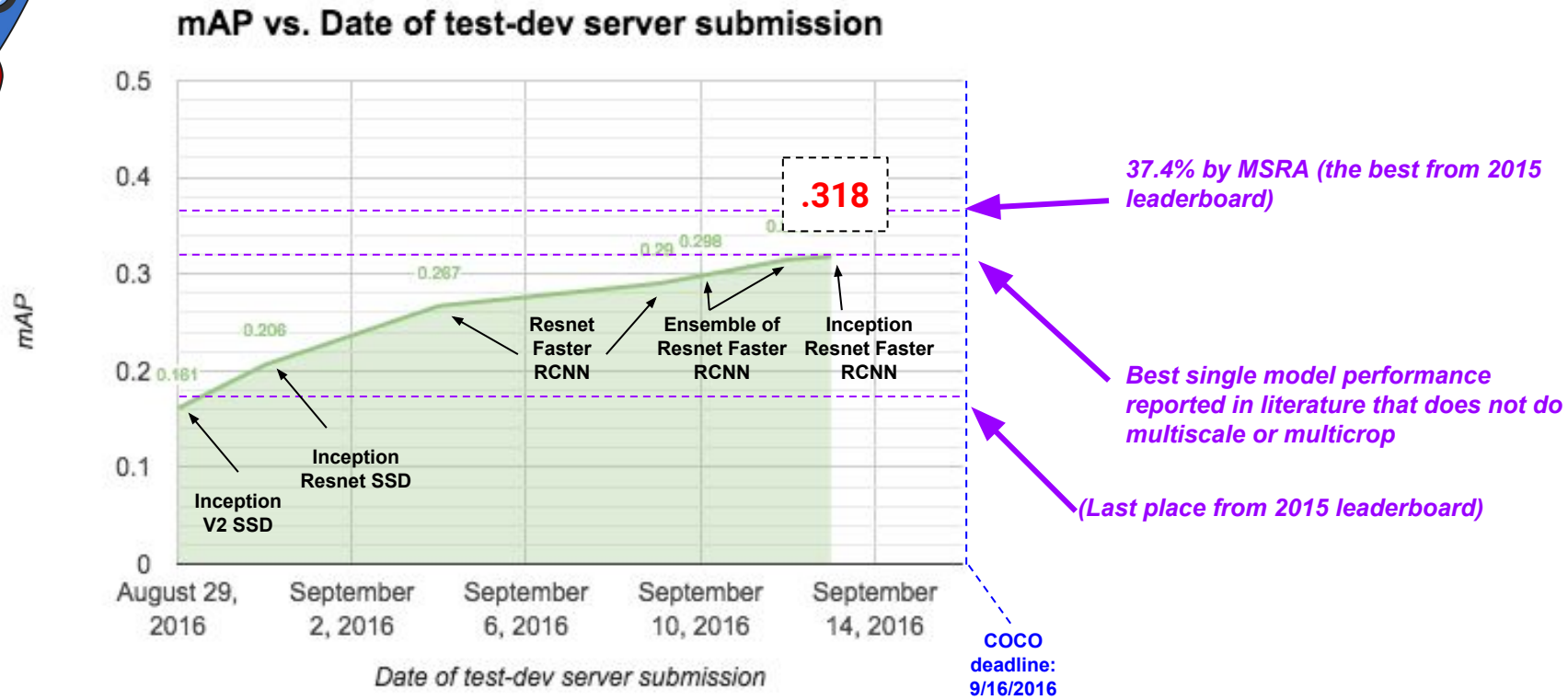
Faster RCNN w/Inception Resnet (v2)



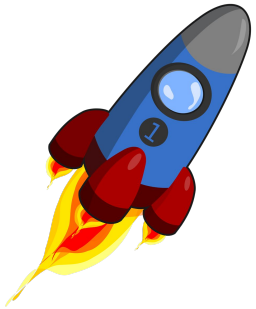
Note: We don't update batch norm parameters during training

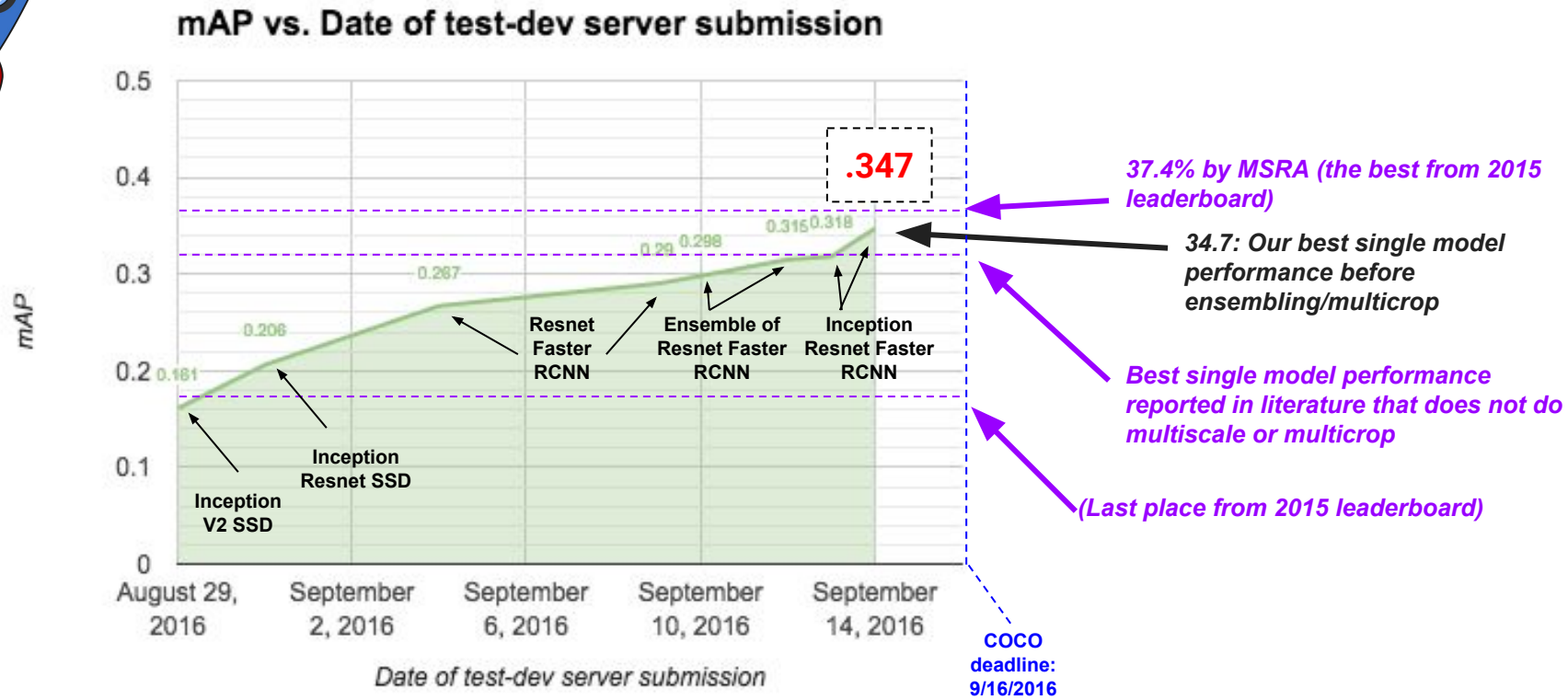
Race to the Top



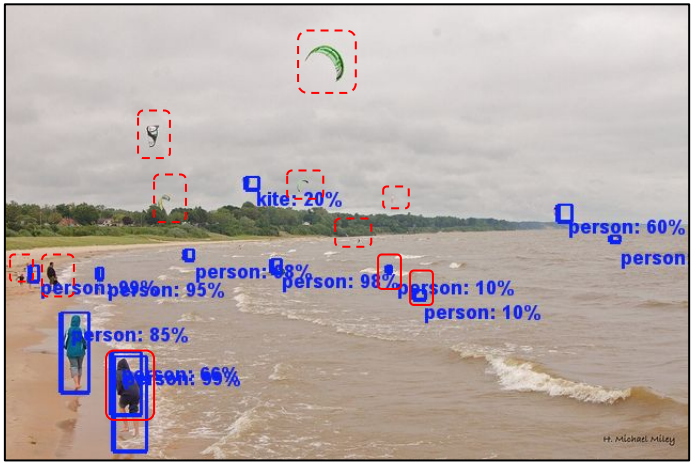


Race to the Top

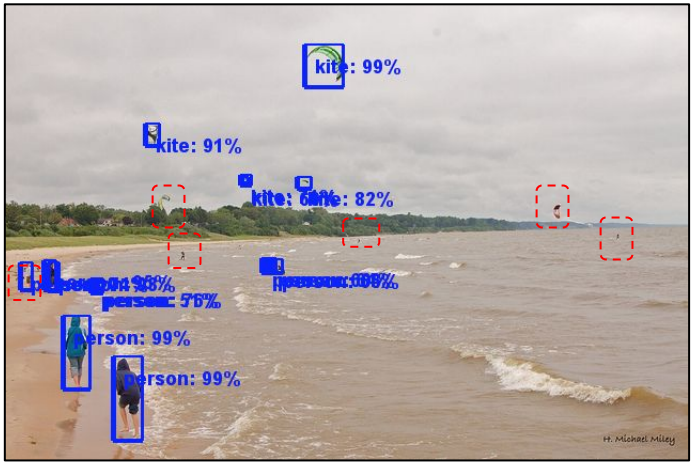




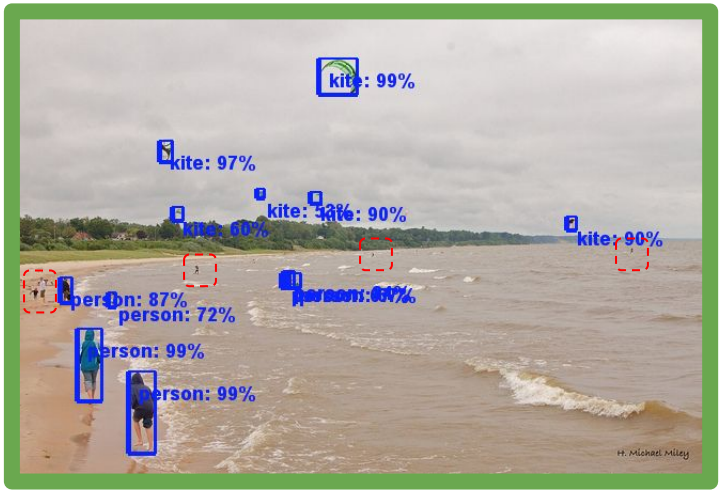
Inception Resnet SSD



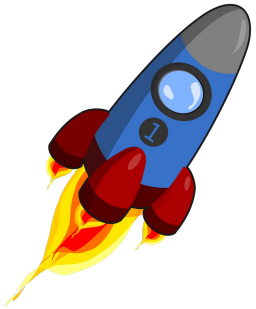
Resnet Faster RCNN

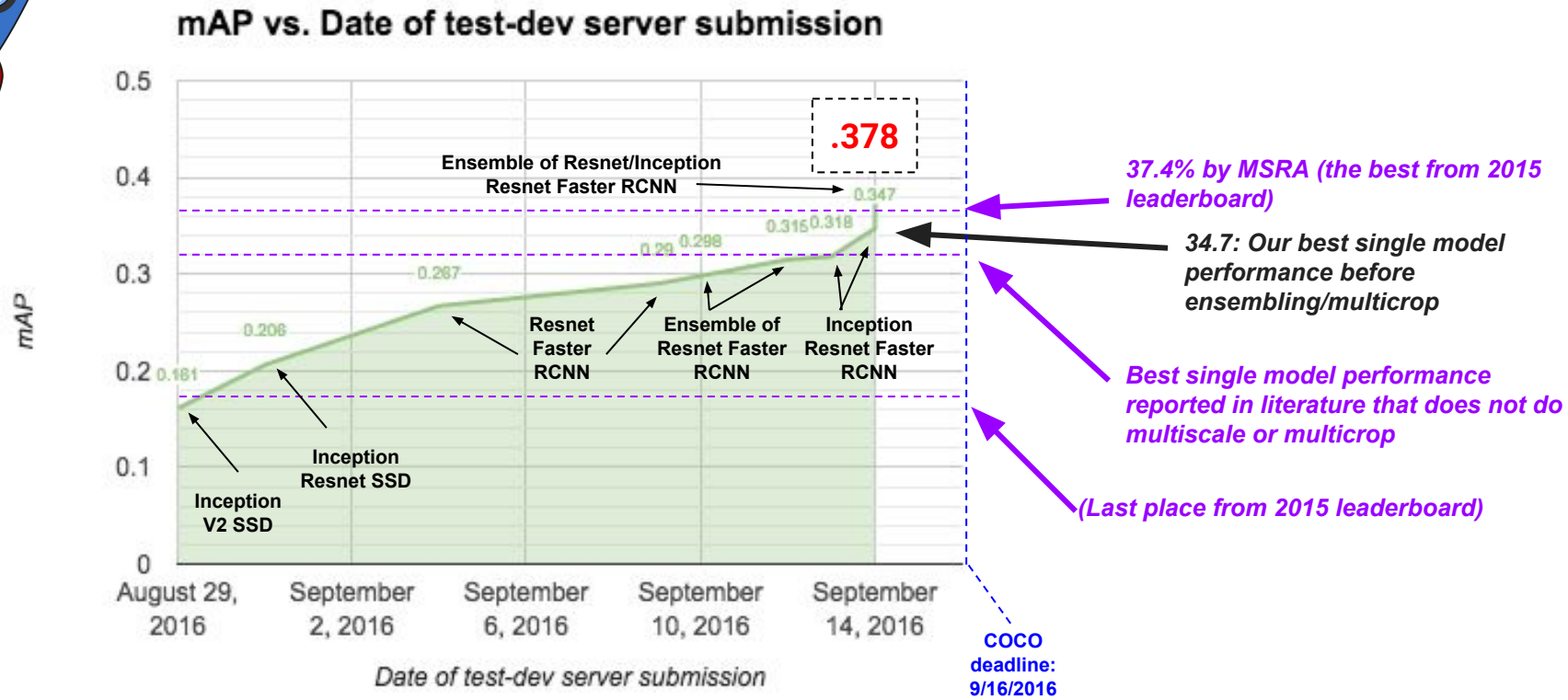


Inception Resnet Faster RCNN



Race to the Top

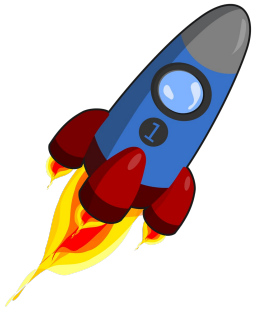


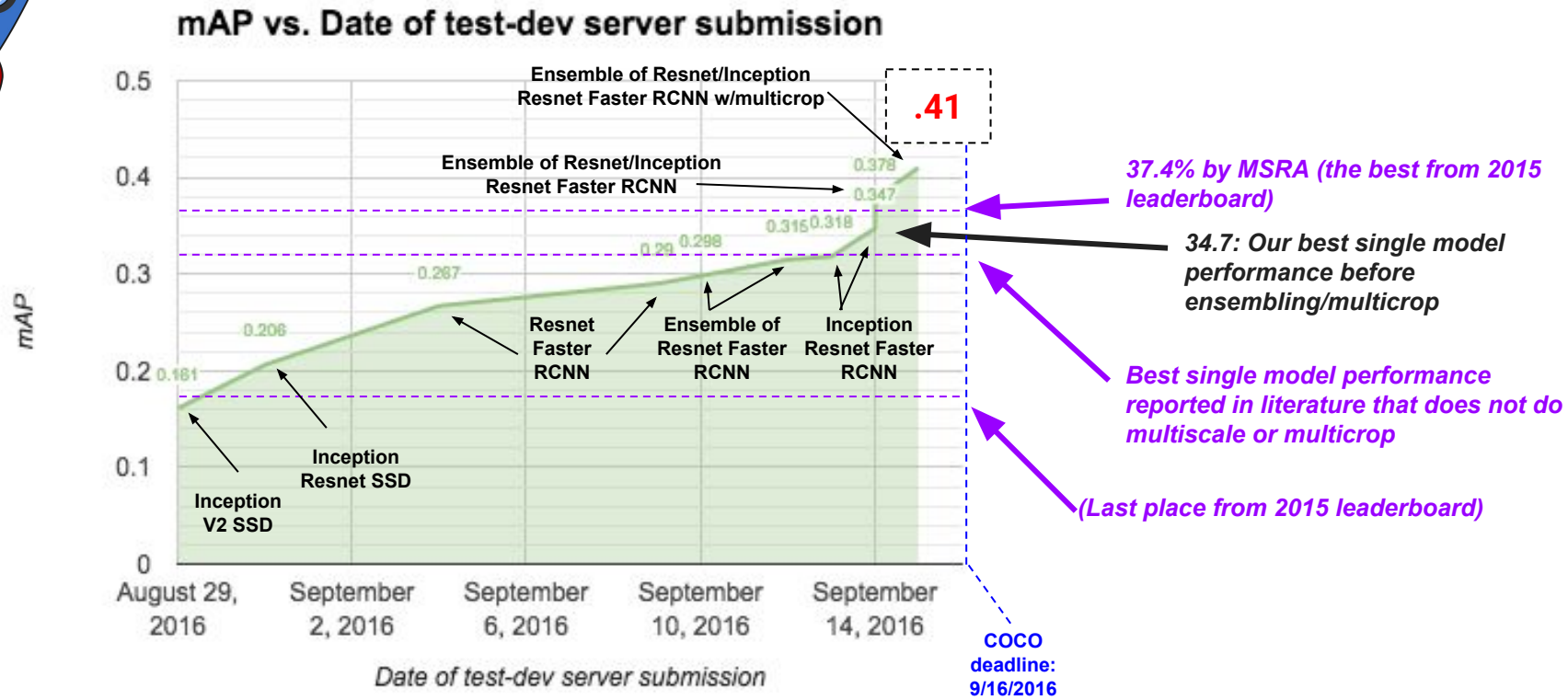


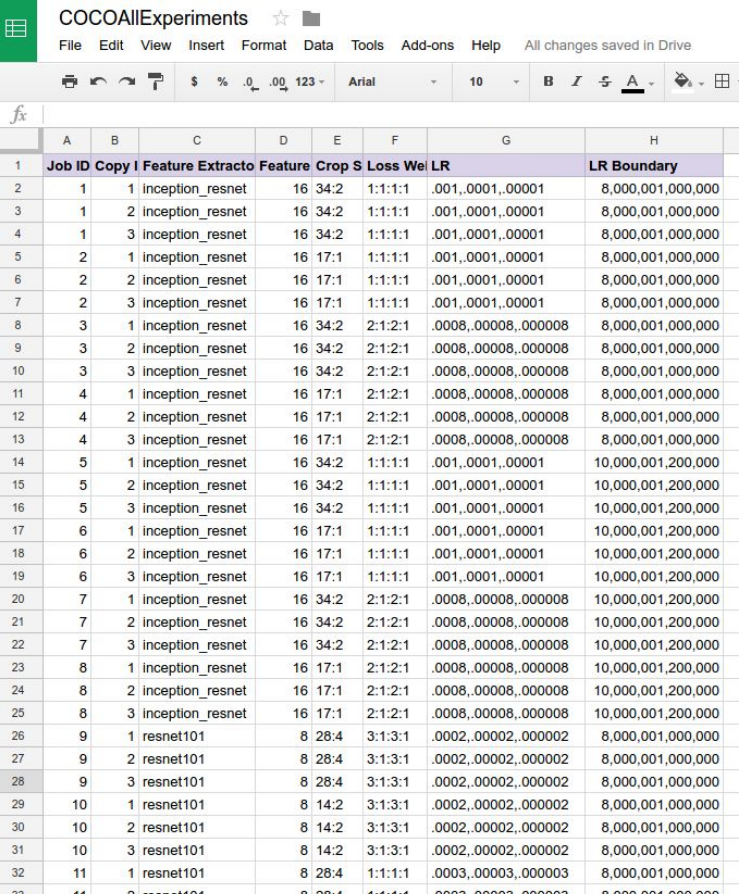
10-crop inference

No multiscale training, horizontal flip, box refinement, box voting, global context or ILSVRC detection data

Race to the Top






COCOAllExperiments

File Edit View Insert Format Data Tools Add-ons Help All changes saved in Drive

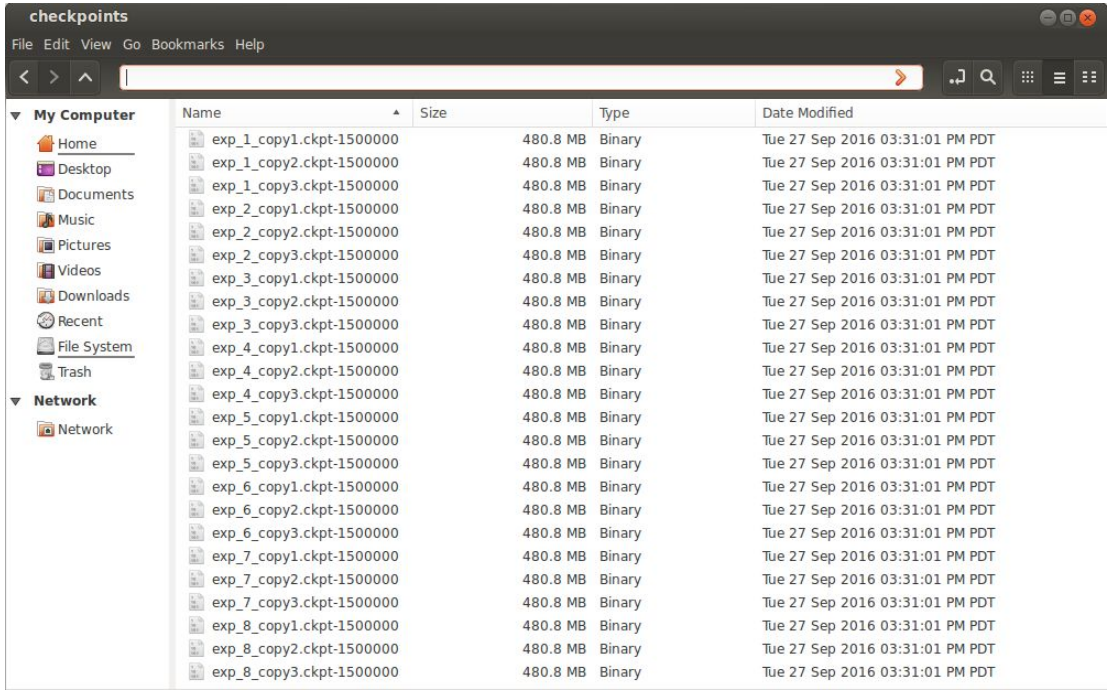
	A	B	C	D	E	F	G	H
1	Job ID	Copy I	Feature Extracto	Feature	Crop S	Loss We	LR	LR Boundary
2	1	1	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
3	1	2	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
4	1	3	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
5	2	1	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
6	2	2	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
7	2	3	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
8	3	1	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
9	3	2	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
10	3	3	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
11	4	1	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
12	4	2	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
13	4	3	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	8,000,001,000,000
14	5	1	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
15	5	2	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
16	5	3	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
17	6	1	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
18	6	2	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
19	6	3	inception_resnet	16	17:1	1:1:1:1	.001,.0001,.00001	10,000,001,200,000
20	7	1	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
21	7	2	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
22	7	3	inception_resnet	16	34:2	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
23	8	1	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
24	8	2	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
25	8	3	inception_resnet	16	17:1	2:1:2:1	.0008,.00008,.000008	10,000,001,200,000
26	9	1	resnet101	8	28:4	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
27	9	2	resnet101	8	28:4	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
28	9	3	resnet101	8	28:4	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
29	10	1	resnet101	8	14:2	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
30	10	2	resnet101	8	14:2	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
31	10	3	resnet101	8	14:2	3:1:3:1	.0002,.00002,.000002	8,000,001,000,000
32	11	1	resnet101	8	28:4	1:1:1:1	.0003,.00003,.000003	8,000,001,000,000

Many Final Experiments

- ❑ Best layer of Inception Resnet for RPN?
- ❑ Use atrous convolution for dense output?
- ❑ Maximize IOU vs Minimize SmoothL1?
- ❑ Best learning rate decay schedule?
- ❑ Best data augmentation operations (e.g. random cropping, random saturation, random contrast)?
- ❑ Best way to balance localization and classification loss functions?
- ❑ etc…

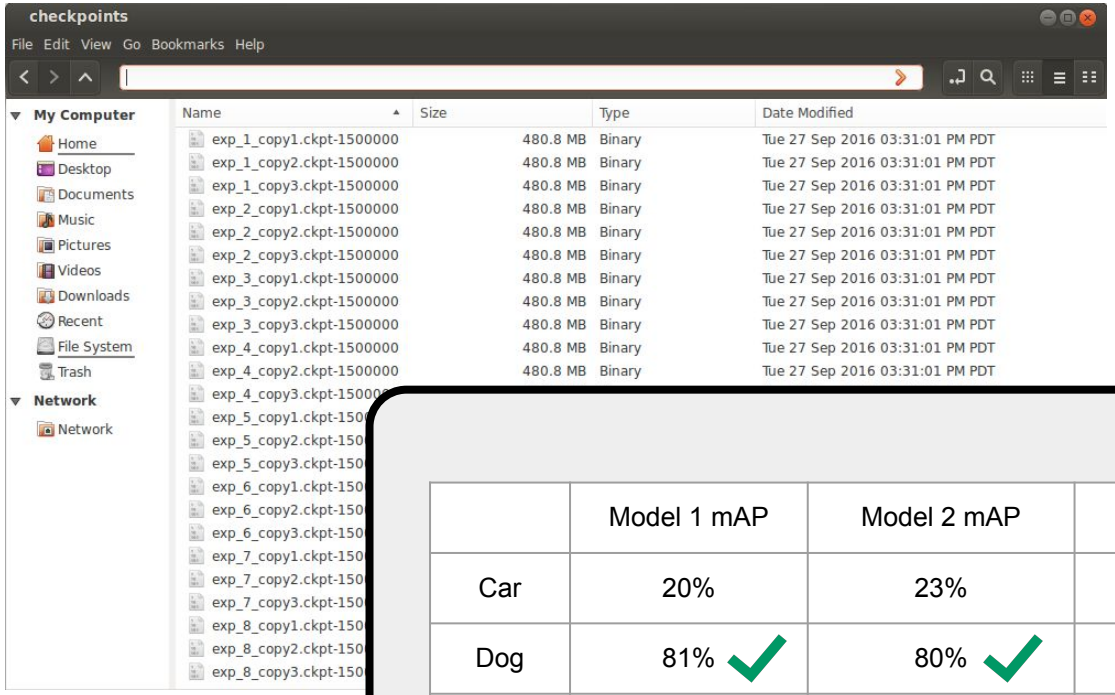
… led to *Model Checkpoint Overload* :(

Model Selection for Ensembling



Take best K models?
Or select diverse K-subset of models?

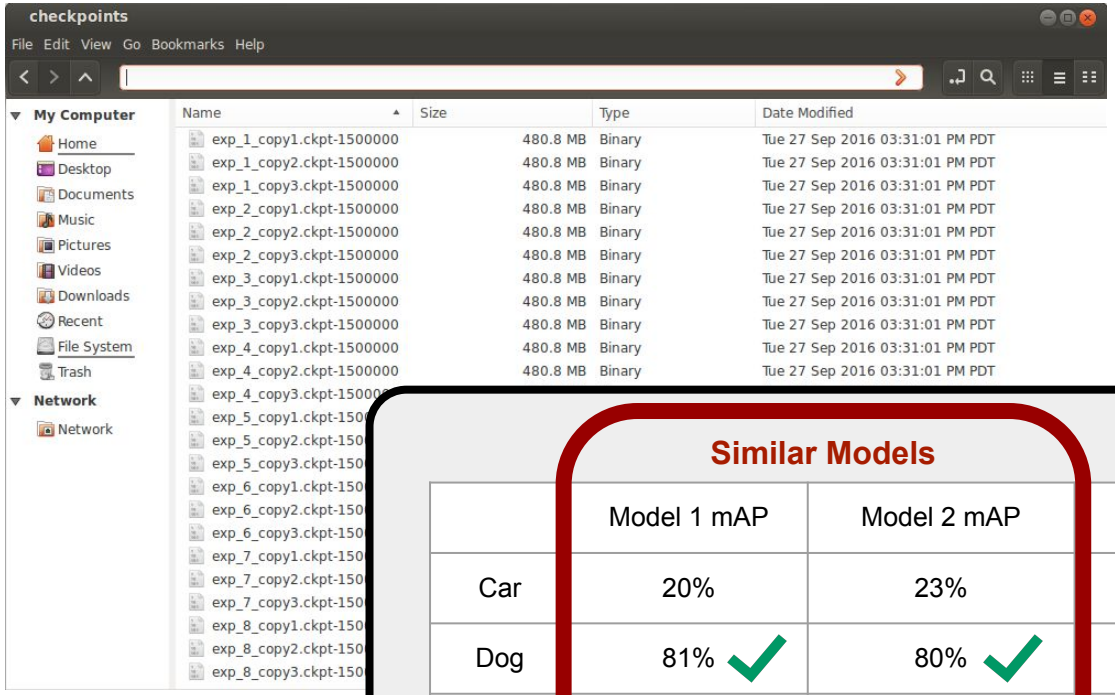
Model Selection for Ensembling



Take best K models?
Or select diverse K-subset of models?

	Model 1 mAP	Model 2 mAP	Model 3 mAP
Car	20%	23%	70% ✓
Dog	81% ✓	80% ✓	15%
Bear	78% ✓	81% ✓	20%
Chair	10%	12%	71% ✓

Model Selection for Ensembling

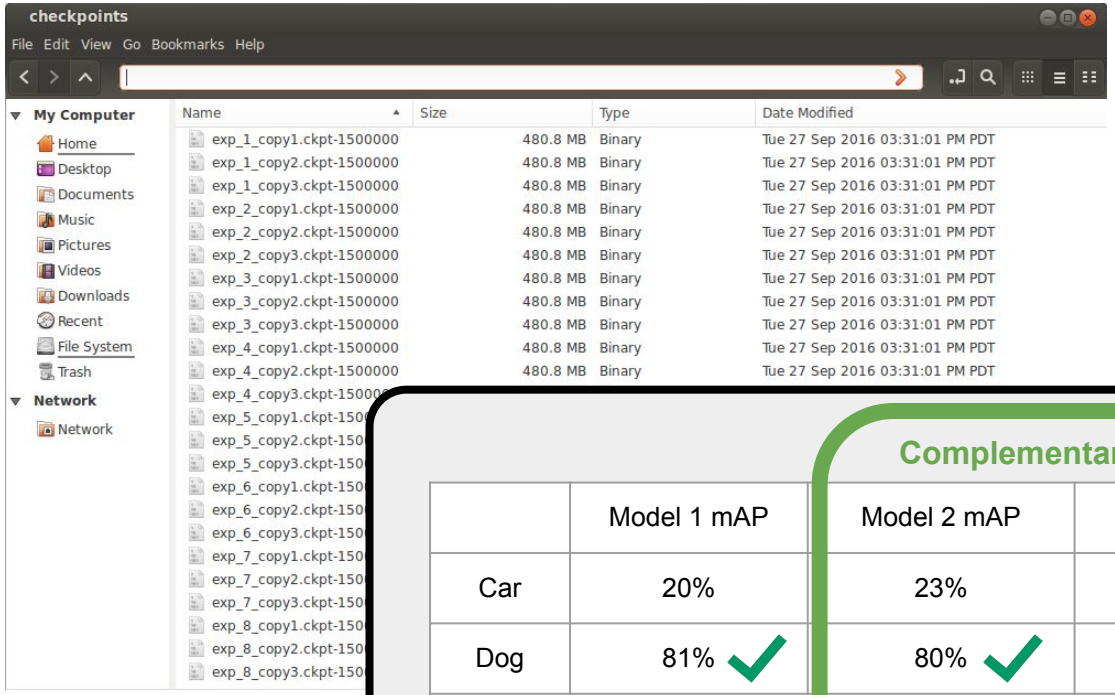


Take best K models?
Or select diverse K-subset of models?

Similar Models

	Model 1 mAP	Model 2 mAP	Model 3 mAP
Car	20%	23%	70% ✓
Dog	81% ✓	80% ✓	15%
Bear	78% ✓	81% ✓	20%
Chair	10%	12%	71% ✓

Model Selection for Ensembling



Take best K models?
Or select diverse K-subset of models?

Complementary Models

	Model 1 mAP	Model 2 mAP	Model 3 mAP
Car	20%	23%	70% ✓
Dog	81% ✓	80% ✓	15%
Bear	78% ✓	81% ✓	20%
Chair	10%	12%	71% ✓

Diversity Matters

***Model NMS* for diverse ensembling**: *Greedily select diverse model collection for ensembling, pruning away models too similar to already selected models.*

Final ensemble selected for challenge submission				
Individual mean AP (on minival)	**Feature Extractor**	**Output Stride**	**Location:Classification loss ratio**	**Location Loss function**
32.93	Resnet 101	8	3:1	SmoothL1
33.3	Resnet 101	8	1:1	SmoothL1
34.75	Inception Resnet	16	1:1	SmoothL1
35	Inception Resnet	16	2:1	SmoothL1+IOU
35.64	Inception Resnet	8	1:1	SmoothL1

Diversity Matters

***Model NMS* for diverse ensembling**: *Greedily select diverse model collection for ensembling, pruning away models too similar to already selected models.*

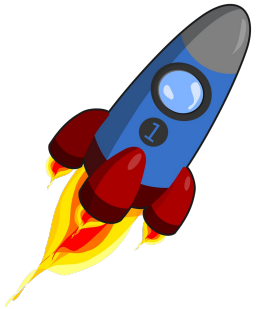
Final ensemble selected for challenge submission				
Individual mean AP (on minival)	**Feature Extractor**	**Output Stride**	**Location:Classification loss ratio**	**Location Loss function**
32.93	Resnet 101	8	3:1	SmoothL1
33.3	Resnet 101	8	1:1	SmoothL1
34.75	Inception Resnet	16	1:1	SmoothL1
35	Inception Resnet	16	2:1	SmoothL1+IOU
35.64	Inception Resnet	8	1:1	SmoothL1

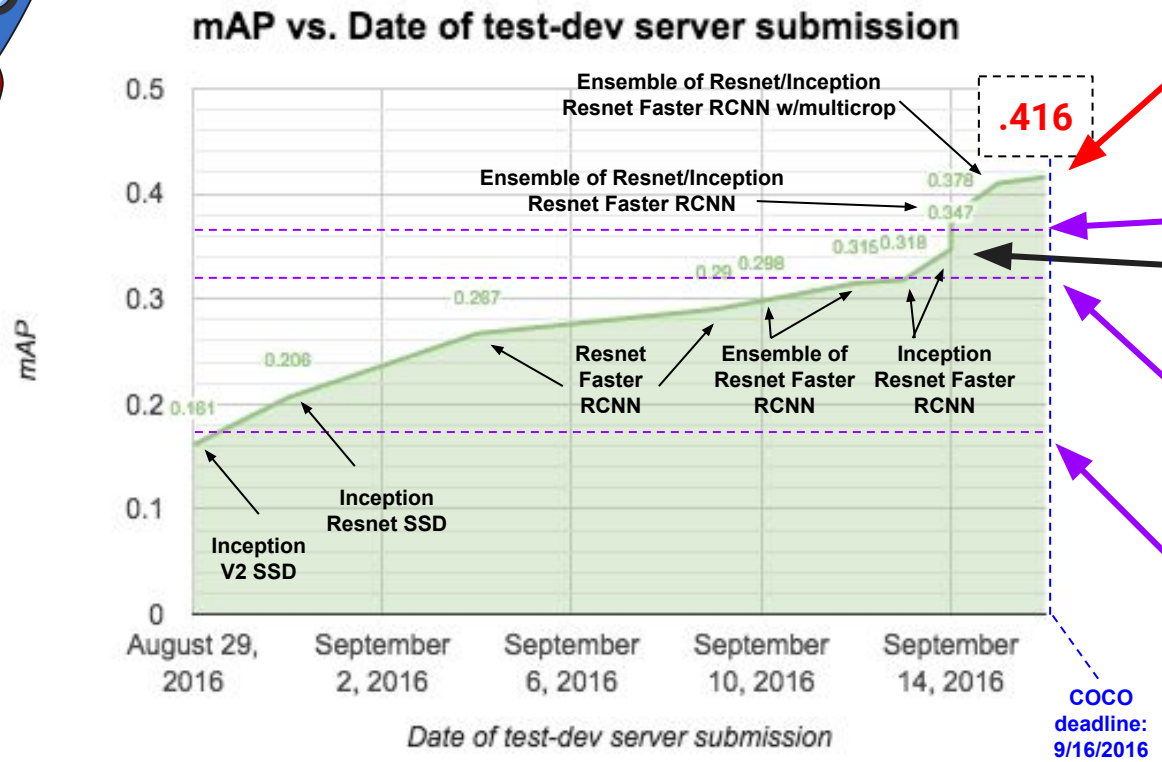
Diversity Matters

***Model NMS* for diverse ensembling**: *Greedily select diverse model collection for ensembling, pruning away models too similar to already selected models.*

Final ensemble selected for challenge submission				
Individual mean AP (on minival)	**Feature Extractor**	**Output Stride**	**Location:Classification loss ratio**	**Location Loss function**
32.93	Resnet 101	8	3:1	SmoothL1
33.3	Resnet 101	8	1:1	SmoothL1
34.75	Inception Resnet	16	1:1	SmoothL1
35	Inception Resnet	16	2:1	SmoothL1+IOU
35.64	Inception Resnet	8	1:1	SmoothL1

Race to the Top





41.6%: Last Google submission to test-dev server with Intelligently-Selected Ensemble of 5 Faster RCNN with Resnet and Inception-Resnet

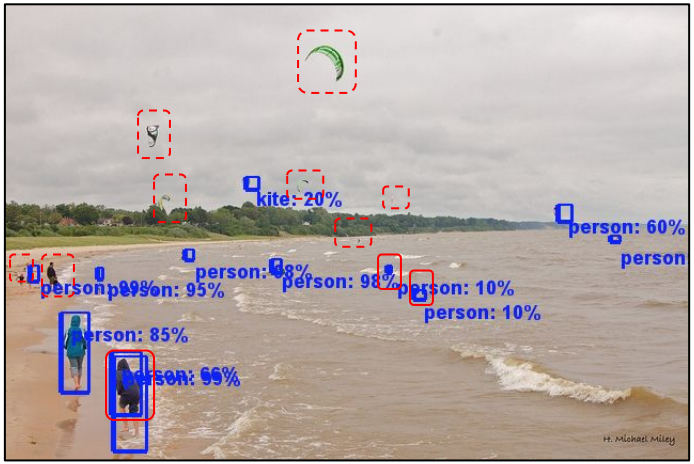
37.4% by MSRA (the best from 2015 leaderboard)

34.7: Our best single model performance before ensembling/multicrop

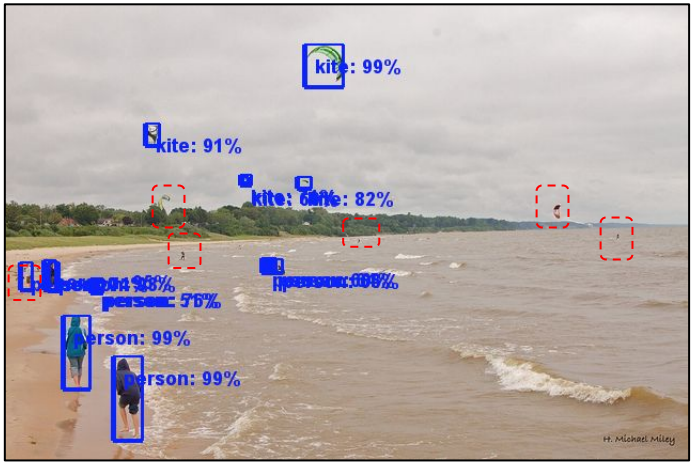
Best single model performance reported in literature that does not do multiscale or multicrop

(Last place from 2015 leaderboard)

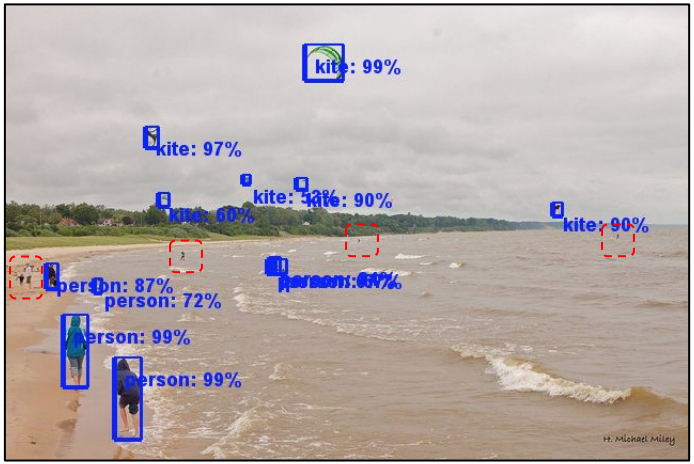
Inception Resnet SSD



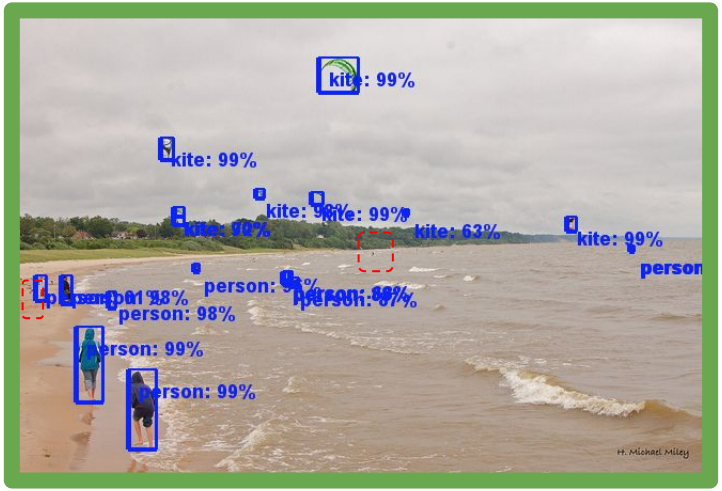
Resnet Faster RCNN



Inception Resnet Faster RCNN



Final ensemble with multicrop inference



Thanks!

SPEAKER
Jonathan Huang (jonathanhuang@google.com)

Object Detection Team
Alireza Fathi, Ian Fischer, Sergio Guadarrama, Jonathan Huang, Anoop Korattikara, Kevin Murphy, Vivek Rathod, Yang Song, Chen Sun, Zbigniew Wojna, Menglong Zhu

And Special Thanks to
Tom Duerig, Dumitru Erhan, Jitendra Malik, George Papandreou, Dominik Roblek, Chuck Rosenberg, Nathan Silberman, Abhinav Srivastava, Rahul Sukthankar, Christian Szegedy, Jasper Uijlings, Jay Yagnik, Xiangxin Zhu

Figure sources (links):

1. Raspberry pi image
2. Black and white kites
3. Kitesurfing on the beach