Object Detection

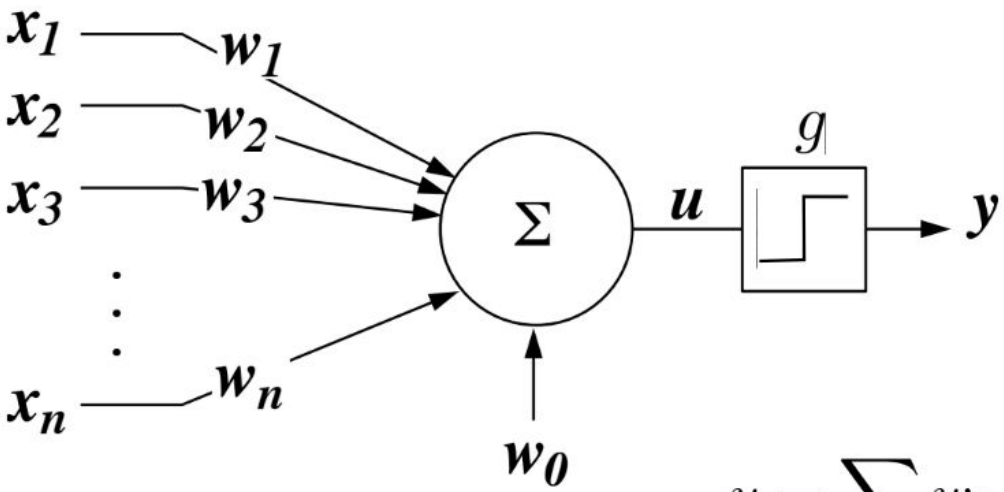
Zbigniew Wojna
University College London
TensorFlight, Inc.

Deep Learning History

TENSORFLIGHT

A brief history of neural networks

1960's



$$u = \sum_i w_i \, x_i$$

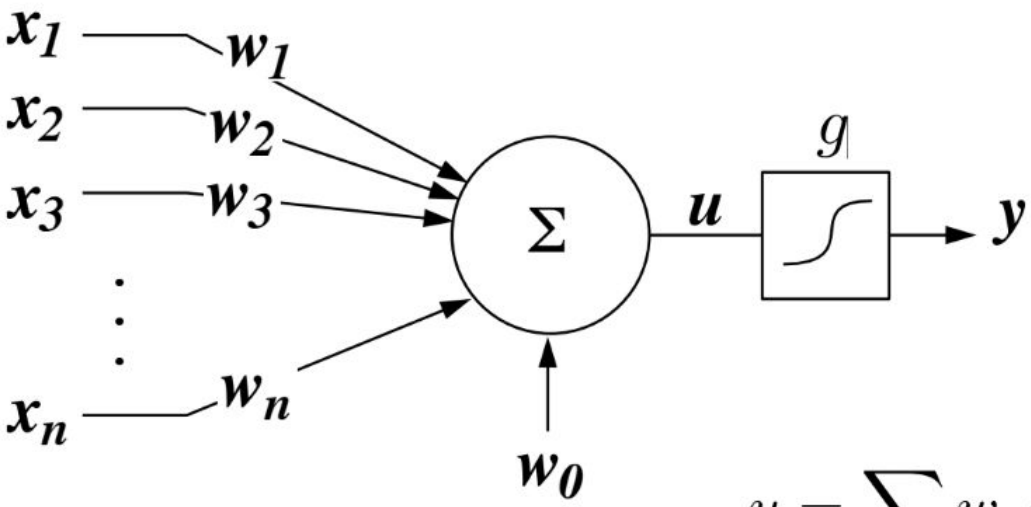
$$y = g(u)$$

https://drive.google.com/a/google.com/file/d/0B_wzP_JlVFcKMUF4aUJNS3F5Tm1pUzhKTEZQV25nYjY1MXVj/view

TENSORFLIGHT

A brief history of neural networks

1980's



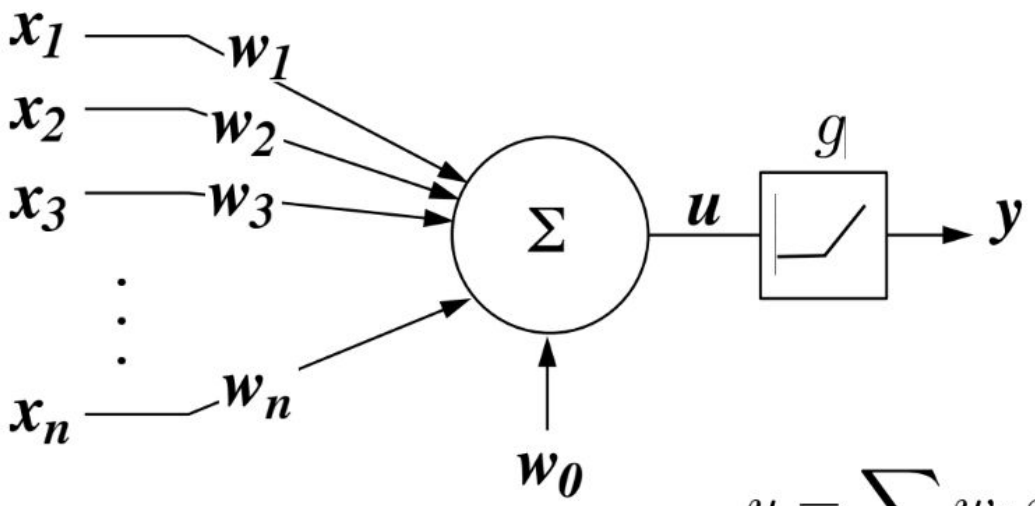
$$u = \sum_i w_i \, x_i$$

$$y = g(u)$$

TENSORFLIGHT

A brief history of neural networks

2000's



$$u = \sum_i w_i \, x_i$$

$$y = g(u)$$

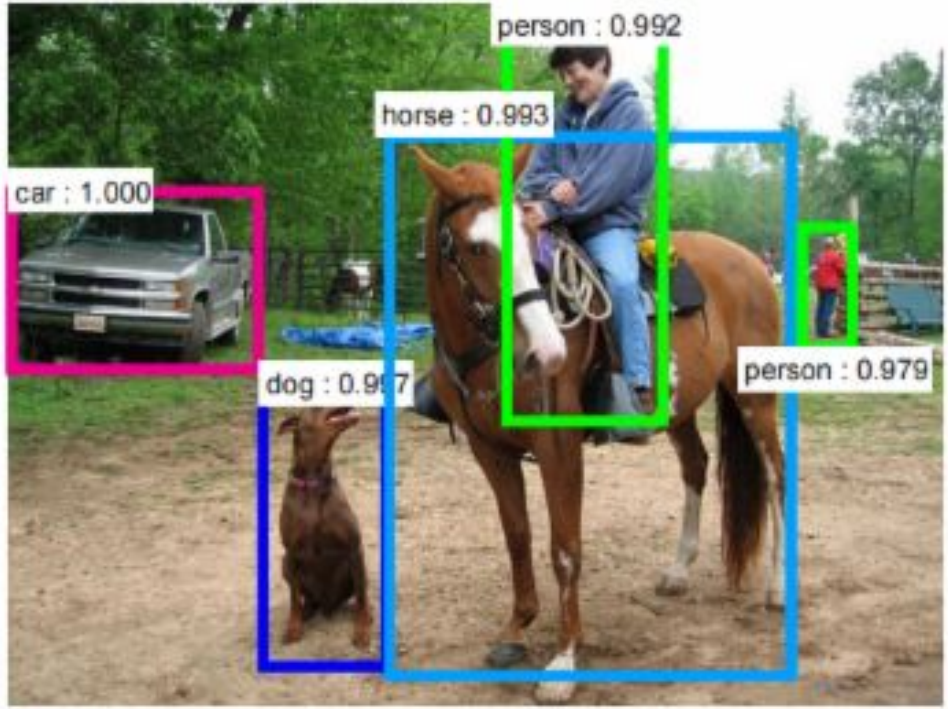
Problems

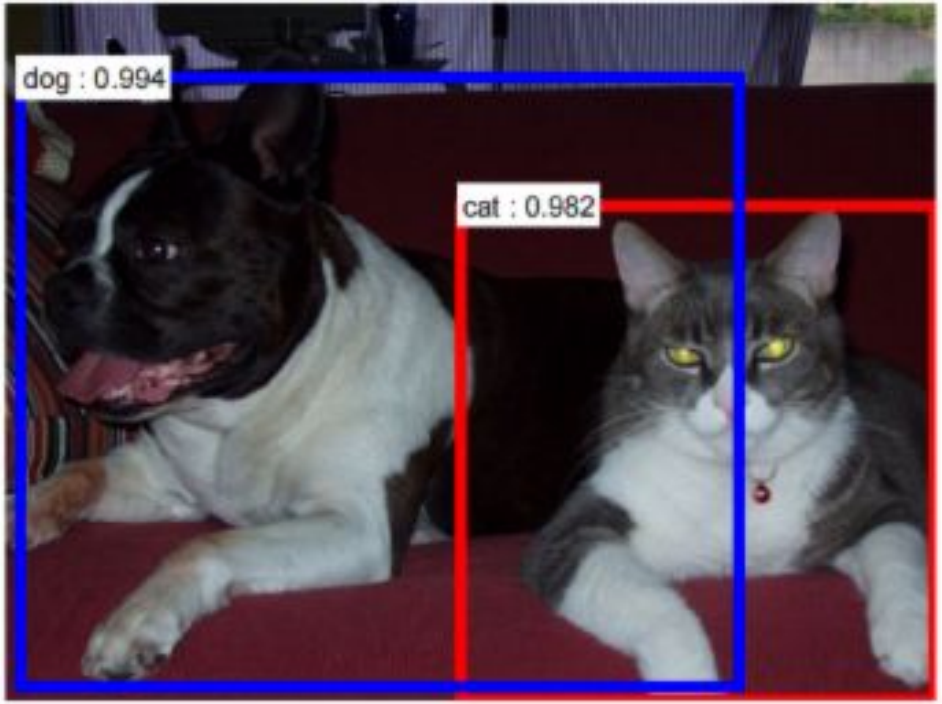
TENSORFLIGHT

Types of problems: Recognition

TENSORFLIGHT

Types of problems: Detection





Types of problems: semantic segmentation

TENSORFLIGHT

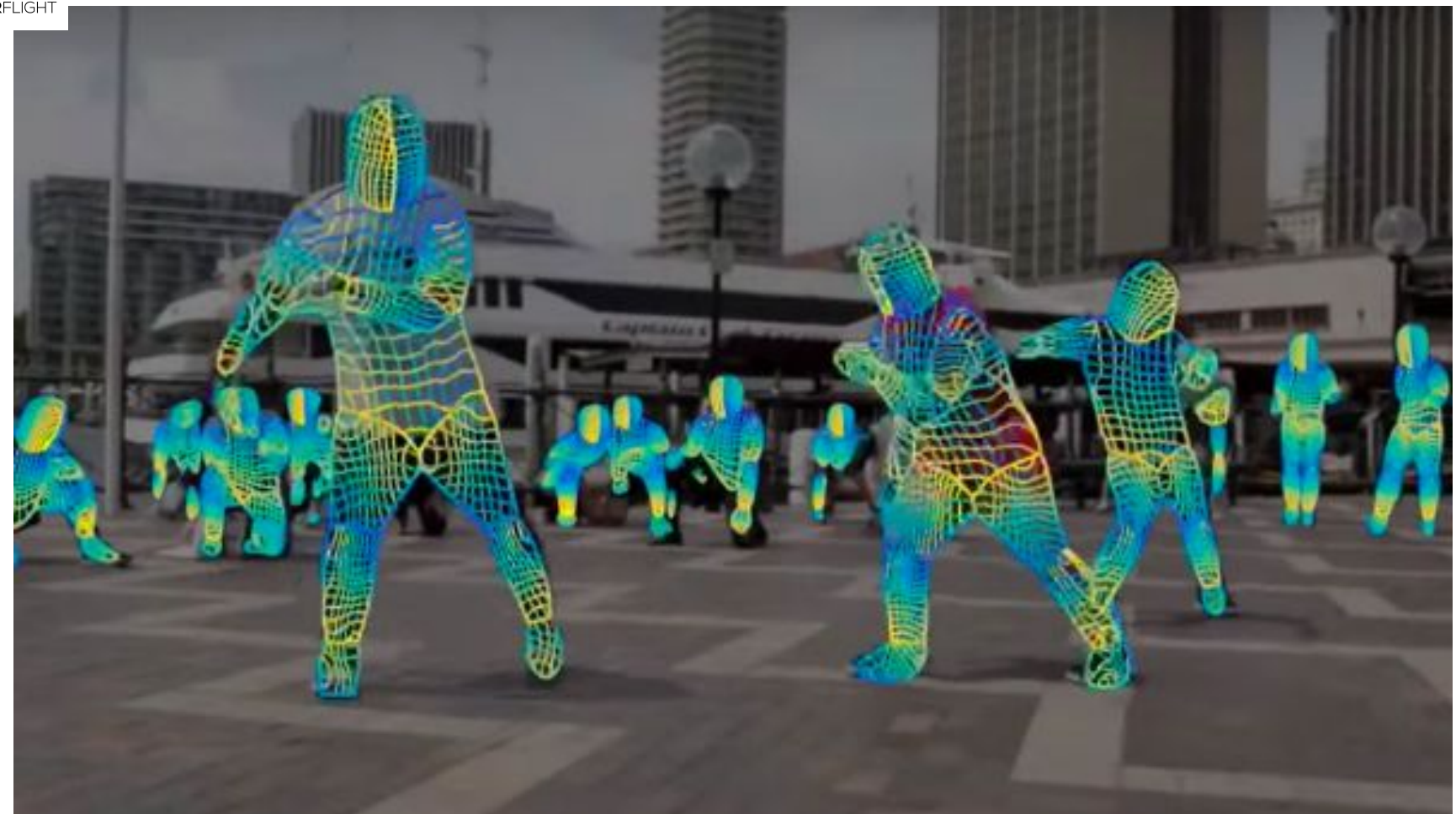
Types of problems: Instance segmentation

TENSORFLIGHT

Types of problems: keypoints detection

TENSORFLIGHT

Types of problems: dense correspondance



TENSORFLIGHT

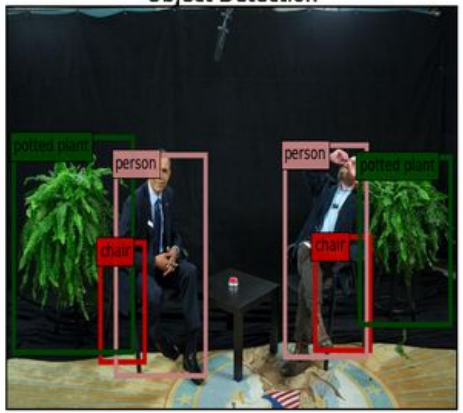
Types of problems: comparison

Image | Class map | Instance map | Part map | Part map (high level)

TENSORFLIGHT

Types of problems: comparison

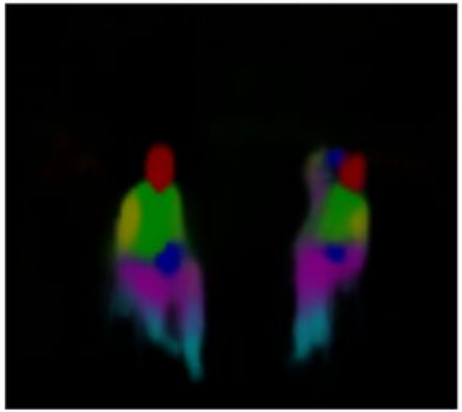
Object Detection



Semantic Segmentation

Semantic Boundaries

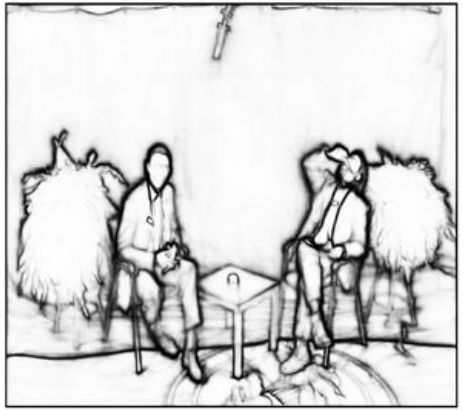
Human Parts



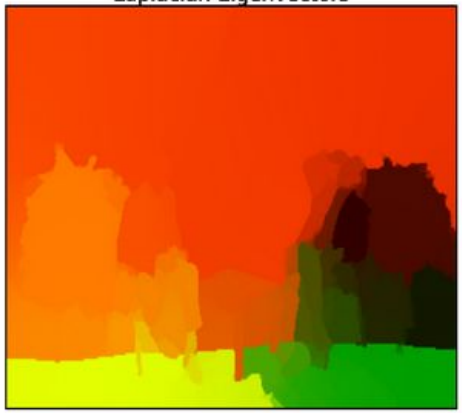
Surface Normals

Saliency

Boundaries



Laplacian Eigenvectors



Dense prediction problems

- **Semantic segmentation**
- Instance segmentation
- Saliency estimation
- **Depth estimation**
- Normal vector estimation
- Stereoscopic images prediction
- Agnostic boundary prediction
- Semantic boundary prediction
- **Agnostic instance boundary detection**
- Semantic instance boundary detection
- Occlusion estimation
- Decoder part in autoencoders

- **Generator network in GANs**
- **Superresolution**
- Image Denoising
- Image Deblurring
- Image Inpainting
- **Image Colorization**
- Optical Flow prediction
- Human parts prediction
- General parts prediction
- Object Detection (densely sampled proposals)
- Key point detection (densely sampled proposals)

Object detection

TENSORFLIGHT

Can you find the head in this image?

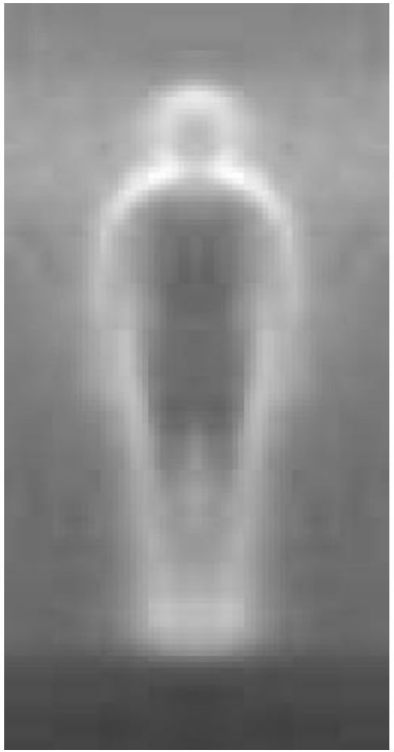
2005

Pedestrian detection (example: Dalal-Triggs, 2005)

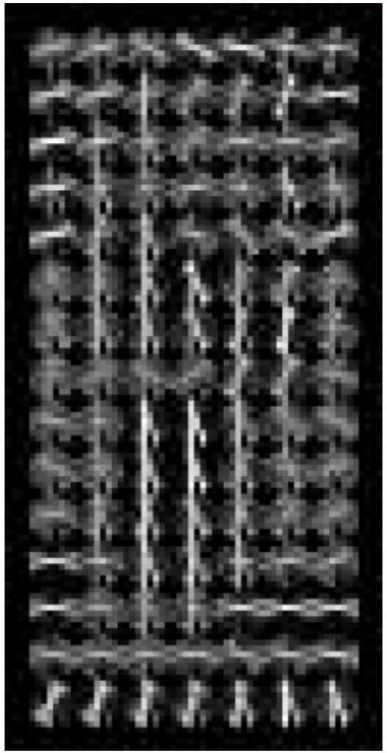
Navneet Dalal and Bill Triggs, Histograms of Oriented Gradients for Human Detection, CVPR05:

1. Extract windows of fixed size (64 x 128) at each position and scale
2. Compute histogram of gradient (HoG) features within each window
3. Compute a score for the window with a linear Support Vector Machine (SVM) classifier
4. Perform non-maximum suppression to remove overlapping detections with lower scores.

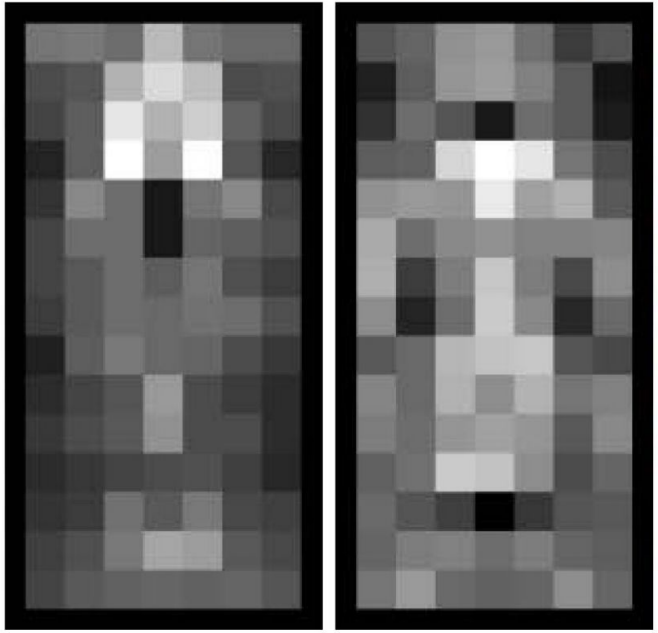
Pedestrian detection (2)



Gradient image



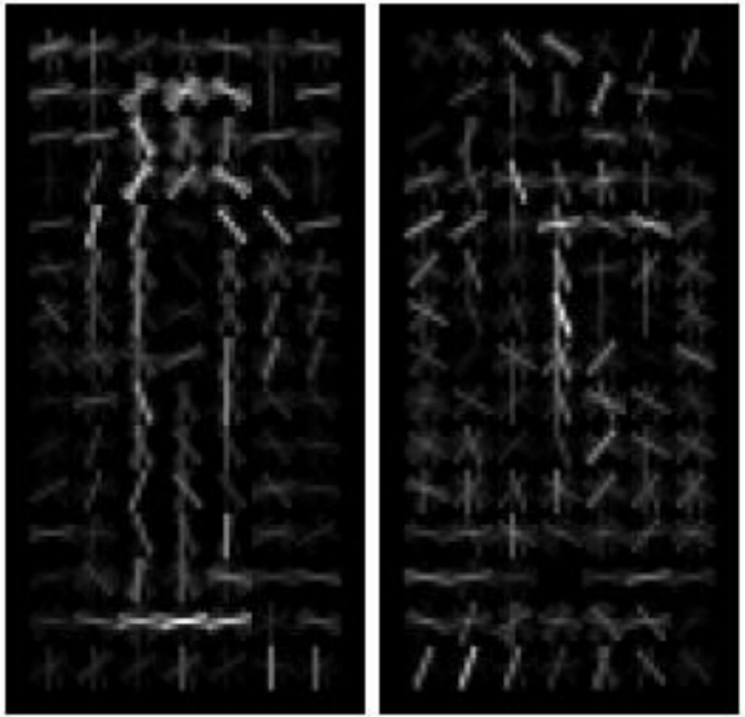
HoG (weighted and interpolated)



SVM weights

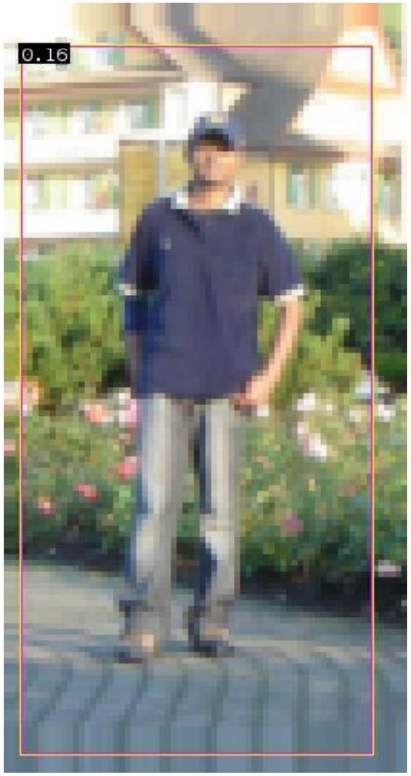
Pedestrian detection (3)

HoG weighted with SVM-weights



Positive weights Negative weights

$$\boldsymbol{w}^t\boldsymbol{x} - b$$



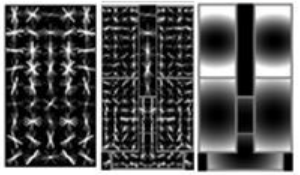
Score = 0.16 > 0 => «Pedestrian»

2010

TENSORFLIGHT

DPM HOG Detector – Object hypothesis

- Bicycle detection process



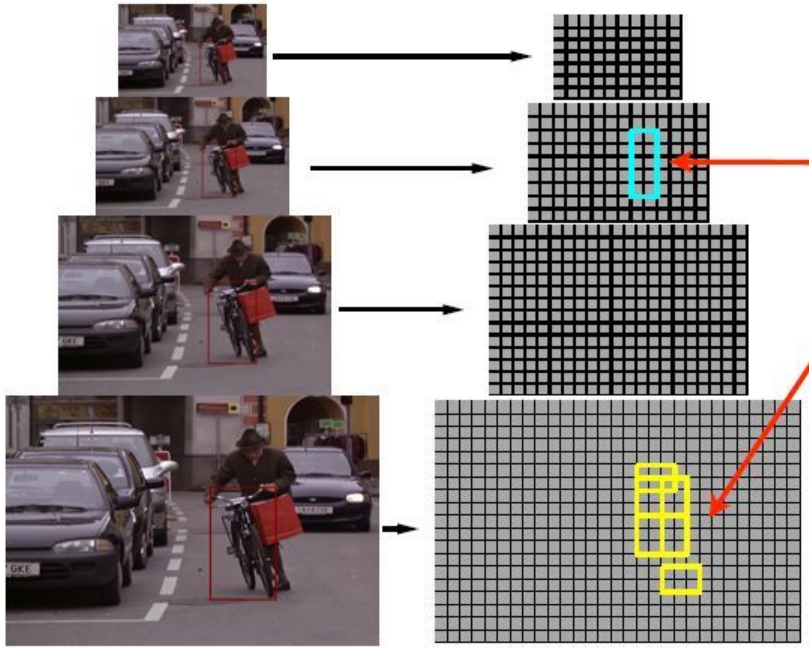


Image pyramid

HOG feature pyramid

$z = (p_0, \ldots, p_n)$

p_0 : location of root

$p_1, \ldots, p_n$: location of parts

Score is sum of filter scores minus deformation costs

P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object Detection with Discriminatively Trained Part Based Models, IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010

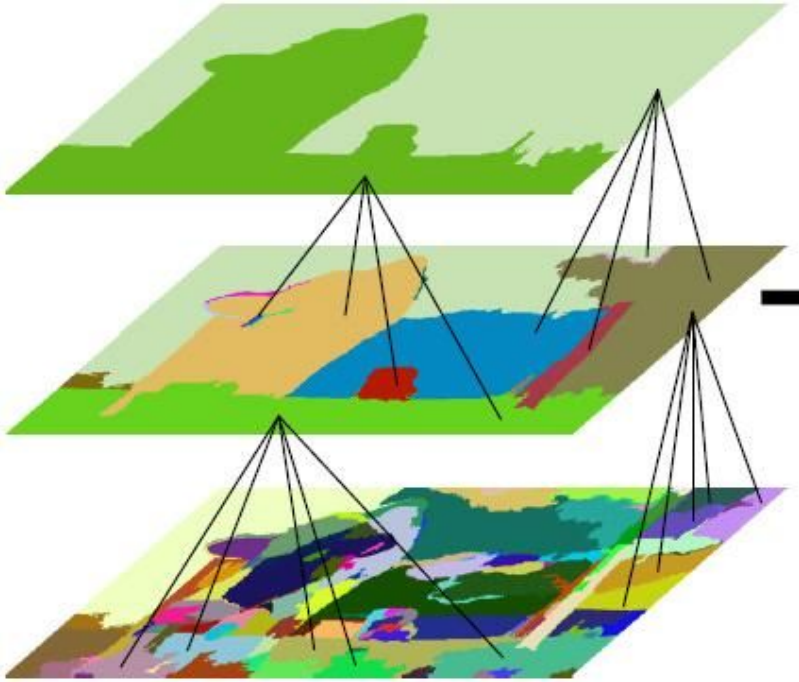
2013 ->

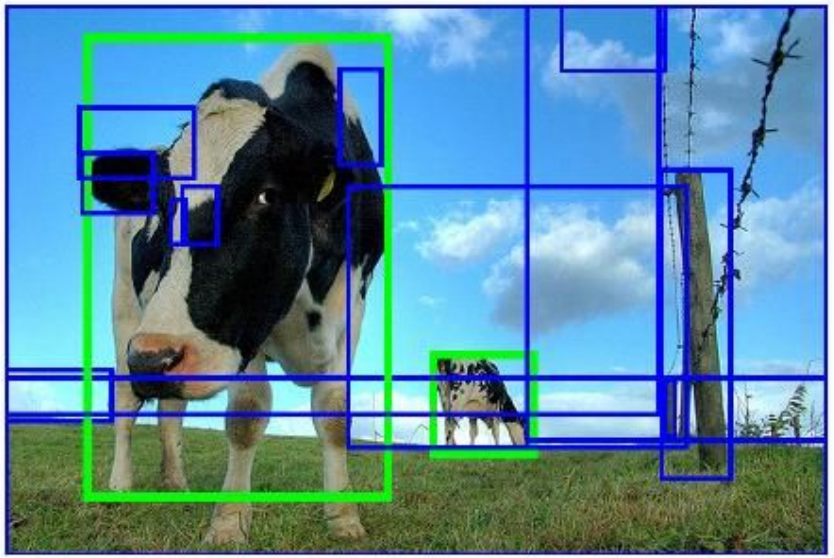
Slides from Stanford

http://cs231n.stanford.edu/slides/2016/winter1516_lecture8.pdf

TENSORFLIGHT

Proposal generation: fast!





https://koen.me/research/pub/uijlings-ijcv2013-draft.pdf

Rich feature hierarchies for accurate object detection and semantic segmentation

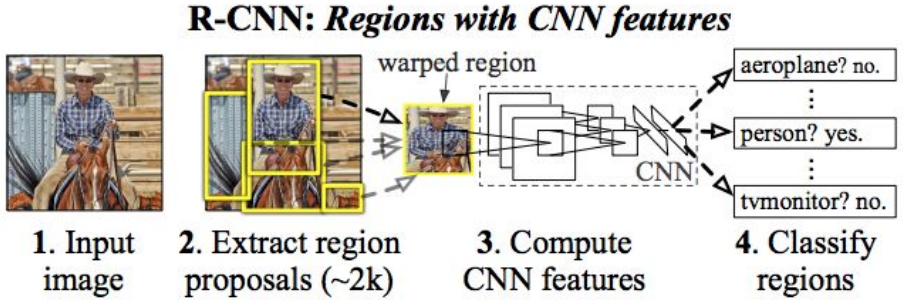


Figure 1: Object detection system overview. Our system (1) takes an input image, (2) extracts around 2000 bottom-up region proposals, (3) computes features for each proposal using a large convolutional neural network (CNN), and then (4) classifies each region using class-specific linear SVMs. R-CNN achieves a mean average precision (mAP) of **53.7% on PASCAL VOC 2010**. For comparison, [39] reports 35.1% mAP using the same region proposals, but with a spatial pyramid and bag-of-visual-words approach. The popular deformable part models perform at 33.4%. On the 200-class **ILSVRC2013 detection dataset, R-CNN's mAP is 31.4%**, a large improvement over OverFeat [34], which had the previous best result at 24.3%.

https://arxiv.org/pdf/1311.2524.pdf

TENSORFLIGHT

Fast R-CNN

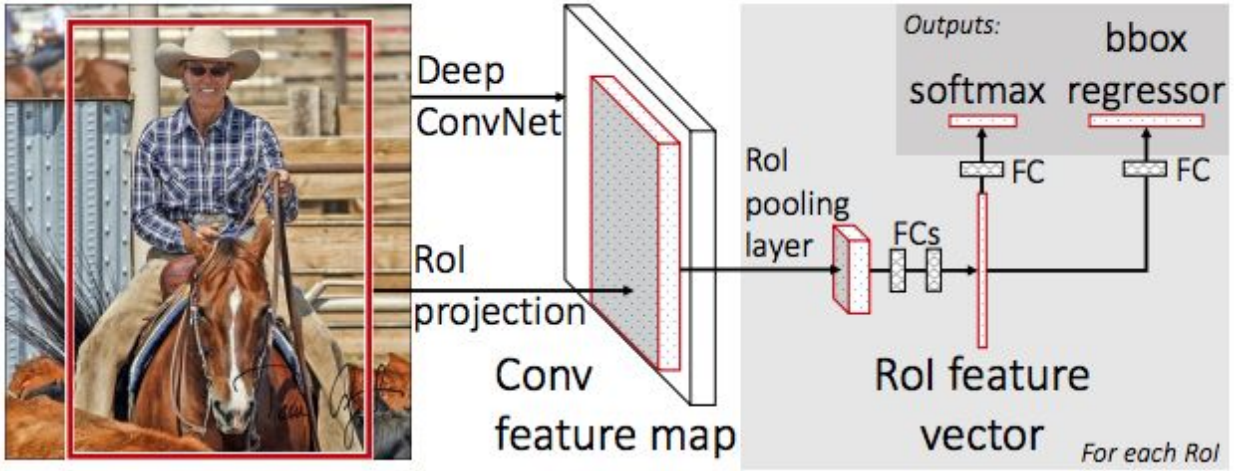


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

https://arxiv.org/pdf/1504.08083.pdf

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

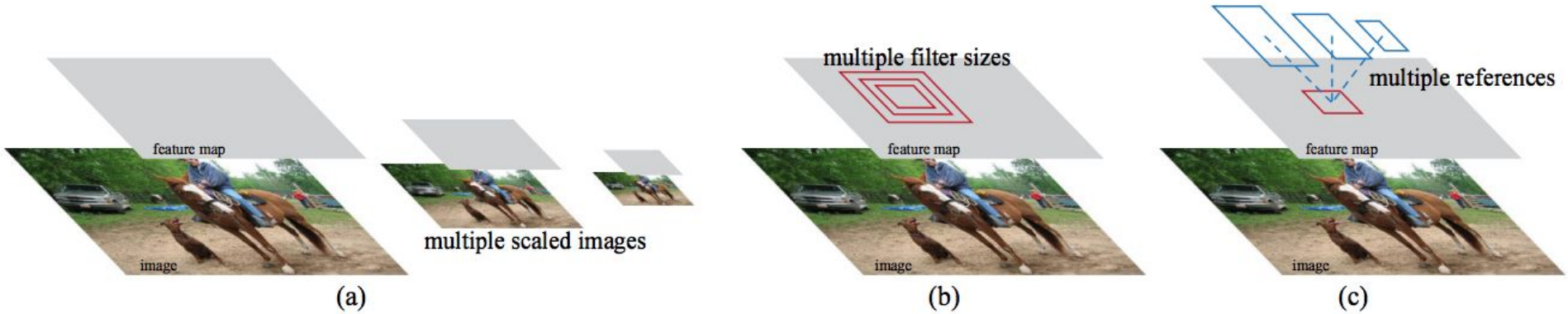


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

https://arxiv.org/pdf/1506.01497.pdf

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

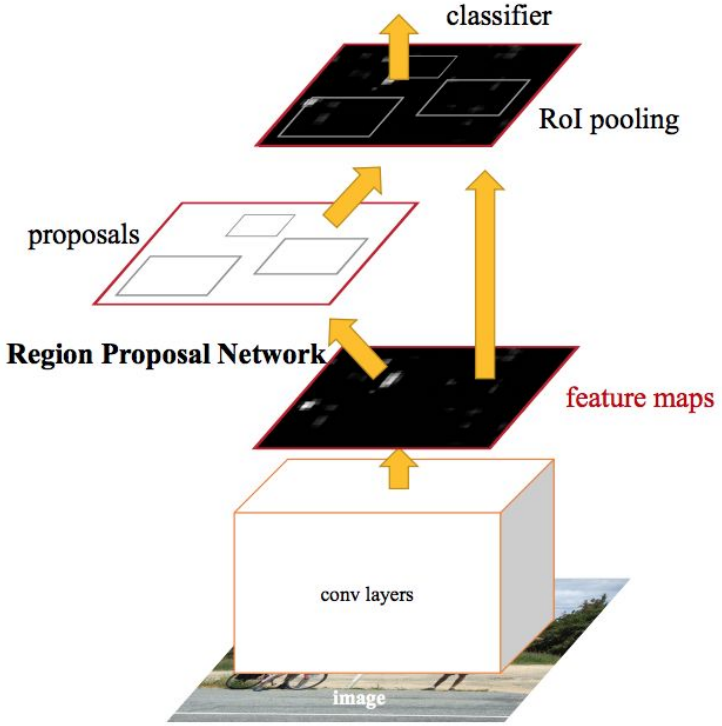


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

https://arxiv.org/pdf/1506.01497.pdf

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

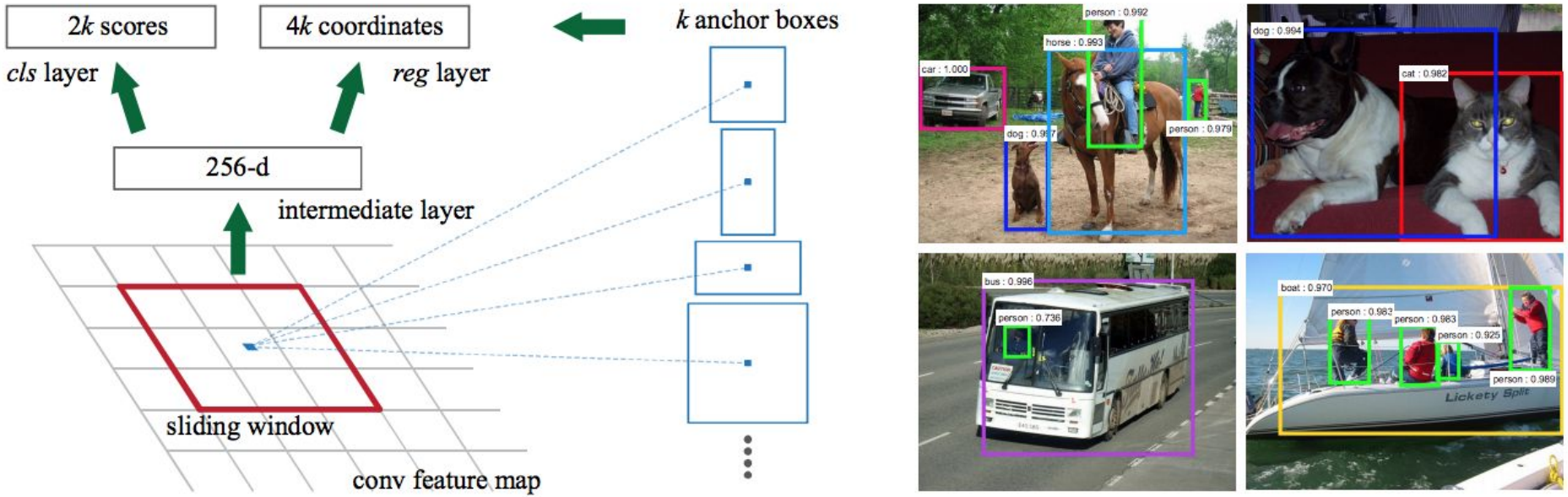


Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

https://arxiv.org/pdf/1506.01497.pdf

Training Region-based Object Detectors with Online Hard Example Mining

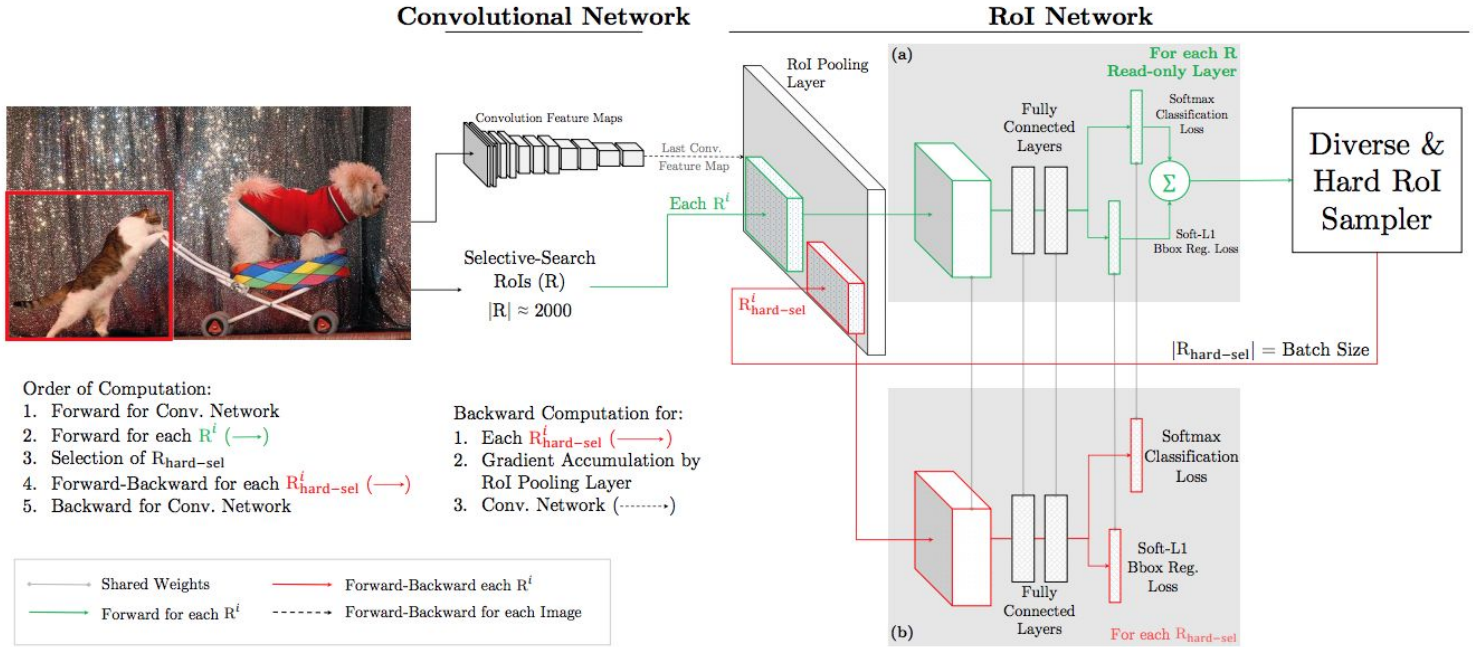
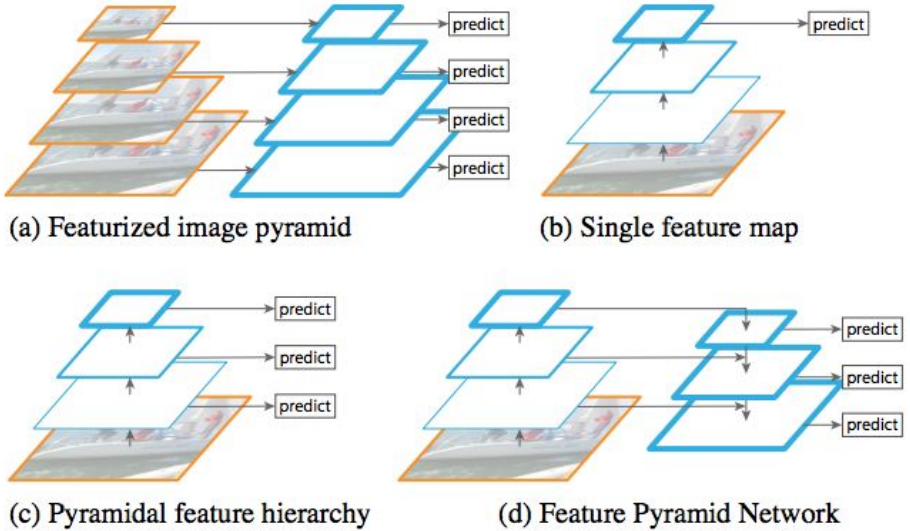


Figure 2: Architecture of the proposed training algorithm. Given an image, and selective search RoIs, the conv network computes a conv feature map. In (a), the *readonly* RoI network runs a forward pass on the feature map and all RoIs (shown in green arrows). Then the Hard RoI module uses these RoI losses to select B examples. In (b), these hard examples are used by the RoI network to compute forward and backward passes (shown in red arrows).

https://arxiv.org/pdf/1604.03540.pdf

Feature Pyramid Networks for Object Detection



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

https://arxiv.org/pdf/1612.03144.pdf

Feature Pyramid Networks for Object Detection

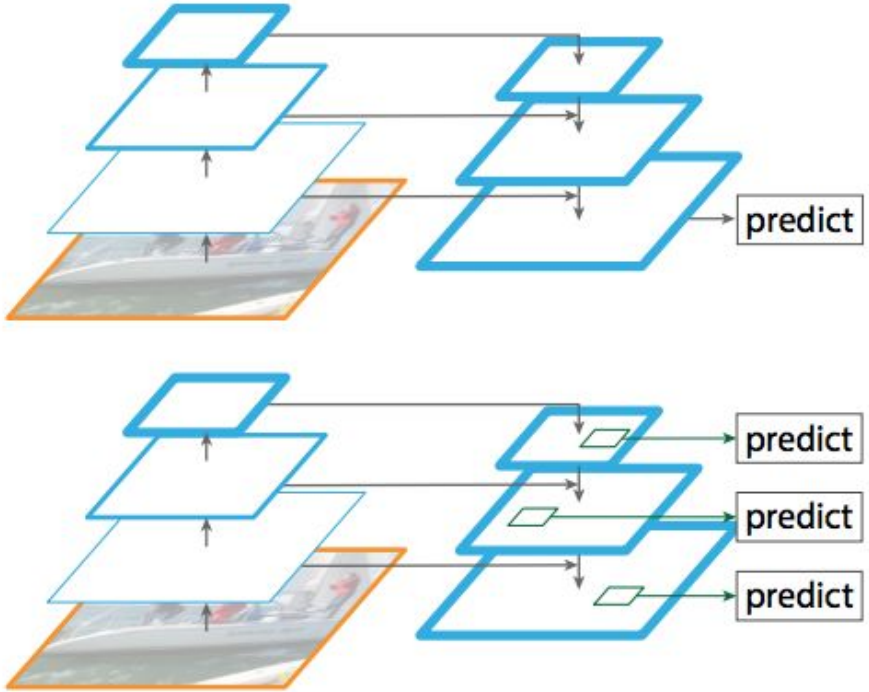


Figure 2. Top: a top-down architecture with skip connections, where predictions are made on the finest level (*e.g.*, [28]). Bottom: our model that has a similar structure but leverages it as a *feature pyramid*, with predictions made independently at all levels.

https://arxiv.org/pdf/1612.03144.pdf

Feature Pyramid Networks for Object Detection

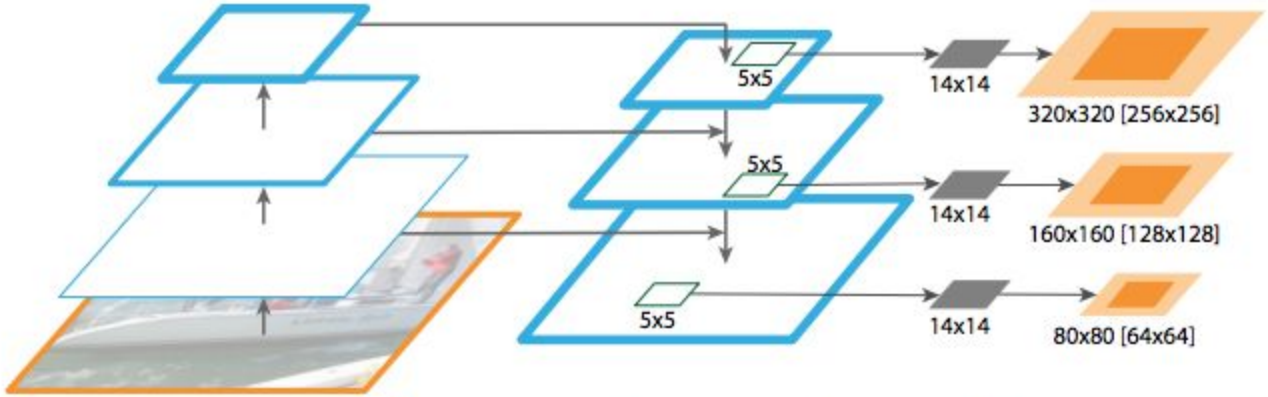
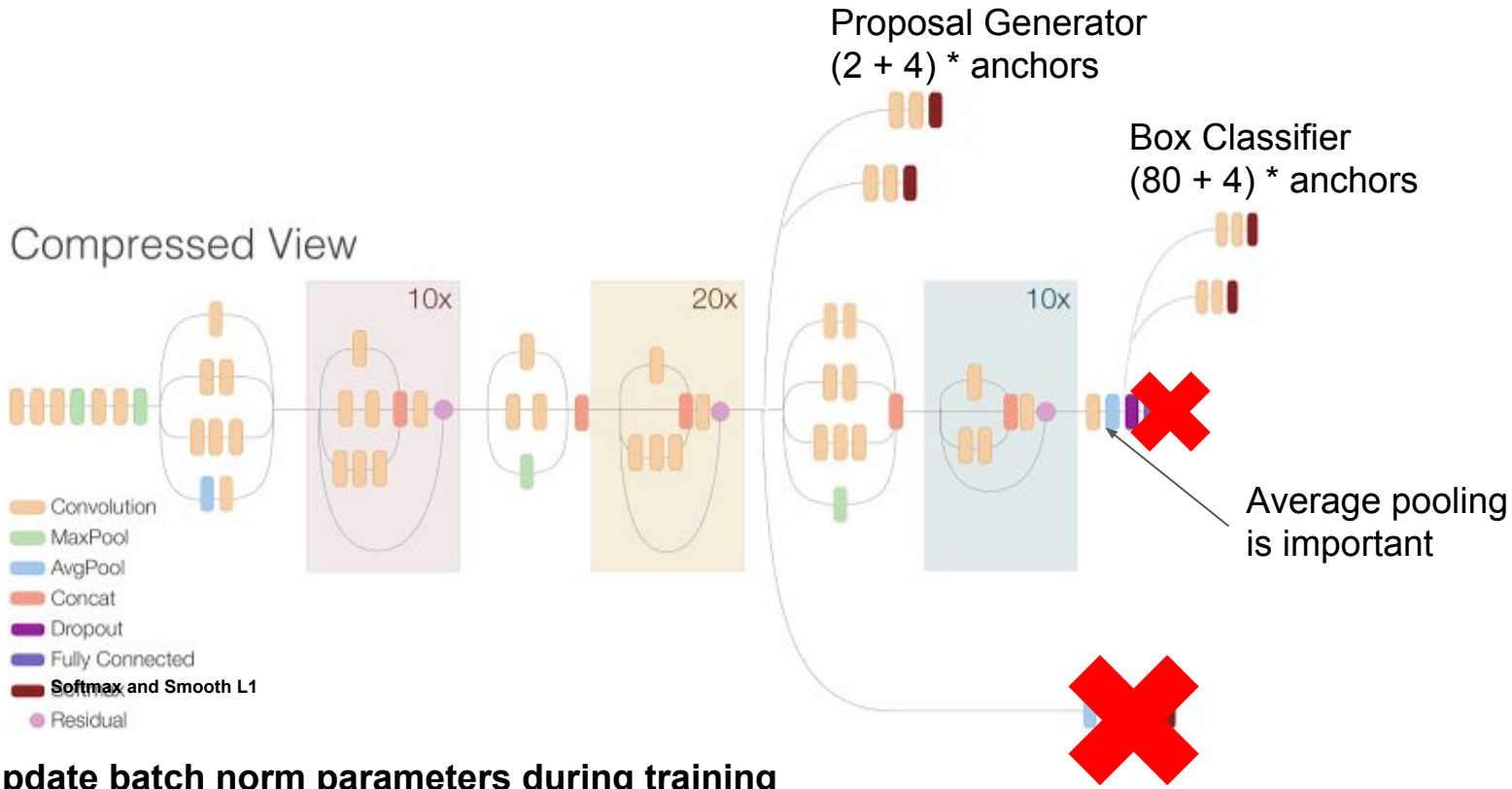


Figure 4. FPN for object segment proposals. The feature pyramid is constructed with identical structure as for object detection. We apply a small MLP on 5×5 windows to generate dense object segments with output dimension of 14×14. Shown in orange are the size of the image regions the mask corresponds to for each pyramid level (levels P_{3-5} are shown here). Both the corresponding image region size (light orange) and canonical object size (dark orange) are shown. Half octaves are handled by an MLP on 7x7 windows ($7 \approx 5\sqrt{2}$), not shown here. Details are in the appendix.

https://arxiv.org/pdf/1612.03144.pdf

Faster RCNN w/Inception Resnet (v2)

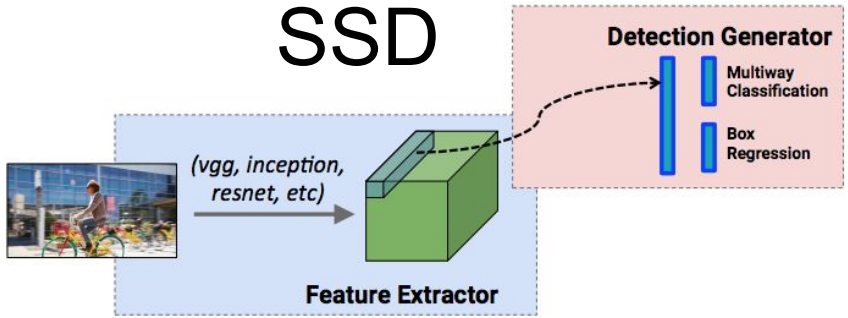


Note: We don't update batch norm parameters during training

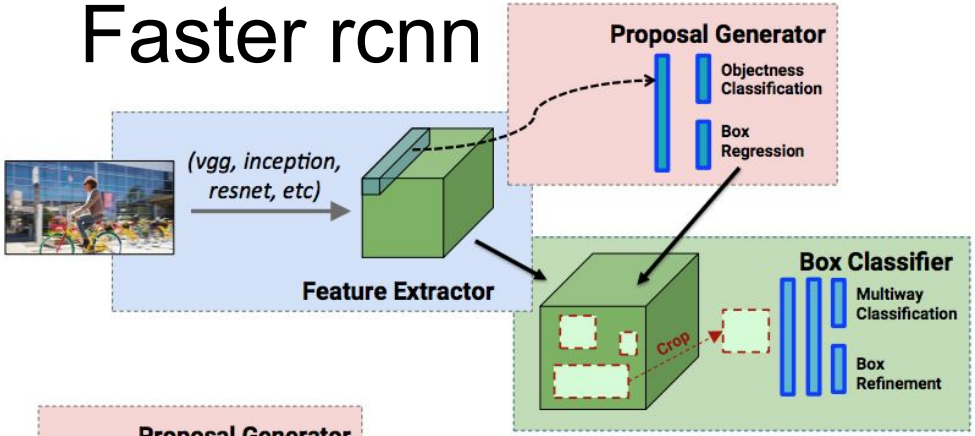
TENSORFLIGHT

Meta architectures

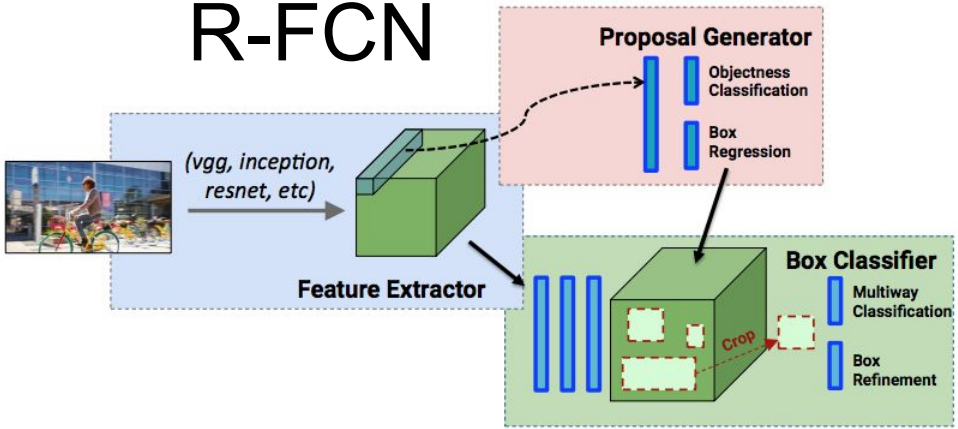
SSD



Faster rcnn



R-FCN



You Only Look Once: Unified, Real-Time Object Detection

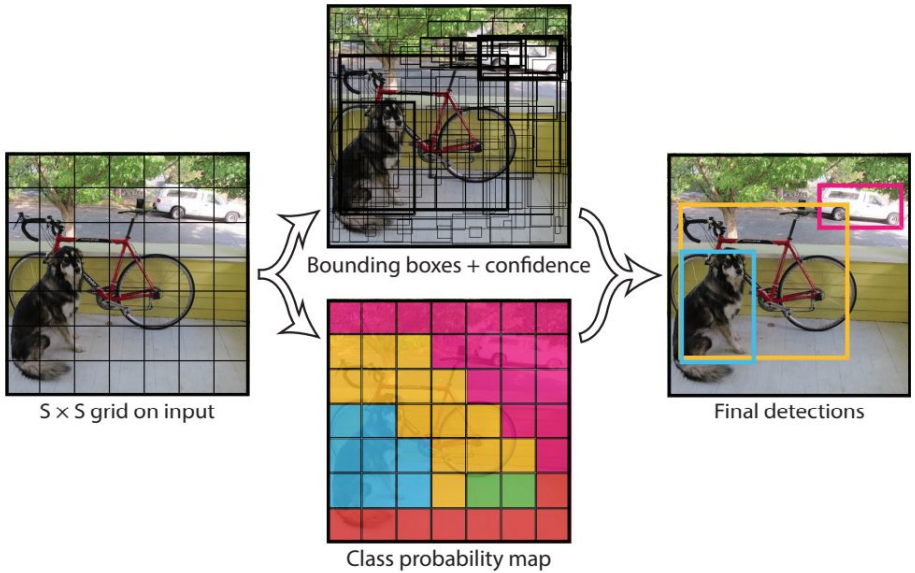


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

https://arxiv.org/pdf/1506.02640.pdf

You Only Look Once: Unified, Real-Time Object Detection

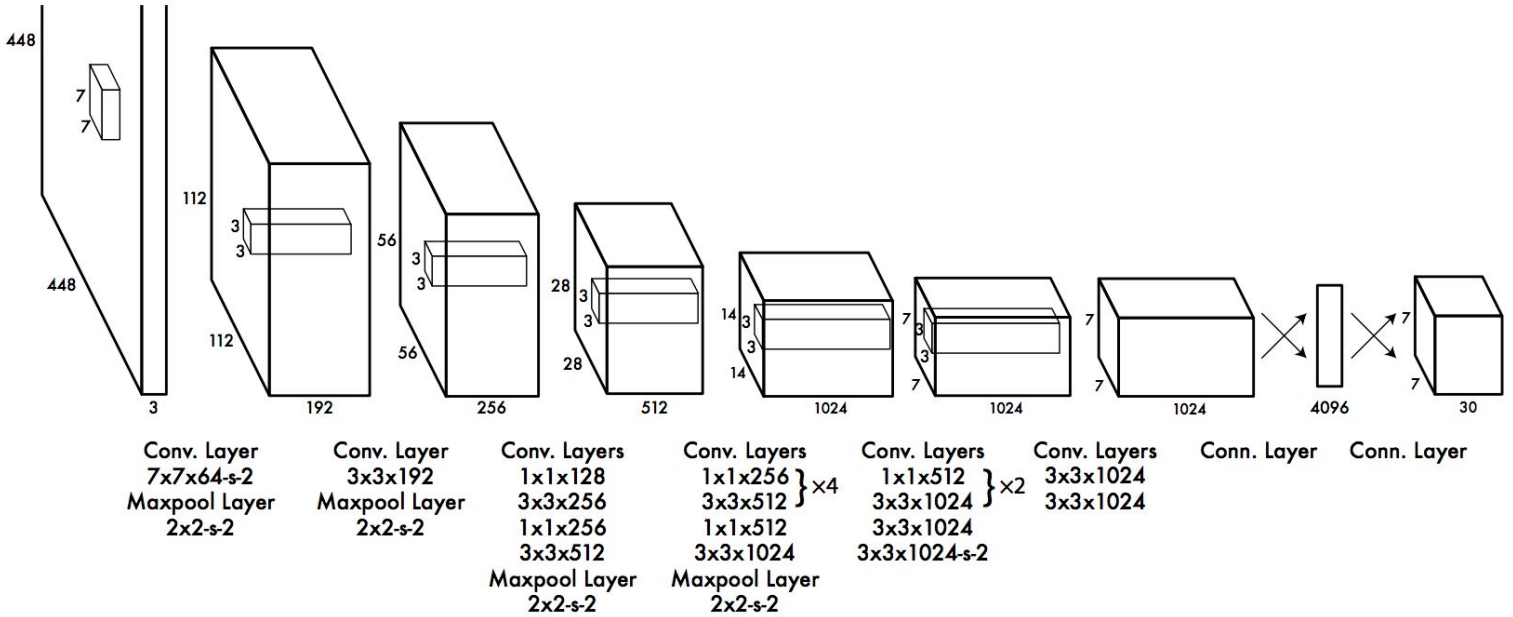


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

https://arxiv.org/pdf/1506.02640.pdf

TENSORFLIGHT

YOLO9000: Better, Faster, Stronger

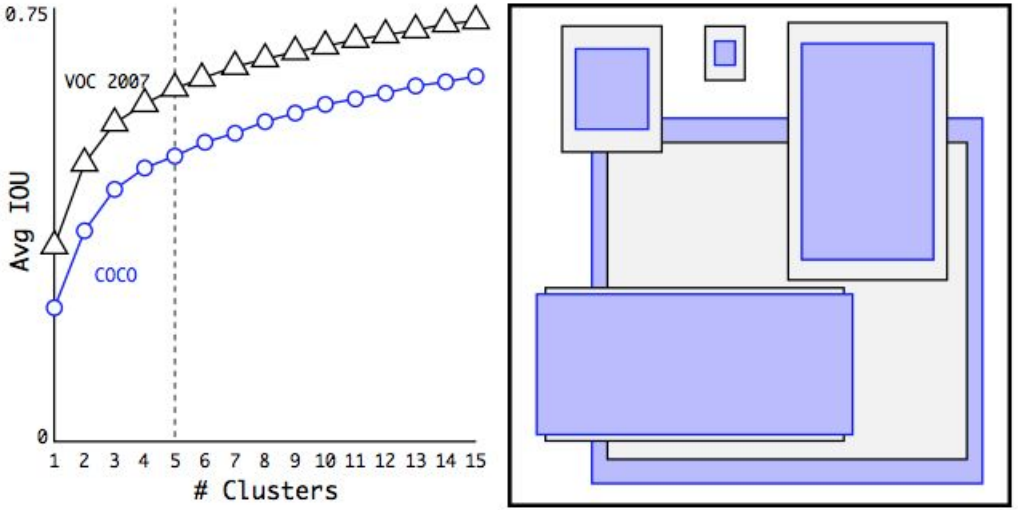


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

https://arxiv.org/pdf/1612.08242.pdf

TENSORFLIGHT

YOLO9000: Better, Faster, Stronger

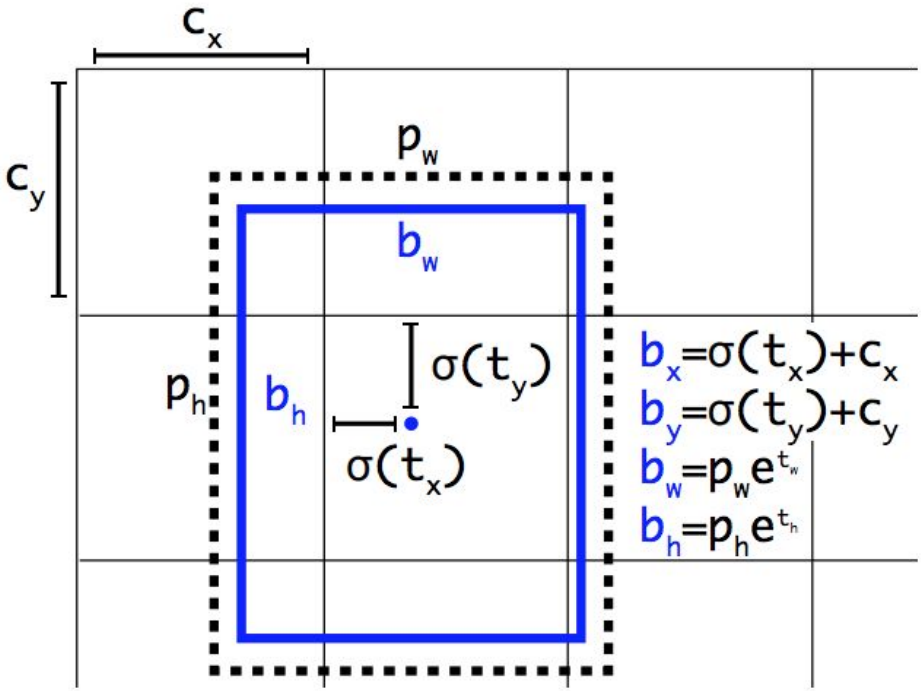


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

https://arxiv.org/pdf/1612.08242.pdf

TENSORFLIGHT

YOLO9000: Better, Faster, Stronger

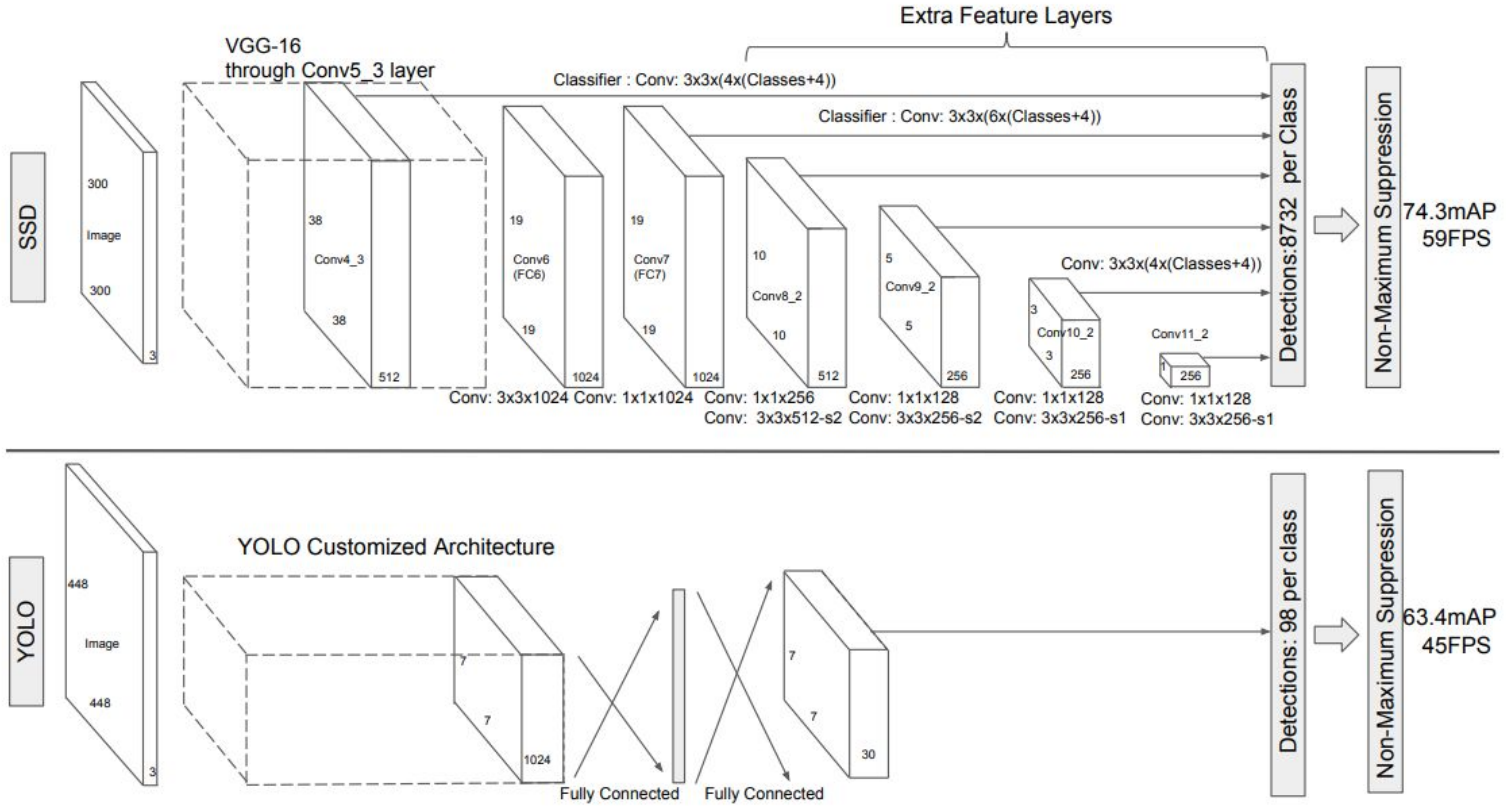
	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

Table 2: The path from YOLO to YOLOv2. Most of the listed design decisions lead to significant increases in mAP. Two exceptions are switching to a fully convolutional network with anchor boxes and using the new network. Switching to the anchor box style approach increased recall without changing mAP while using the new network cut computation by 33%.

https://arxiv.org/pdf/1612.08242.pdf

TENSORFLIGHT

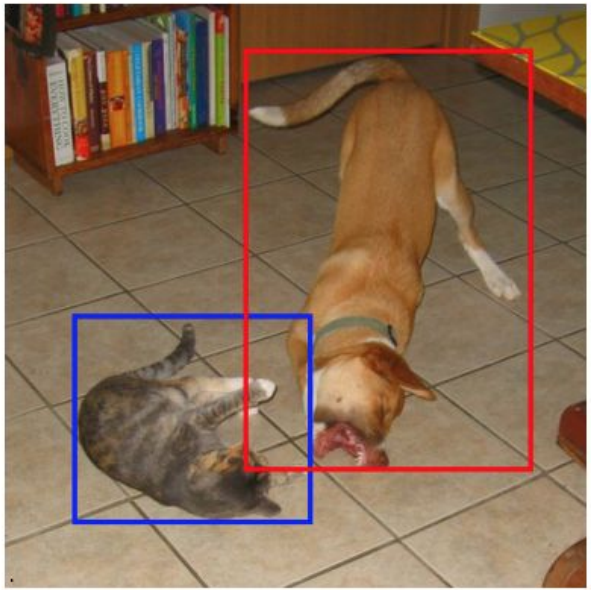
SSD: Single Shot MultiBox Detector



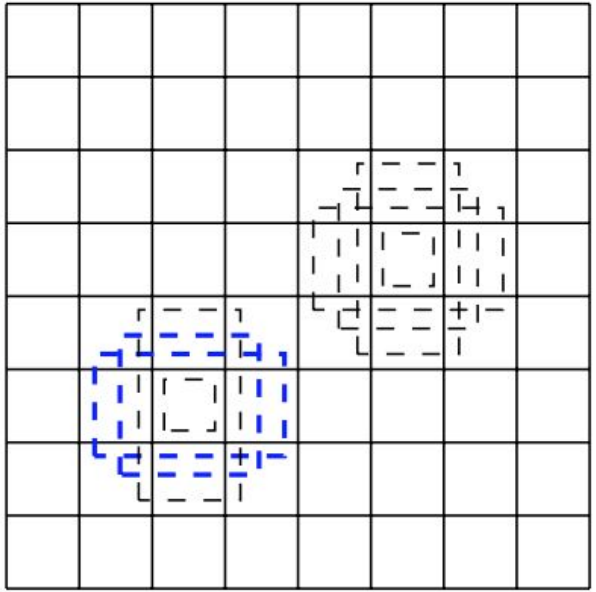
https://arxiv.org/pdf/1512.02325.pdf

TENSORFLIGHT

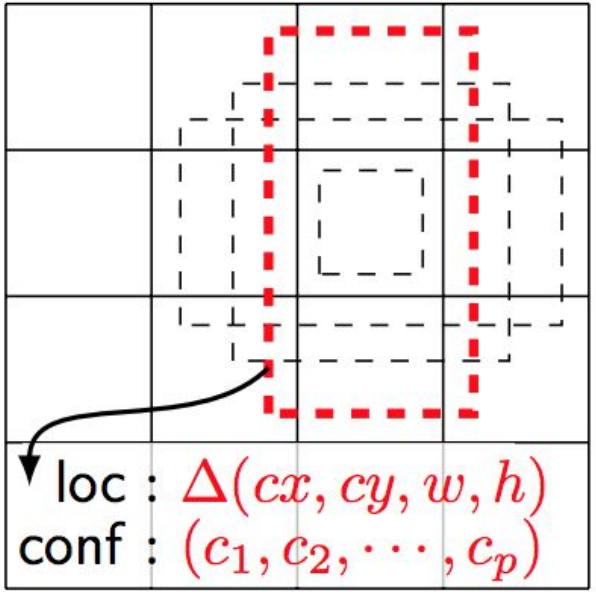
SSD: Single Shot MultiBox Detector



(a) Image with GT boxes



(b) 8×8 feature map



(c) 4×4 feature map

DSSD : Deconvolutional Single Shot Detector

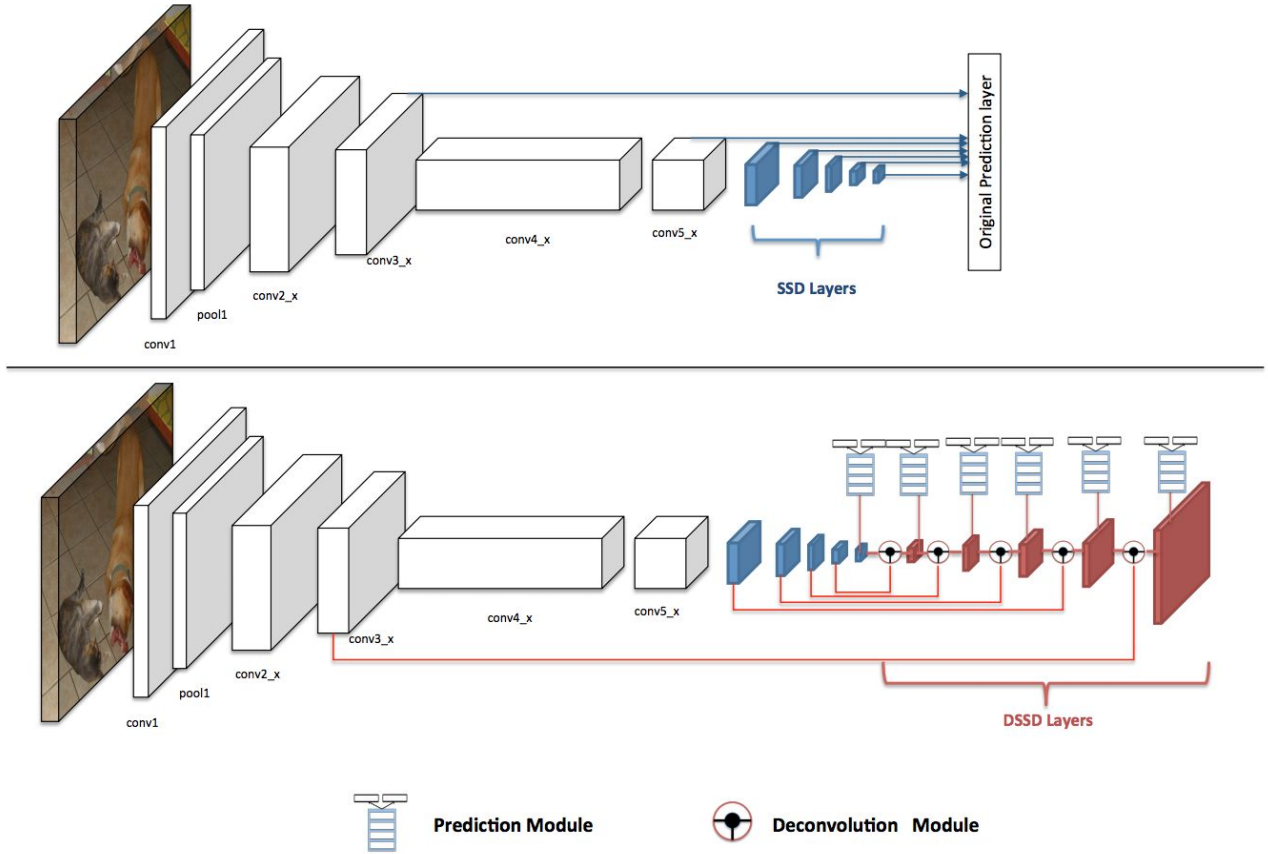


Figure 1: **Networks of SSD and DSSD on residual network.** The blue modules are the layers added in SSD framework, and we call them SSD Layers. In the bottom figure, the red layers are DSSD layers.

https://arxiv.org/pdf/1701.06659.pdf

R-FCN: Object Detection via Region-based Fully Convolutional Networks

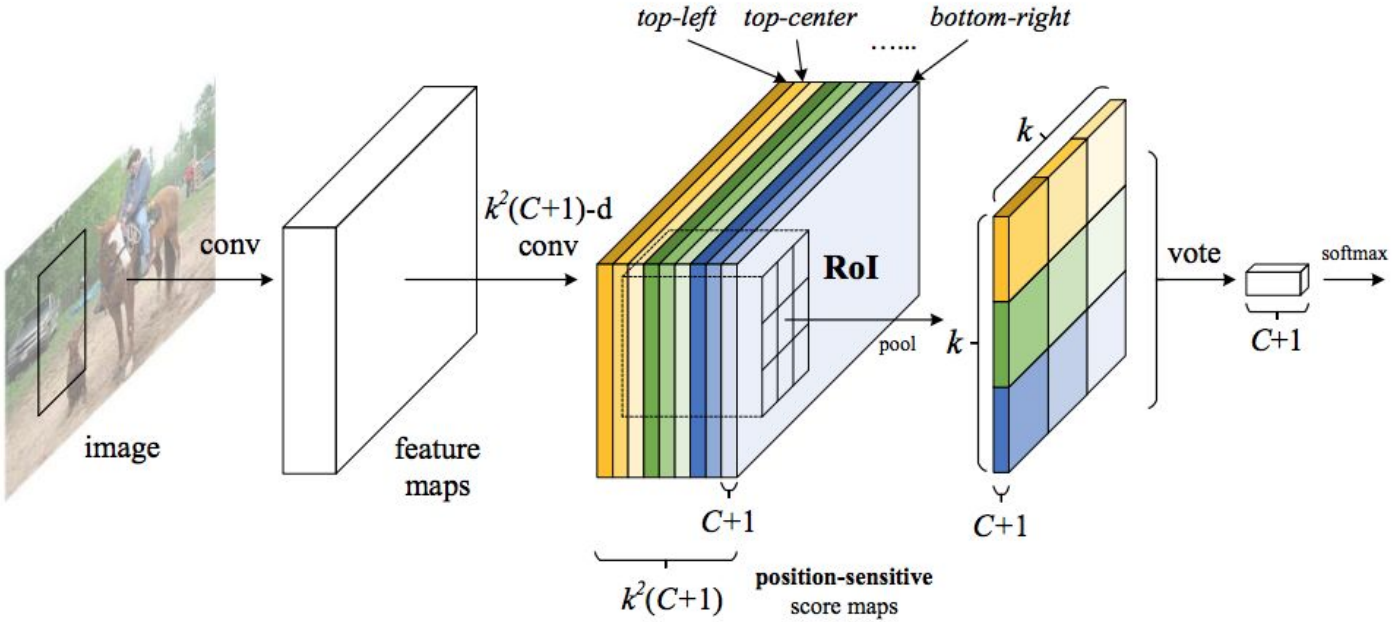


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

https://arxiv.org/pdf/1605.06409.pdf

R-FCN: Object Detection via Region-based Fully Convolutional Networks

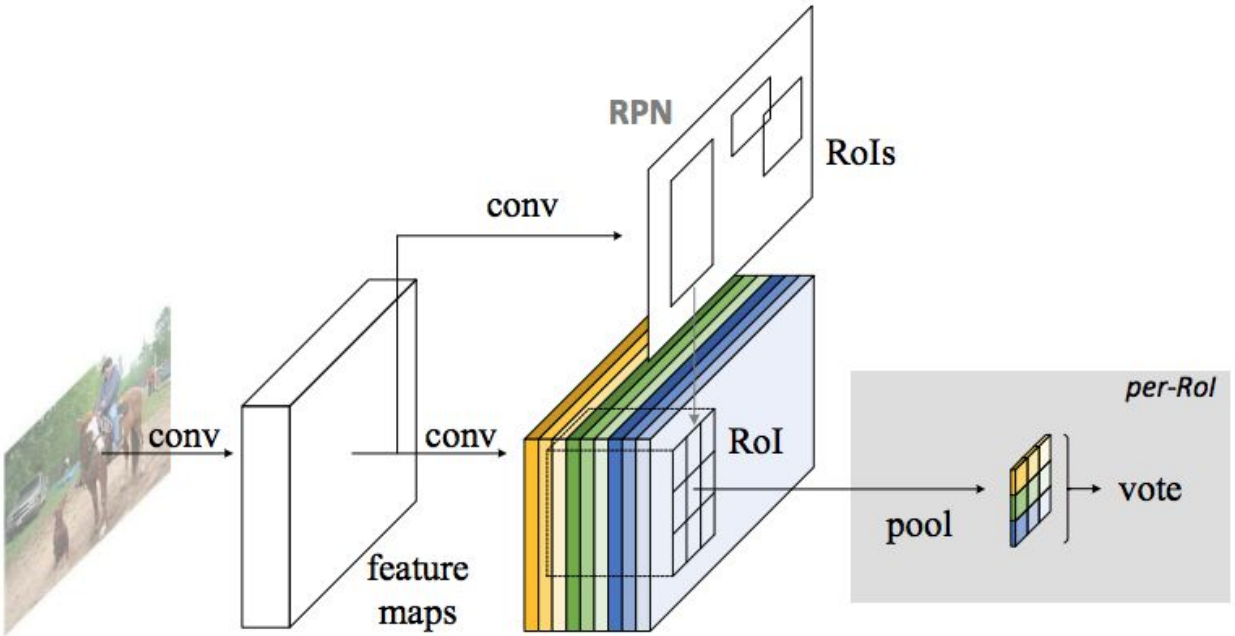


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

https://arxiv.org/pdf/1605.06409.pdf

R-FCN: Object Detection via Region-based Fully Convolutional Networks

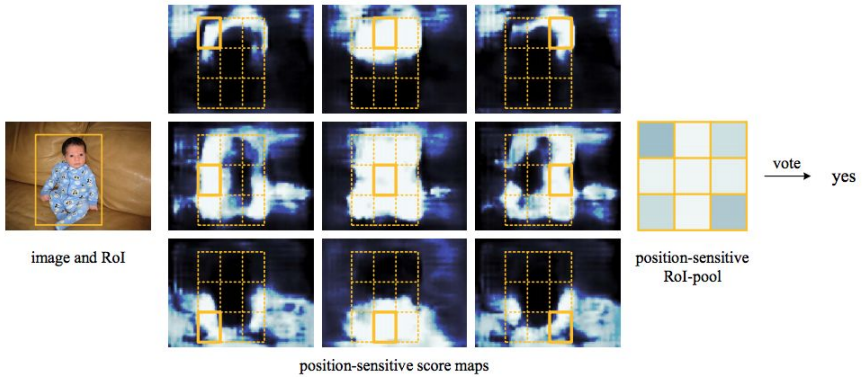


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

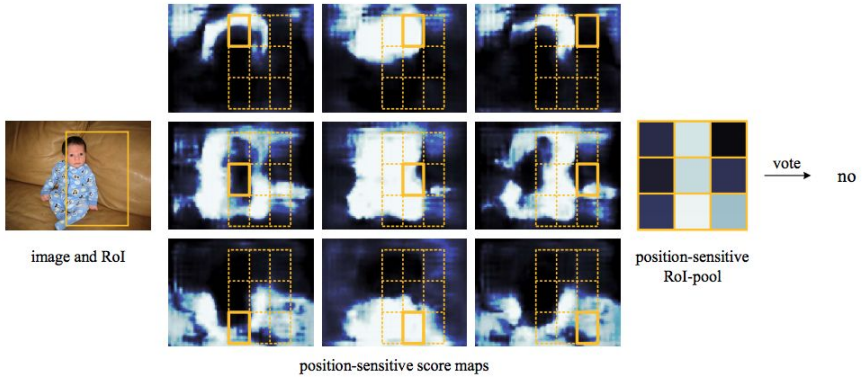


Figure 4: Visualization when an RoI does not correctly overlap the object.

https://arxiv.org/pdf/1605.06409.pdf

Deformable Convolutional Networks

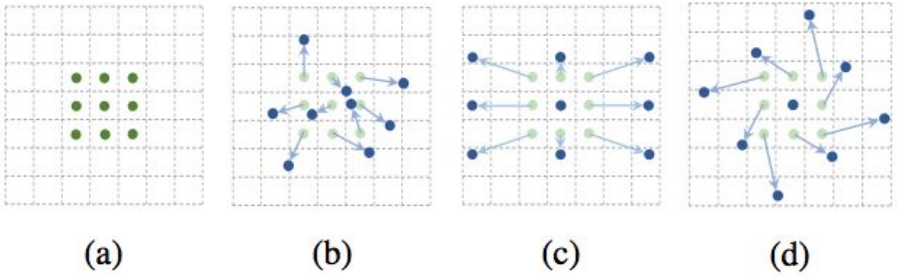


Figure 1: Illustration of the sampling locations in 3×3 standard and deformable convolutions. (a) regular sampling grid (green points) of standard convolution. (b) deformed sampling locations (dark blue points) with augmented offsets (light blue arrows) in deformable convolution. (c)(d) are special cases of (b), showing that the deformable convolution generalizes various transformations for scale, (anisotropic) aspect ratio and rotation.

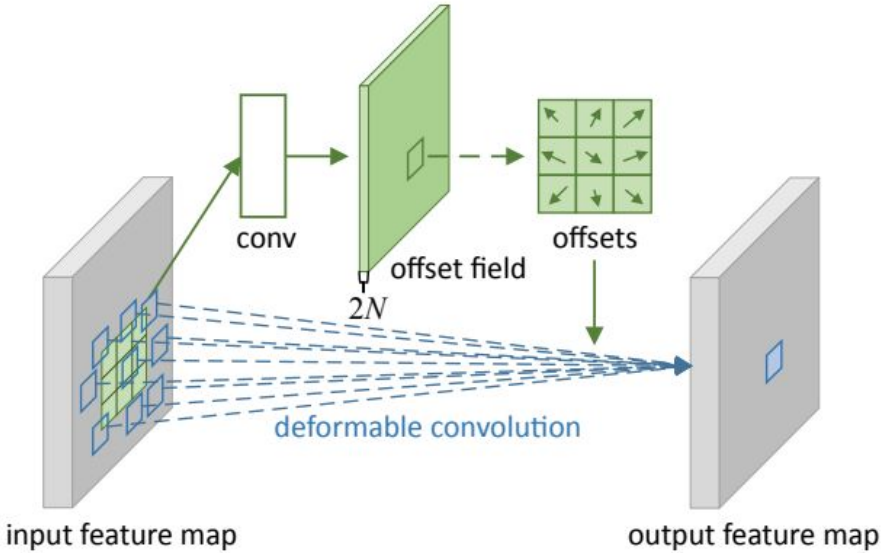


Figure 2: Illustration of 3×3 deformable convolution.

https://arxiv.org/pdf/1703.06211.pdf

TENSORFLIGHT

Deformable Convolutional Networks

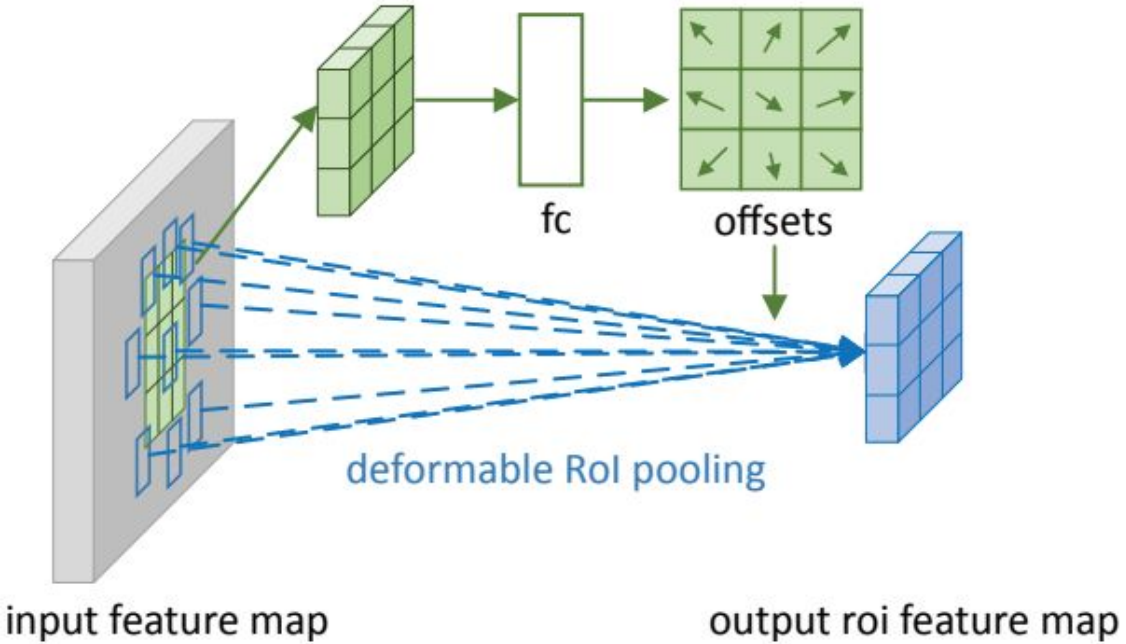


Figure 3: Illustration of 3×3 deformable RoI pooling.

https://arxiv.org/pdf/1703.06211.pdf

Deformable Convolutional Networks

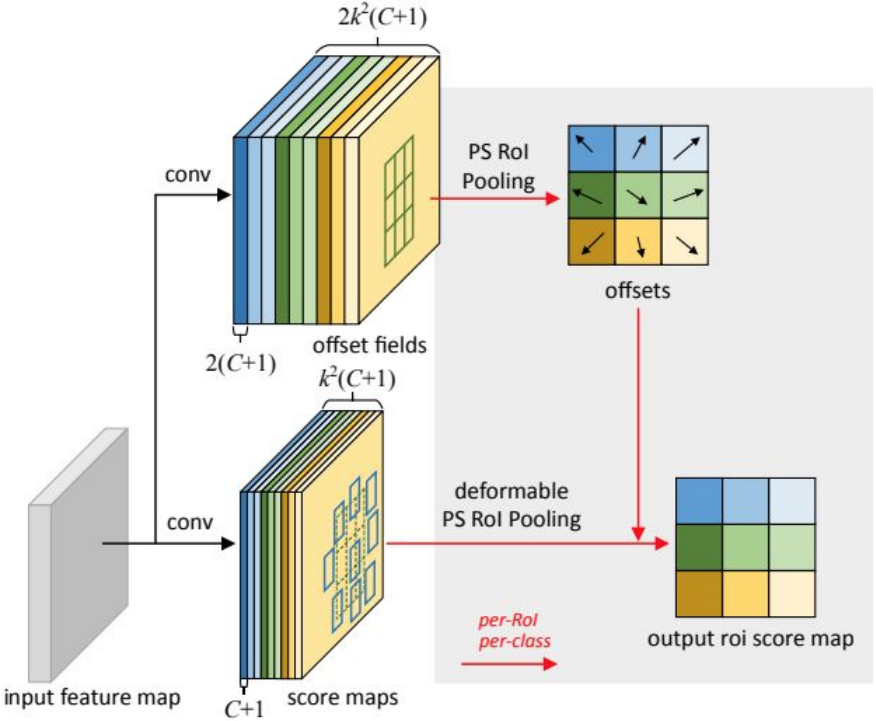
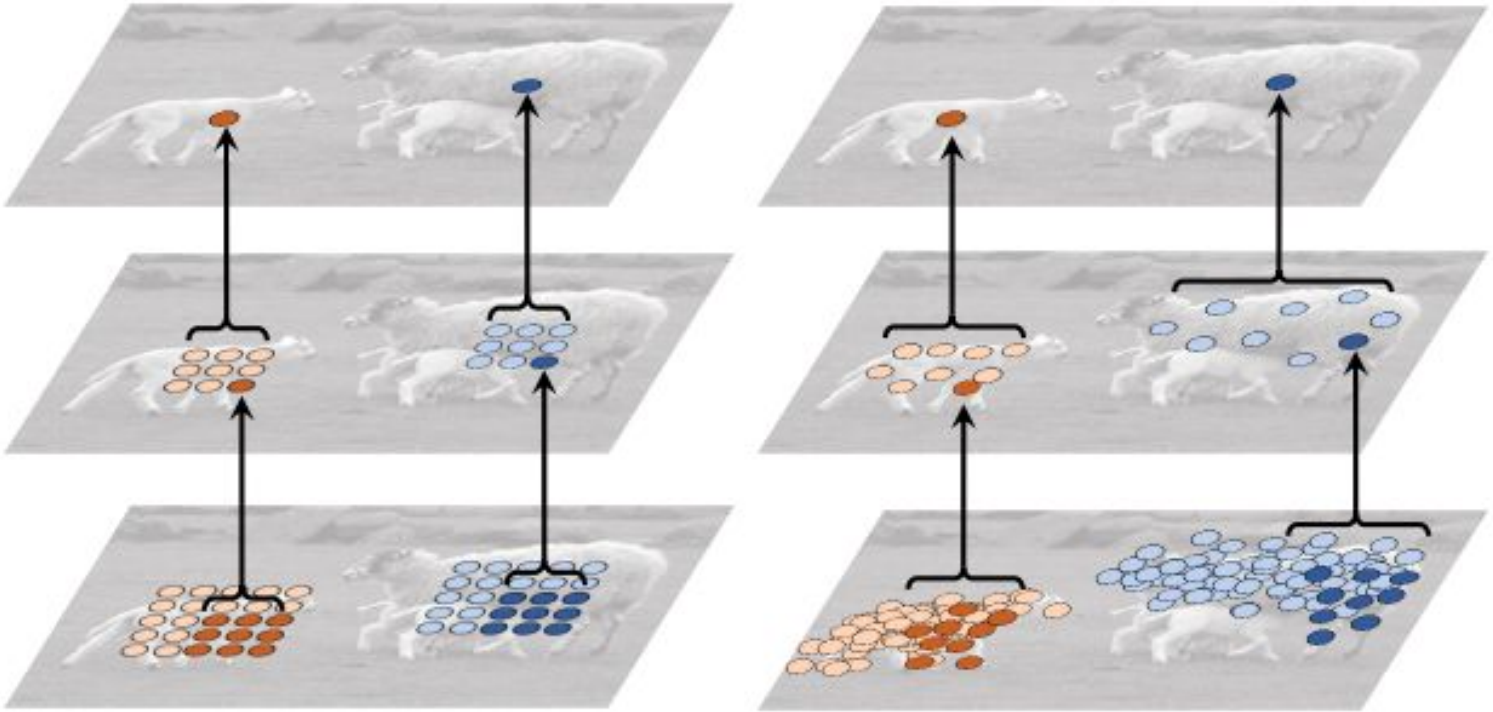


Figure 4: Illustration of 3×3 deformable PS RoI pooling.

https://arxiv.org/pdf/1703.06211.pdf

TENSORFLIGHT

Deformable Convolutional Networks



(a) standard convolution

(b) deformable convolution

https://arxiv.org/pdf/1703.06211.pdf

TENSORFLIGHT

Deformable Convolutional Networks

Figure 6: Each image triplet shows the sampling locations ($9^3 = 729$ red points in each image) in three levels of 3×3 deformable filters (see Figure 5 as a reference) for three activation units (green points) on the background (left), a small object (middle), and a large object (right), respectively.

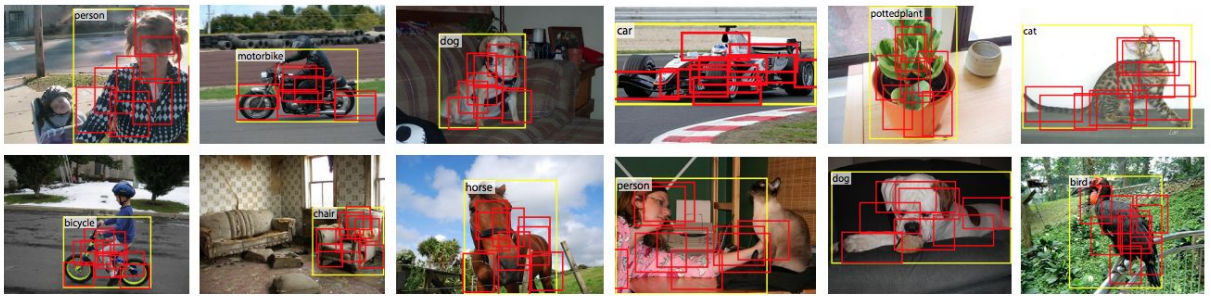


Figure 7: Illustration of offset parts in deformable (positive sensitive) RoI pooling in R-FCN [7] and 3×3 bins (red) for an input RoI (yellow). Note how the parts are offset to cover the non-rigid objects.

https://arxiv.org/pdf/1703.06211.pdf

Improving Object Detection With One Line of Code

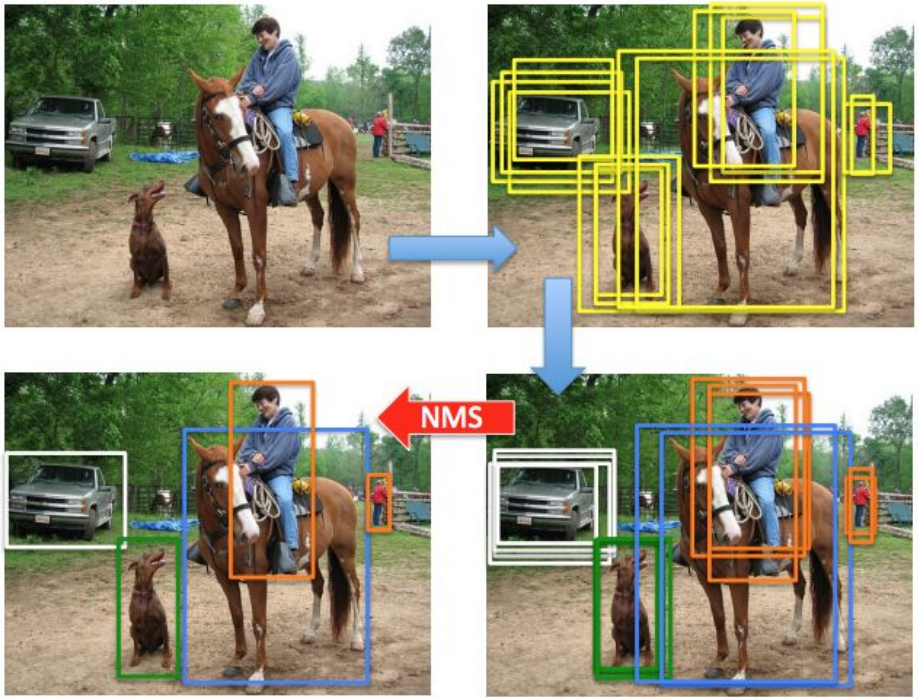


Figure 3. In object detection, first category independent region proposals are generated. These region proposals are then assigned a score for each class label using a classification network and their positions are updated slightly using a regression network. Finally, non-maximum-suppression is applied to obtain detections.

https://arxiv.org/pdf/1704.04503.pdf

Improving Object Detection With One Line of Code

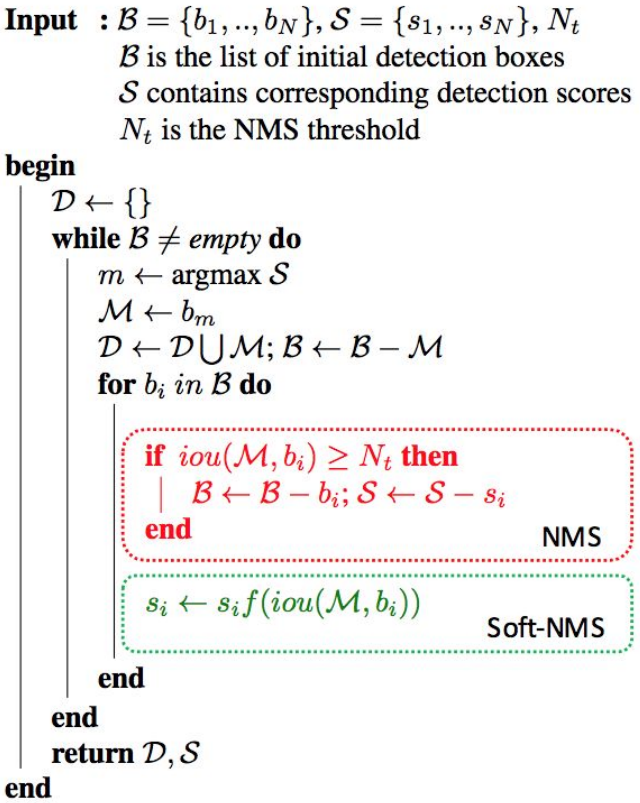


Figure 2. The pseudo code in red is replaced with the one in green in Soft-NMS. We propose to revise the detection scores by scaling them as a linear or Gaussian function of overlap.

https://arxiv.org/pdf/1704.04503.pdf

TENSORFLIGHT

Focal Loss for Dense Object Detection

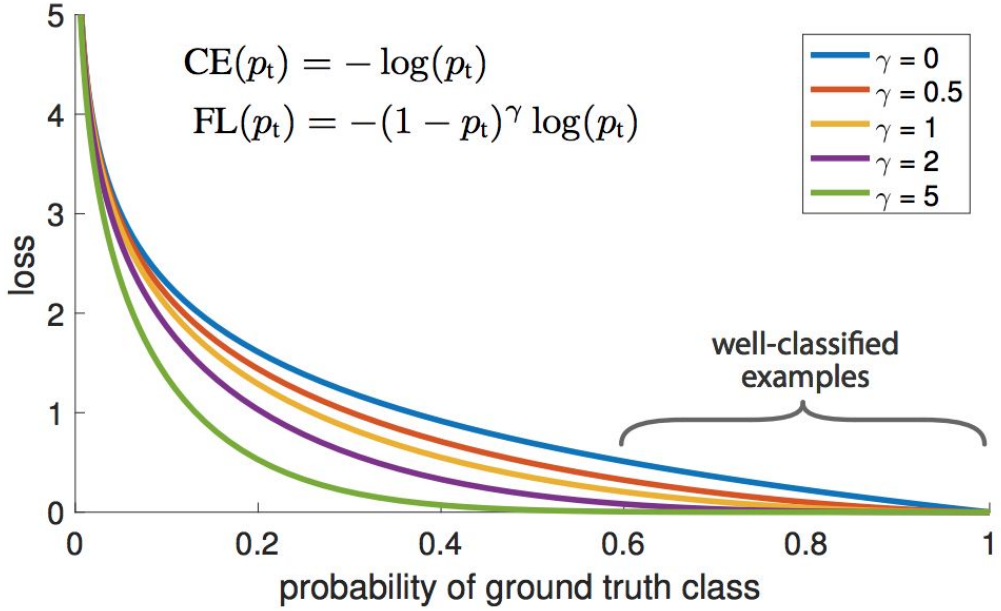


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.

https://arxiv.org/pdf/1708.02002.pdf

Focal Loss for Dense Object Detection

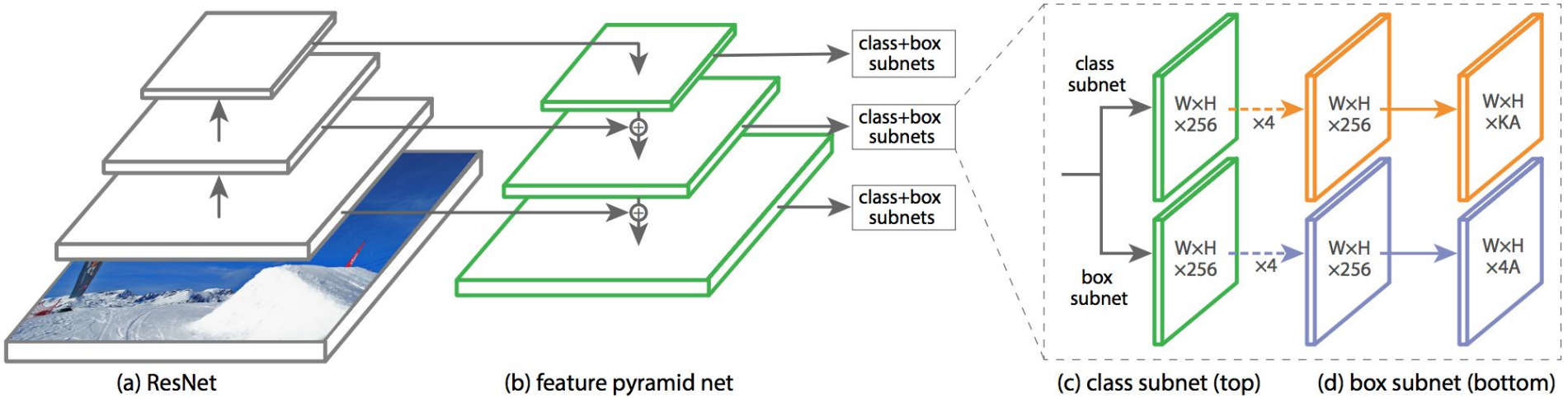


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

https://arxiv.org/pdf/1708.02002.pdf

Offer: Object detection project with TensorFlight

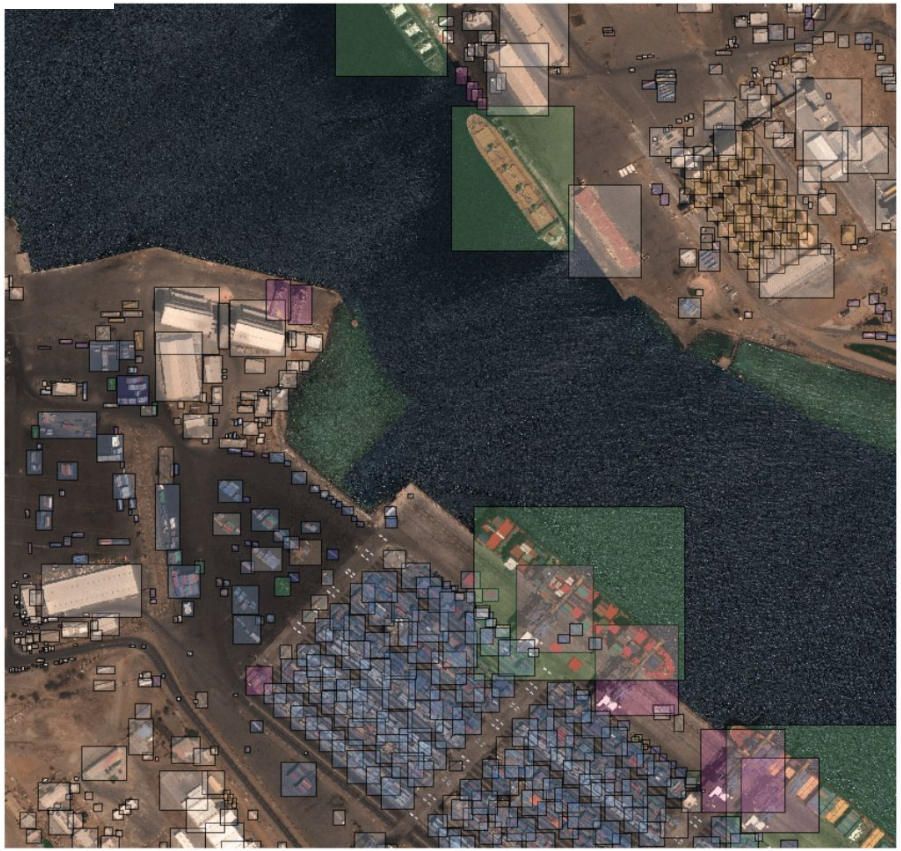


Figure 9: A fully annotated image from xView. Classes are denoted by different bounding box shadings. All imagery in this figure is from Digital Globe.

http://challenge.xviewdataset.org/challenge-description

Thank You

TENSORFLIGHT

TensorFlight is working with master and PhD students on the joint research projects aiming for publication and students thesis.

We are also hiring full time deep learning engineers and backend developers.

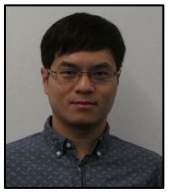
Bonus

Speed/accuracy trade-offs for modern convolutional object detectors


Jonathan Huang


Vivek Rathod


Chen Sun


Menglong Zhu


Anoop Korattikara


Alireza Fathi


Ian Fischer


Yang Song


Sergio Guadarrama


Kevin Murphy

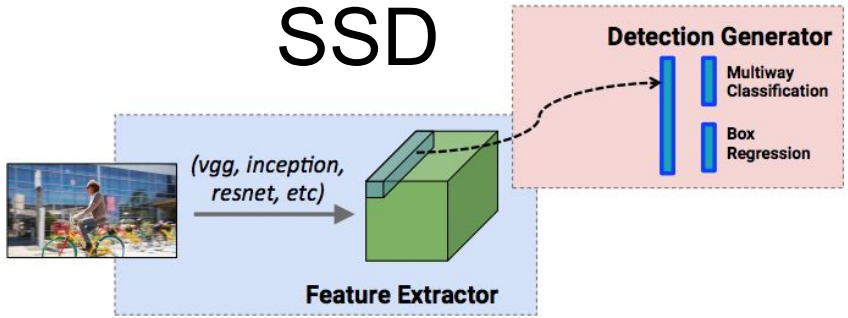
MS COCO 2016 object detection results

	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

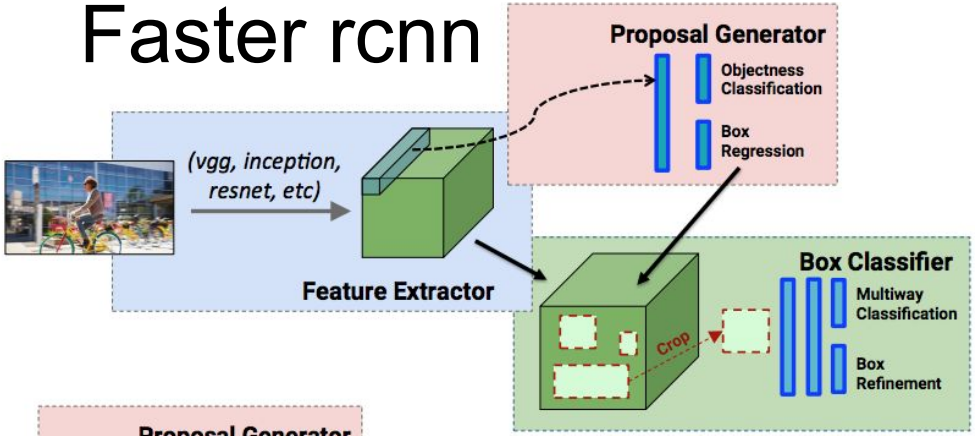
TENSORFLIGHT

Meta architectures

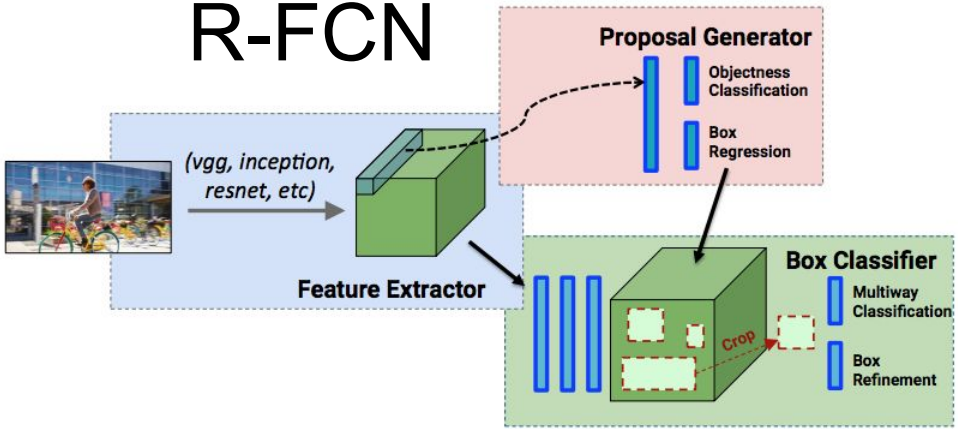
SSD



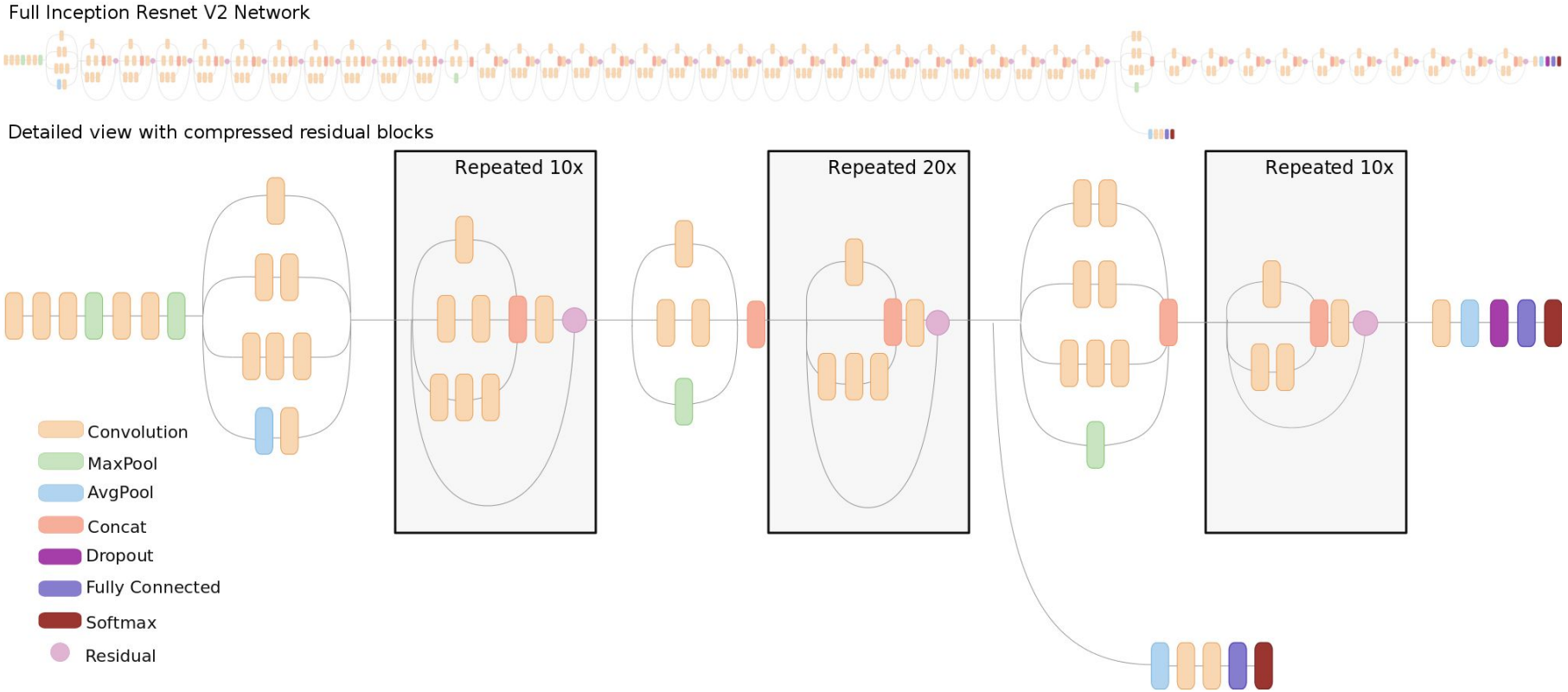
Faster rcnn



R-FCN

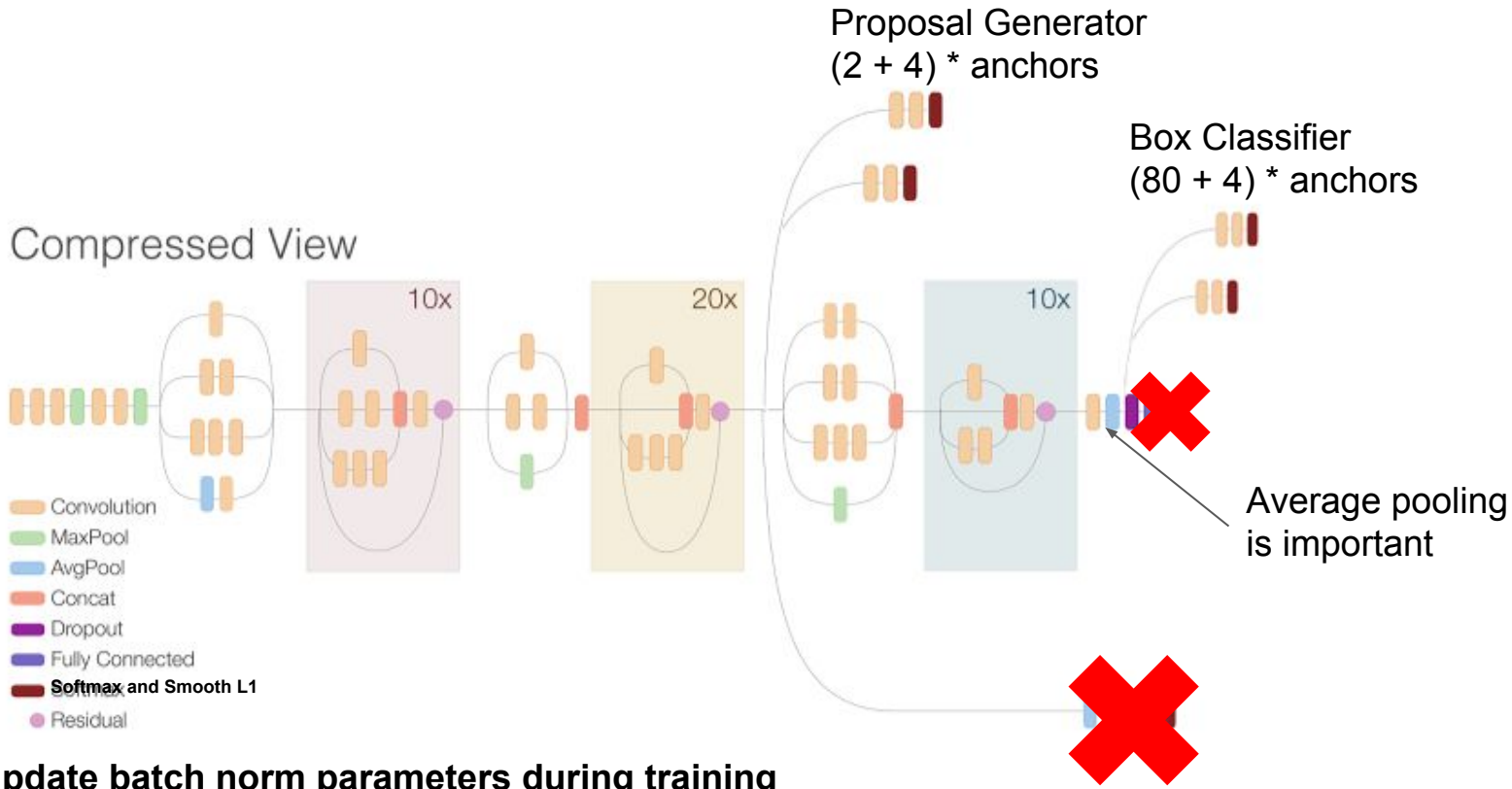


Inception Resnet (v2) Feature Extractor



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

Faster RCNN w/Inception Resnet (v2)



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

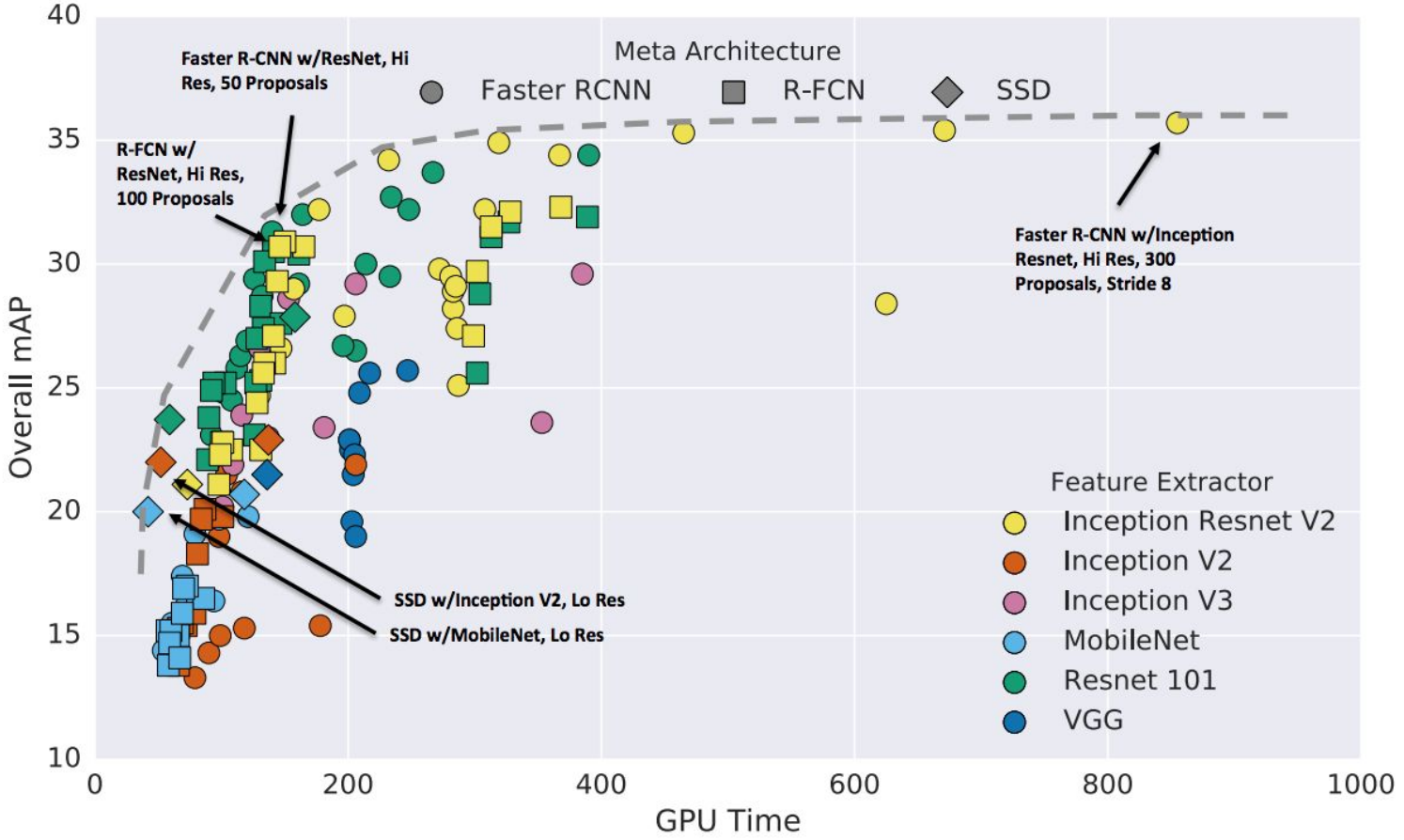
TENSORFLIGHT

10-crop inference

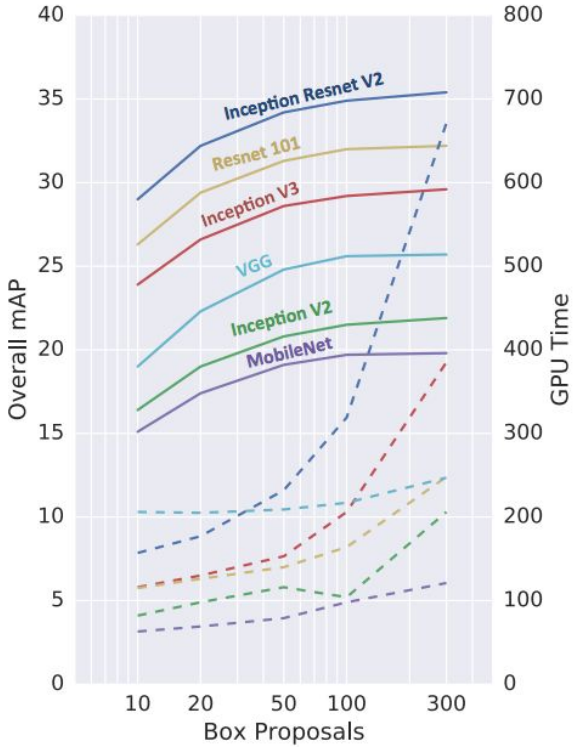
- No multiscale training
- No horizontal flip
- No box refinement
- No box voting
- No global context
- No ILSVRC detection data

TENSORFLIGHT

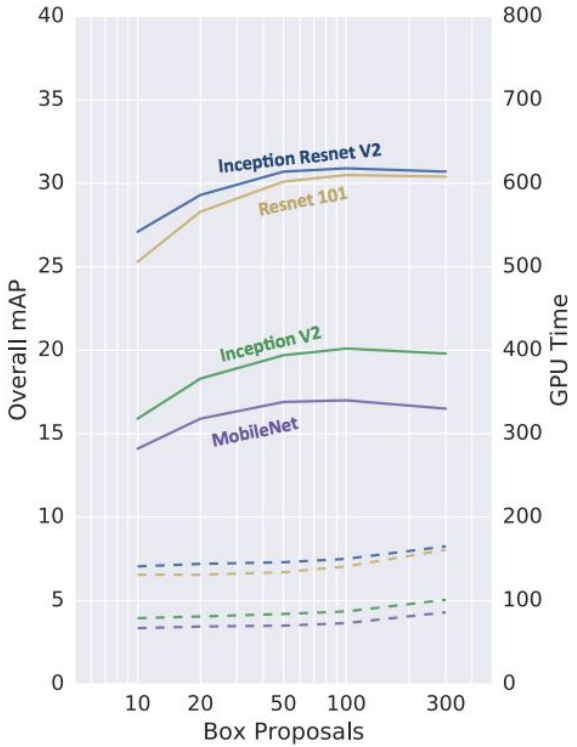
Accuracy vs Time



No of proposals vs Accuracy vs Time

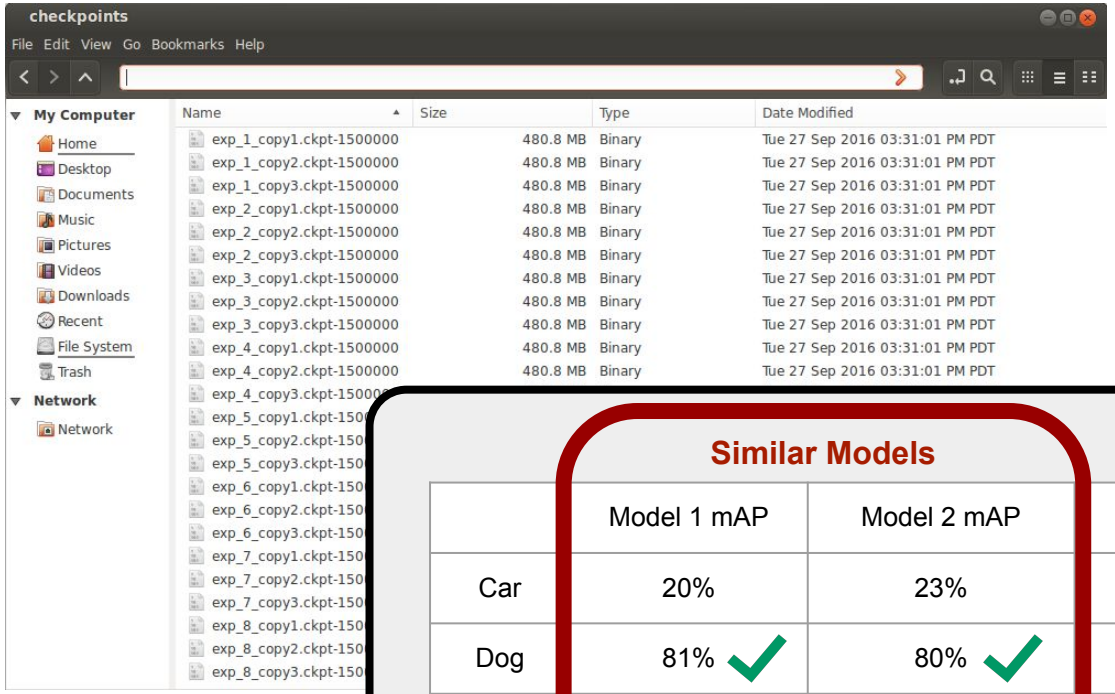


(a) FRCNN



(b) RFCN

Model Selection for Ensembling

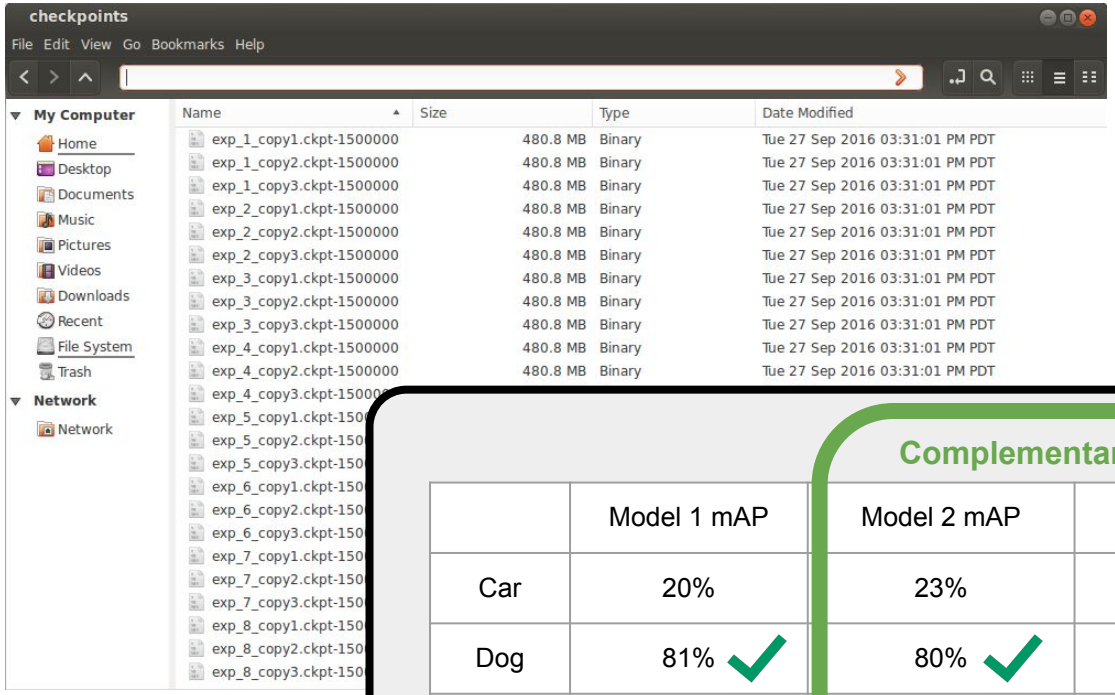


Take best K models?
Or select diverse K-subset of models?

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% ✓

Comparison of works

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