Object Detection @ Google

Jonathan Huang (jonathanhuang@google.com)

SIDAS 2017 Industry Keynote, 15 Sept 2017

Presenting the work of **many** people @Google

Google Research and Machine Intelligence

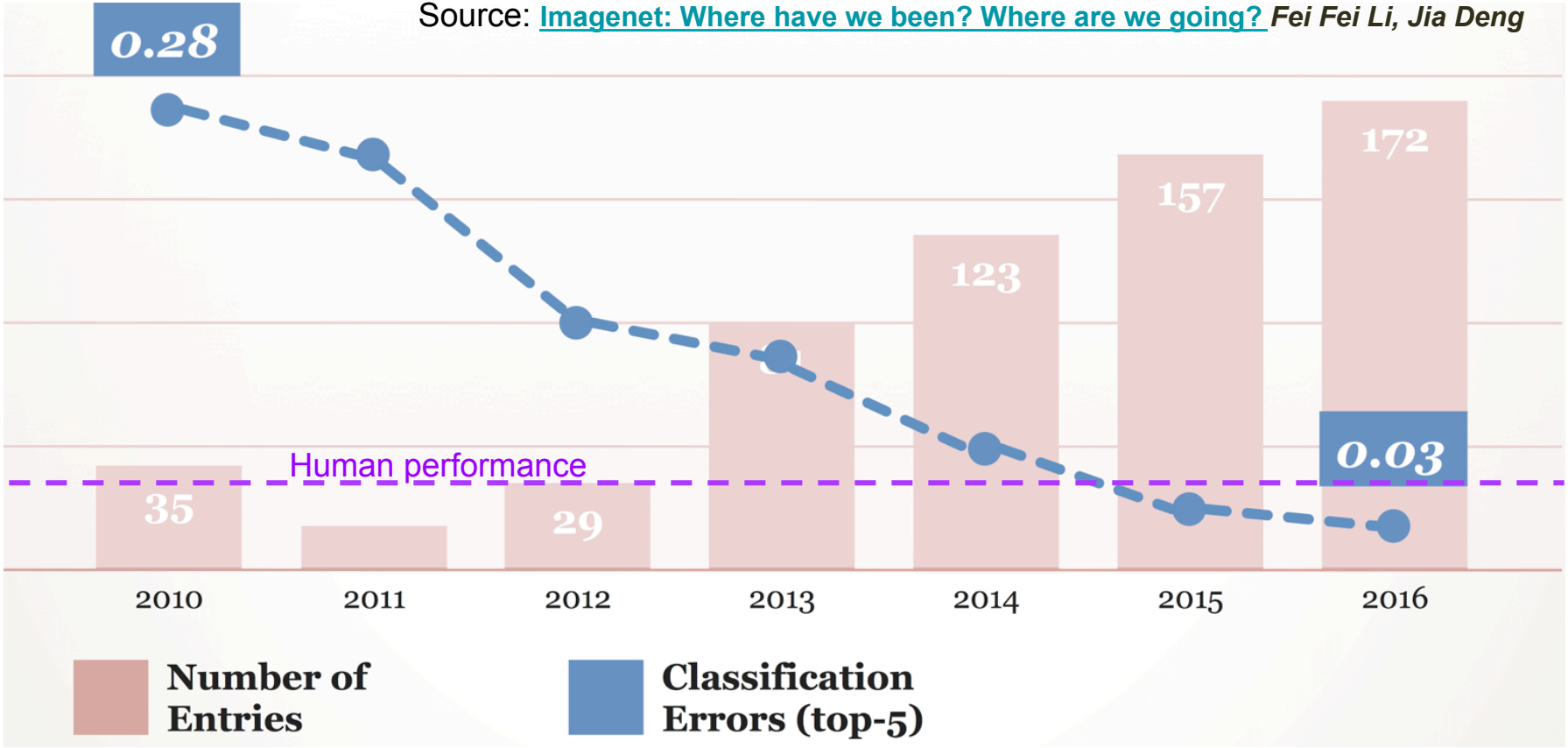
Imagenet Challenge

Given an image, predict one of 1000 different classes

Image credit: www.cs.toronto.edu/~fritz/absps/imagenet.pdf

Imagenet Progress Over the Years

Source: Imagenet: Where have we been? Where are we going? *Fei Fei Li, Jia Deng*



IS THIS A

CAT or DOG?

CAT DOG

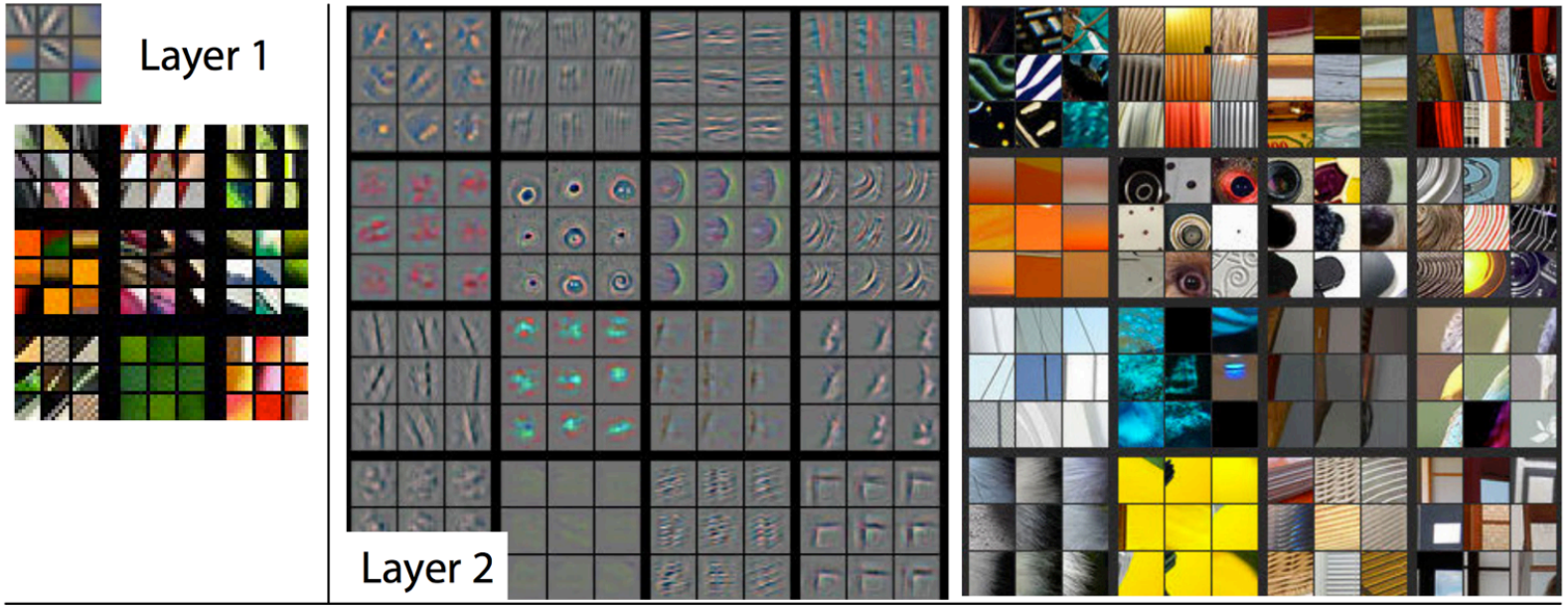
OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

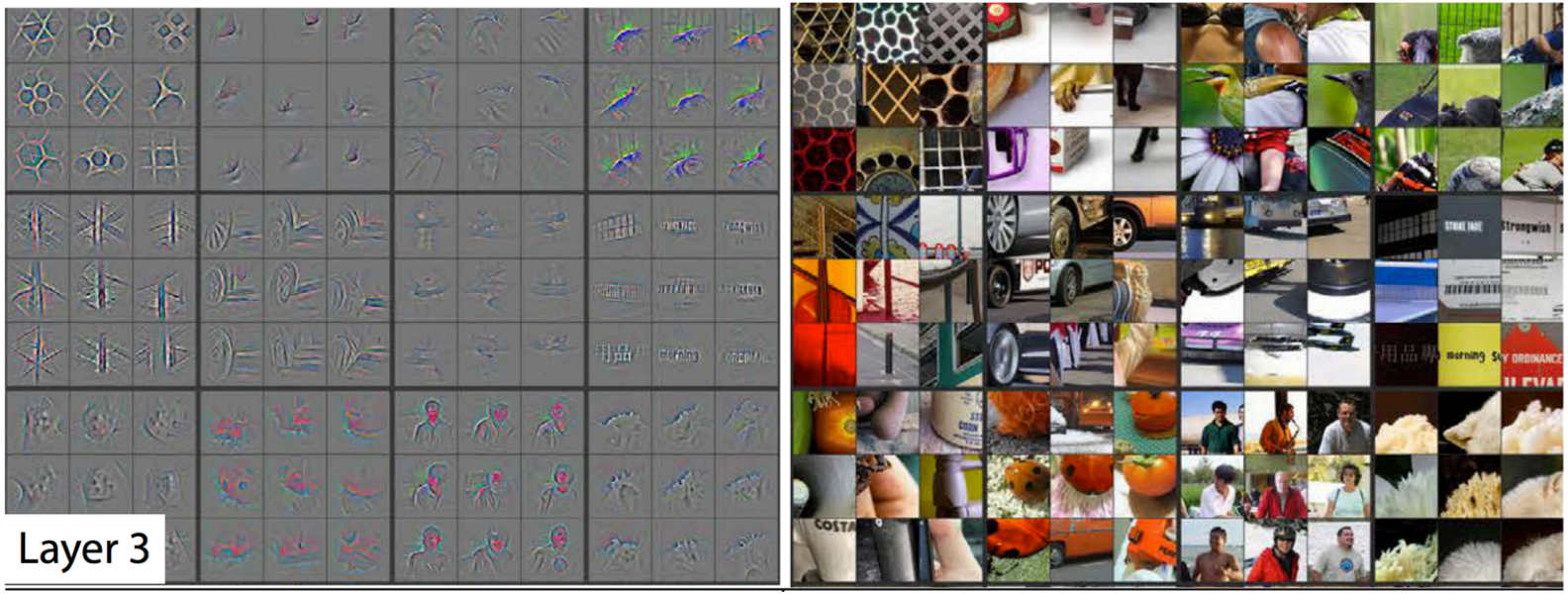
DEEP *neural* NETWORK

Convolutions, Sliding Windows, Patterns!



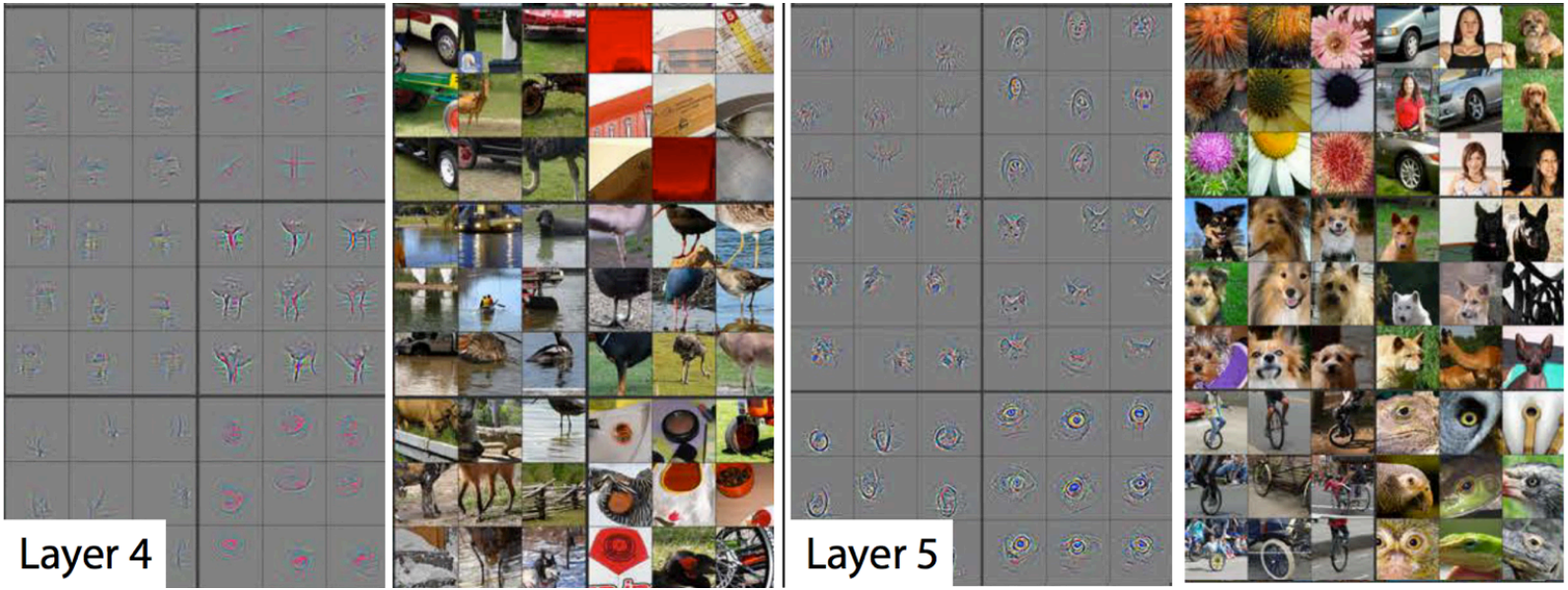
Source: [Visualizing and Understanding Convolutional Networks](#) *M.D. Zeiler, R. Fergus*

Convolutions, Sliding Windows, Patterns!



Source: [Visualizing and Understanding Convolutional Networks](#) *M.D. Zeiler, R. Fergus*

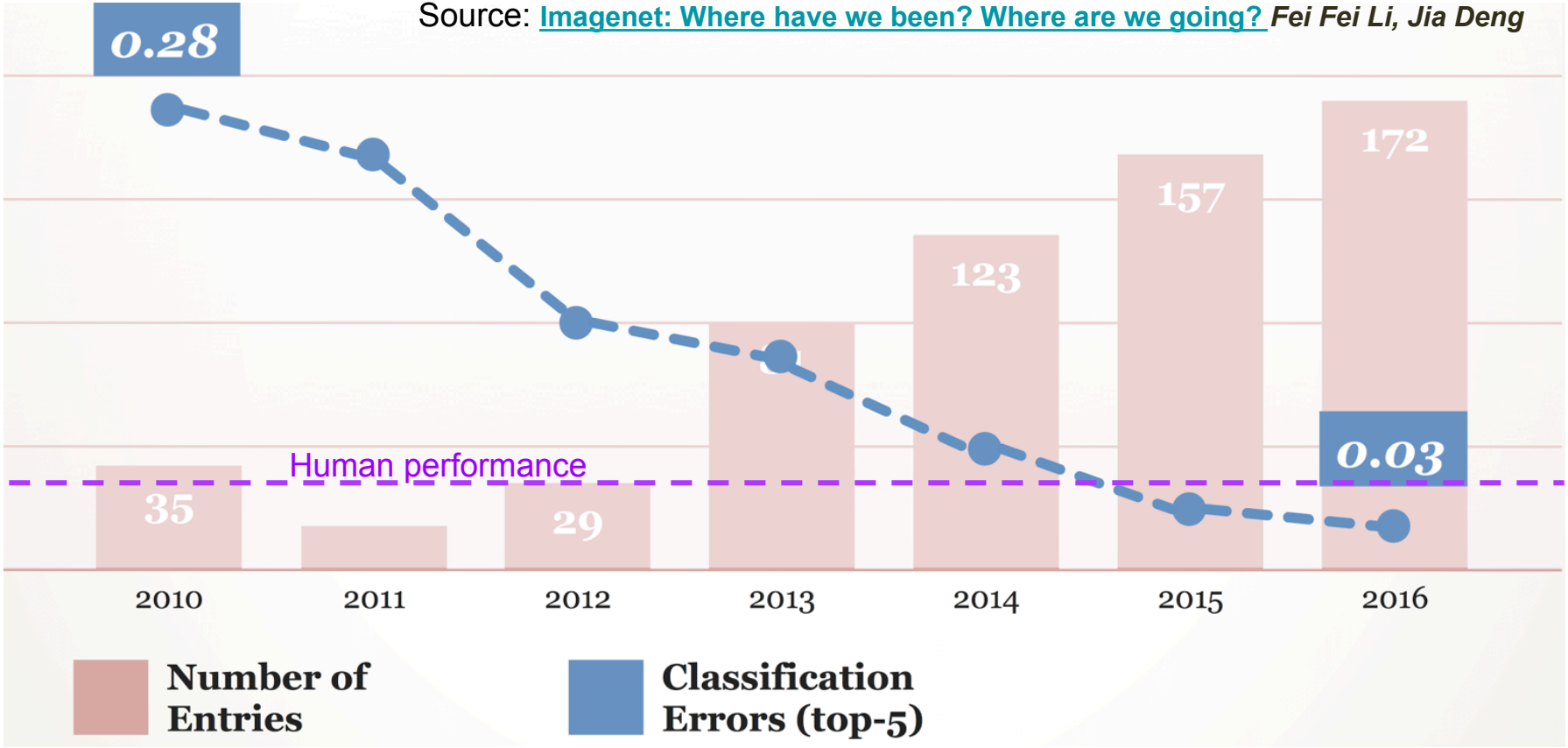
Convolutions, Sliding Windows, Patterns!



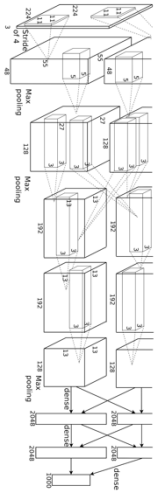
Source: [Visualizing and Understanding Convolutional Networks](#) *M.D. Zeiler, R. Fergus*

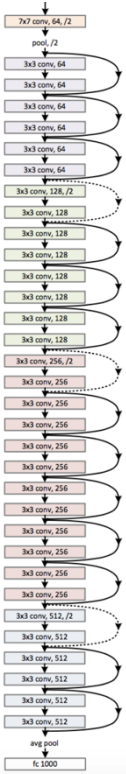
Imagenet Progress Over the Years

Source: Imagenet: Where have we been? Where are we going? *Fei Fei Li, Jia Deng*



“Famous Architectures”





AlexNet
[Krizhevsky et al, 2012]

VGG
[Simonyan and Zisserman, 2014]

Inception
[Szegedy et al, 2014]

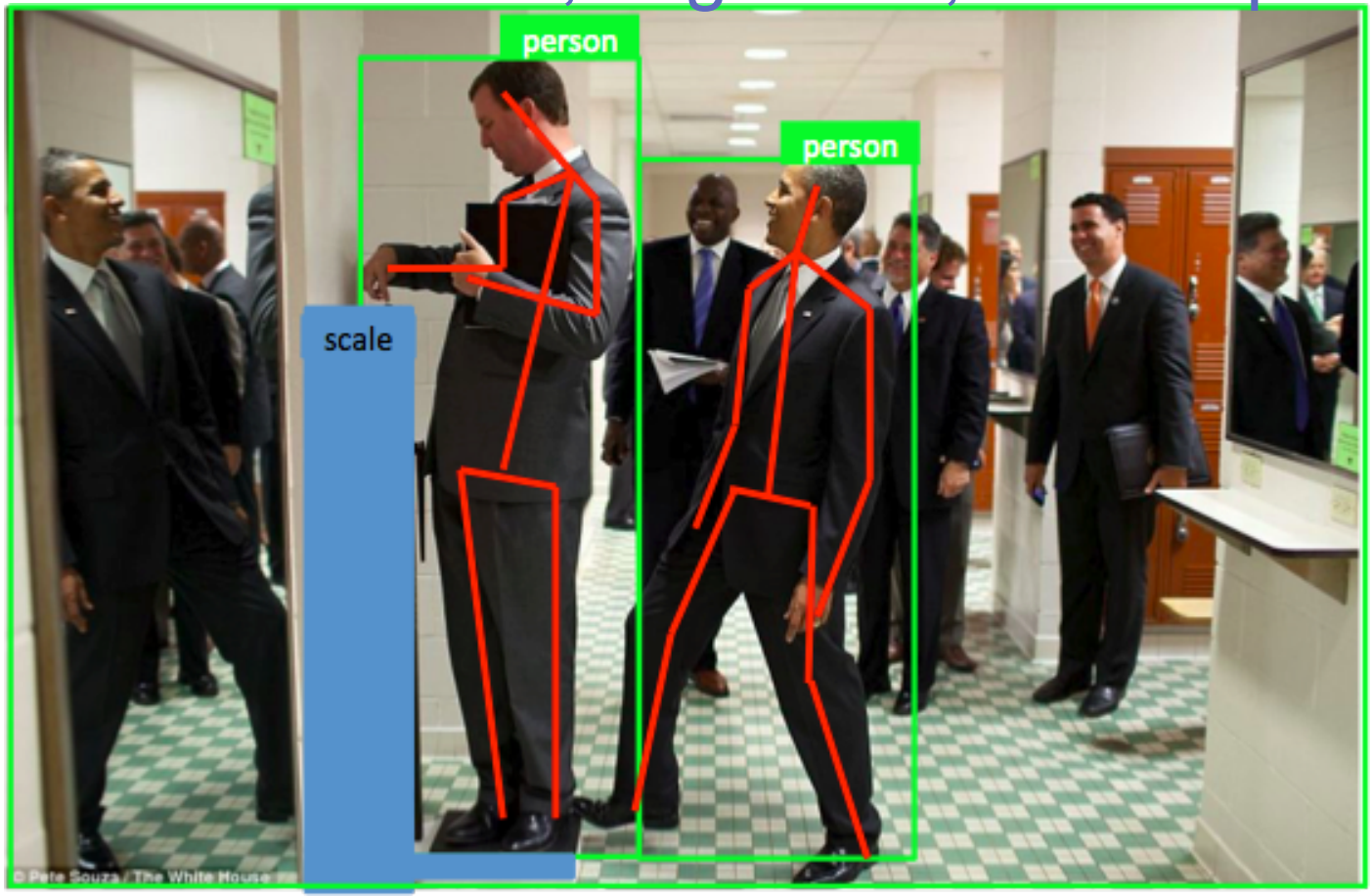
ResNet
[He et al, 2015]

Inception Resnet
[Szegedy et al, 2015]

Today's image taggers just return a bag of words...

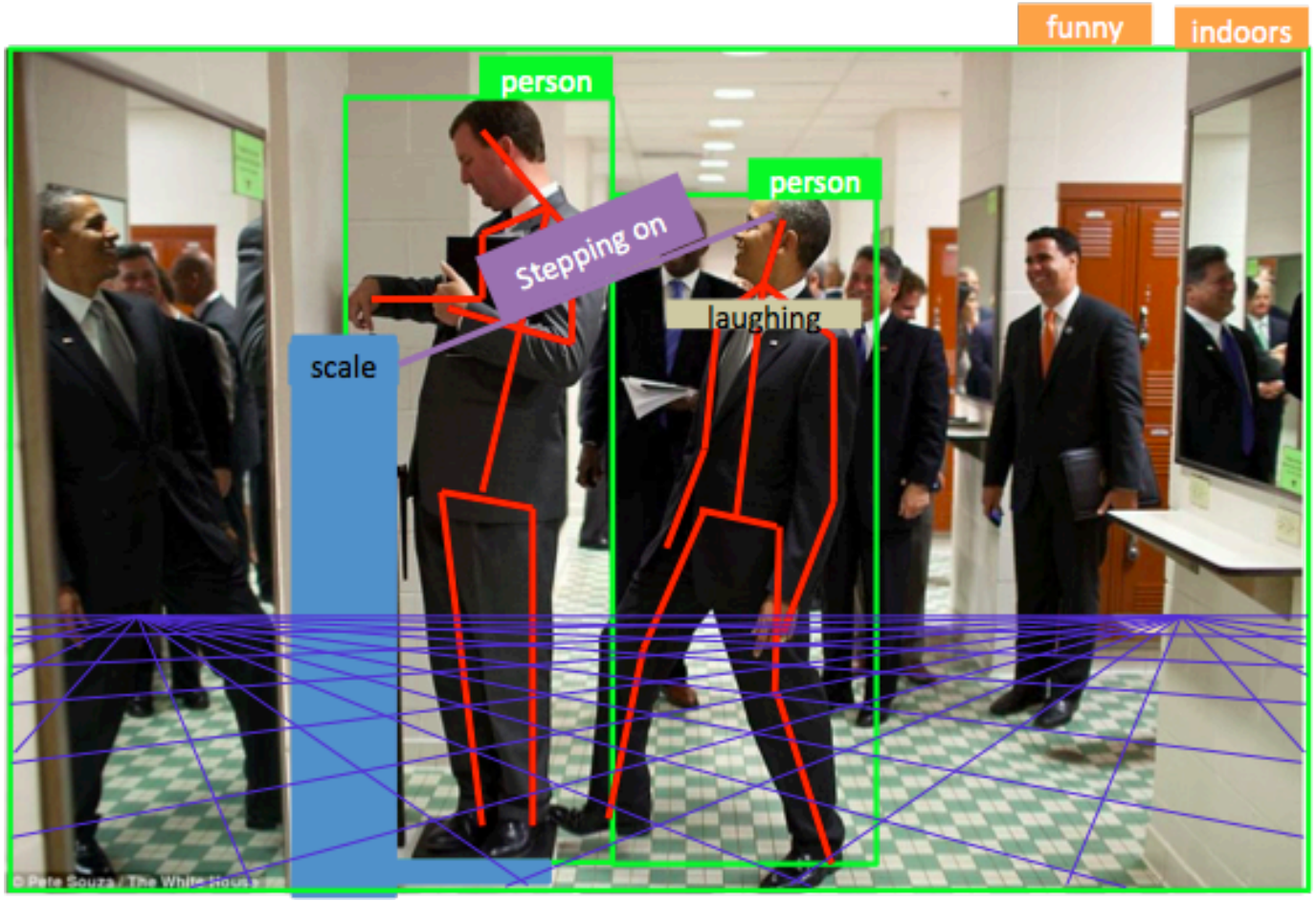
Source: http://karpathy.github.io/2012/10/22/state-of-computer-vision/

What we need: boxes, segments, human pose...

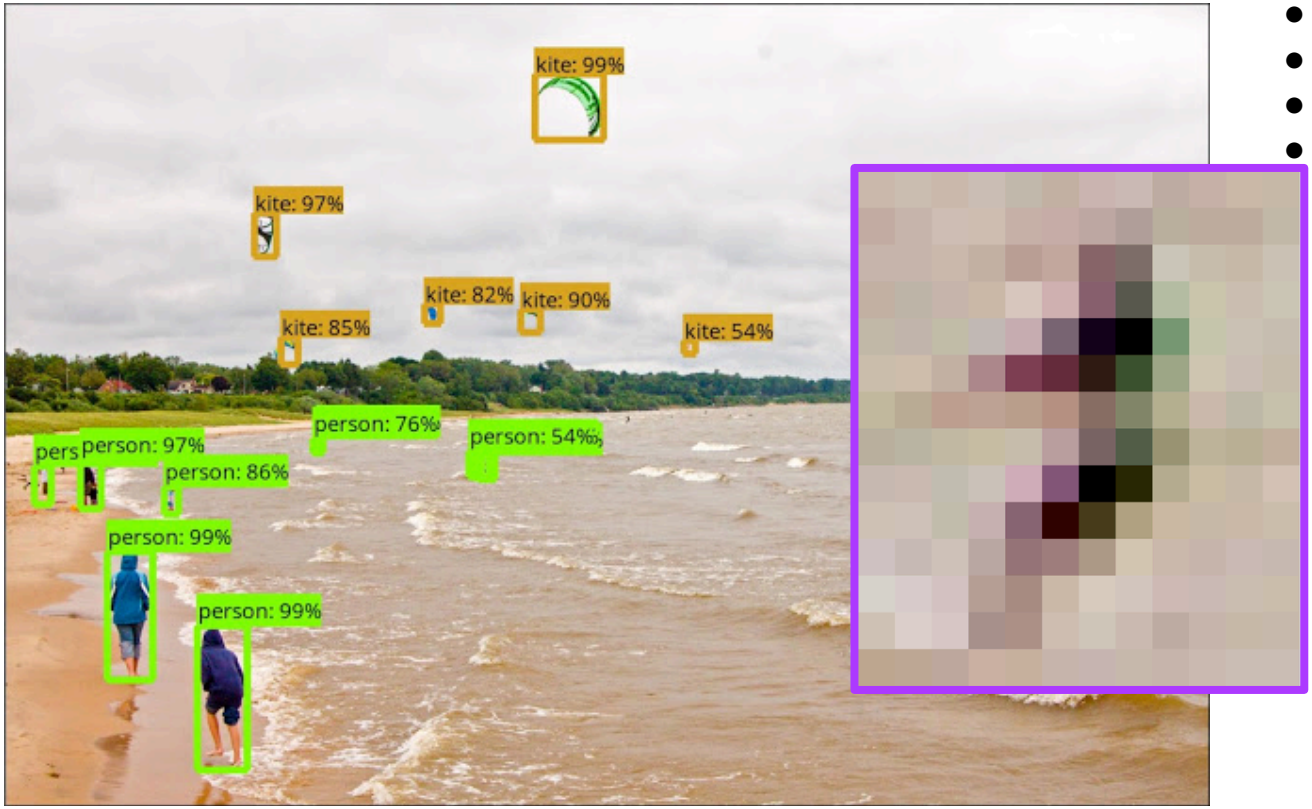


Based on a figure from Jia Deng

And finally: attributes, relations, 3-d!



From Classification to Detection

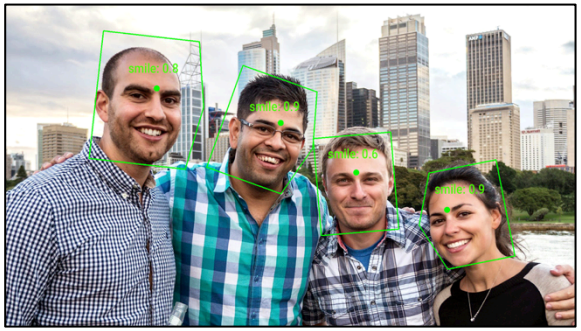


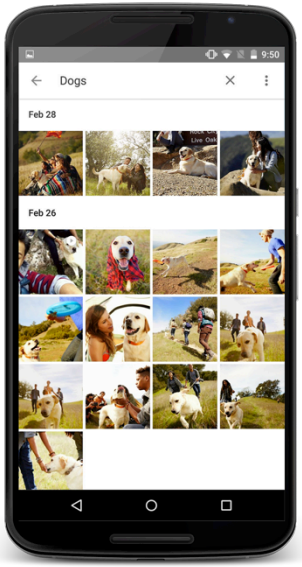
Challenges

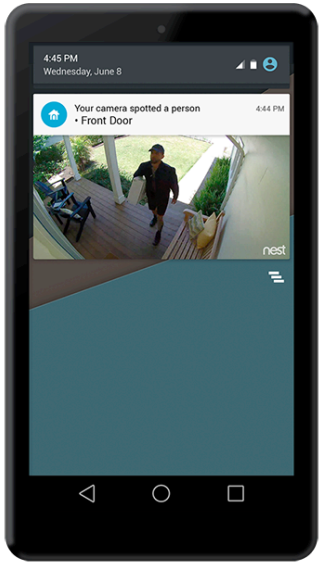
- Variable # outputs
- Localization
- Running time
- Even with perfect localization, need to classify based on much fewer pixels than in Imagenet setting, requires context!

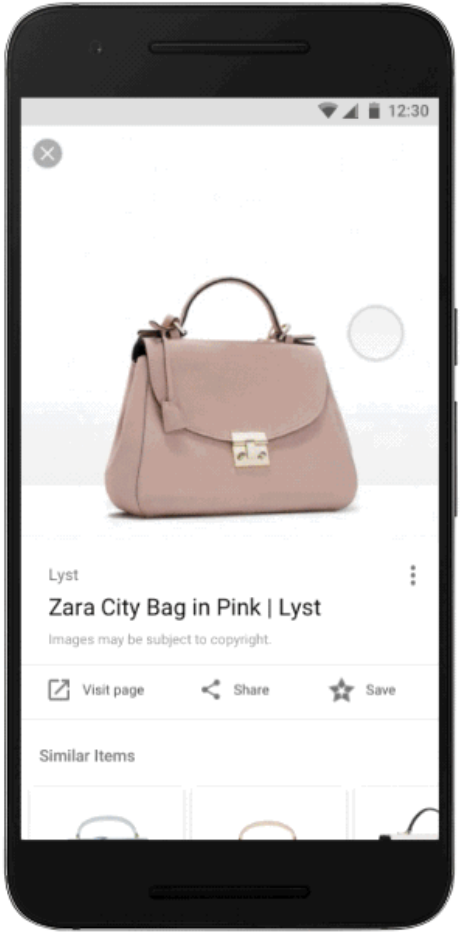
Photo credit: Michael Mina

Object Detection @ Google

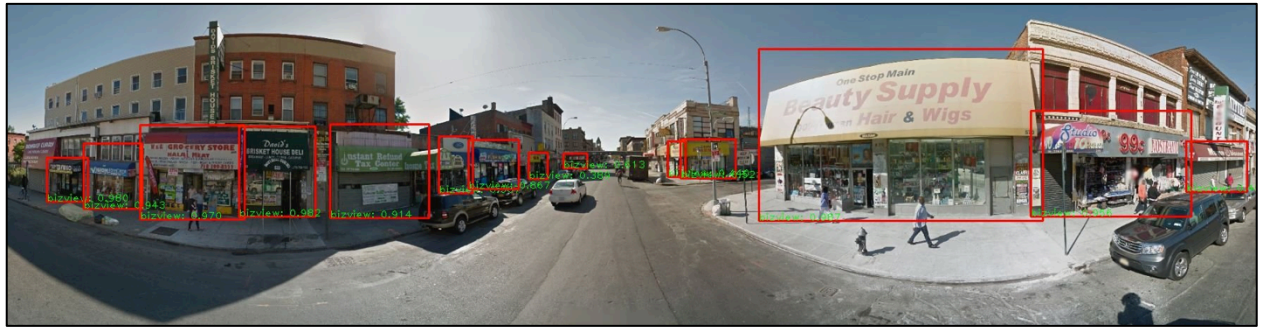








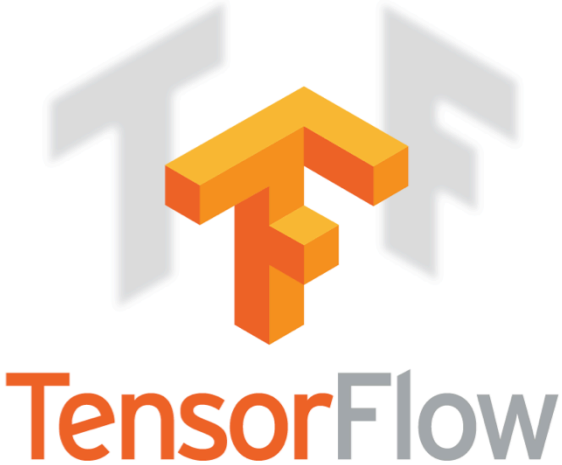
Object Detection @ Google



This Talk

- **The Tensorflow Object Detection API**
- Pushing the envelope on Accuracy
  - Better Models
  - Better Data
- Pushing the envelope on Speed
  - Lightweight Models
  - Adaptive Inference

Object Detection in Tensorflow



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


Your laptop


Datacenters


Mobile

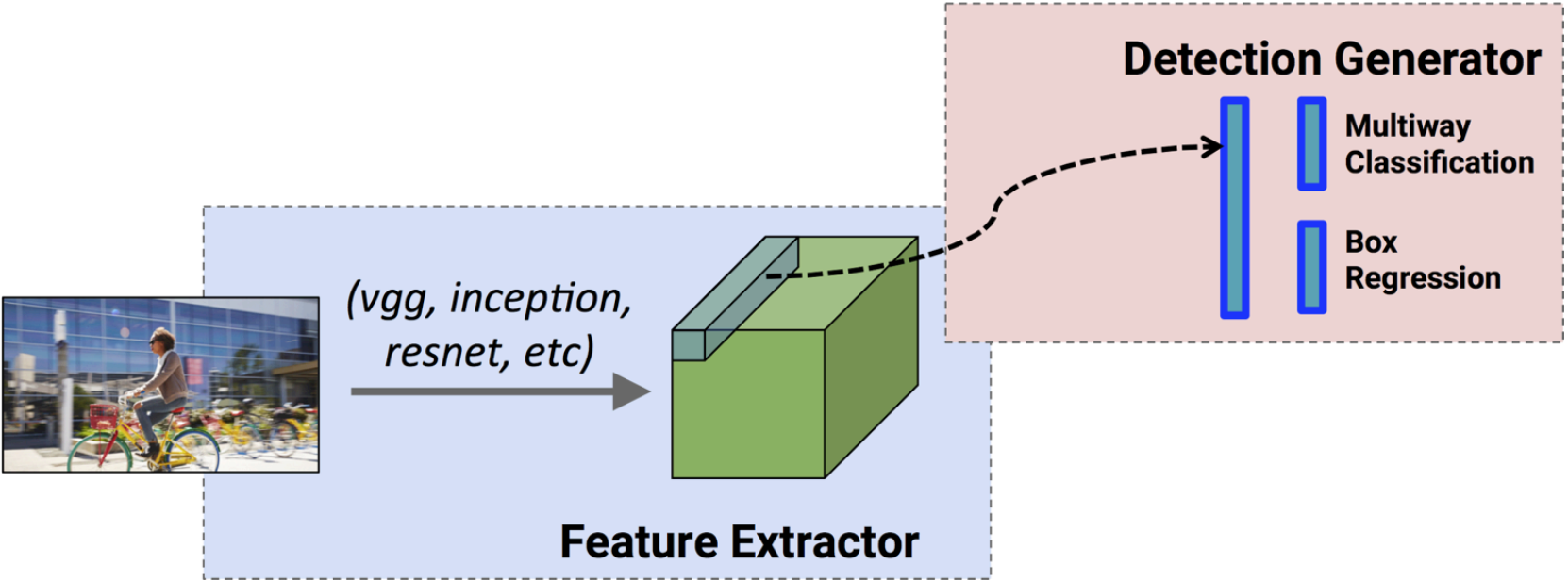

Raspberry Pi


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...

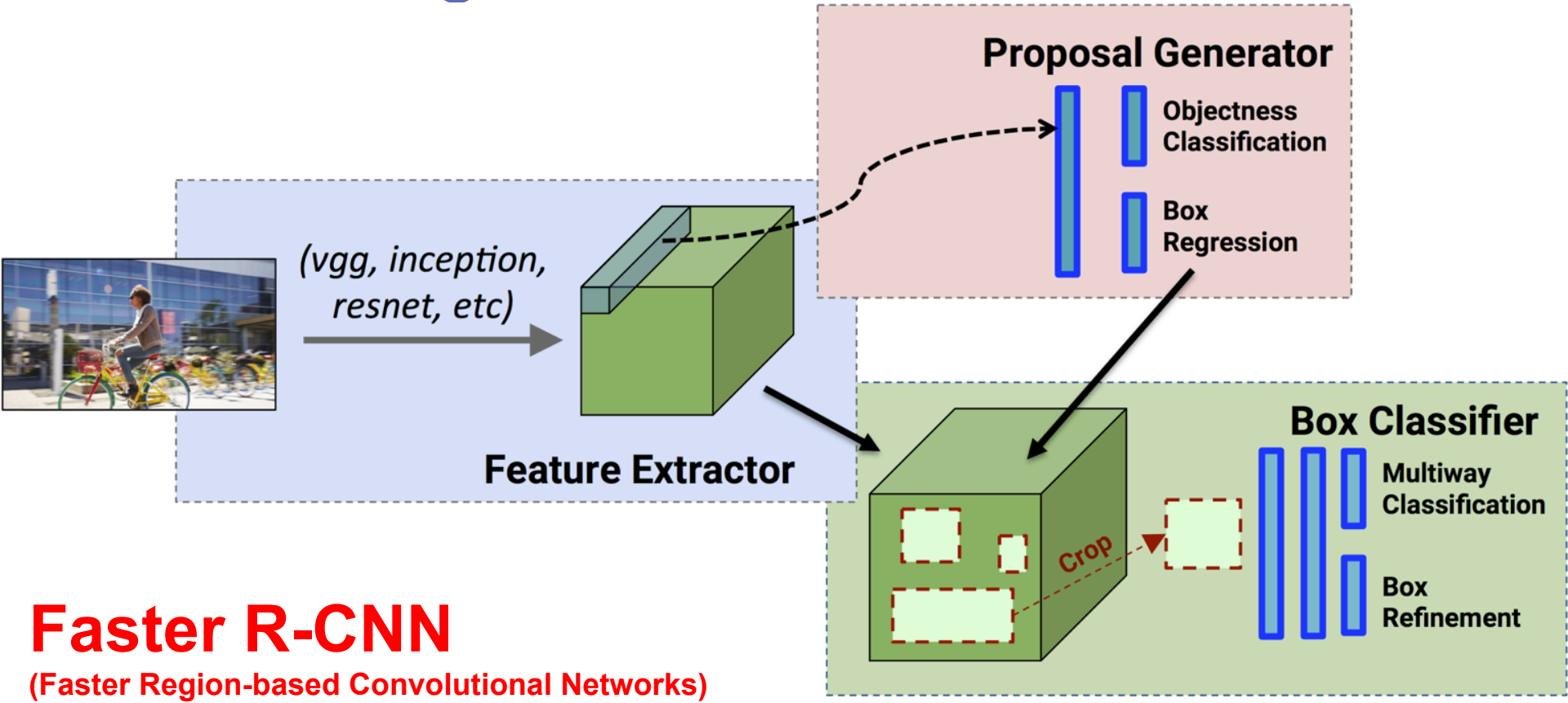
A convergence of architectures



SSD

(Single Shot Detector --- encapsulates Multibox, YOLO, YOLO v2)

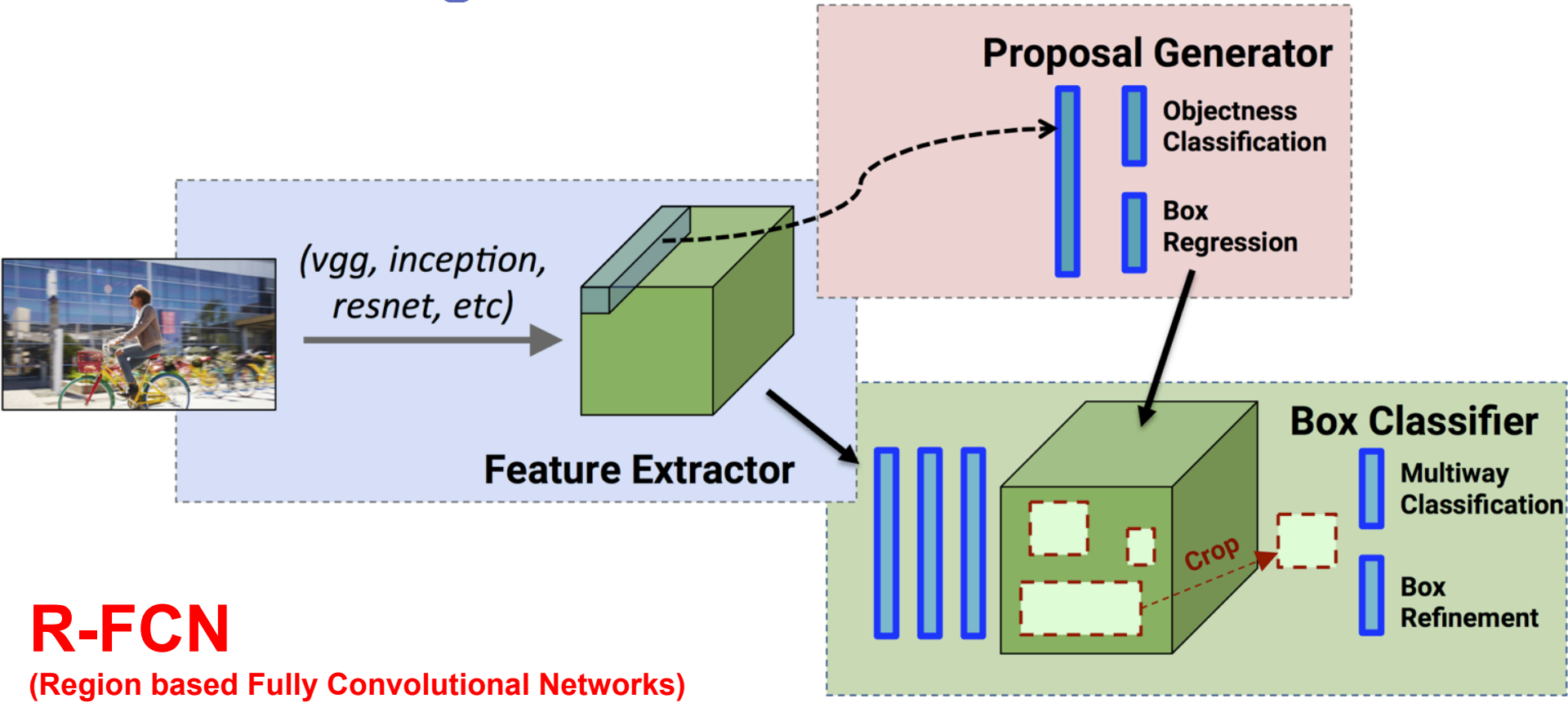
A convergence of architectures



Faster R-CNN

(Faster Region-based Convolutional Networks)

A convergence of architectures



R-FCN

(Region based Fully Convolutional Networks)

Recent works in the detection literature

Paper	Meta-architecture	Feature Extractor	Matching	Box Encoding $\phi(b_a, a)$	Location Loss functions
Szegedy et al. [40]	SSD	InceptionV3	Bipartite	$[x_0, y_0, x_1, y_1]$	L_2
Redmon et al. [29]	SSD	Custom (GoogLeNet inspired)	Box Center	$[x_c, y_c, \sqrt{w}, \sqrt{h}]$	L_2
Ren et al. [31]	Faster R-CNN	VGG	Argmax	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	SmoothL_1
He et al. [13]	Faster R-CNN	ResNet-101	Argmax	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	SmoothL_1
Liu et al. [26] (v1)	SSD	InceptionV3	Argmax	$[x_0, y_0, x_1, y_1]$	L_2
Liu et al. [26] (v2, v3)	SSD	VGG	Argmax	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	SmoothL_1
Dai et al [6]	R-FCN	ResNet-101	Argmax	$[\frac{x_c}{w_a}, \frac{y_c}{h_a}, \log w, \log h]$	SmoothL_1

But which is the best?

Answer: there is no best

TensorFlow Object Detection API

Meta-architecture	SSD, Faster R-CNN, R-FCN
Feature Extractor	Mobilenet, VGG, Inception V2, Inception V3, Resnet-50, Resnet-101, Resnet-152, Inception Resnet v2
Learning schedule	Manually Stepped, Exponential Decay, etc
Location Loss function	L2, L1, Huber, IOU
Classification Loss function	SigmoidCrossEntropy, SoftmaxCrossEntropy

CONFIG_PATH="tensorflow/models/samples/configs/faster_rcnn_resnet101_pets.config"

Model parameters

```
model {
  faster_rcnn {
    num_classes: 37
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
    first_stage_objectness_loss_weight: 1.0
```

37 dog/cat breeds

high resolution input images

Faster R-CNN, Resnet 101

Train/Eval parameters

```
train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED/model.ckpt"
  from_detection_checkpoint: true
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

eval_config: {
  num_examples: 2000
}
```

manually stepped learning rate schedule

Detection checkpoint pretrained on COCO

Data sources

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED/pet_train.record"
  }
  label_map_path: "PATH_TO_BE_CONFIGURED/
pet_label_map.pbtxt"
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED/pet_val.record"
  }
  label_map_path: "PATH_TO_BE_CONFIGURED/
pet_label_map.pbtxt"
}
```

CONFIG_PATH="tensorflow/models/samples/configs/faster_rcnn_resnet101_**pascal**.config"

Model parameters

```
model {
  faster_rcnn {
    num_classes: 20
    image_resizer {
      keep_aspect_ratio_resizer {
        min_dimension: 600
        max_dimension: 1024
      }
    }
    feature_extractor {
      type: 'faster_rcnn_resnet101'
      first_stage_features_stride: 16
    }
    first_stage_anchor_generator {
      grid_anchor_generator {
        scales: [0.25, 0.5, 1.0, 2.0]
        aspect_ratios: [0.5, 1.0, 2.0]
        height_stride: 16
        width_stride: 16
      }
    }
    first_stage_box_predictor_conv_hyperparams {
      op: CONV
      regularizer {
        l2_regularizer {
          weight: 0.0
        }
      }
      initializer {
        truncated_normal_initializer {
          stddev: 0.01
        }
      }
    }
    first_stage_nms_score_threshold: 0.0
    first_stage_nms_iou_threshold: 0.7
    first_stage_max_proposals: 300
    first_stage_localization_loss_weight: 2.0
    first_stage_objectness_loss_weight: 1.0
```

Train/Eval parameters

```
train_config: {
  batch_size: 1
  optimizer {
    momentum_optimizer: {
      learning_rate: {
        manual_step_learning_rate {
          initial_learning_rate: 0.0003
          schedule {
            step: 0
            learning_rate: .0003
          }
          schedule {
            step: 900000
            learning_rate: .00003
          }
          schedule {
            step: 1200000
            learning_rate: .000003
          }
        }
      }
      momentum_optimizer_value: 0.9
    }
    use_moving_average: false
  }
  gradient_clipping_by_norm: 10.0
  fine_tune_checkpoint: "PATH_TO_BE_CONFIGURED/model.ckpt"
  from_detection_checkpoint: true
  data_augmentation_options {
    random_horizontal_flip {
    }
  }
}

eval_config: {
  num_examples: 2000
}
```

Data sources

```
train_input_reader: {
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED/pascal_train.record"
  }
  label_map_path: "PATH_TO_BE_CONFIGURED/
pascal_label_map.pbtxt"
}

eval_input_reader: {
  tf_record_input_reader {
    input_path: "PATH_TO_BE_CONFIGURED/pascal_val.record"
  }
  label_map_path: "PATH_TO_BE_CONFIGURED/
pascal_label_map.pbtxt"
}
```

This repository Search Pull requests Issues Marketplace Gist

tensorflow / models

Unwatch 1,311 Unstar 17,218 Fork 6,704

Code Issues 196 Pull requests 29 Projects 0 Wiki Insights

Branch: master models / object_detection / Create new file Upload files Find file History

derekjchow committed with sguada Make Record scripts python3 compatible. (#1614) Latest commit 057203e 2 hours ago

..		
anchor_generators	Add Tensorflow Object Detection API. (#1561)	6 days ago
box_coders	Add Tensorflow Object Detection API. (#1561)	6 days ago
builders	Fix compatibility for model_builder_test.py (#1571)	4 days ago
core	Add Tensorflow Object Detection API. (#1561)	6 days ago
data	Add Tensorflow Object Detection API. (#1561)	6 days ago
data_decoders	Add Tensorflow Object Detection API. (#1561)	6 days ago
g3doc	Fix ML Engine Dashboard link (#1599)	a day ago
matchers	Add Tensorflow Object Detection API. (#1561)	6 days ago
meta_architectures	Add Tensorflow Object Detection API. (#1561)	6 days ago
models	Use spatial_squeeze=False for ResNet feature extractors. (#1586)	4 days ago
protos	Add Tensorflow Object Detection API. (#1561)	6 days ago
samples	Reduce batchsize from 32->24 for SSD configs.	5 days ago
test_images	Add Tensorflow Object Detection API. (#1561)	6 days ago
utils	Change visualizer font and jupyter notebook line thickness (#1589)	4 days ago
BUILD	Add Tensorflow Object Detection API. (#1561)	6 days ago
CONTRIBUTING.md	Add Tensorflow Object Detection API. (#1561)	6 days ago
README.md	Clean up documentation. (#1563)	5 days ago
__init__.py	Add Tensorflow Object Detection API. (#1561)	6 days ago
create_pascal_tf_record.py	Make Record scripts python3 compatible. (#1614)	2 hours ago
create_pascal_tf_record_test.py	Add Tensorflow Object Detection API. (#1561)	6 days ago
create_pet_tf_record.py	Make Record scripts python3 compatible. (#1614)	2 hours ago
eval.py	Add Tensorflow Object Detection API. (#1561)	6 days ago
eval_util.py	Add Tensorflow Object Detection API. (#1561)	6 days ago
evaluator.py	Add Tensorflow Object Detection API. (#1561)	6 days ago
export_inference_graph.py	Add Tensorflow Object Detection API. (#1561)	6 days ago

README.md

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]

Maintainers

- Jonathan Huang, github: jch1
- Vivek Rathod, github: tombstone
- Derek Chow, github: derekjchow

Tensorflow detection model zoo

We provide a collection of detection models pre-trained on the COCO dataset. These models can be useful for out-of-the-box inference if you are interested in categories already in COCO (e.g., humans, cars, etc). They are also useful for initializing your models when training on novel datasets.

In the table below, we list each such pre-trained model including:

- a model name that corresponds to a config file that was used to train this model in the `samples/configs` directory,
- a download link to a tar.gz file containing the pre-trained model,
- model speed (one of {slow, medium, fast}),
- detector performance on COCO data as measured by the COCO mAP measure. Here, higher is better, and we only report bounding box mAP rounded to the nearest integer.
- Output types (currently only `Boxes`)

You can un-tar each tar.gz file via, e.g.,:

```
tar -xzvf ssd_mobilenet_v1_coco.tar.gz
```

Inside the un-tar'ed directory, you will find:

- a graph proto (`graph.pbtxt`)
- a checkpoint (`model.ckpt.data-00000-of-00001`, `model.ckpt.index`, `model.ckpt.meta`)
- a frozen graph proto with weights baked into the graph as constants (`frozen_inference_graph.pb`) to be used for out of the box inference (try this out in the Jupyter notebook!)

Model name	Speed	COCO mAP	Outputs
ssd_mobilenet_v1_coco	fast	21	Boxes
ssd_inception_v2_coco	fast	24	Boxes
rfcn_resnet101_coco	medium	30	Boxes
faster_rcnn_resnet101_coco	medium	32	Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37	Boxes

API Organization

Front Ends

Train, Eval, Export for serving, Demos (Jupyter Notebook, Android)

High Level API

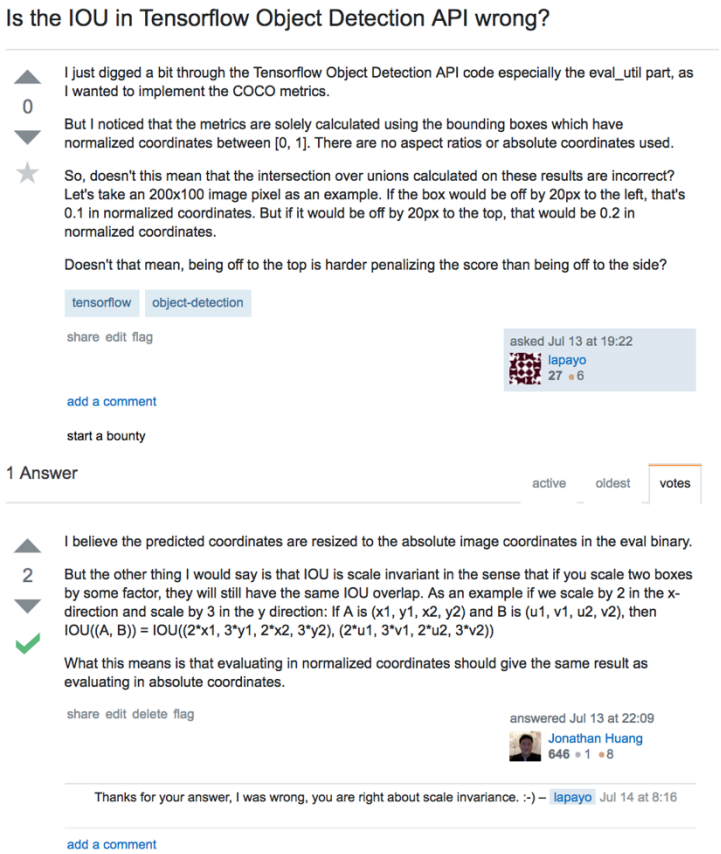
Meta-architectures (SSD, Faster R-CNN, …)
Models (SSD w/Mobilenet, Faster R-CNN w/Resnet, ...)

Low Level API

Box operations (e.g., IOU, Clipping, Non max suppression), Box Representations, Target Assignment, Visualization ops,

API can be used at any of these levels

Ongoing Community Engagement...



Is the IOU in Tensorflow Object Detection API wrong?

0

I just digged a bit through the Tensorflow Object Detection API code especially the eval_util part, as I wanted to implement the COCO metrics.

But I noticed that the metrics are solely calculated using the bounding boxes which have normalized coordinates between [0, 1]. There are no aspect ratios or absolute coordinates used.

So, doesn't this mean that the intersection over unions calculated on these results are incorrect? Let's take an 200x100 image pixel as an example. If the box would be off by 20px to the left, that's 0.1 in normalized coordinates. But if it would be off by 20px to the top, that would be 0.2 in normalized coordinates.

Doesn't that mean, being off to the top is harder penalizing the score than being off to the side?

tensorflow object-detection

share edit flag

asked Jul 13 at 19:22
lapayo
27 ●6

add a comment

start a bounty

1 Answer

active oldest votes

2

I believe the predicted coordinates are resized to the absolute image coordinates in the eval binary.

But the other thing I would say is that IOU is scale invariant in the sense that if you scale two boxes by some factor, they will still have the same IOU overlap. As an example if we scale by 2 in the x-direction and scale by 3 in the y direction: If A is (x1, y1, x2, y2) and B is (u1, v1, u2, v2), then IOU((A, B)) = IOU((2*x1, 3*y1, 2*x2, 3*y2), (2*u1, 3*v1, 2*u2, 3*v2))

What this means is that evaluating in normalized coordinates should give the same result as evaluating in absolute coordinates.

share edit delete flag

answered Jul 13 at 22:09
Jonathan Huang
646 ●1 ●8

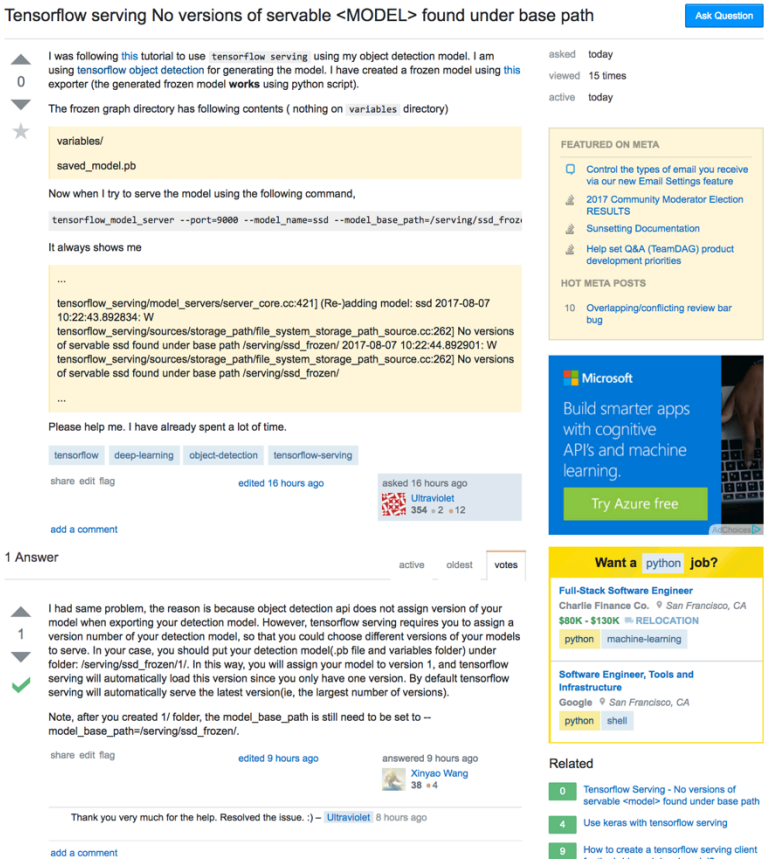
Thanks for your answer, I was wrong, you are right about scale invariance. :-) – lapayo Jul 14 at 8:16

add a comment

We provide regular support on StackOverflow (and Github)

Ongoing Community Engagement...

Community members support each other too :)



Tensorflow serving No versions of servable <MODEL> found under base path

Ask Question

I was following this tutorial to use `tensorflow serving` using my object detection model. I am using tensorflow object detection for generating the model. I have created a frozen model using this exporter (the generated frozen model **works** using python script).

The frozen graph directory has following contents (nothing on `variables` directory)

```
variables/

saved_model.pb
```

Now when I try to serve the model using the following command,

```
tensorflow_model_server --port=9000 --model_name=ssd --model_base_path=/serving/ssd_froz
```

It always shows me

```
...

tensorflow_serving/model_servers/server_core.cc:421] (Re-)adding model: ssd 2017-08-07
10:22:43.892834: W
tensorflow_serving/sources/storage_path/file_system_storage_path_source.cc:262] No versions
of servable ssd found under base path /serving/ssd_frozen/ 2017-08-07 10:22:44.892901: W
tensorflow_serving/sources/storage_path/file_system_storage_path_source.cc:262] No versions
of servable ssd found under base path /serving/ssd_frozen/

...
```

Please help me. I have already spent a lot of time.

tensorflow deep-learning object-detection tensorflow-serving

share edit flag

edited 16 hours ago

asked 16 hours ago
Ultraviolet
354 ● 2 ● 12

add a comment

asked today
viewed 15 times
active today

FEATURED ON META

- Control the types of email you receive via our new Email Settings feature
- 2017 Community Moderator Election RESULTS
- Sunsetting Documentation
- Help set Q&A (TeamDAG) product development priorities

HOT META POSTS

10 Overlapping/conflicting review bar bug

1 Answer

active oldest votes

I had same problem, the reason is because object detection api does not assign version of your model when exporting your detection model. However, tensorflow serving requires you to assign a version number of your detection model, so that you could choose different versions of your models to serve. In your case, you should put your detection model(.pb file and variables folder) under folder: /serving/ssd_frozen/1/. In this way, you will assign your model to version 1, and tensorflow serving will automatically load this version since you only have one version. By default tensorflow serving will automatically serve the latest version(ie, the largest number of versions).

Note, after you created 1/ folder, the model_base_path is still need to be set to --model_base_path=/serving/ssd_frozen/.

share edit flag

edited 9 hours ago

answered 9 hours ago
Xinyao Wang
38 ● 4

Thank you very much for the help. Resolved the issue. :) – Ultraviolet 8 hours ago

add a comment

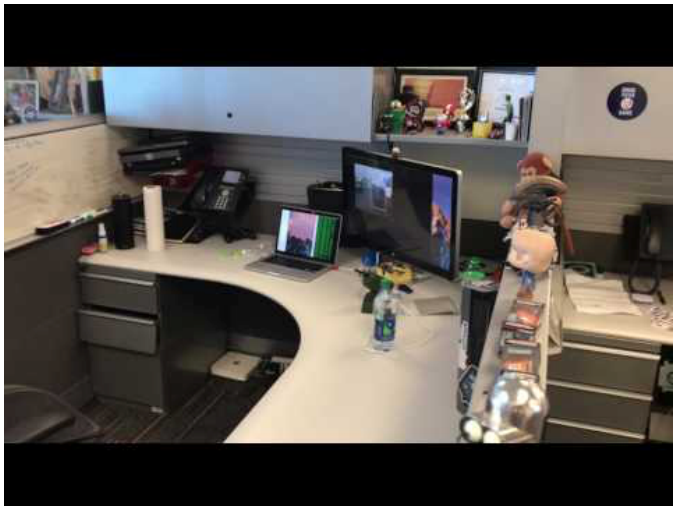
Related

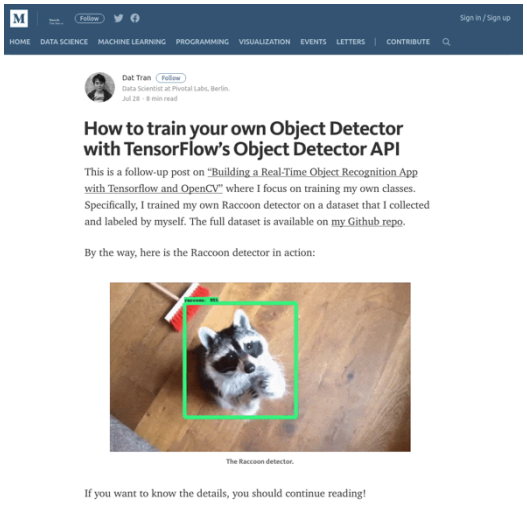
0 Tensorflow Serving - No versions of servable <model> found under base path

4 Use keras with tensorflow serving

9 How to create a tensorflow serving client

Community Creations!





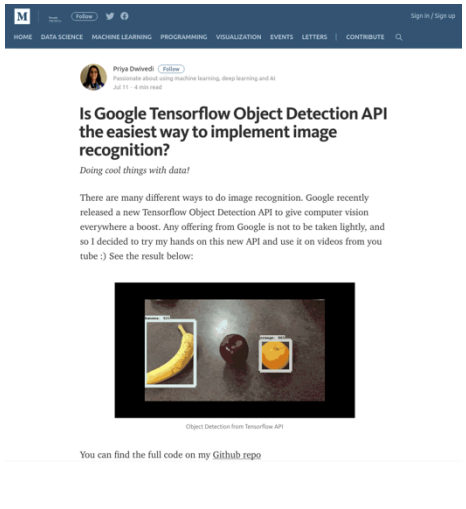
How to train your own Object Detector with TensorFlow's Object Detector API

This is a follow-up post on "Building a Real-Time Object Recognition App with Tensorflow and OpenCV" where I focus on training my own classes. Specifically, I trained my own Raccoon detector on a dataset that I collected and labeled by myself. The full dataset is available on my Github repo.

By the way, here is the Raccoon detector in action:

The Raccoon detector.

If you want to know the details, you should continue reading!



Is Google Tensorflow Object Detection API the easiest way to implement image recognition?

Doing cool things with data!

There are many different ways to do image recognition. Google recently released a new Tensorflow Object Detection API to give computer vision everywhere a boost. Any offering from Google is not to be taken lightly, and so I decided to try my hands on this new API and use it on videos from you tube :) See the result below:

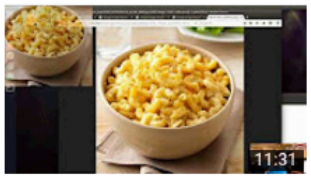
Object Detection from Tensorflow API

You can find the full code on my Github repo

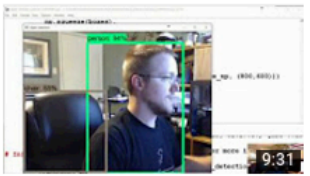
Testing Custom Object Detector - TensorFlow Object Detection A...
4,524 views • 3 days ago

Training Custom Object Detector - TensorFlow Object Detection
3,007 views • 3 days ago

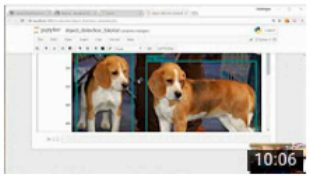
Creating TFRecords - TensorFlow Object Detection API Tutorial p.4
3,145 views • 3 days ago



Tracking Custom Objects - TensorFlow Object Detection A...
4,600 views • 3 days ago



Adapting to video feed - TensorFlow Object Detection A...
15,157 views • 6 days ago



Intro - TensorFlow Object Detection API Tutorial p.1
16,571 views • 1 week ago

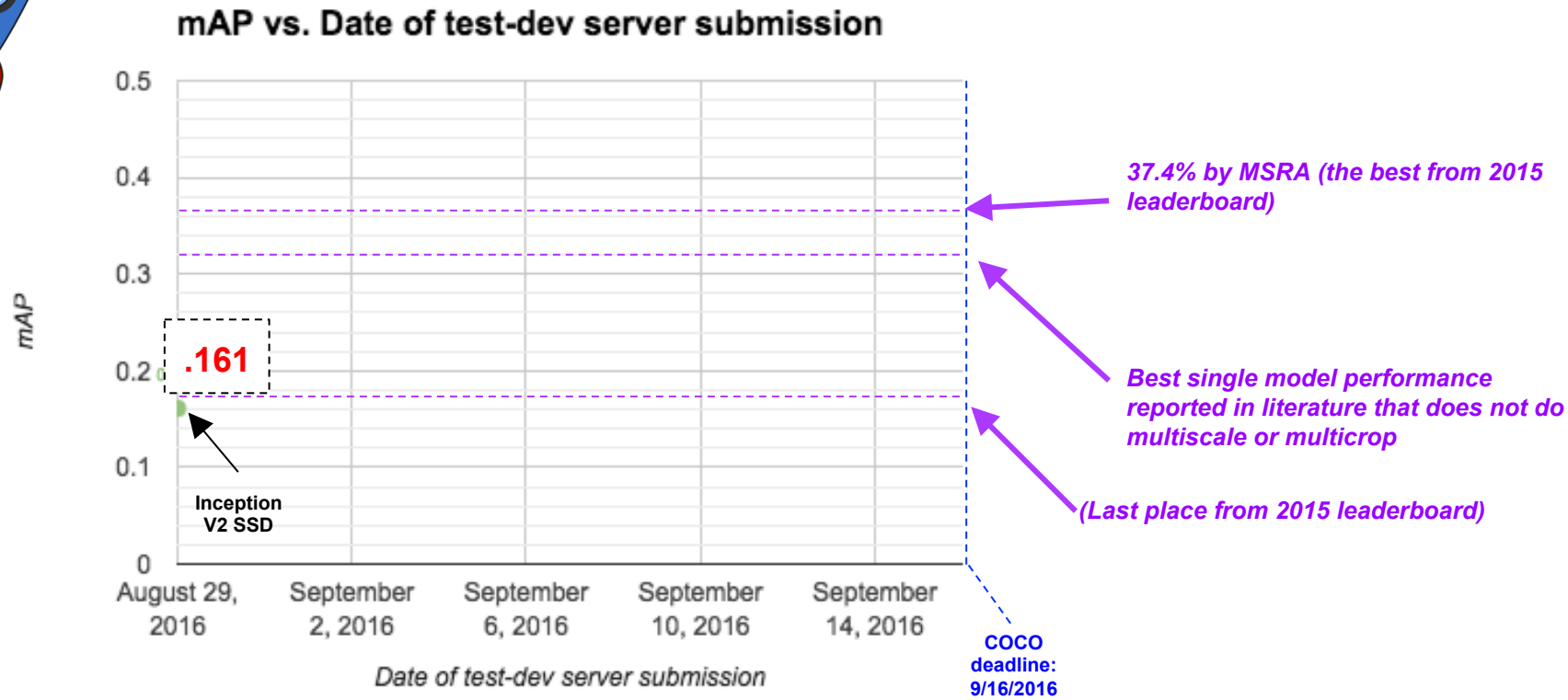
This Talk

- The Tensorflow Object Detection API
- **Pushing the envelope on Accuracy**
  - **Better Models**
  - Better Data
- Pushing the envelope on Speed
  - Lightweight Models
  - Adaptive Inference

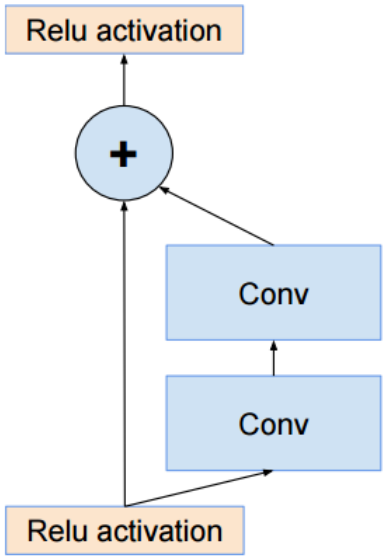
Race to the Top

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

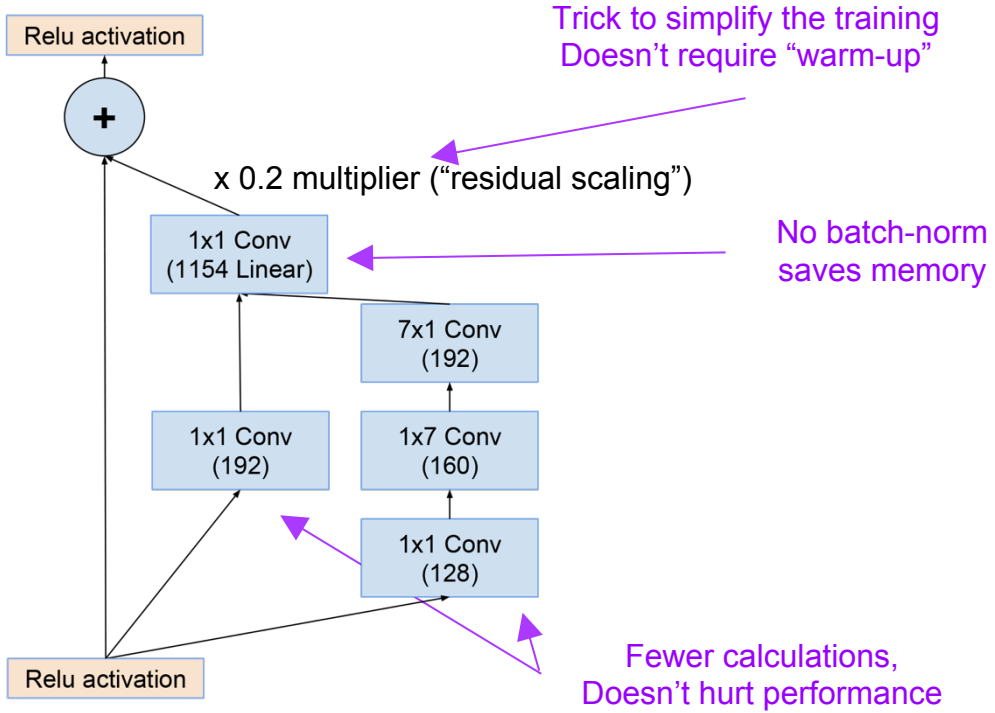
Liu, Wei, et al. "**SSD: Single Shot MultiBox Detector**." *arXiv preprint arXiv:1512.02325* (2015).



Residual Blocks vs. Inception Resnet Blocks



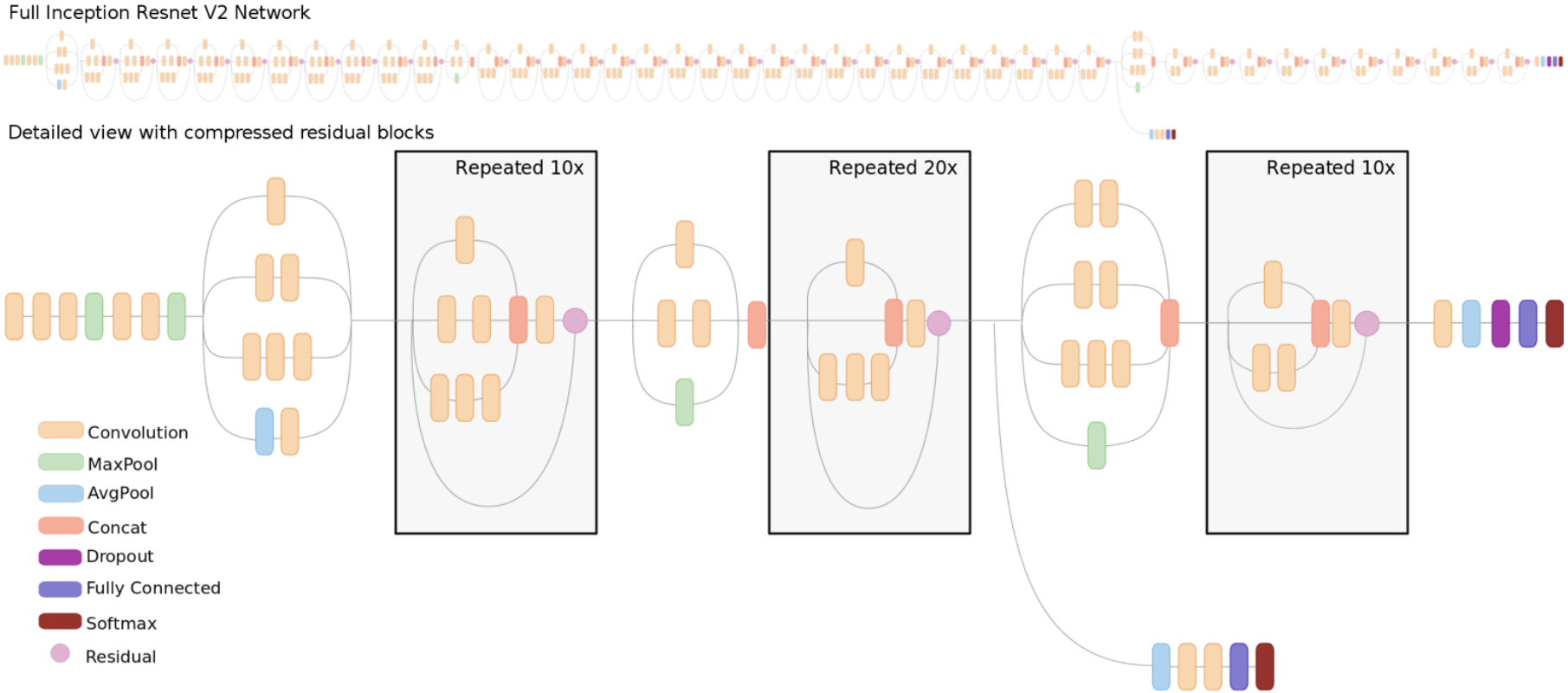
Residual Block



Inception Resnet Block

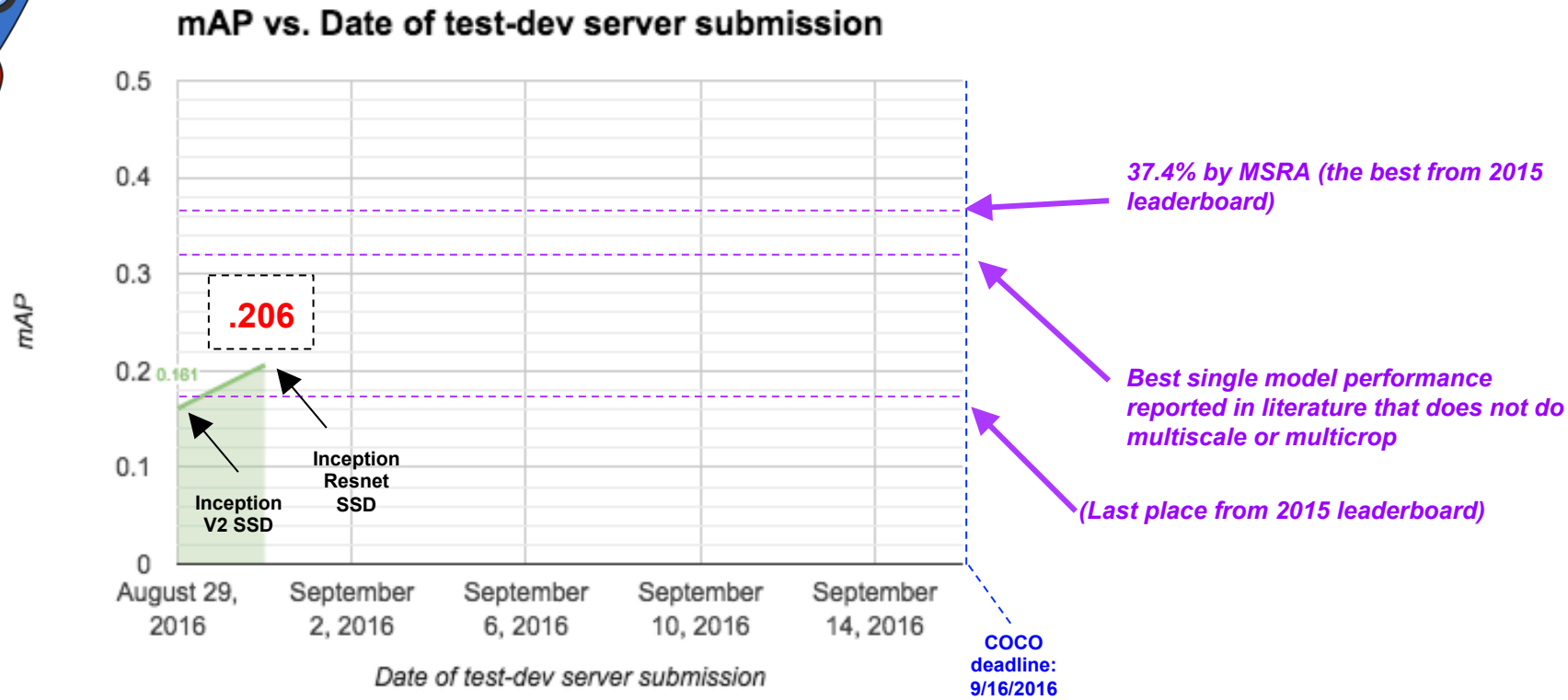
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning by Szegedy et al.

Inception Resnet (v2) Feature Extractor



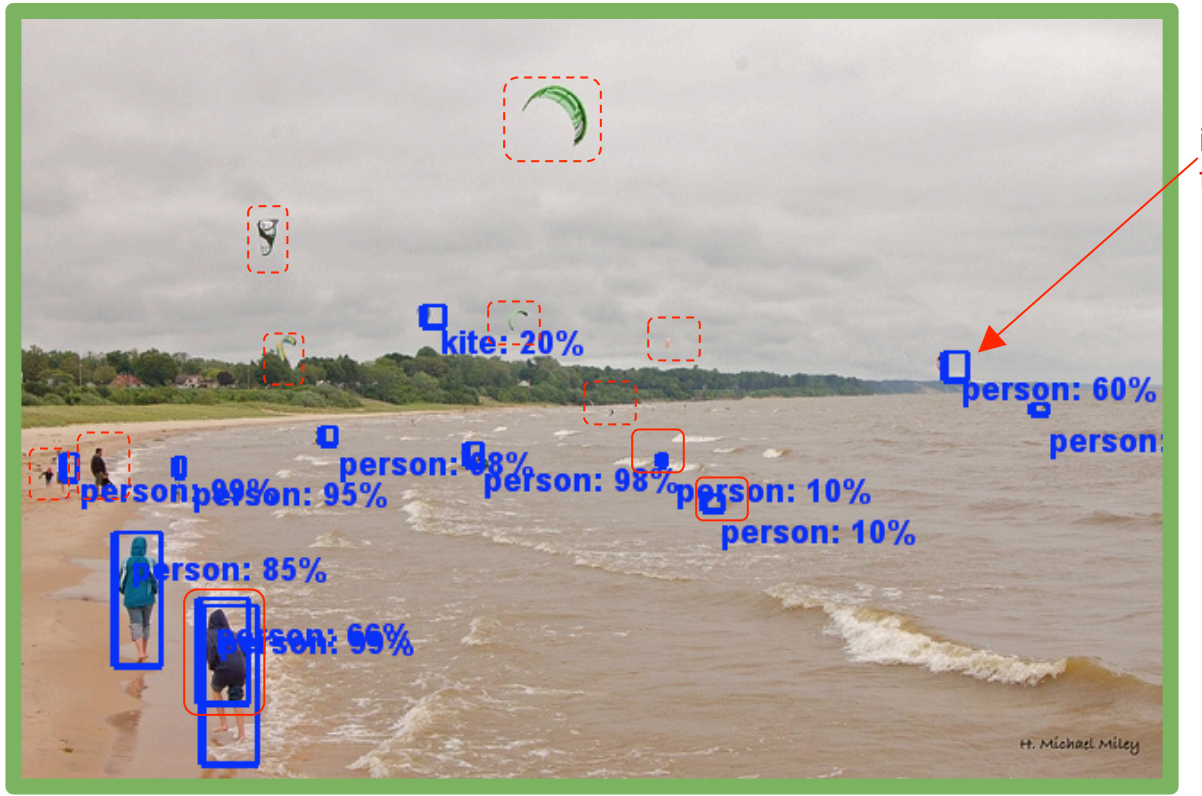
Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning by Szegedy et al.

Race to the Top



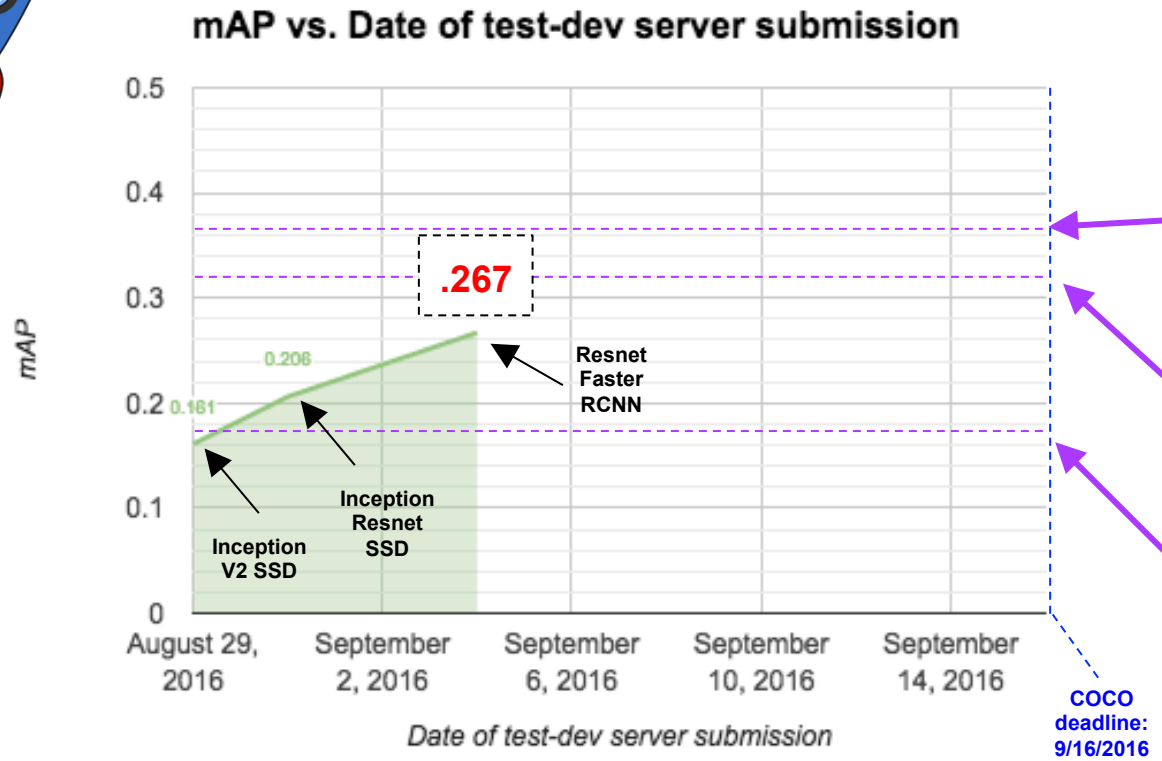

H. Michael Miley

Inception Resnet SSD



inaccurate localization
for small objects

Race to the Top

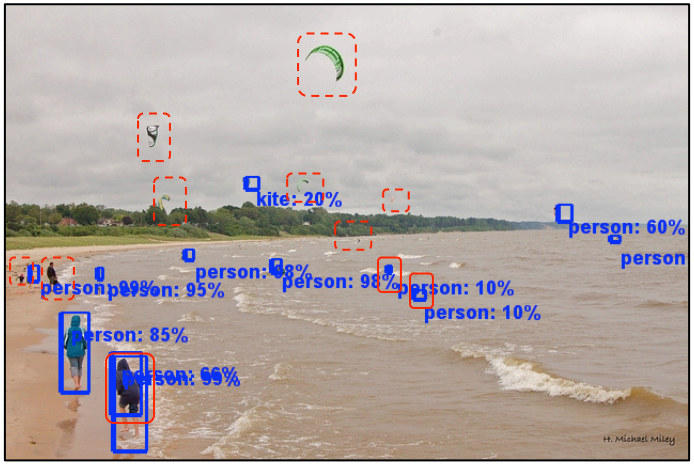


37.4% by MSRA (the best from 2015 leaderboard)

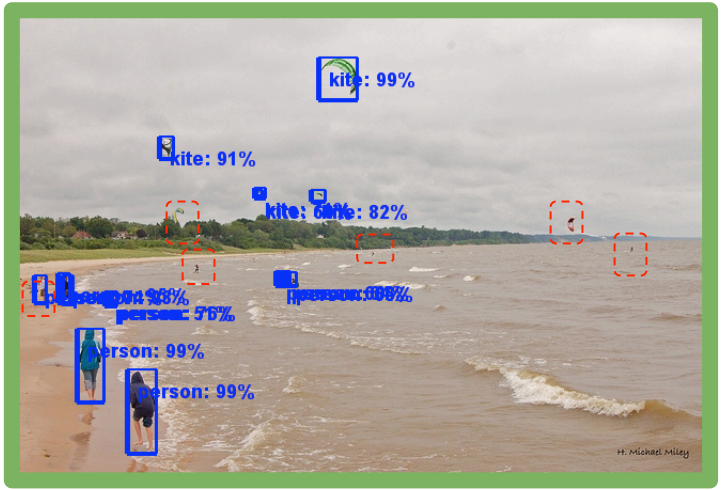
Best single model performance reported in literature that does not do multiscale or multicrop

(Last place from 2015 leaderboard)

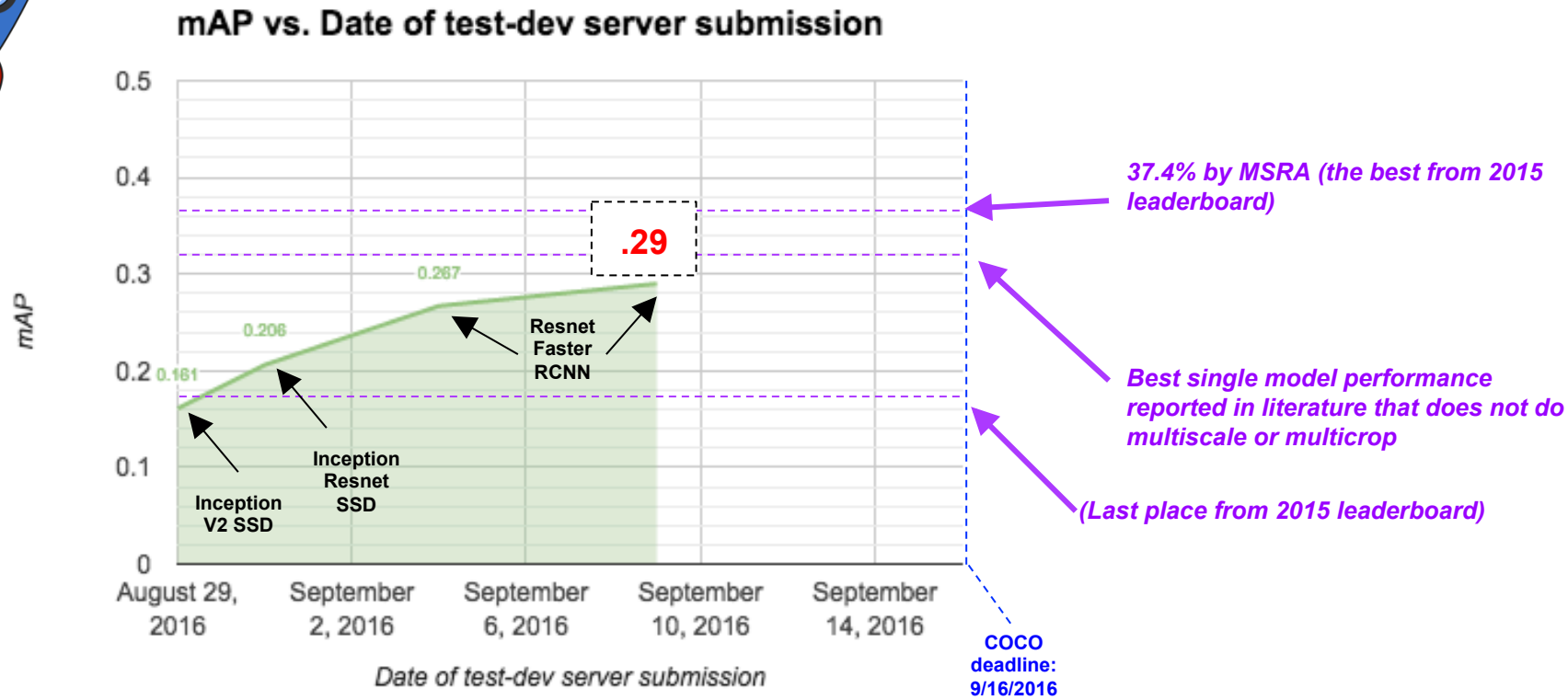
Inception Resnet SSD



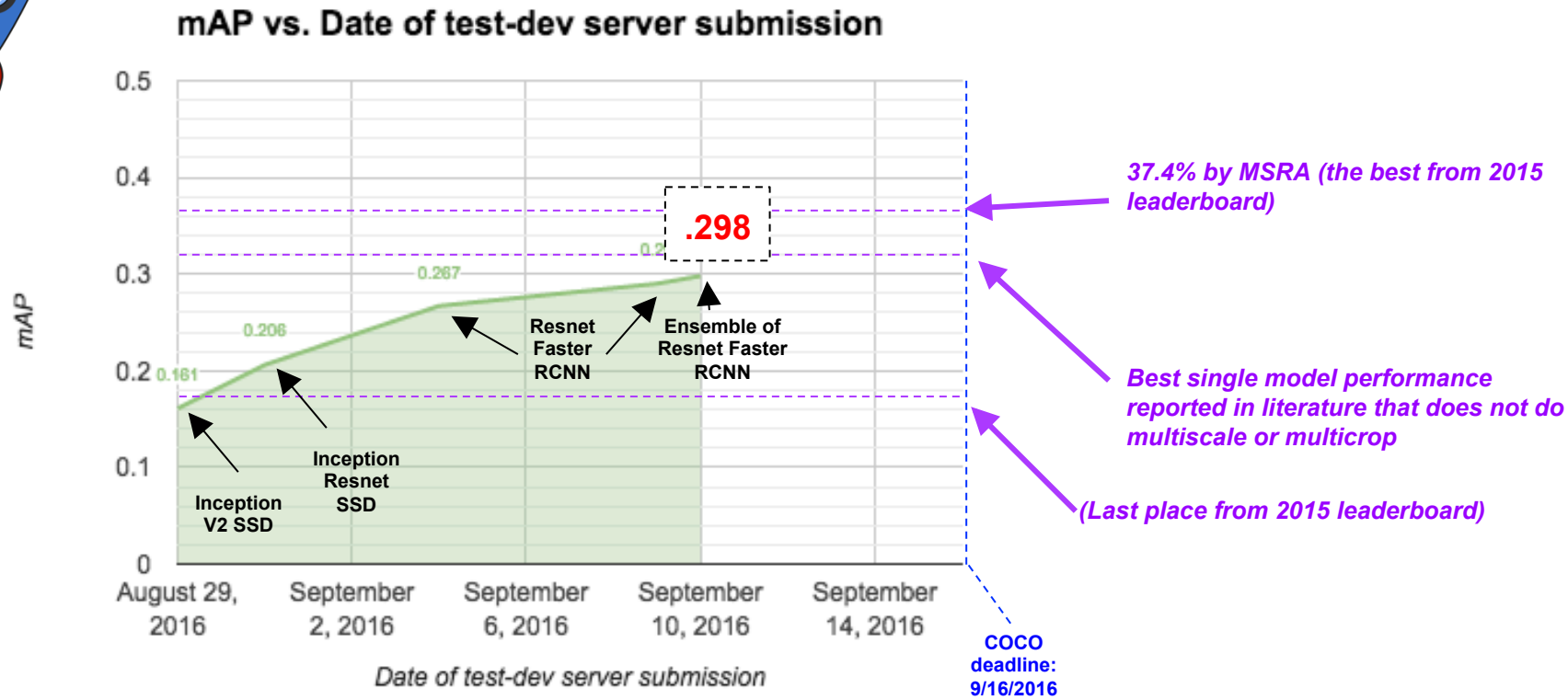
Resnet Faster RCNN



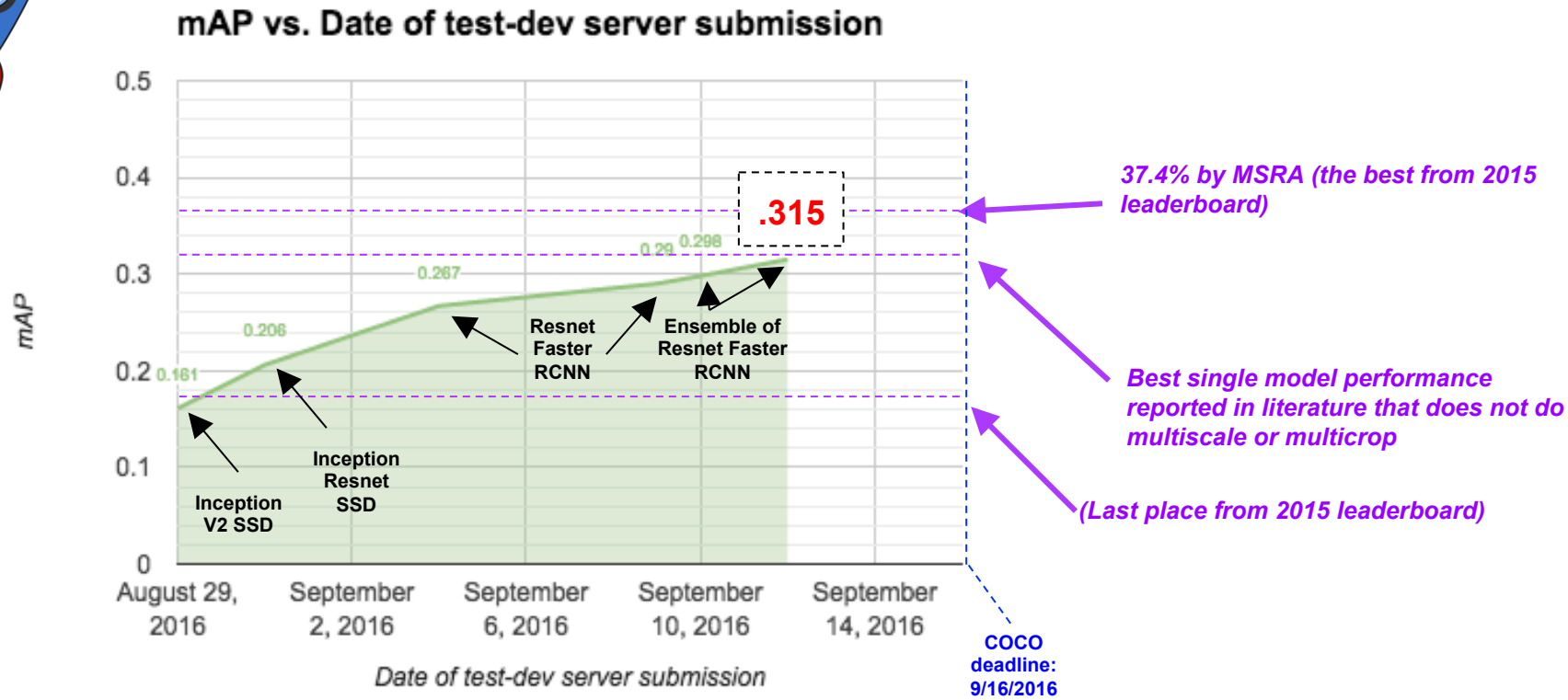
Race to the Top



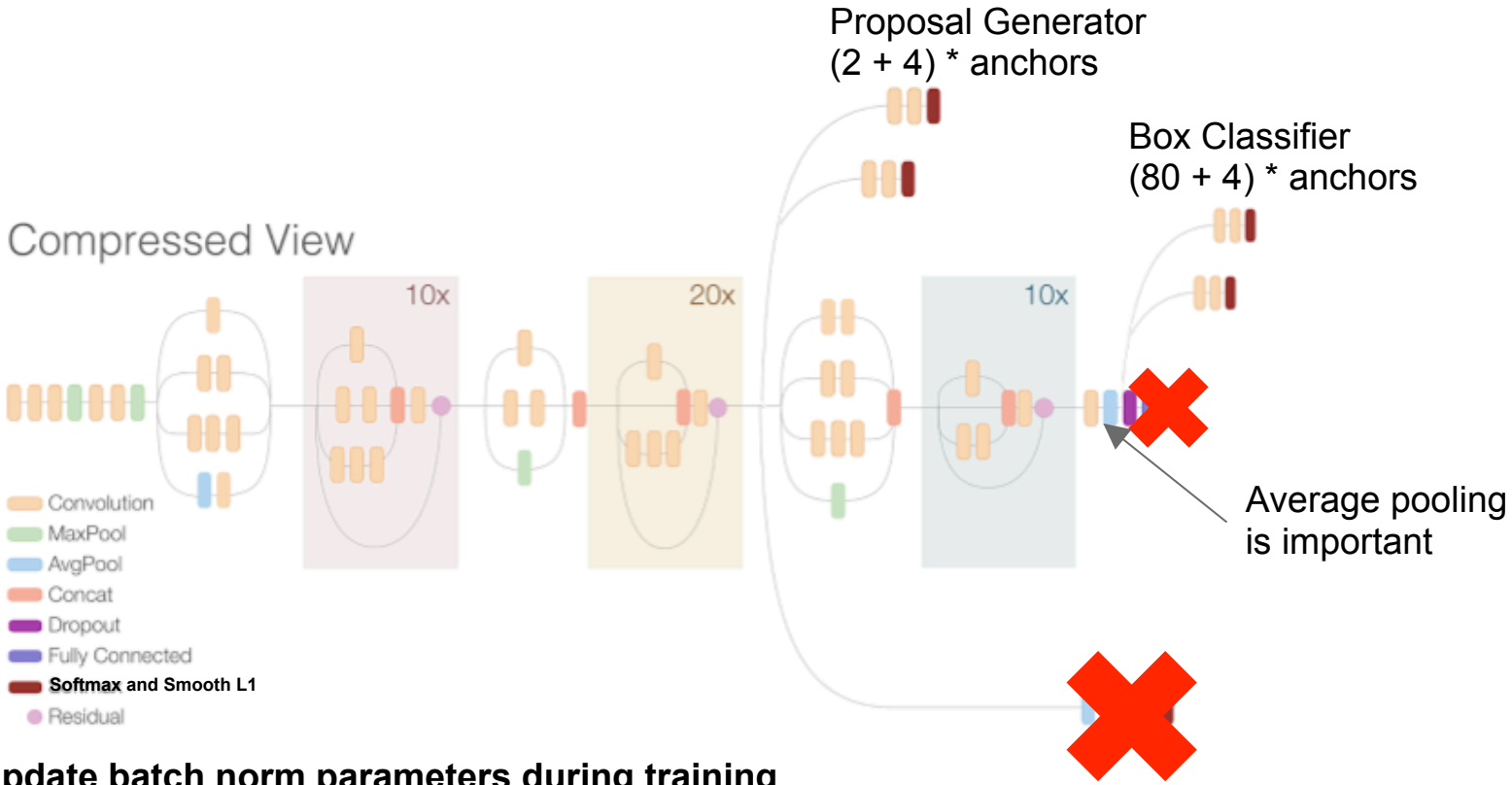
Race to the Top



Race to the Top

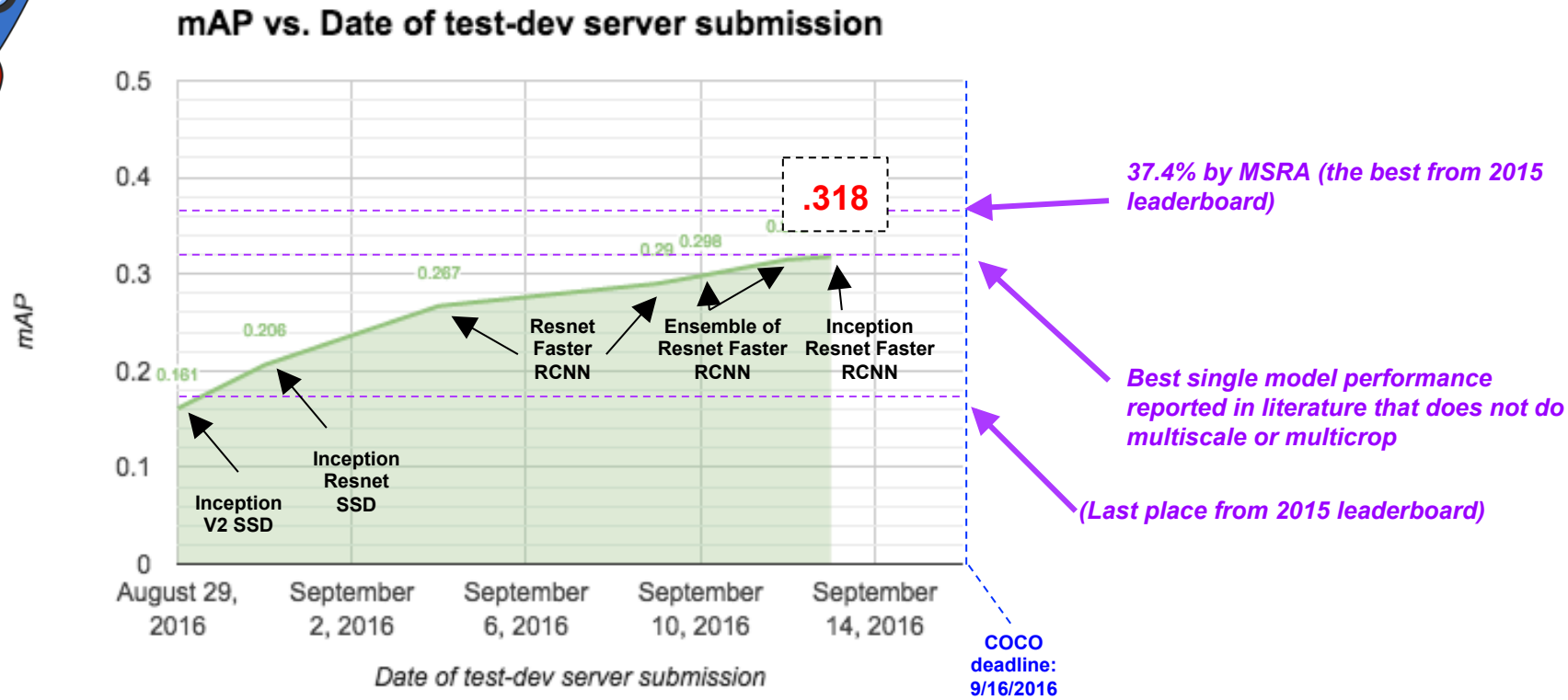


Faster RCNN w/Inception Resnet (v2)

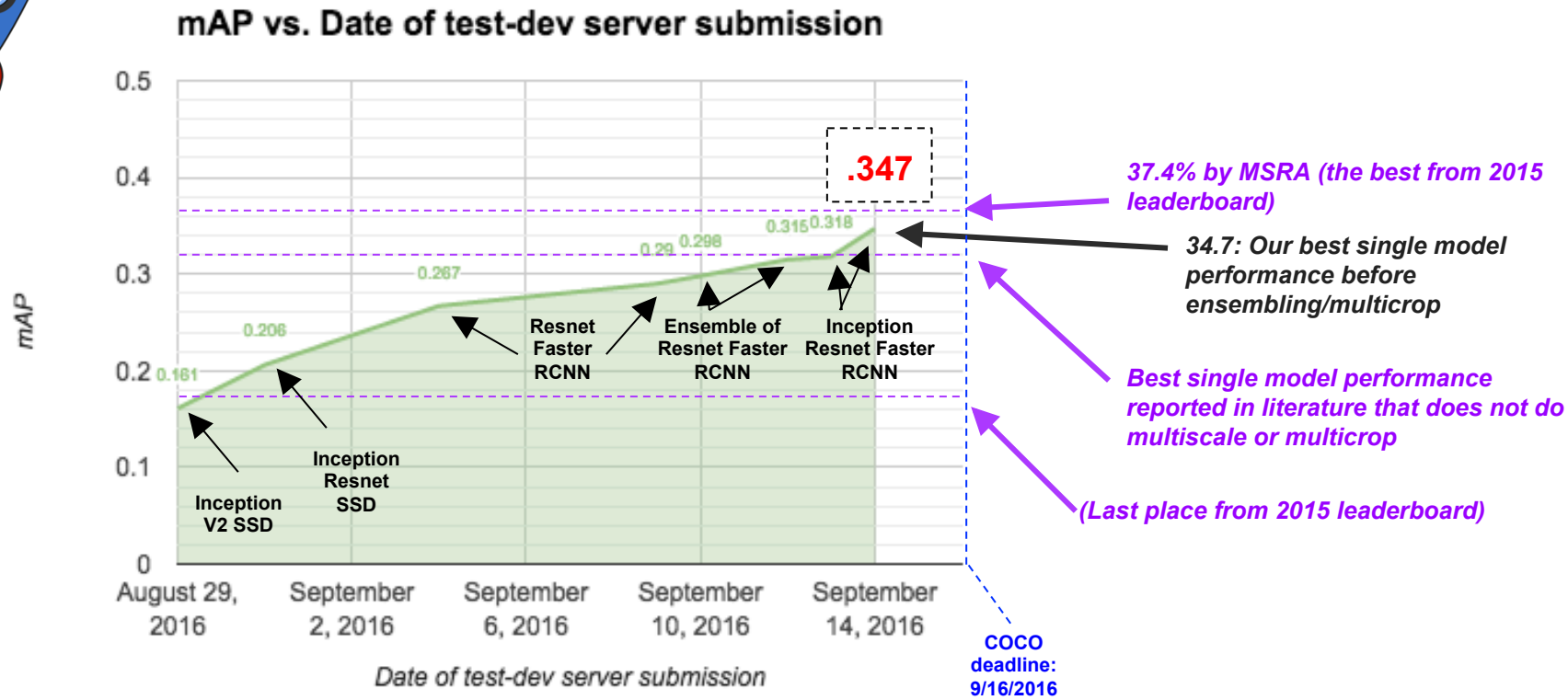


Note: We don't update batch norm parameters during training

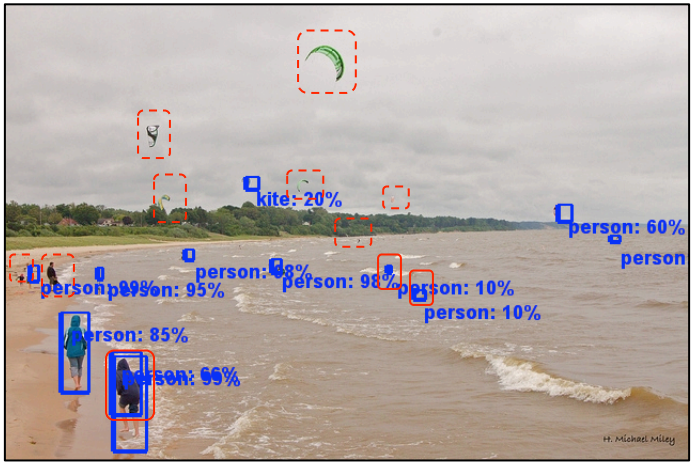
Race to the Top



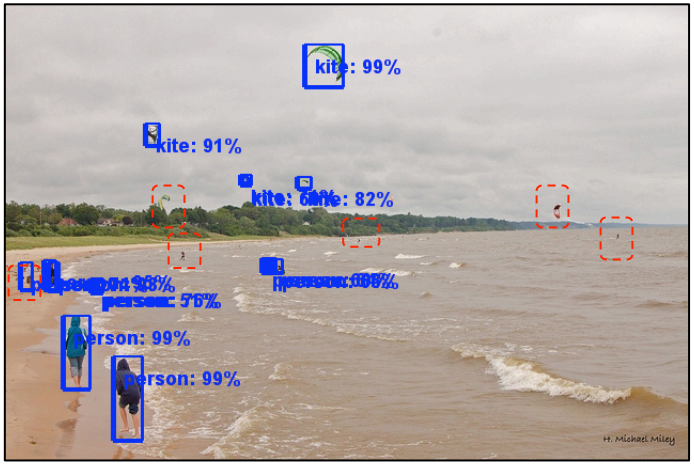
Race to the Top



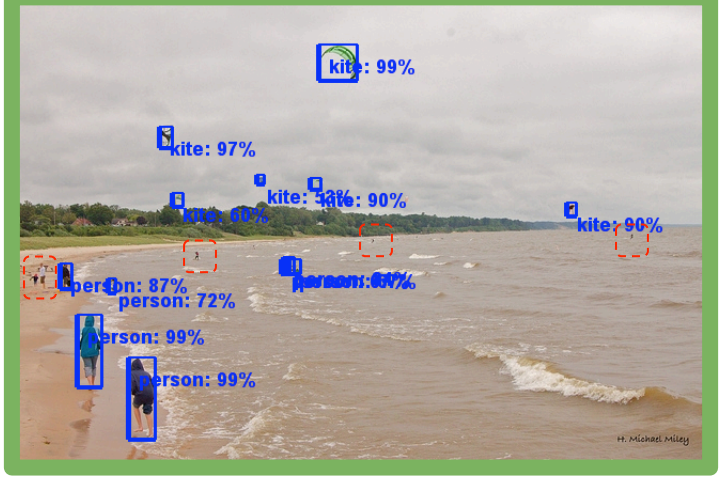
Inception Resnet SSD



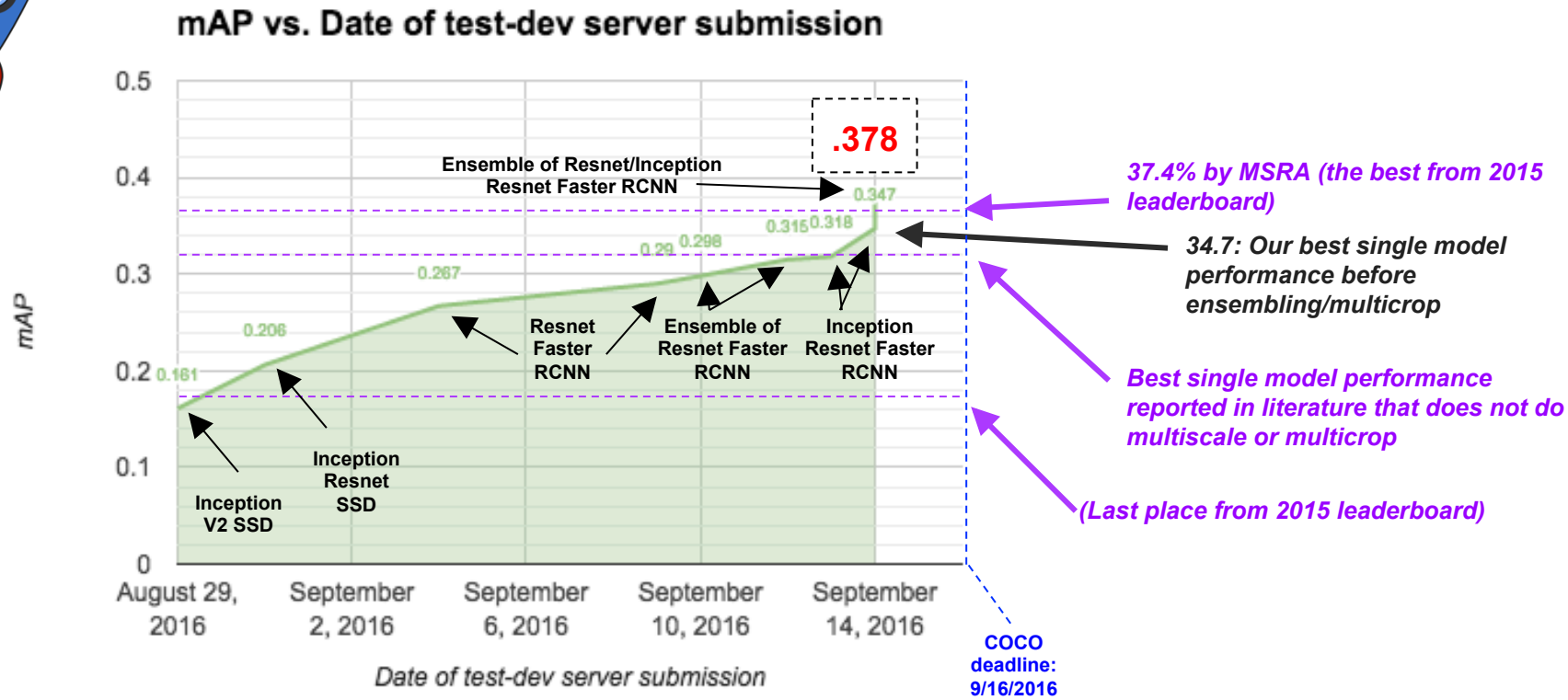
Resnet Faster RCNN



Inception Resnet Faster RCNN



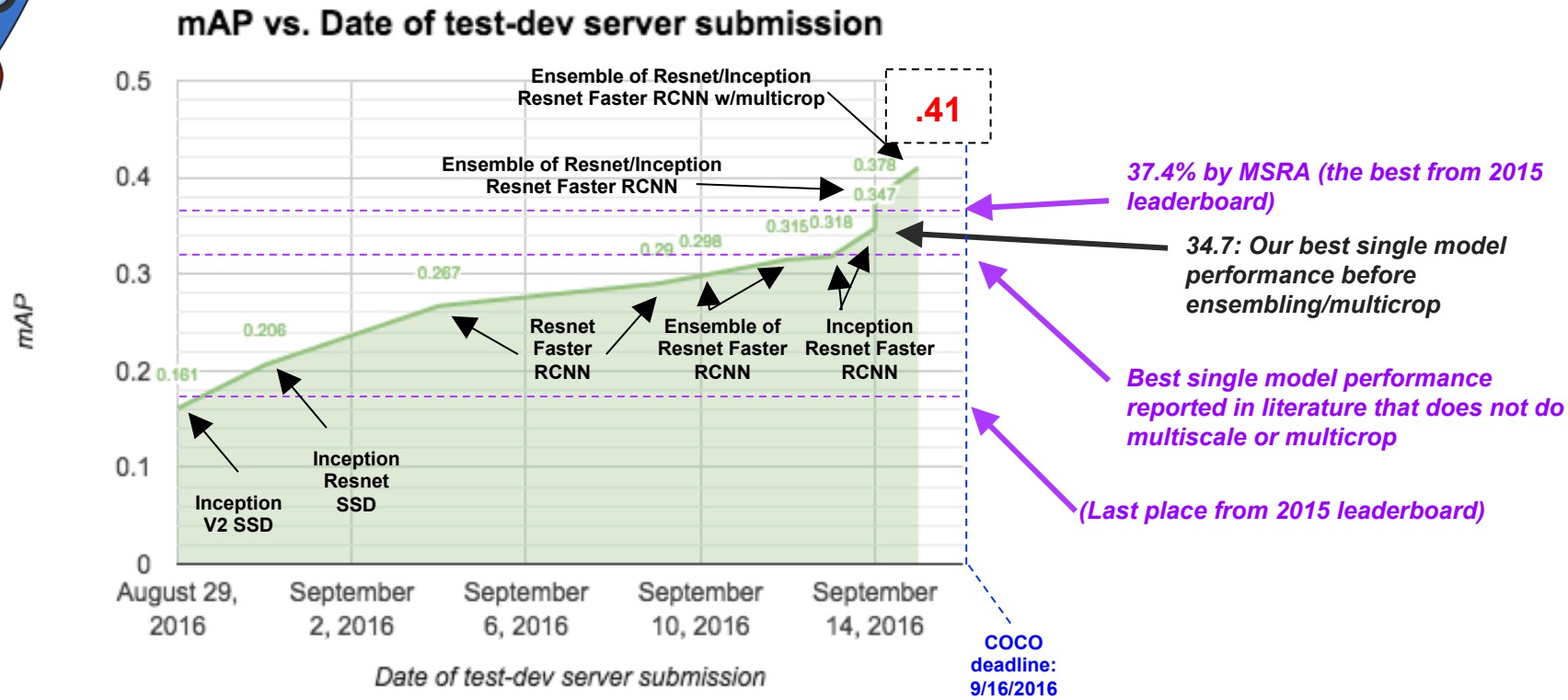
Race to the Top

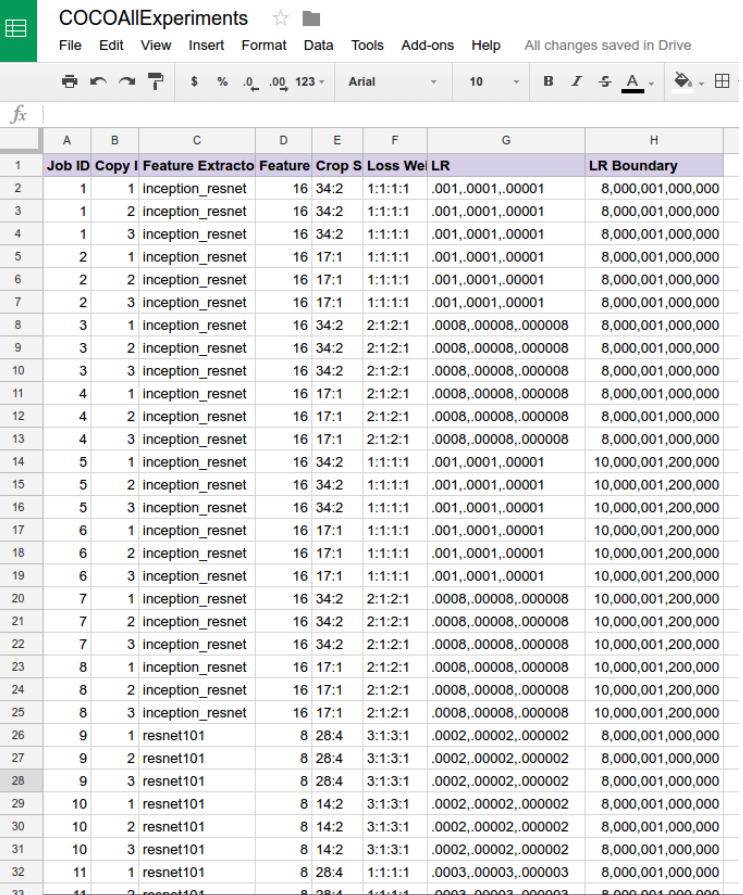


10-crop inference

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

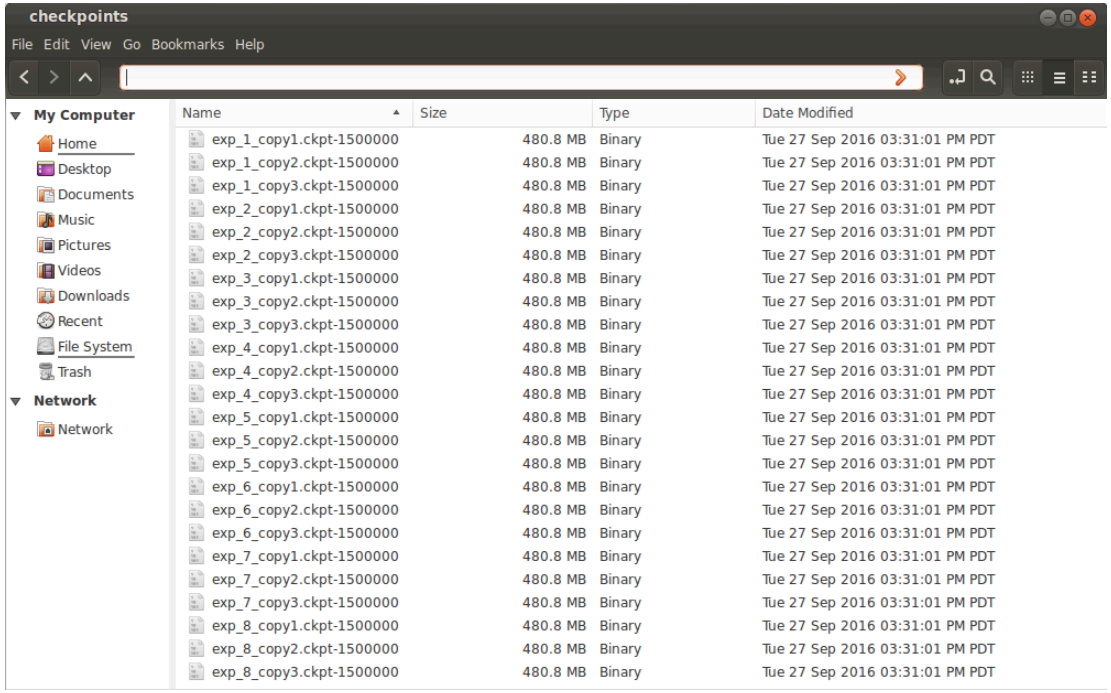
	A	B	C	D	E	F	G	H
1	Job ID	Copy I	Feature Extracto	Feature	Crop S	Loss We	LR	LR Boundary
2	1	1	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
3	1	2	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
4	1	3	inception_resnet	16	34:2	1:1:1:1	.001,.0001,.00001	8,000,001,000,000
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)?
- ❏ etc.

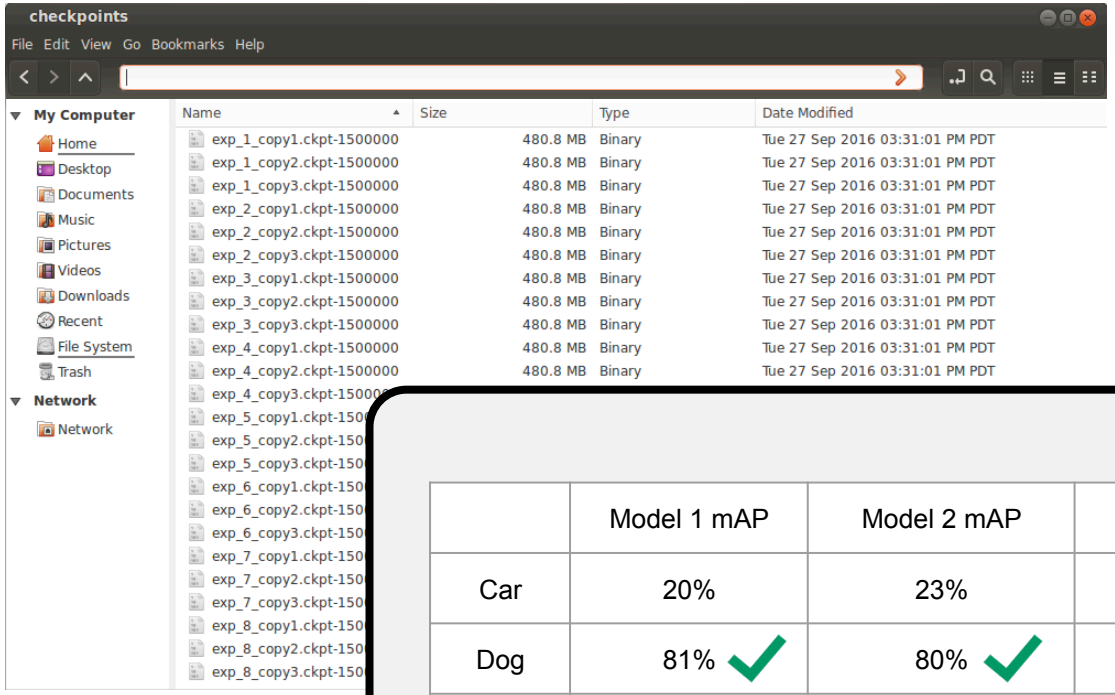
… led to *Model Checkpoint Overload* :(

Model Selection for Ensembling



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

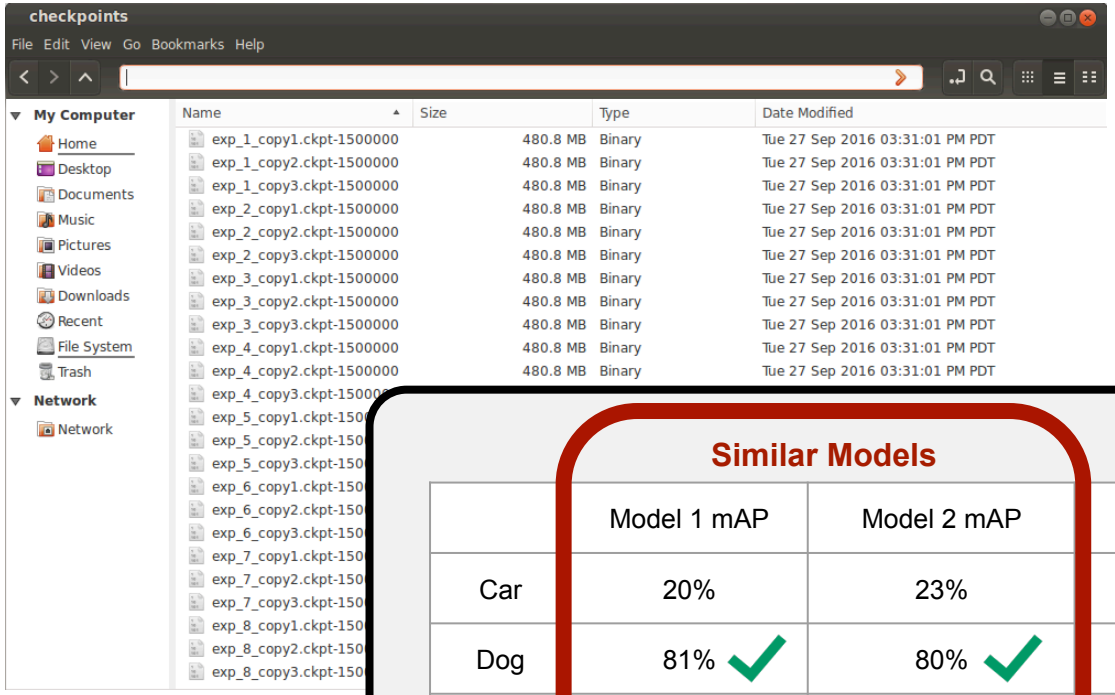
Model Selection for Ensembling



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

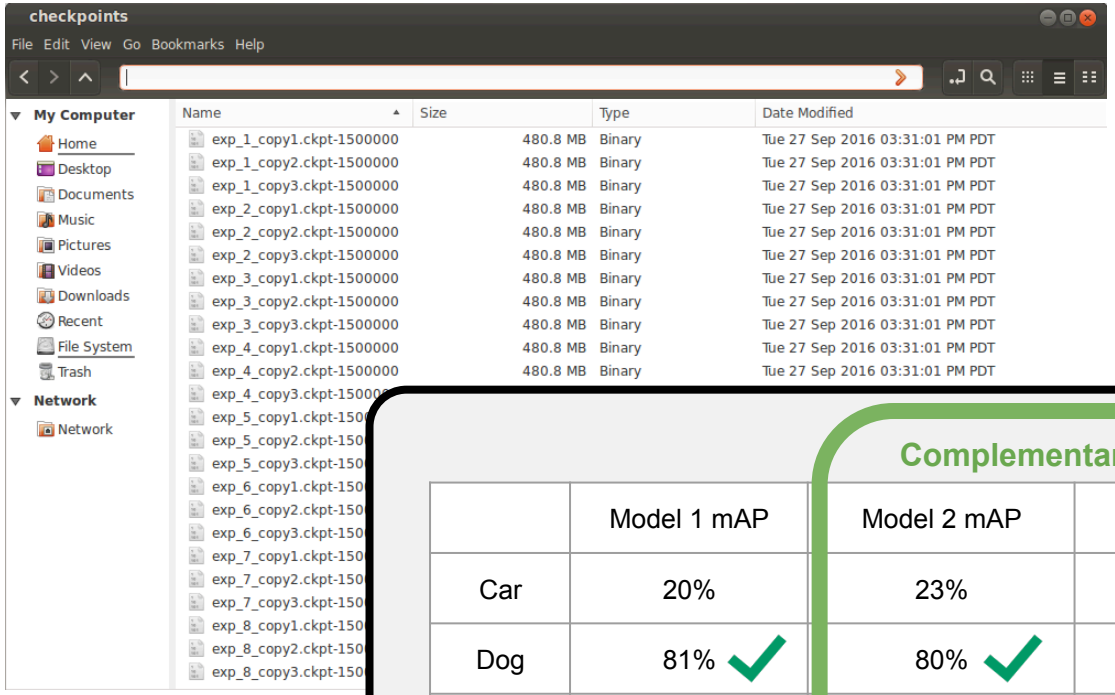


Take best K models? Or select diverse K-subset of models?

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.*

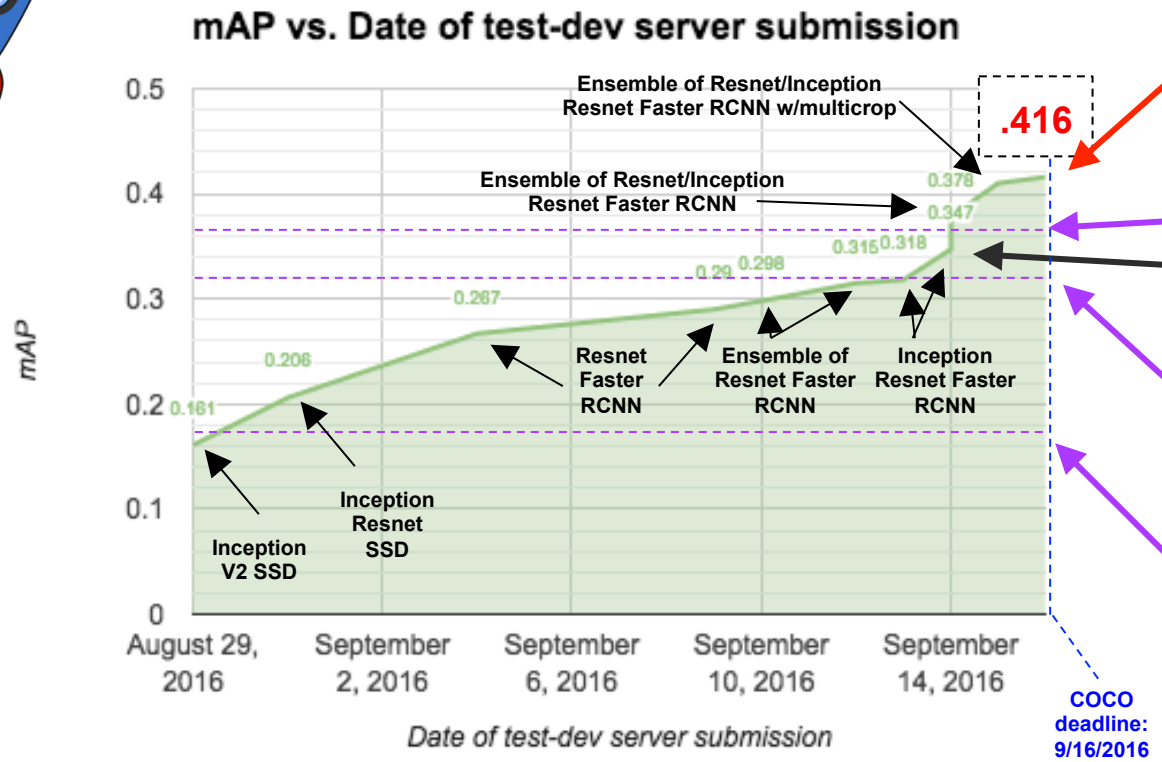
Final ensemble selected for challenge submission				
Individual mean AP (on minival)	**Feature Extractor**	**Output Stride**	**Location:Classification loss ratio**	**Location Loss function**
32.93	Resnet 101	8	3:1	SmoothL1
33.3	Resnet 101	8	1:1	SmoothL1
34.75	Inception Resnet	16	1:1	SmoothL1
35	Inception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

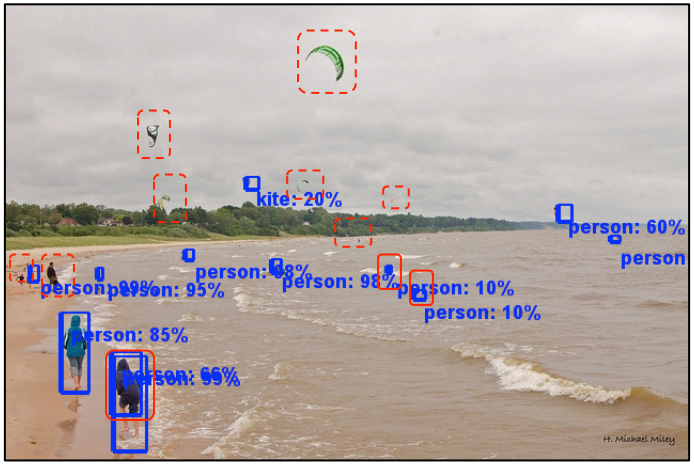
37.4% by MSRA (the best from 2015 leaderboard)

34.7: Our best single model performance before ensembling/multicrop

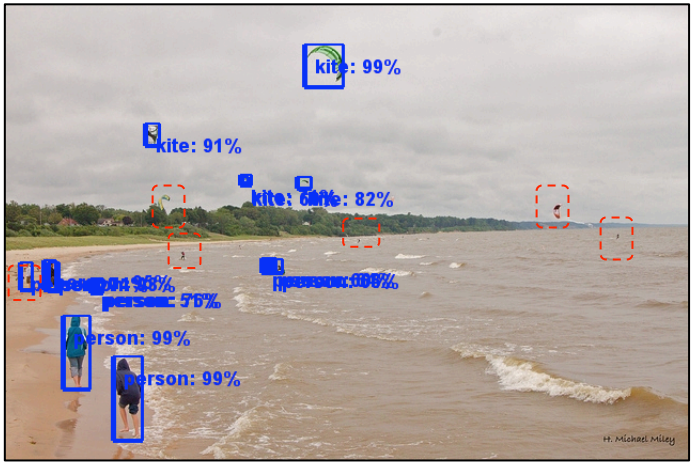
Best single model performance reported in literature that does not do multiscale or multicrop

(Last place from 2015 leaderboard)

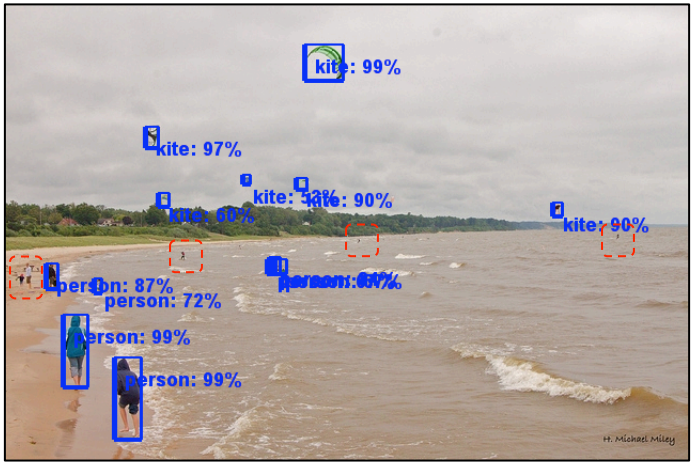
Inception Resnet SSD



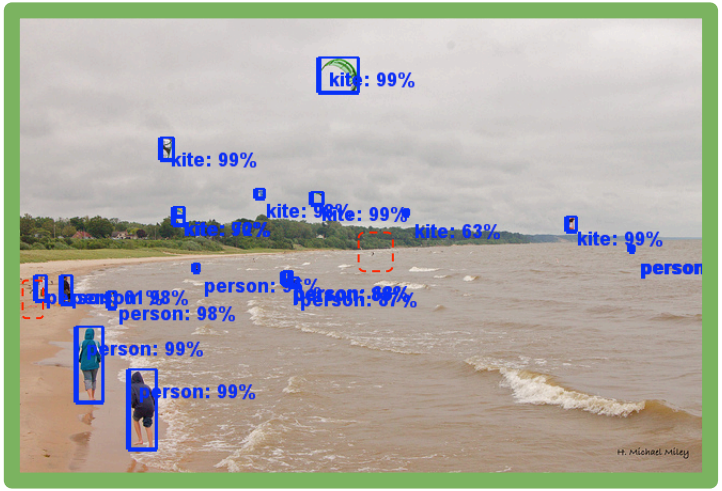
Resnet Faster RCNN



Inception Resnet Faster RCNN



Final ensemble with multicrop inference



This Talk

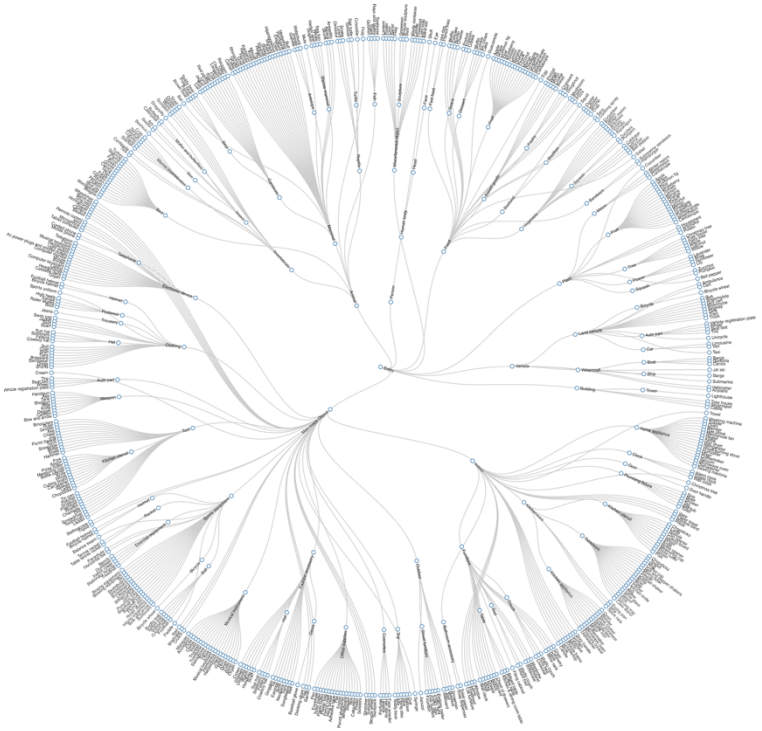
- The Tensorflow Object Detection API
- **Pushing the envelope on Accuracy**
  - Better Models
  - **Better Data**
- Pushing the envelope on Speed
  - Lightweight Models
  - Adaptive Inference

JFT-300M: a massive dataset from ImageSearch data

- 300M web images
- 375M image label pairs

JFT-300M: a massive dataset from ImageSearch data

- 300M web images
- 375M image label pairs
- ~ 19K categories



JFT-300M: a massive dataset from ImageSearch data

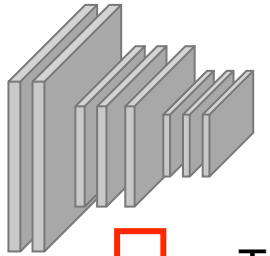
- 300M web images
- 375M image label pairs
- ~ 18K categories
- ~ 20% label noise
- Unknown recall
- Long-tail distribution

Tortoise:

vs.

Revisiting Unreasonable Effectiveness of Data in Deep Learning Era [Sun et al, '17]

JFT 300M

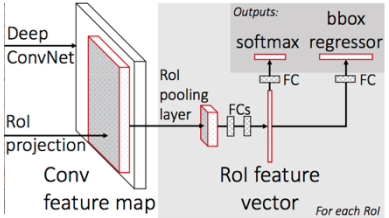


18K labels

Transfer weights

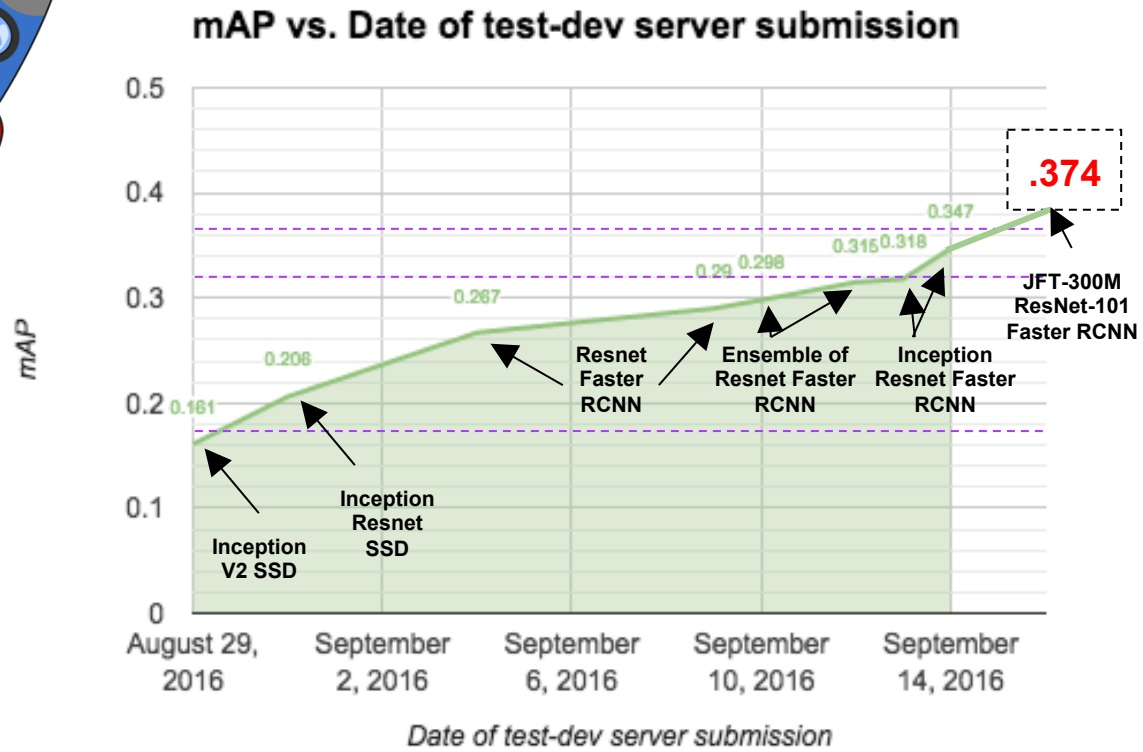
PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

COCO
Common Objects in Context



Detections

Better Representation Learning Helps!

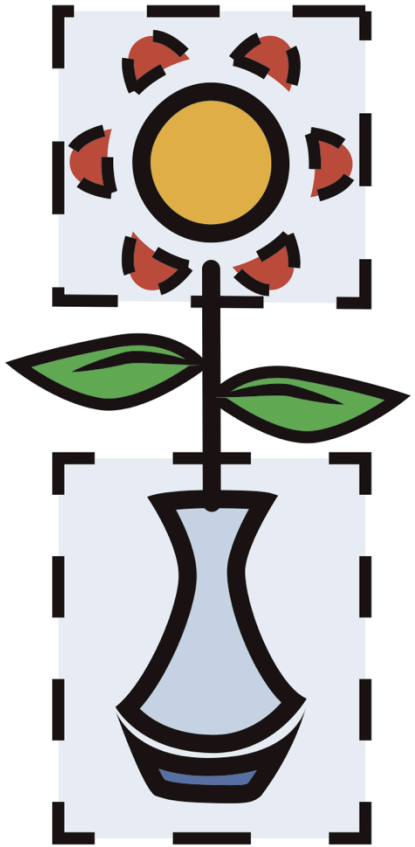


Using a JFT-300M pre-trained checkpoint to replace ImageNet ones:

- 2.7% gain over best single model
- 3.1% gain over comparable ResNet model

Open Images

GITHUB.COM/OPENIMAGES



- **5000 classes**
- Validation and Test sets:
  - **41k images**
- Training set:
  - **9M images**

Open Images

GITHUB.COM/OPENIMAGES

balcony, stairs, facade, iron, door, interior design, gate, architecture, handrail, baluster, window, arch

cutlery, tableware, metal, tool, spoon, fork

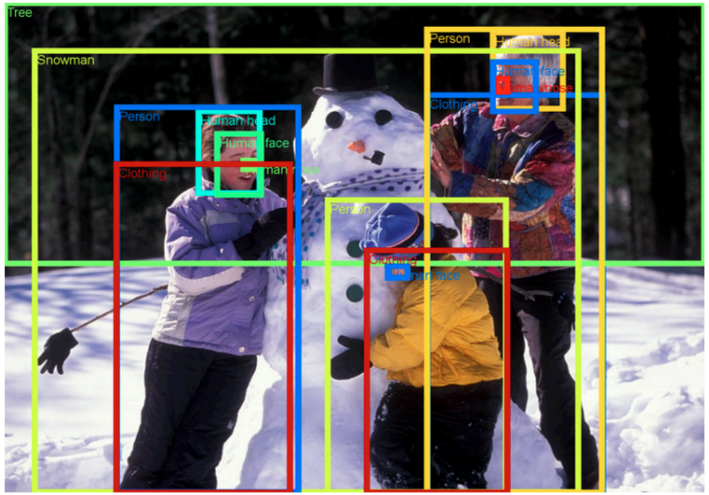
Open Images

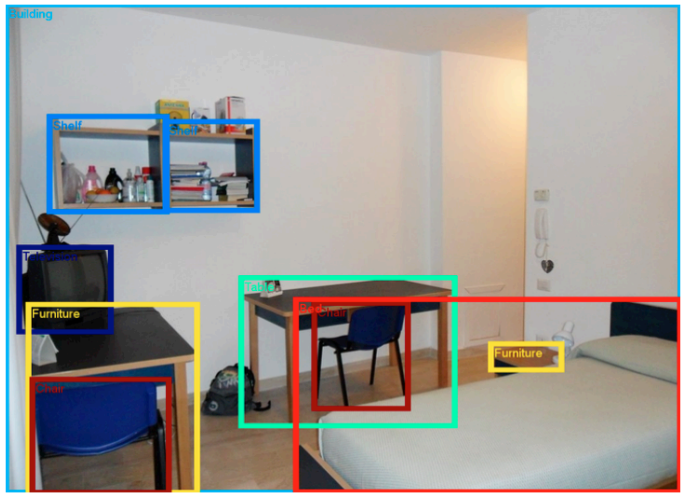
GITHUB.COM/OPENIMAGES

- **600 boxable object classes**
- Validation and Test sets:
  - **830k bounding-boxes on 167k images**
  - all object instances manually boxed
- Training set:
  - **1.2M bounding-boxes**
  - 1 box per class per image
  - produced by human verification technique

Open Images

GITHUB.COM/OPENIMAGES

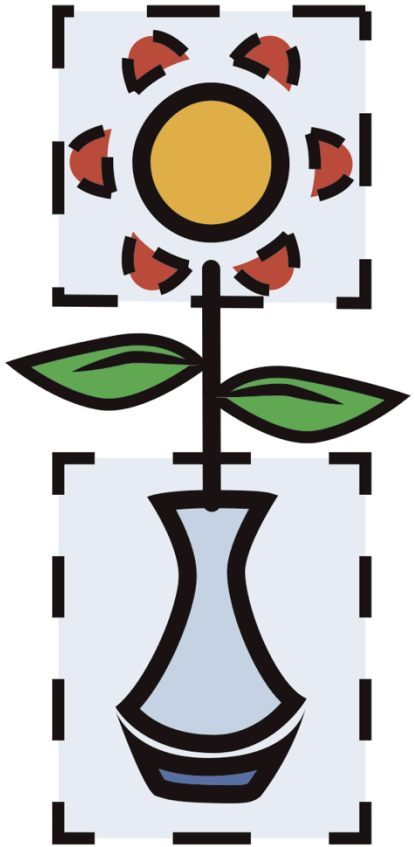




Annotated images from the Open Images dataset. Left: FAMILY MAKING A SNOWMAN by mwvchamber, Right: STANZA STUDENTI.S.S. ANNUNZIATA byersupalermo. Both images used under CC BY 2.0 license.

Open Images

GITHUB.COM/OPENIMAGES



Stay tuned for pre-trained Tensorflow Object Detection models on the Open Images dataset...

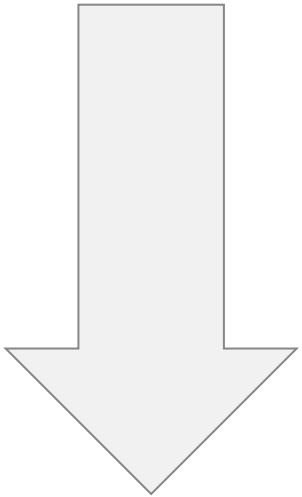
"Krasin I., Duerig T., Alldrin N., Ferrari V., Abu-El-Haija S., Kuznetsova A., Rom H., Uijlings J., Popov S., Veit A., Belongie S., Gomes V., Gupta A., Sun C., Chechik G., Cai D., Feng Z., Narayanan D., Murphy K. OpenImages: A public dataset for large-scale multi-label and multi-class image classification, 2017. Available from https://github.com/openimages".

This Talk

- The Tensorflow Object Detection API
- Pushing the envelope on Accuracy
  - Better Models
  - Better Data
- **Pushing the envelope on Speed**
  - **Lightweight Models**
  - Adaptive Inference

Vision Models are Getting Bigger and Better

- AlexNet
- GoogleNet, VGG
- Inception V2, V3
- ResNet
- Inception ResNet



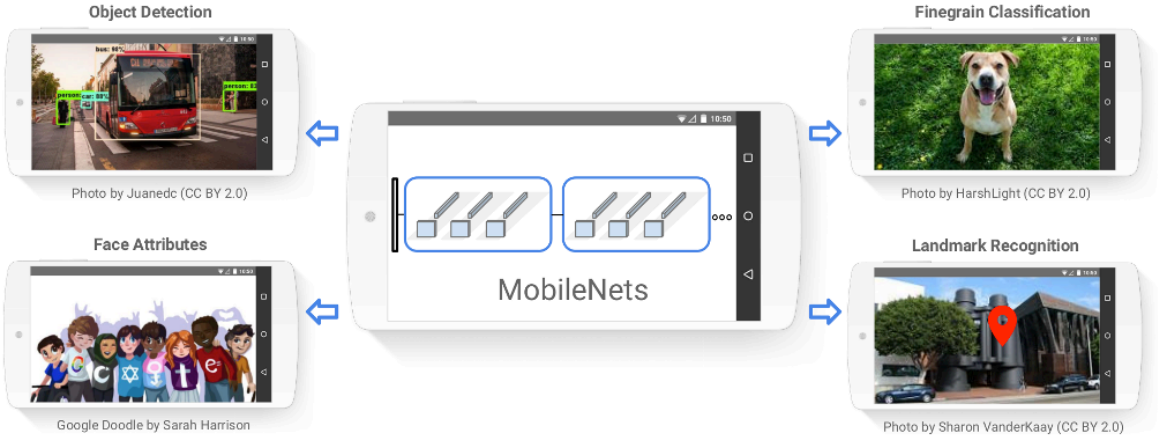
Bigger and Better

Example: GoogleNet

[slide credit: Andrew Howard]

Mobile Vision at Google

Long-Term Goal: Enable Computer Vision on Mobile Devices.



This means optimizing for:

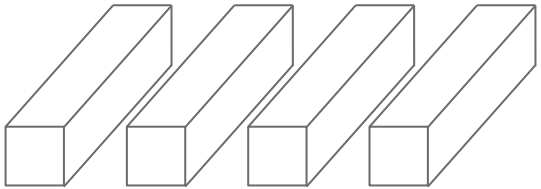
- ~~Latency~~
- Power Consumption
- Memory Footprint
- Disk Footprint - Binary and Model Size

[slide credit: Andrew Howard]

Convolution vs Depthwise Separable Convolutions

Convolution:

Filter and Combine in 1 Step:

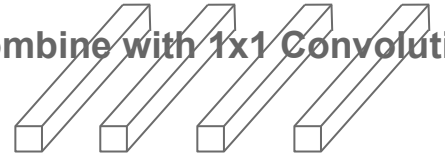


Depthwise Separable Convolution:

Filter and Combine in 2 Steps:

Filter with Depthwise Convolutions:

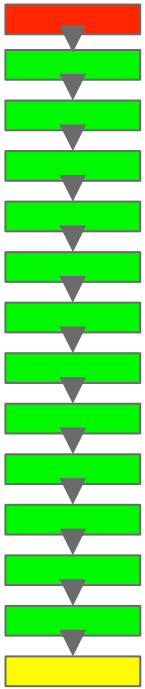
Combine with 1x1 Convolutions:



- Breaks the computational dependence between the size of filters and the number of output channels.
- In practice for 3x3 Convolution, saves ~8x computation at the cost of ~1% accuracy.

[slide credit: Andrew Howard]

MobileNet Architecture



Type	Output Depth	Output Resolution
Convolution	32	112
Separable Convolution	64	112
Separable Convolution	128	56
Separable Convolution	128	56
Separable Convolution	256	28
Separable Convolution	256	28
Separable Convolution	512	14
Separable Convolution	512	14
Separable Convolution	512	14
Separable Convolution	512	14
Separable Convolution	512	14
Separable Convolution	512	14
Separable Convolution	1024	7
Separable Convolution	1024	7
Avg Pool + FC	1000	1

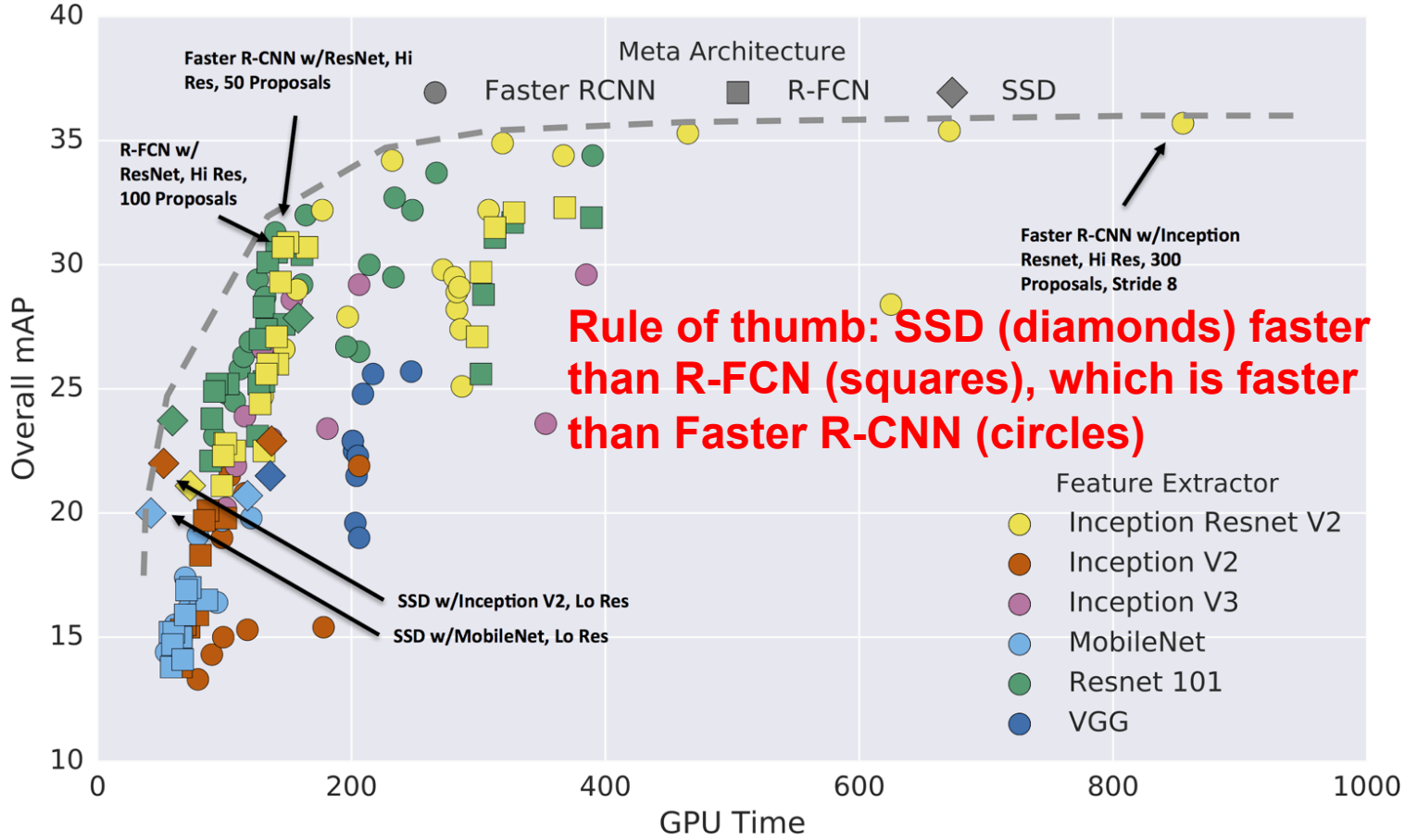
- 95% of computation is 1x1 convolutions efficiently implemented with GEMMs.

[slide credit: Andrew Howard]

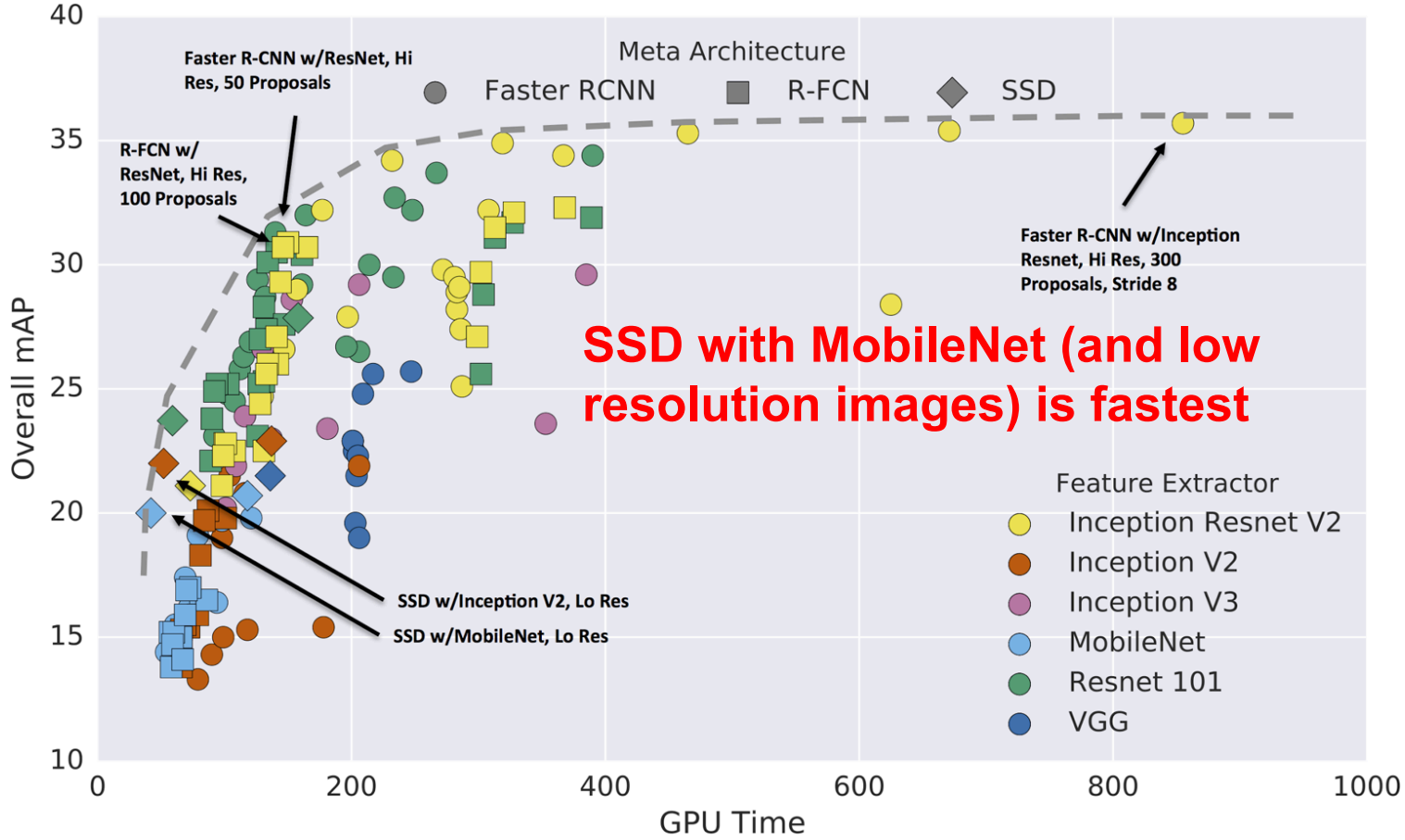
MobileNet on Imagenet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

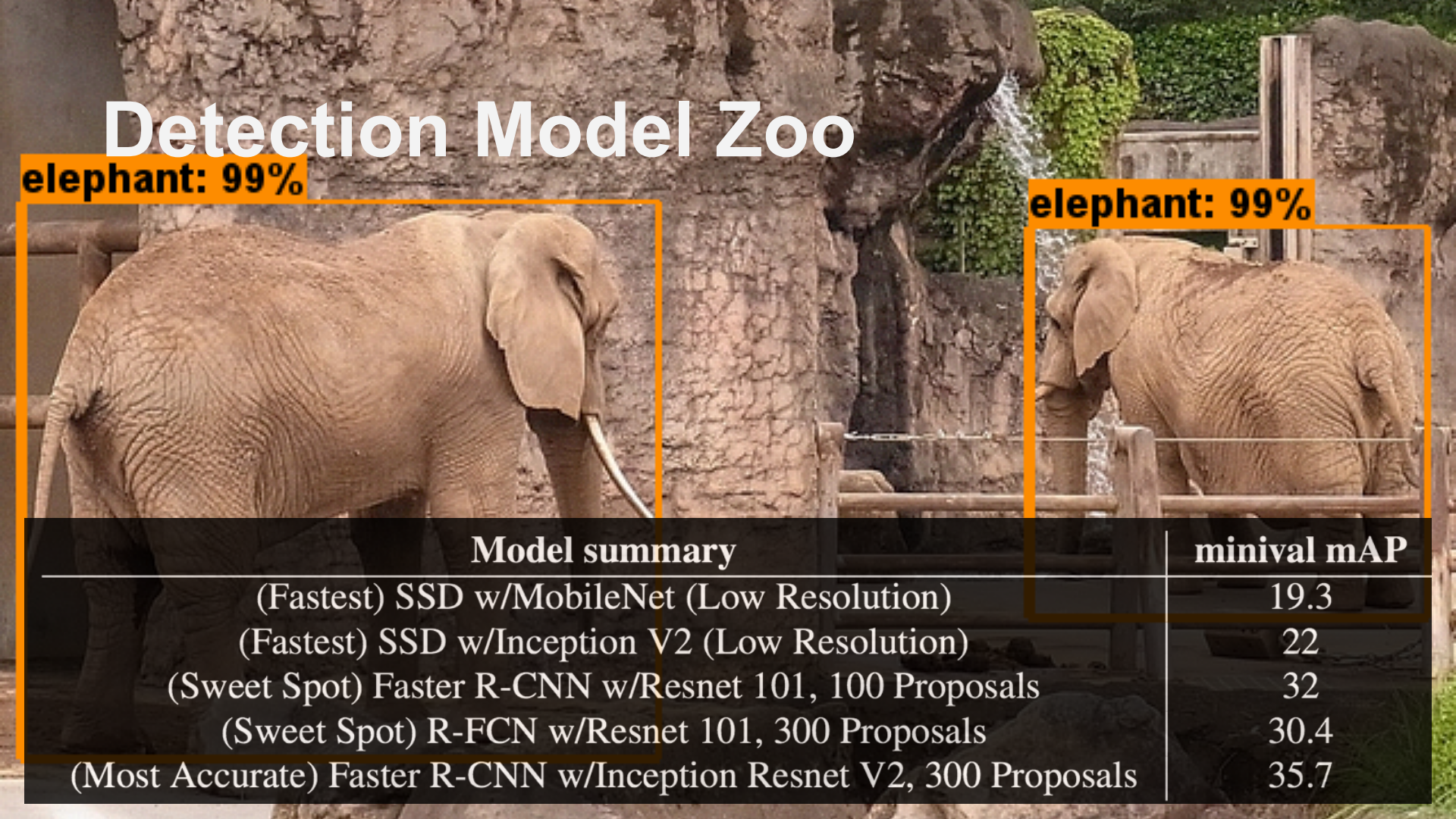


Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. Speed/accuracy trade-offs for modern convolutional object detectors. CVPR 2017



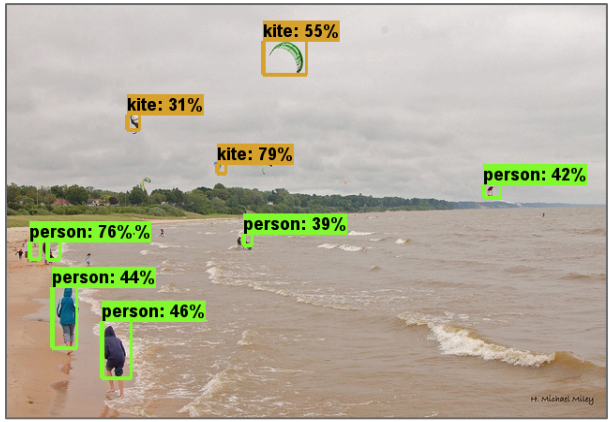
Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., ... & Murphy, K. Speed/accuracy trade-offs for modern convolutional object detectors. CVPR 2017

Detection Model Zoo

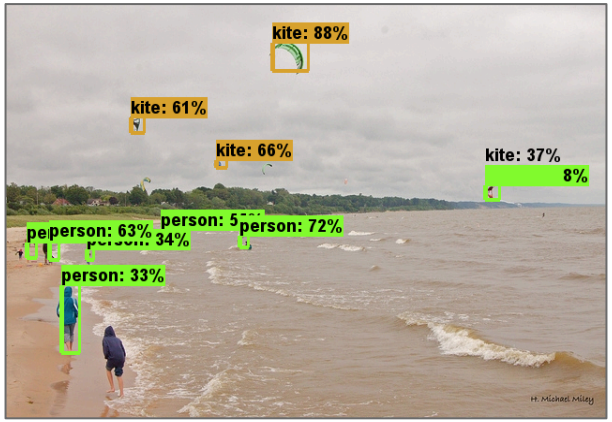


Model summary	minival mAP
(Fastest) SSD w/MobileNet (Low Resolution)	19.3
(Fastest) SSD w/Inception V2 (Low Resolution)	22
(Sweet Spot) Faster R-CNN w/Resnet 101, 100 Proposals	32
(Sweet Spot) R-FCN w/Resnet 101, 300 Proposals	30.4
(Most Accurate) Faster R-CNN w/Inception Resnet V2, 300 Proposals	35.7

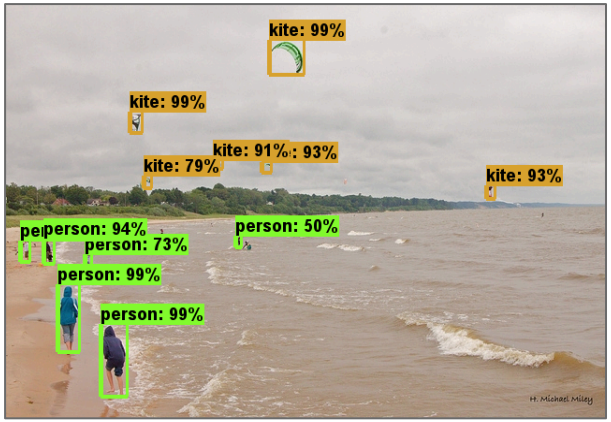
SSD w/MobileNet (Low Resolution)



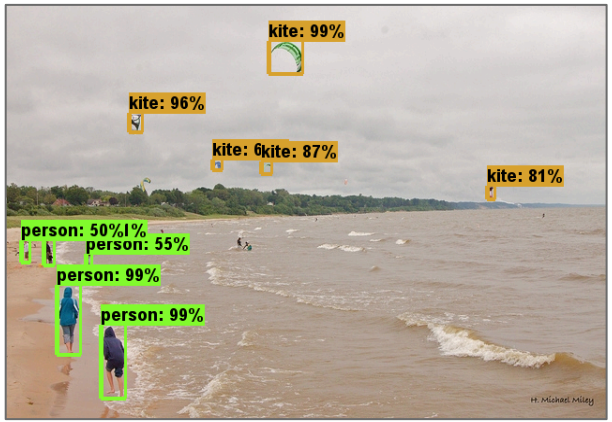
SSD w/Inception V2 (Low Resolution)



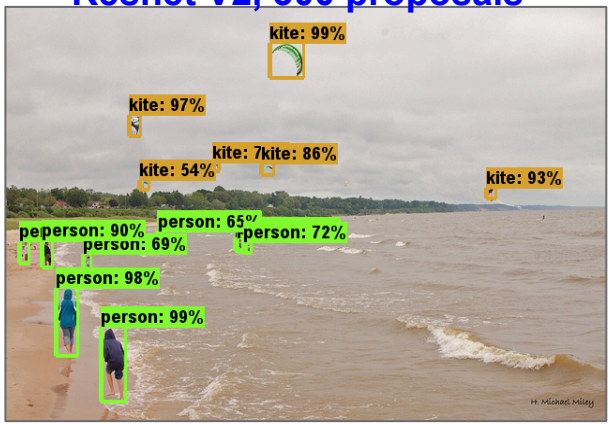
Faster R-CNN w/Resnet101, 100 proposals



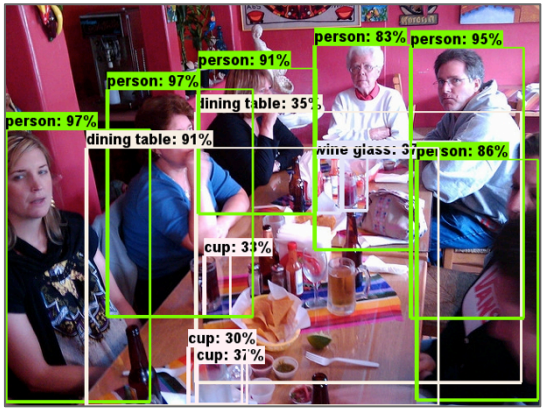
RFCN w/Resnet101, 300



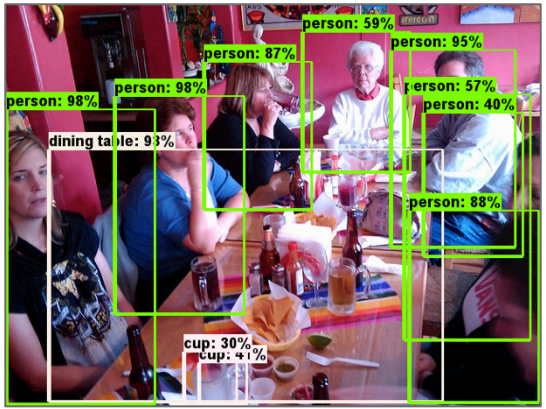
Faster R-CNN w/Inception Resnet V2, 300 proposals



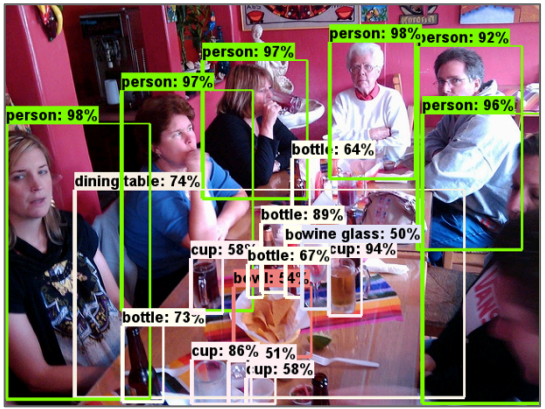
SSD w/MobileNet (Low Resolution)



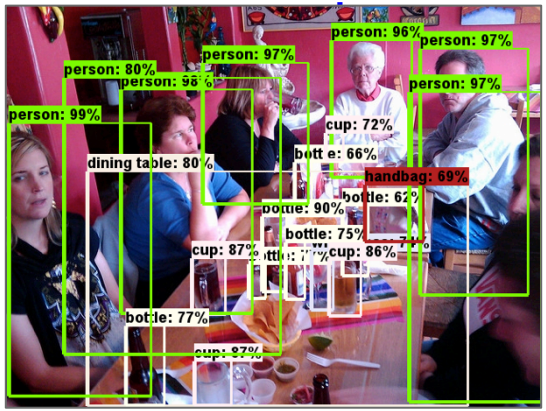
SSD w/Inception V2 (Low Resolution)



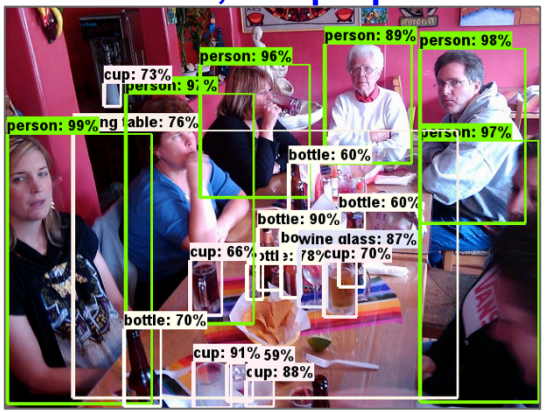
Faster R-CNN w/Resnet101, 100 proposals



RFCN w/Resnet101, 300



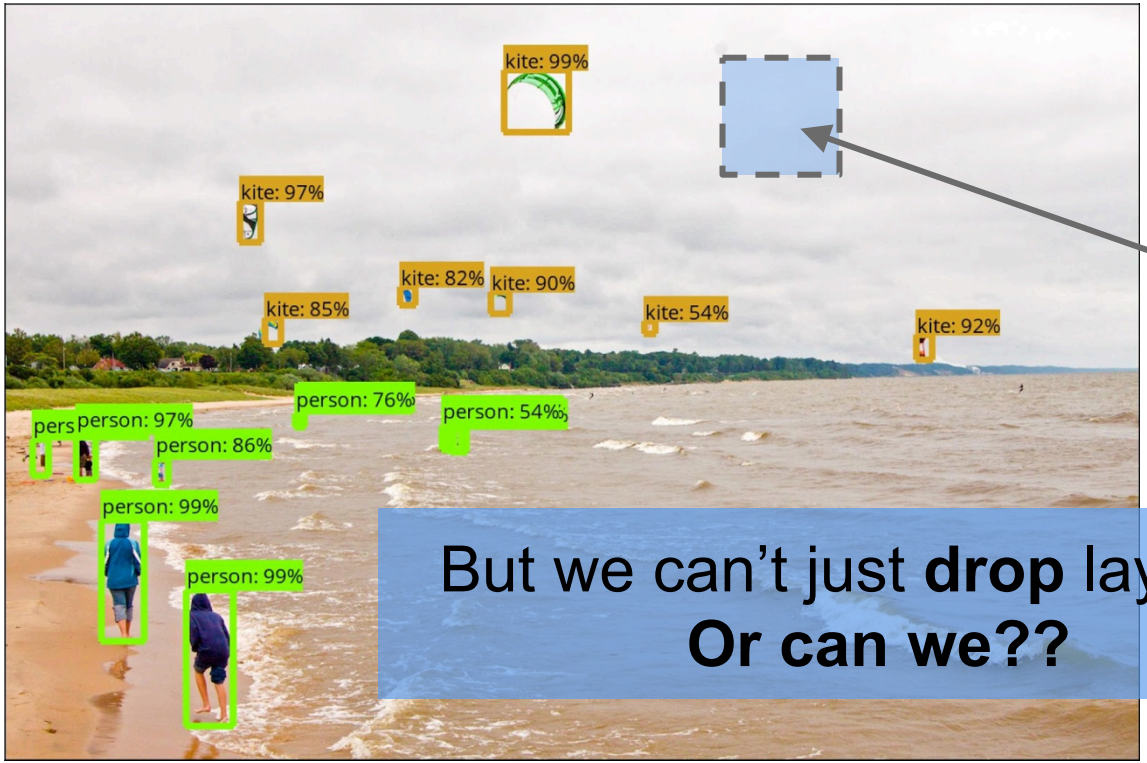
Faster R-CNN w/Inception Resnet V2, 300 proposals



This Talk

- The Tensorflow Object Detection API
- Pushing the envelope on Accuracy
  - Better Models
  - Better Data
- **Pushing the envelope on Speed**
  - Lightweight Models
  - **Adaptive Inference**

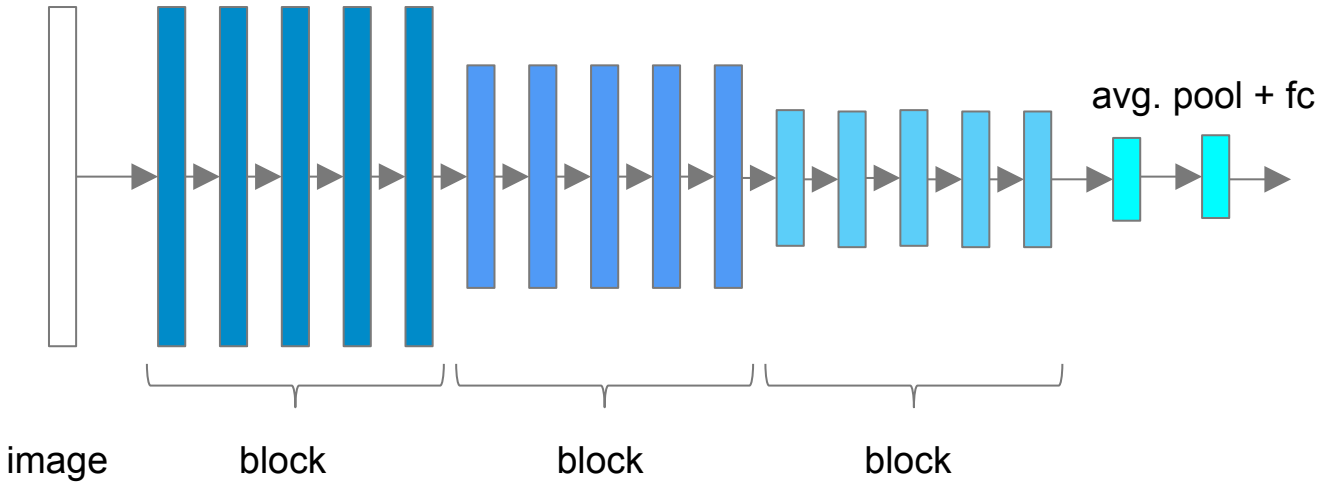
Idea: Not all parts of an image are equally interesting!



Do we really need to pass this region through ~100 layers of our trained CNN?

But we can't just **drop** layers…
Or can we??

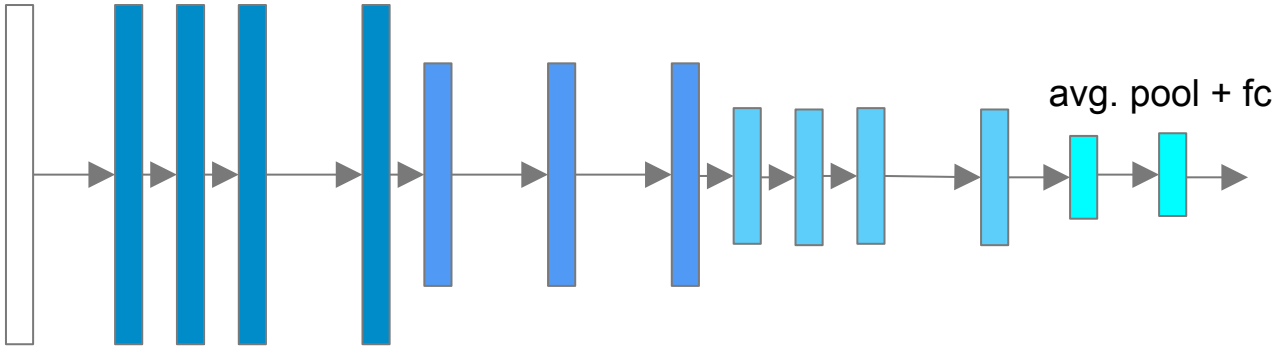
Residual Networks (ResNet)



Residual unit: $y = \mathcal{F}(x) + x$

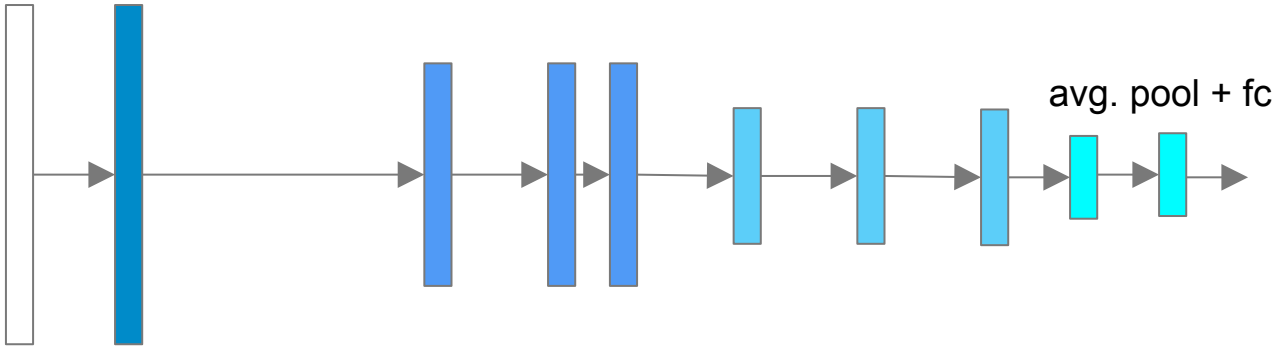
He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

ResNet with Stochastic Depth



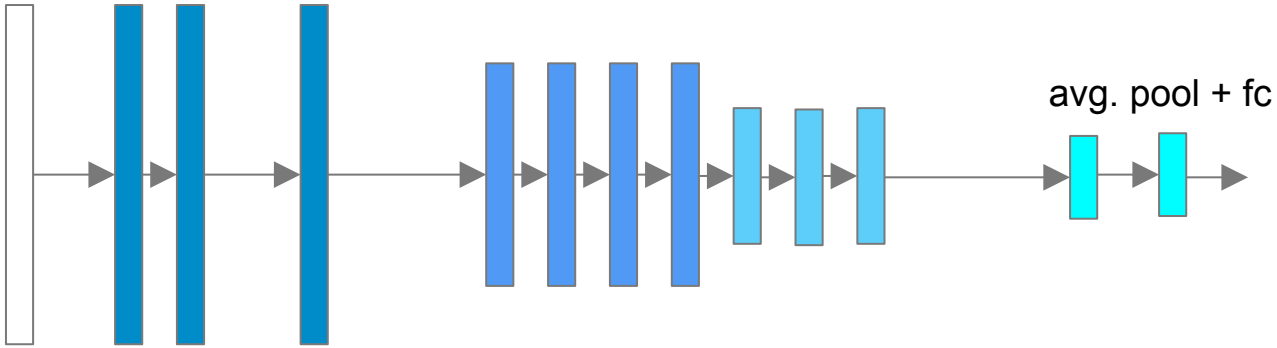
Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016, October). Deep networks with stochastic depth. In *European Conference on Computer Vision* (pp. 646-661). Springer International Publishing.

ResNet with Stochastic Depth



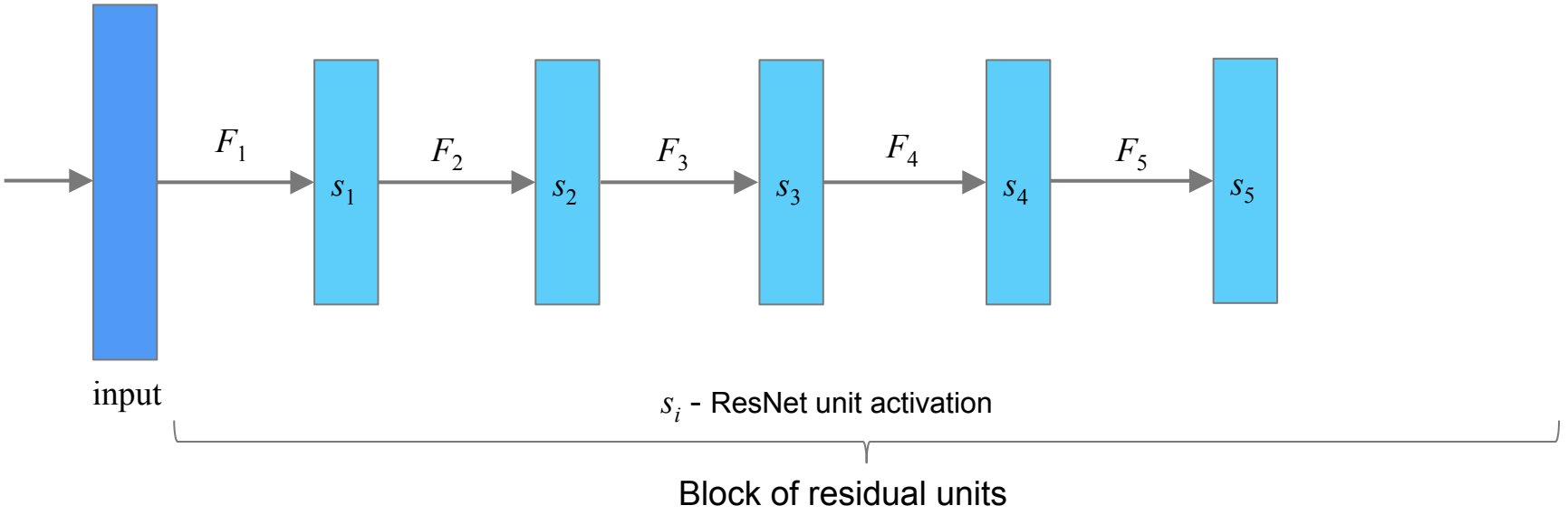
Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016, October). Deep networks with stochastic depth. In *European Conference on Computer Vision* (pp. 646-661). Springer International Publishing.

ResNet with Stochastic Depth



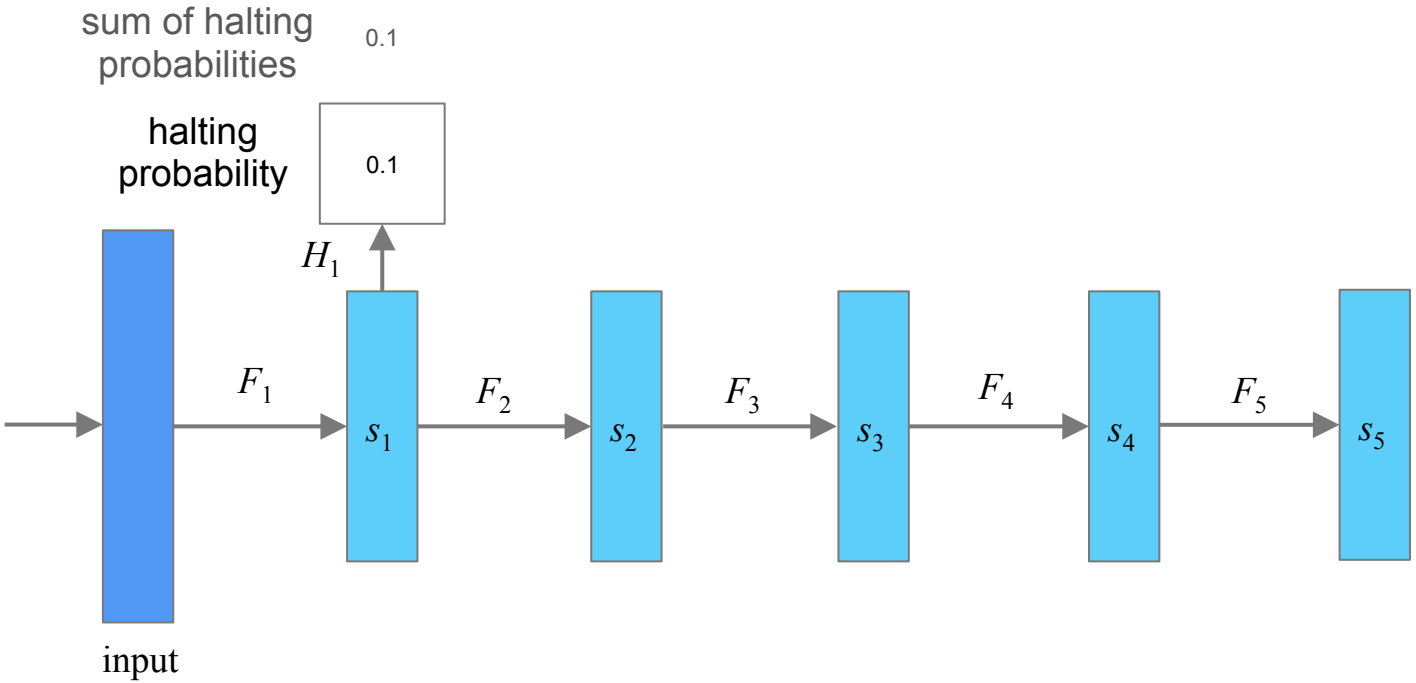
Huang, G., Sun, Y., Liu, Z., Sedra, D., & Weinberger, K. Q. (2016, October). Deep networks with stochastic depth. In *European Conference on Computer Vision* (pp. 646-661). Springer International Publishing.

Adaptive Computation Time for Residual Networks

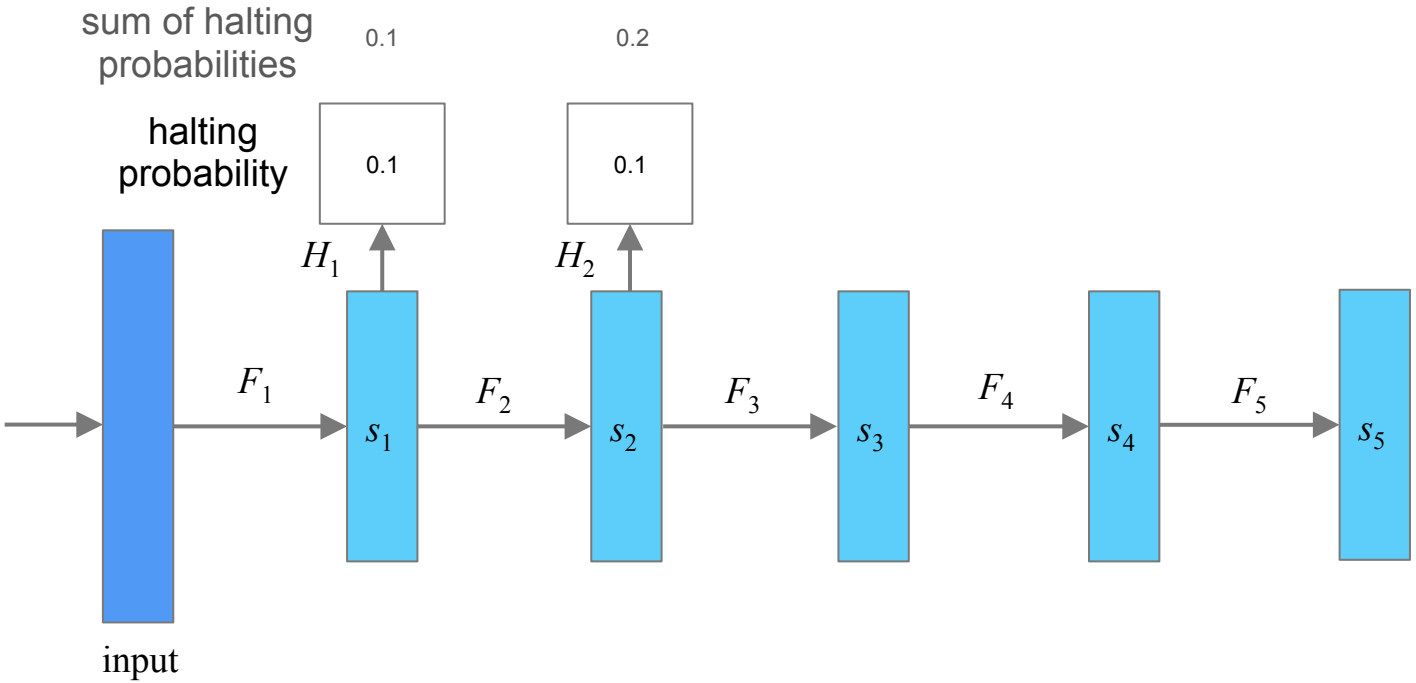


Figurnov, M., Collins, M. D., Zhu, Y., Zhang, L., Huang, J., Vetrov, D., & Salakhutdinov, R. Spatially adaptive computation time for residual networks. CVPR 2017

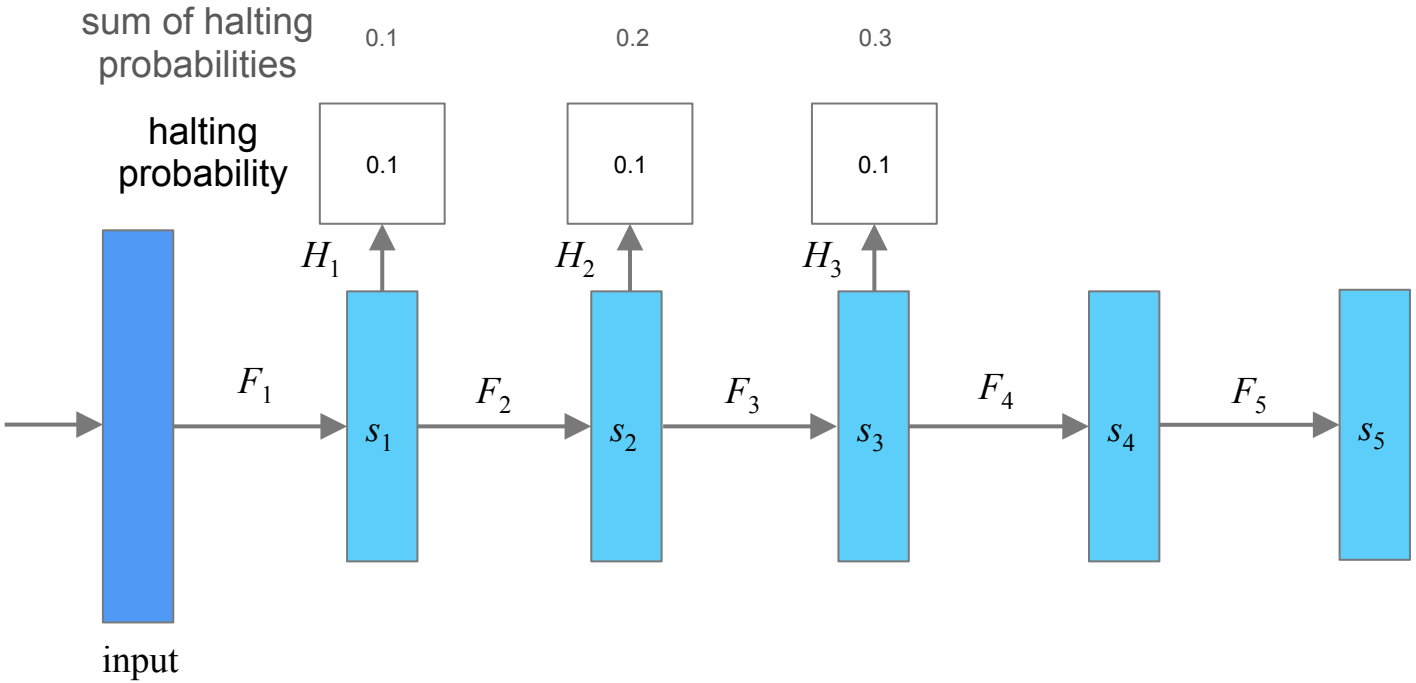
Adaptive Computation Time for Residual Networks



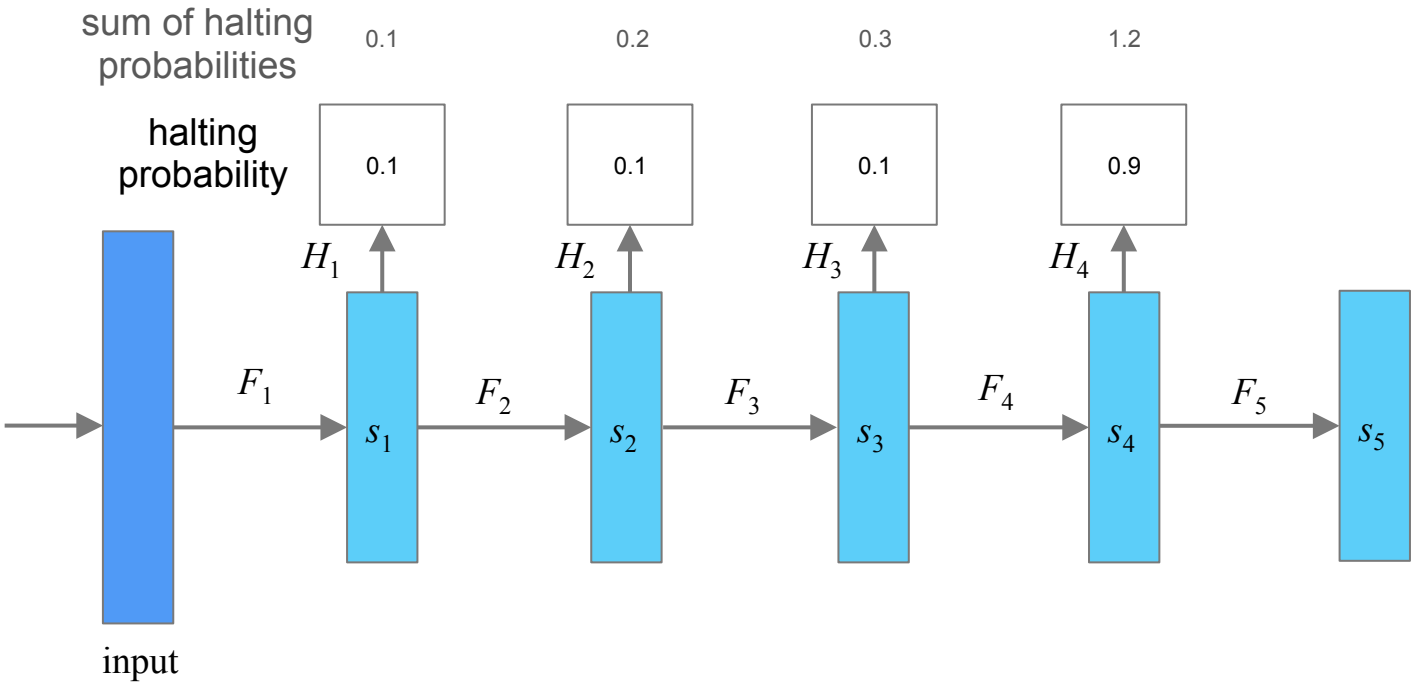
Adaptive Computation Time for Residual Networks



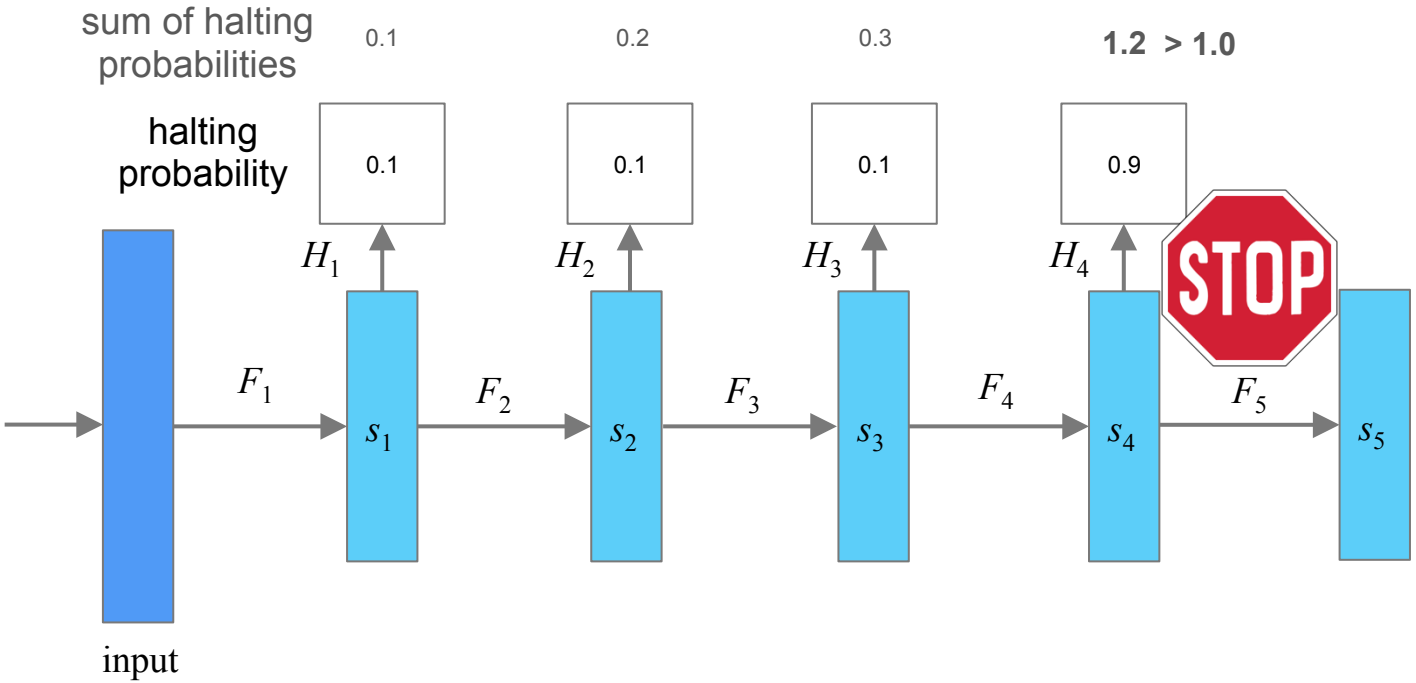
Adaptive Computation Time for Residual Networks



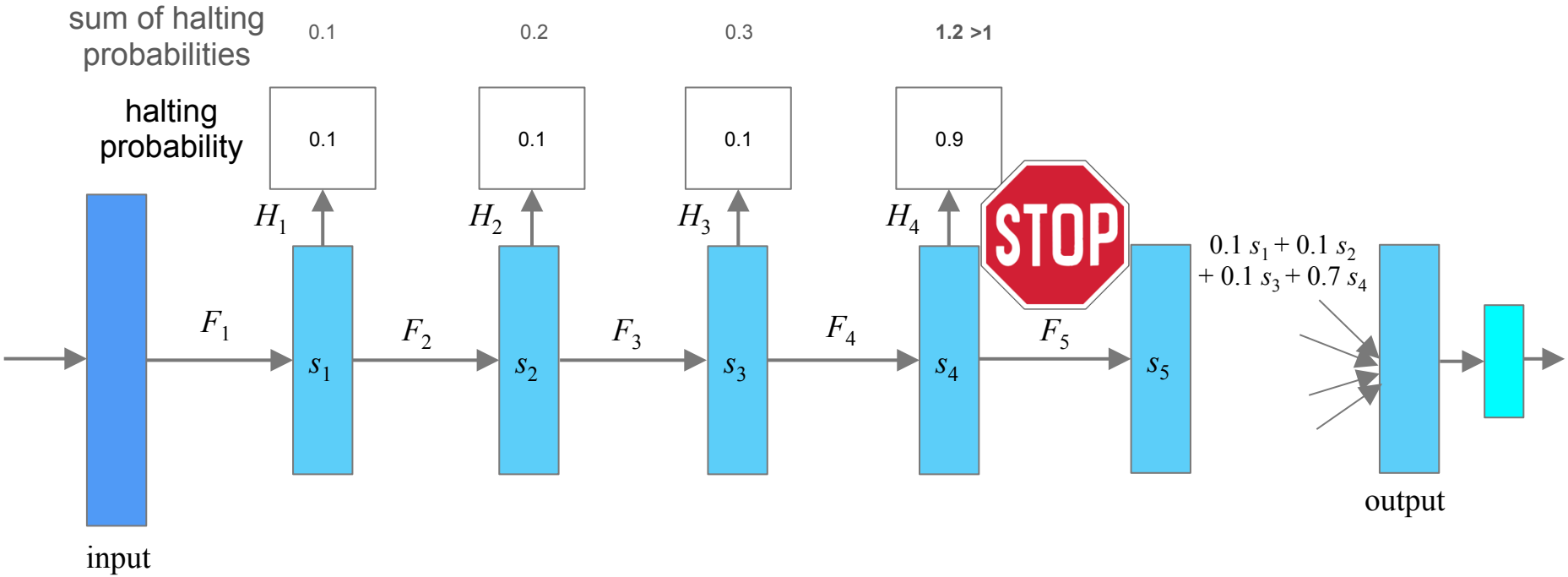
Adaptive Computation Time for Residual Networks



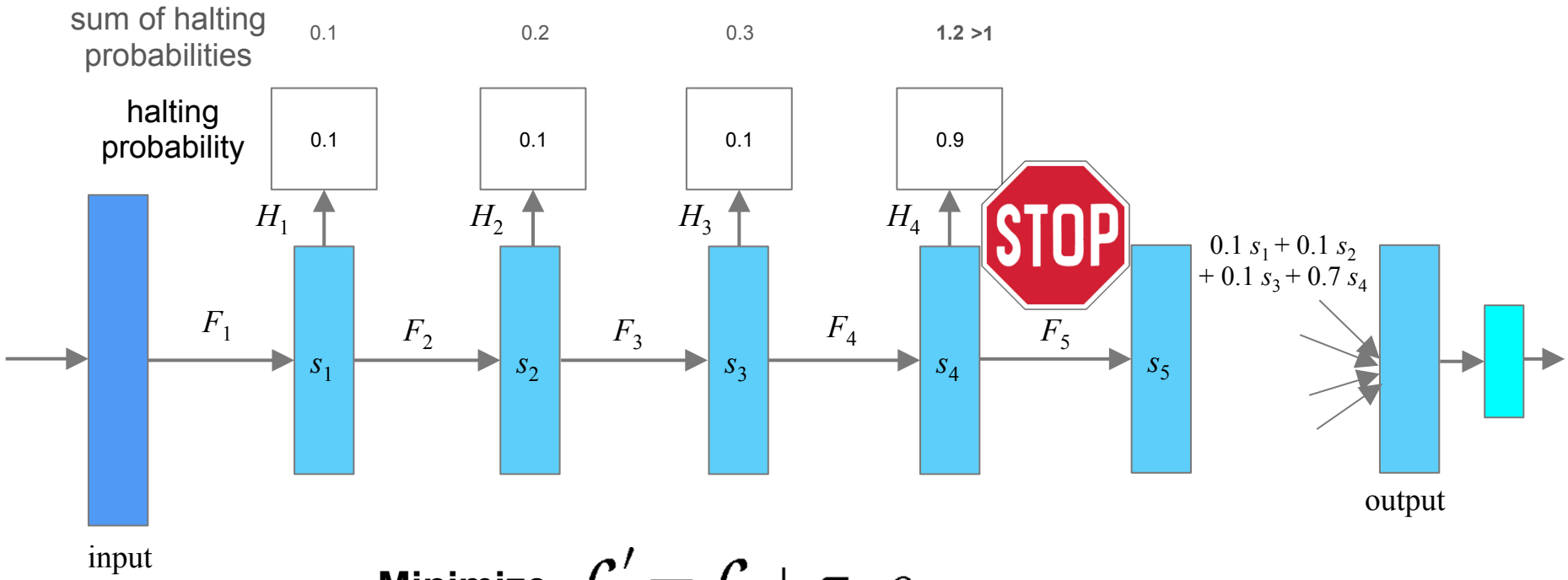
Adaptive Computation Time for Residual Networks



Adaptive Computation Time for Residual Networks



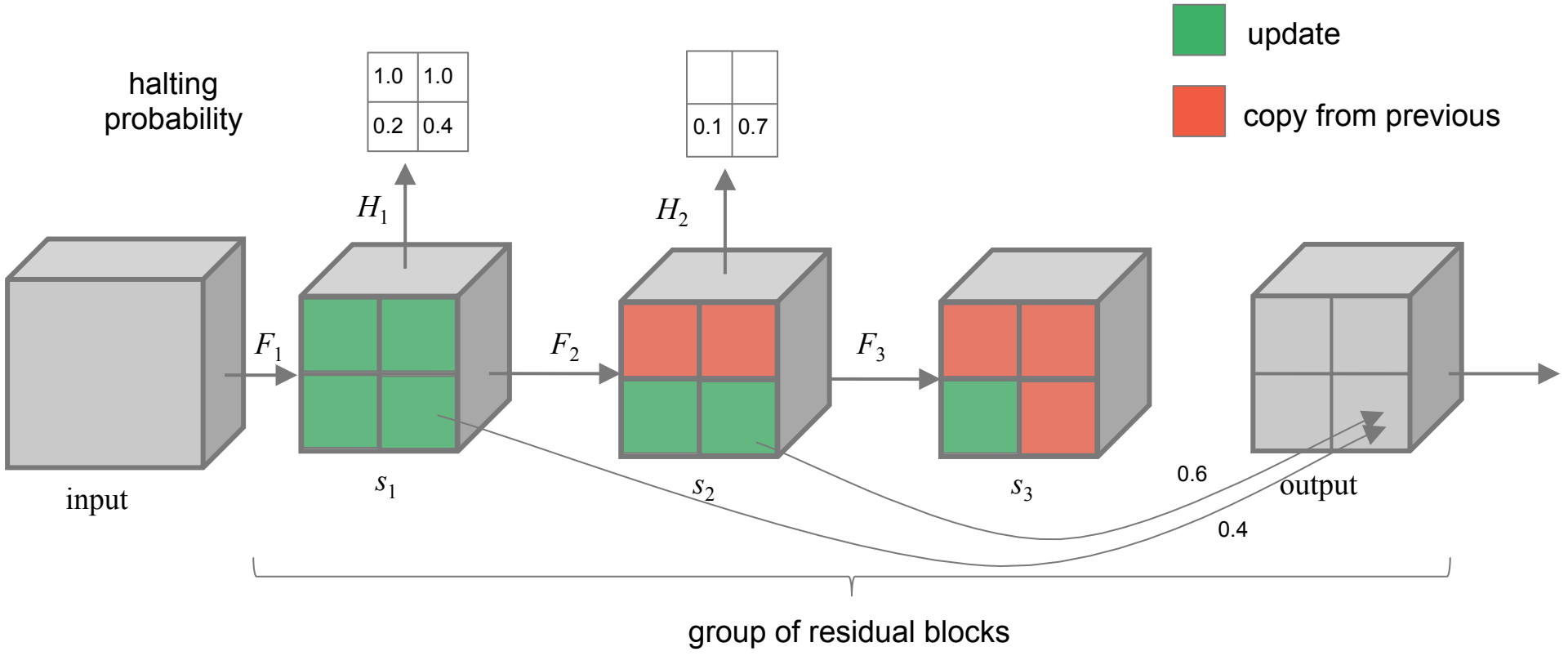
Adaptive Computation Time for Residual Networks



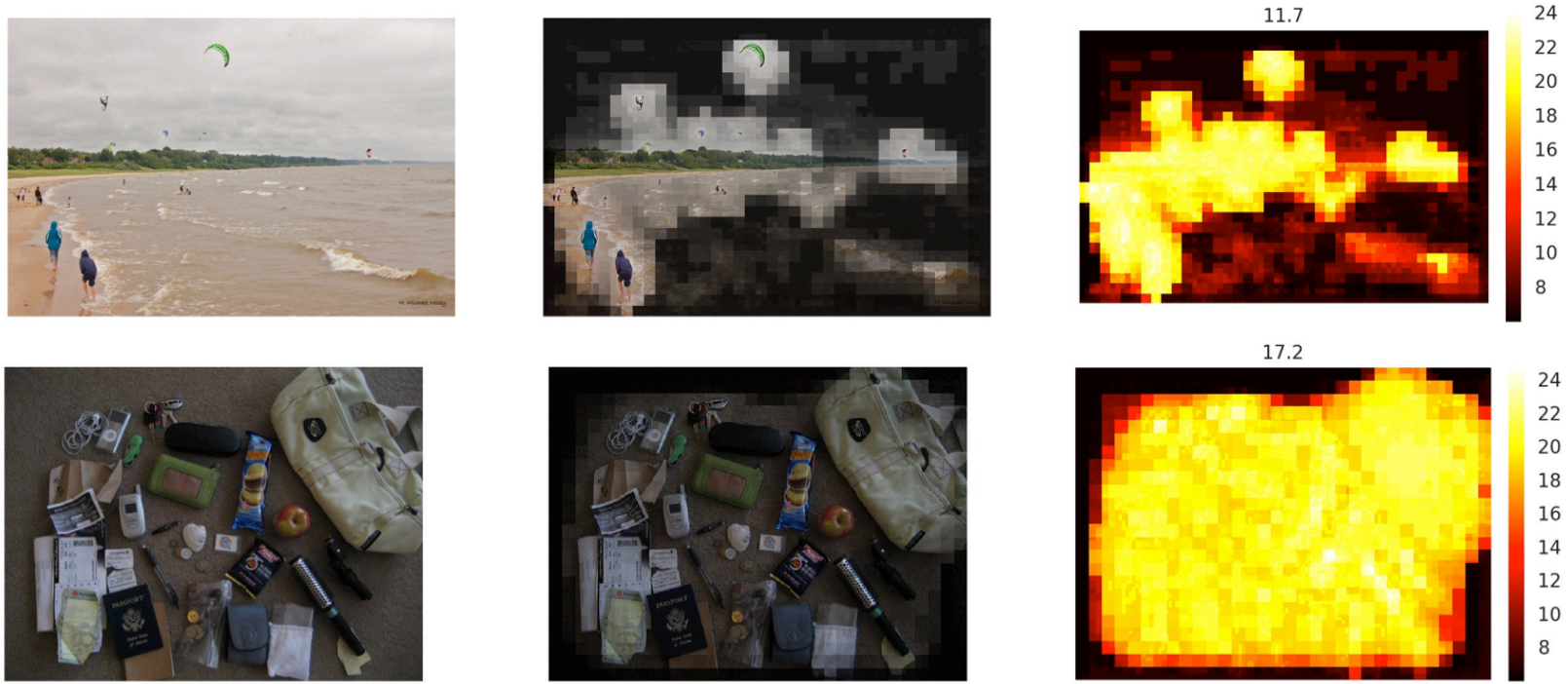
Minimize $\mathcal{L}' = \mathcal{L} + \tau \cdot \rho$

Ponder cost: (differentiable) upper bound on number of layers processed

Spatially Adaptive Computation Time for Residual Networks



Spatially Adaptive Computation Time for Residual Networks [Figurnov et al '17]



When used with Faster R-CNN, similar accuracy as full ResNet, only 50% of the FLOPs!

Thanks!

SPEAKER
Jonathan Huang (jonathanhuang@google.com)

Special thanks to
Derek Chow, Alireza Fathi, Ian Fischer, Sergio Guadarrama, Jonathan Huang, Anoop Korattikara, Kevin Murphy, Vivek Rathod, Yang Song, Chen Sun, Matt Tang, Zbigniew Wojna, Menglong Zhu, 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, Andrew Howard, Michael Figurnov, Vittorio Ferrari, Victor Gomes, Michael Hays