Deep Learning and Its Applications in Signal Processing

Lesson 4: Convolutional Neural Networks for Signal Processing

Liang Dong, ECE

Baylor University
SCHOOL OF ENGINEERING & COMPUTER SCIENCE
Department of Electrical & Computer Engineering

A Classic CNN Architecture – LeNet-5

- Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Haffner proposed a convolutional neural network (CNN) architecture for handwritten and machine-printed character recognition in 1998.

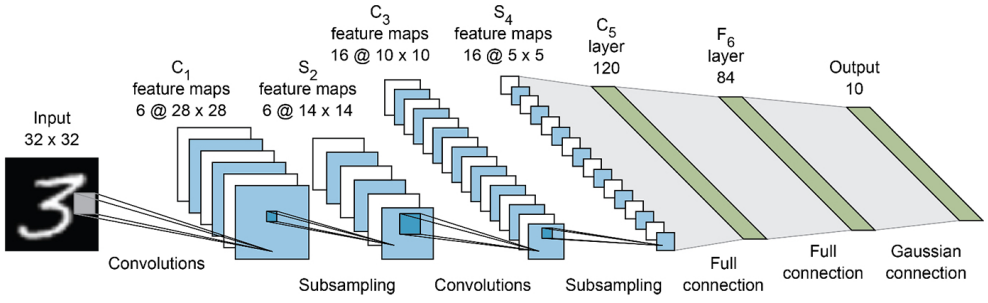


Figure: LeNet-5 Architecture

Website of LeNet-5 convolutional neural networks
Website of the MNIST database of handwritten digits

A Classic CNN Architecture – LeNet-5

- The LeNet-5 architecture consists of two sets of convolutional and average pooling layers, followed by a flattening convolutional layer, then two fully-connected layers and finally a softmax classifier.

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Figure: Summary Table of LeNet-5 Architecture

Code Examples of LeNet-5

- TensorFlow and Keras code examples of LeNet-5 on the MNIST and CIFAR-10 datasets.

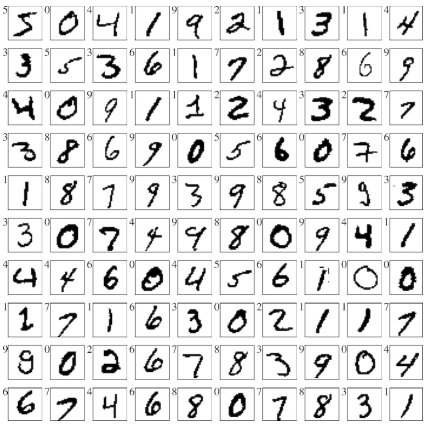


Figure: MNIST dataset

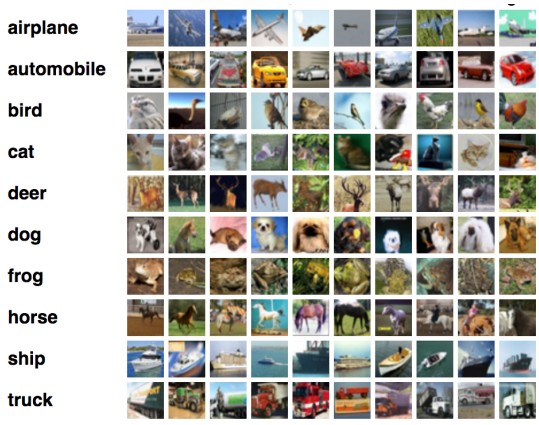


Figure: CIFAR-10 dataset

Common CNN Models

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) since 2010

Winners:

- 2012 AlexNet (error rate 15.3% ← the runner up 26.1%)
- 2014 VGGNet (error rate 7.32%), GoogLeNet (error rate 6.66%)
- 2015 ResNet – Deep Residual Network (error rate 3.57%)
  The first time a machine surpassed humans in recognizing ImageNet data!

IMAGENET is a dataset of over 15 millions labeled high-resolution images with around 22,000 categories. ILSVRC uses a subset of ImageNet. Roughly 1.2 million training images, 50,000 validation images and 150,000 testing images.

AlexNet

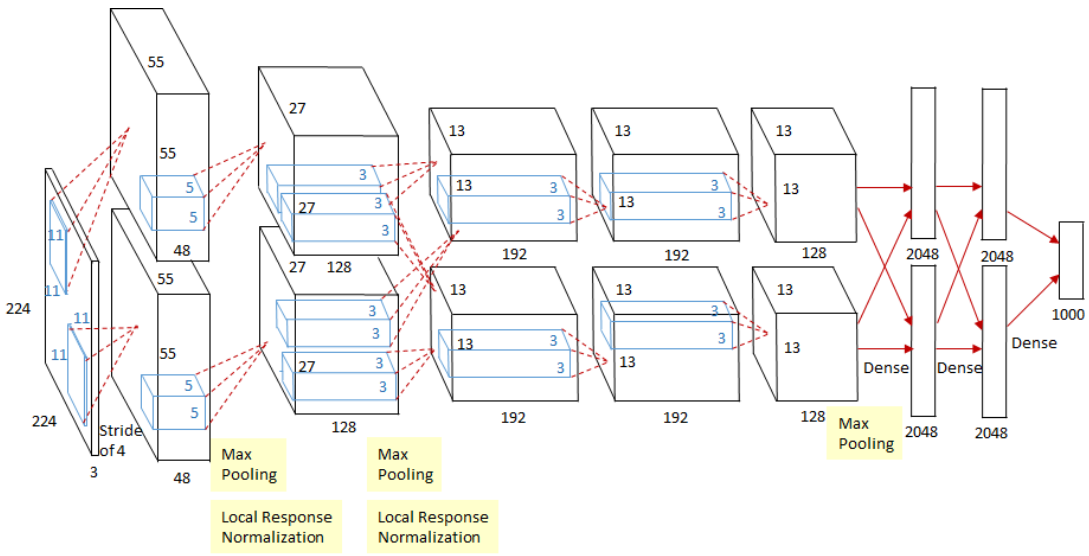


Figure: AlexNet Architecture

AlexNet

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	227x227x3	-	-	-
1	Convolution	96	55 x 55 x 96	11x11	4	relu
	Max Pooling	96	27 x 27 x 96	3x3	2	relu
2	Convolution	256	27 x 27 x 256	5x5	1	relu
	Max Pooling	256	13 x 13 x 256	3x3	2	relu
3	Convolution	384	13 x 13 x 384	3x3	1	relu
4	Convolution	384	13 x 13 x 384	3x3	1	relu
5	Convolution	256	13 x 13 x 256	3x3	1	relu
	Max Pooling	256	6 x 6 x 256	3x3	2	relu
6	FC	-	9216	-	-	relu
7	FC	-	4096	-	-	relu
8	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Figure: Summary Table of AlexNet Architecture

AlexNet

- ReLU is introduced in AlexNet. (Before Alexnet, Tanh was used.) ReLU is six times faster than Tanh to reach 25% training error rate.
- Local Response Normalization. (Nowadays, batch normalization is used instead of local response normalization.)
- Overlapping Pooling. Pooling with stride smaller than the kernel size.
- Two forms of data augmentation.
  – Image translation and horizontal reflection (mirroring)
  – Altering the intensity
- Dropout. Dropout with probability of 0.5 is used at the first two fully-connected layers.
- Batch size: 128; Momentum v: 0.9; Weight Decay: 0.0005; Learning rate: 0.01, reduced by 10 manually when validation error rate stopped improving, and reduced 3 times.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, "ImageNet classification with deep convolutional neural networks," *J Commun. ACM*, Vol. 60, No 6, pp. 84–90, 2017.

VGGNet

- VGGNet was invented by VGG (Visual Geometry Group) from University of Oxford.
- 1st runner-up of the classification task and the winner of the localization task in ILSVRC 2014.
- There are many other models built on top of VGGNet or based on the 3×3 convolutional-layer idea of VGGNet.

Karen Simonyan and Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, 2014.

VGGNet

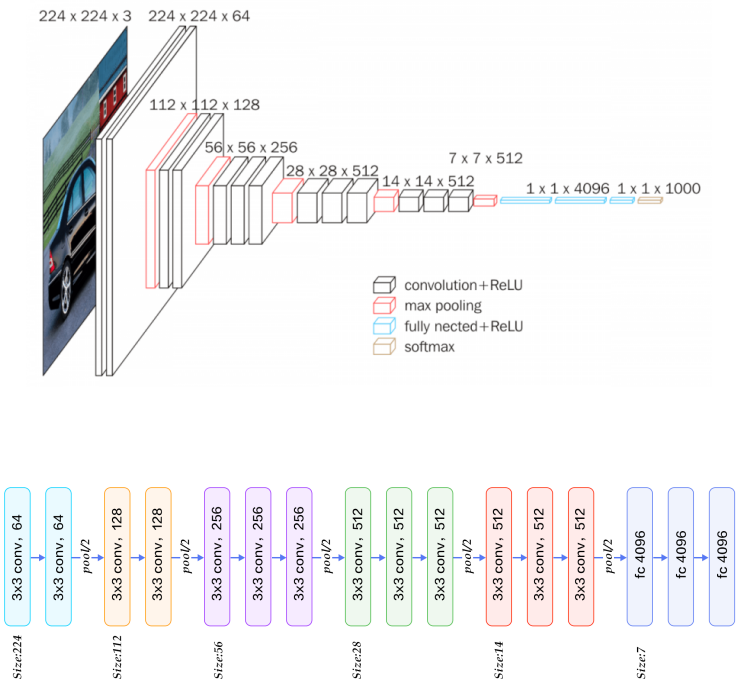
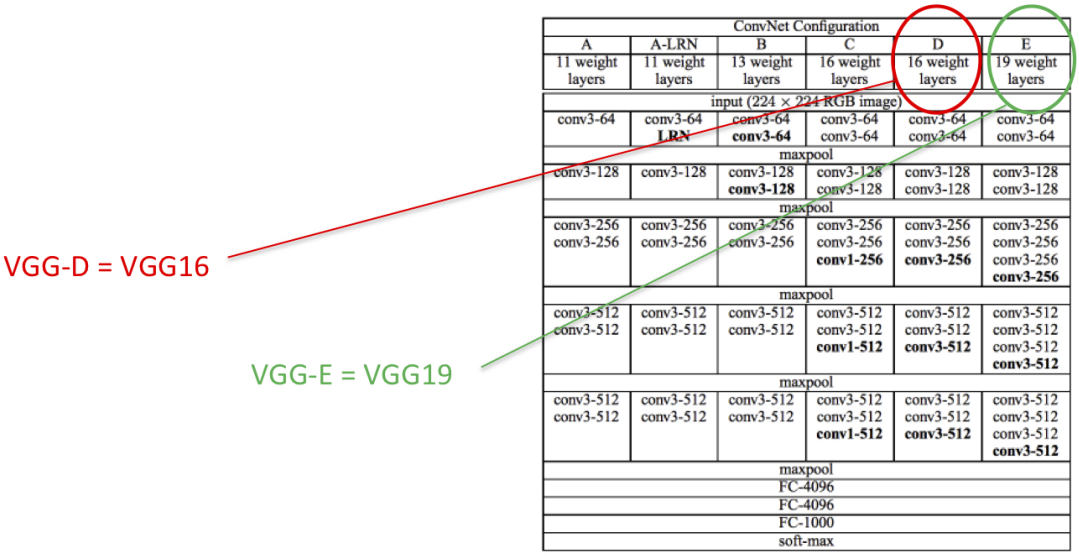


Figure: VGG-16 Architecture

VGGNet

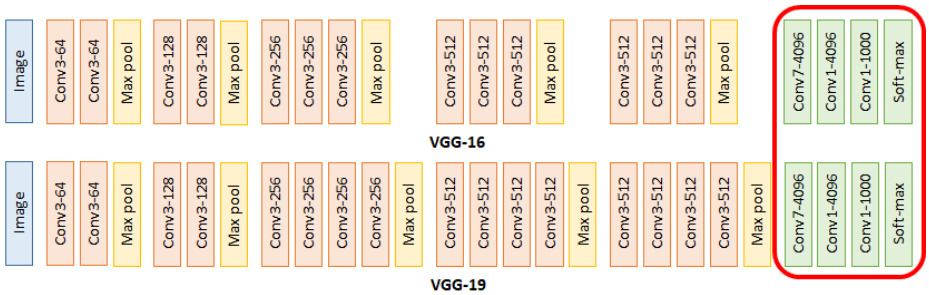


ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure: Summary Table of VGG Architecture

VGGNet

- Multi-Scale Training. An image is scaled with smaller-size equal to a range from 256 to 512, then cropped to 224×224. It is more accurate for test-image objects with different sizes.
- Multi-Scale Testing. By scaling the test image to different sizes, it increases the chance of correct classification.
- Dense Testing (Convolutionalized Testing). During testing, the first FC is replaced by 7×7 conv. The second and third FCs are replaced by 1×1 conv.



GoogLeNet

- It is from Google and pays tribute to Prof. Yan LeCun's LeNet.
- It contains 1×1 conv at the middle of the network.
- Global average pooling is used at the end of the network instead of fully connected layers.
- Inception Module. To have different sizes/types of convolutions for the same input and stacking the outputs. The name "Inception" comes from [2] and a famous Internet meme: *We need to go Deeper*.

[1] Christian Szegedy, et al., "Going Deeper with Convolutions," *arXiv*, 2014.
[2] Min Lin, Qiang Chen, and Shuicheng Yan, "Network In Network," *arXiv*, 2013.

GoogLeNet

- Network in Network

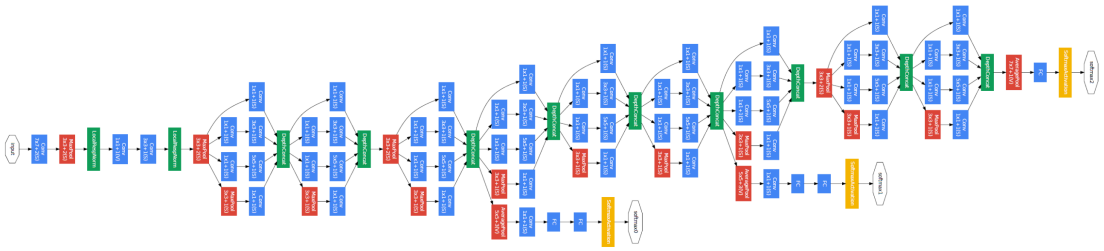
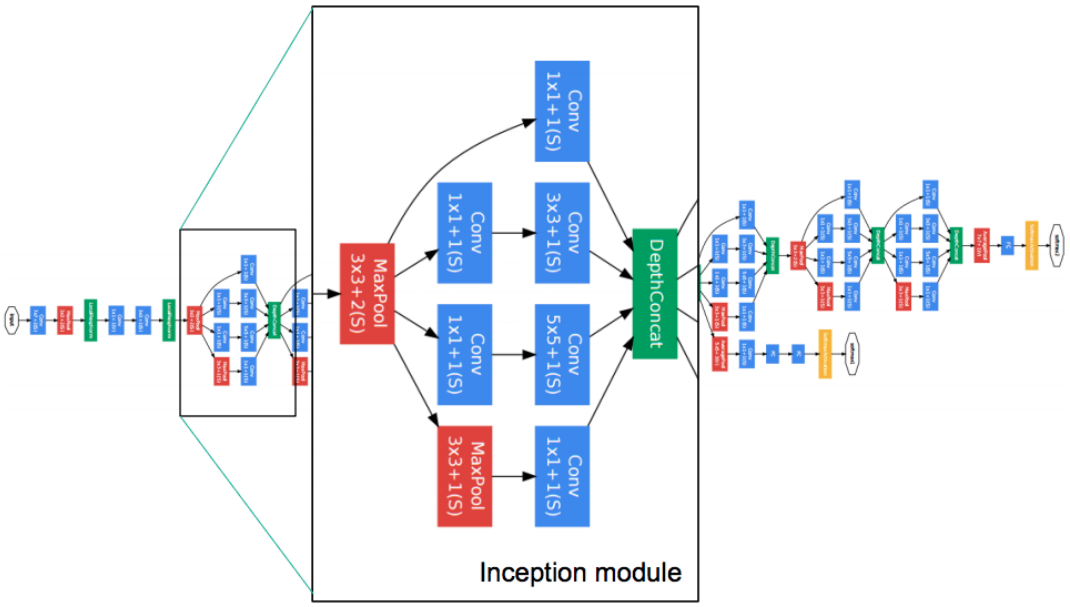


Figure: GoogLeNet Architecture

GoogLeNet

- Inception Module



GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure: Summary Table of GoogLeNet Architecture

GoogLeNet

- 1×1 convolution is used with ReLU. To introduce more non-linearity and to reduce the dimension hence reducing the computation.

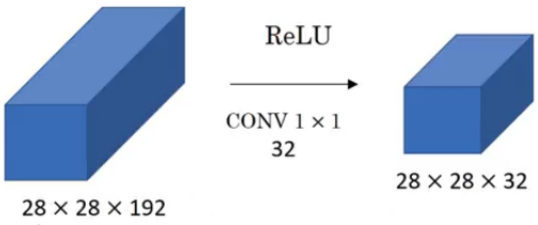
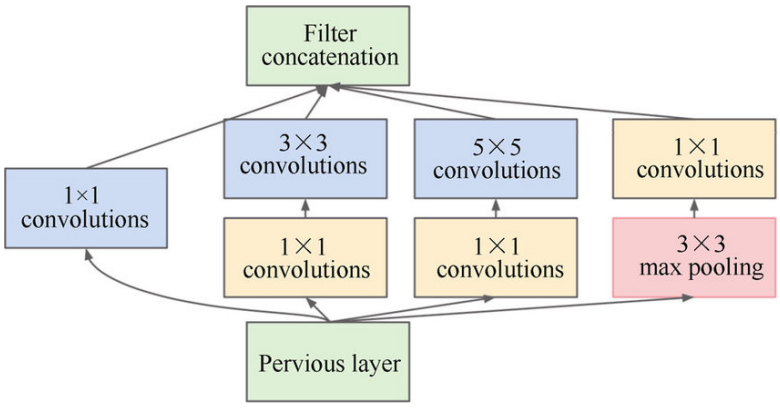


Figure: The number of filters goes from 192 to 32.

GoogLeNet

- Inception Module. 1×1 conv, 3×3 conv, 5×5 conv, and 3×3 max pooling are performed simultaneously on the previous input.
- Different kinds of features are extracted. All feature maps are concatenated as the output.
- 1×1 convolution is inserted into the inception module for dimension reduction.

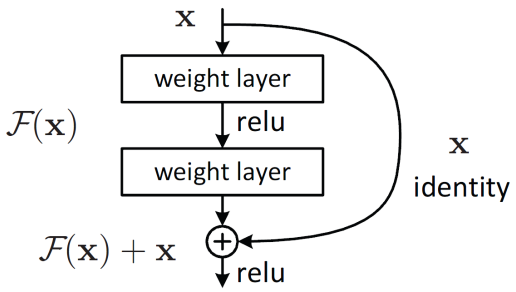


GoogLeNet

- Global Average Pooling. Global average pooling is used near the end of the network by averaging each feature map from 7×7 to 1×1.
- Multi-Scale Testing
- Multi-Crop Testing

Deep Residual Network – ResNet

- The residual network (ResNet) can alleviate the problem of training very deep neural networks.
- Skip Connection in residual networks



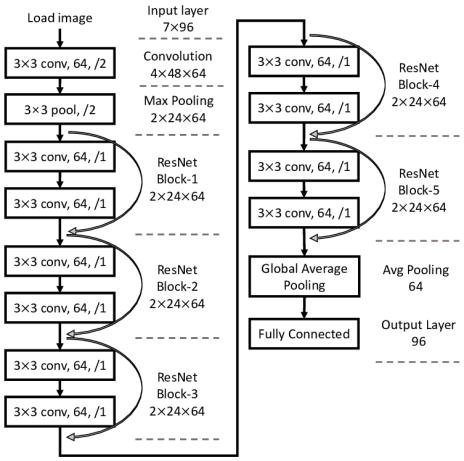
$$\mathbf{y} = F(\mathbf{x}, \{\mathbf{W}_i\}) + \mathbf{W}_s\mathbf{x}$$

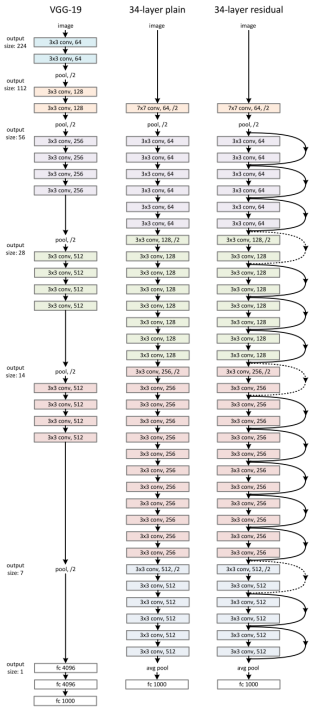
where $\mathbf{W}_s$ is a linear projection if $F(\mathbf{x})$ and $\mathbf{x}$ have different dimensions.

Kaiming He, et al., "Deep Residual Learning for Image Recognition," *arXiv*, 2015.

Deep Residual Network – ResNet

- An ensemble of deep residual networks achieved a 3.57% error rate on ImageNet. Winner of the ILSVRC 2015 classification competition.





Deep Residual Network – ResNet

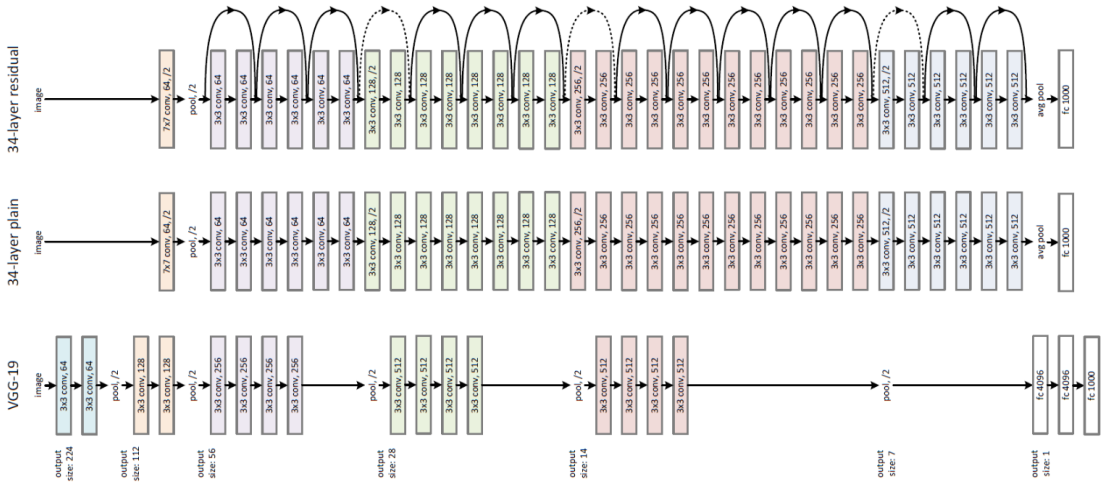
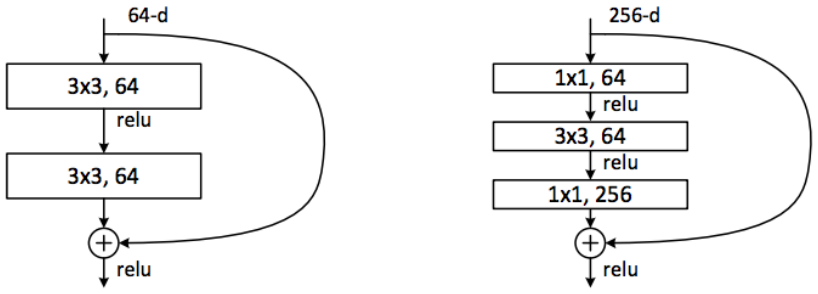


Figure: ResNet Architecture

34-layer ResNet compared with 34-layer plain network and VGG-19.

Deep Residual Network – ResNet

- Bottleneck Design to reduce complexity.
  The 1×1 conv layers are added to the start and the end of a residual block. It reduces the number of parameters while not degrading the performance too much.



- With the bottleneck design, 34-layer ResNet becomes 50-layer ResNet.

Deep Residual Network – ResNet

- ResNet preserves the gradient through the identity mapping.
- It combats the vanishing gradient problem for very deep neural networks.

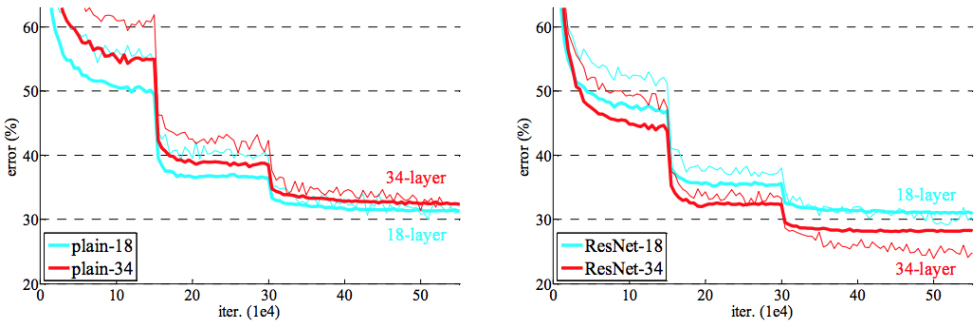
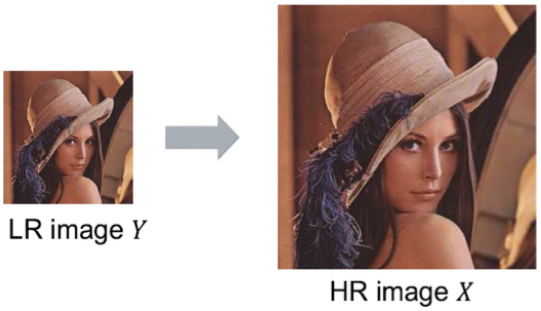


Figure: The 34-Layer ResNet outperforms the 18-Layer ResNet by 2.8%.

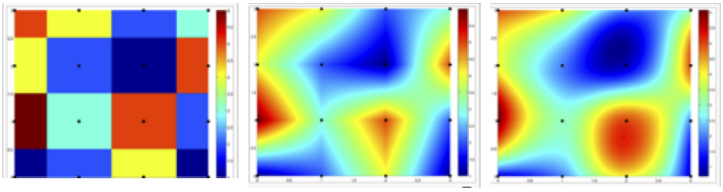
Example: Super Resolution with ResNet

- Super-Resolution: Obtaining a high resolution (HR) image from a low resolution (LR) image.

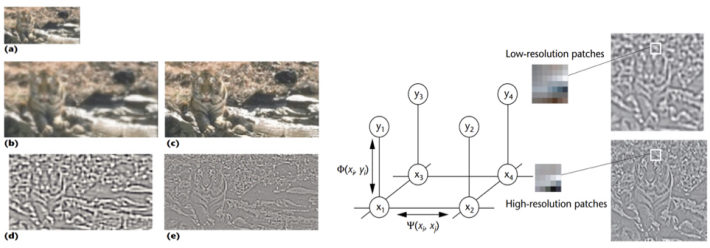


Example: Super Resolution Traditional Methods

- Using Interpolation – Bilinear, Bicubic, Splines
  Results overly smoothed edges and ringing artifacts

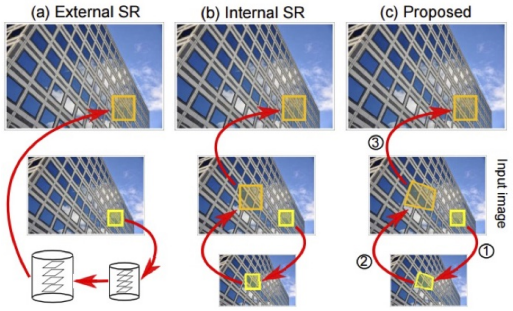


- Using External Database – Lots of HR + LR patch pairs

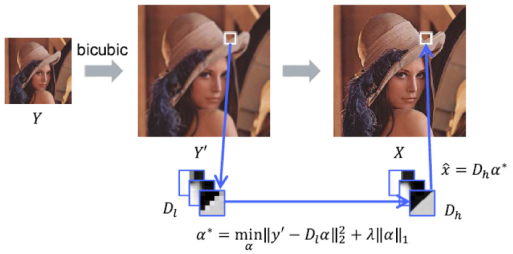


Example: Super Resolution Traditional Methods

- Using Redundancy at different locations and across different scales



- Using Predefined Dictionary. Sparse coding algorithm



Example: Super Resolution with ResNet

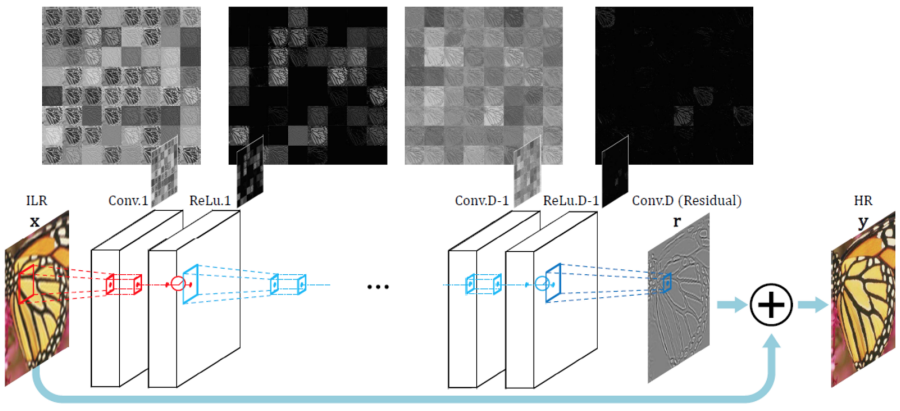
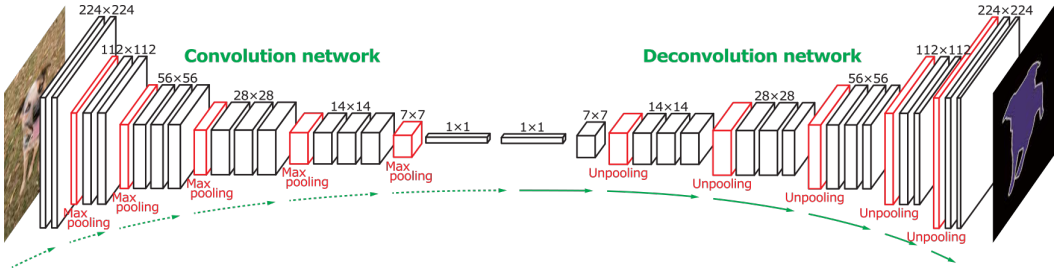


Figure: An interpolated low-resolution (ILR) image goes through layers and transforms into a high-resolution (HR) image. The network predicts a residual image and the addition of ILR and the residual gives the desired output.

J. Kim, J. K. Lee, and K. M. Lee, "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," *IEEE Conf. on Computer Vision & Pattern Recognition (CVPR)*, 2016.

Fully Convolutional Neural Net for Semantic Segmentation



- The fully convolutional networks take input of arbitrary size and produce correspondingly-sized output with inference and learning.

- The fully convolutional networks can be used for Semantic Segmentation.

E. Shelhamer, J. Long, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, April 2017.

Fully Convolutional Neural Network

- The deconvolution or upsampling is accomplished by using fractionally strided convolutions or transposed convolutions at a fractional value, e.g., 0.5.

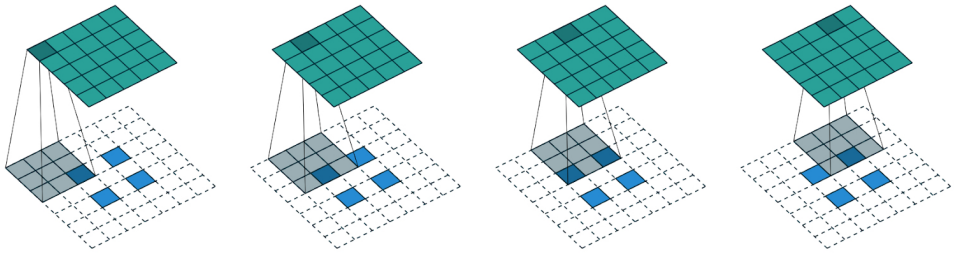
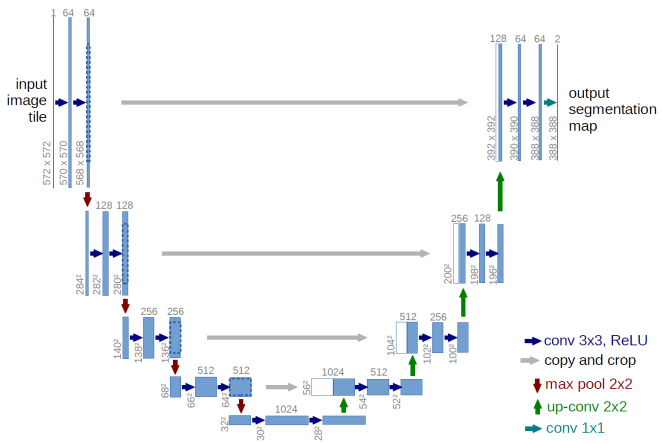
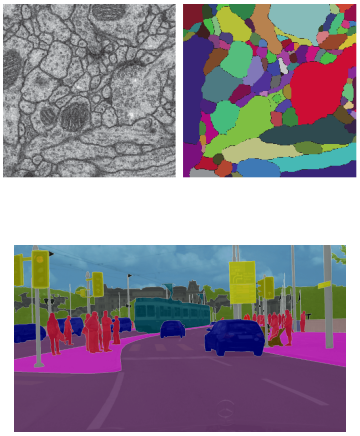


Figure: The blue pixels are the original 2×2 pixels being expanded to 5×5 pixels. All white pixels are zeros.

Vincent Dumoulin and Francesco Visin, "A guide to convolution arithmetic for deep learning," *arXiv*, 2016.

Fully Convolutional Neural Network – U-Net

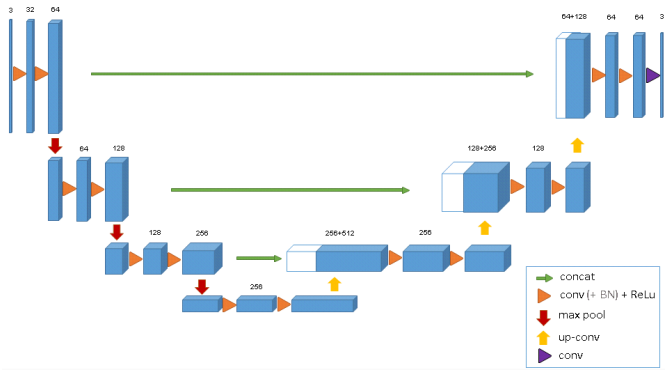


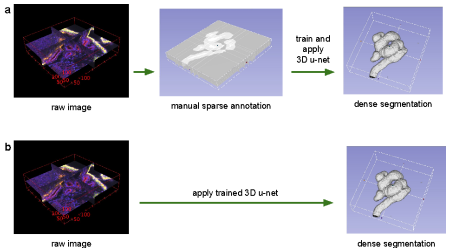


- U-Net is a fully convolutional neural network that was developed for biomedical image segmentation.
- The output is of similar size as the input.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv*, 2015.

Fully Convolutional Neural Network – 3D U-Net





Özgün Çiçek, et al., "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," *arXiv*, 2016.

CNN for Object Detection

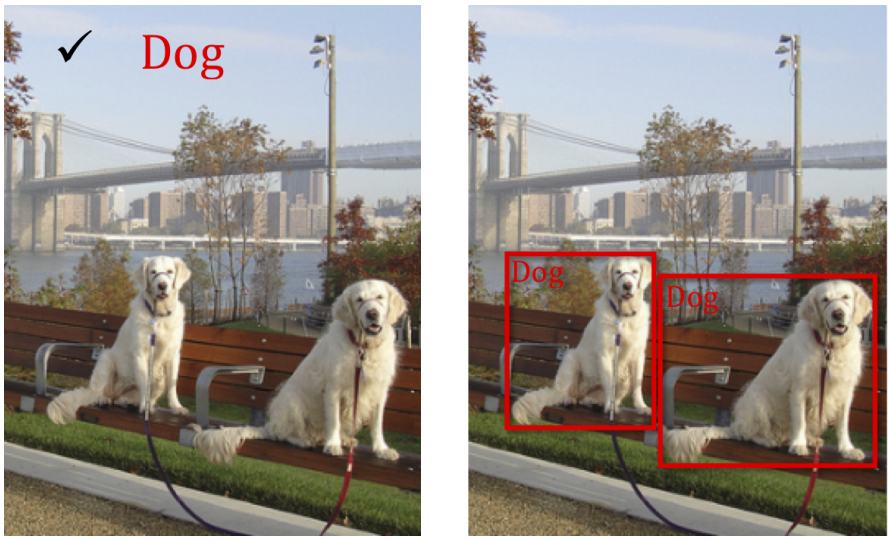
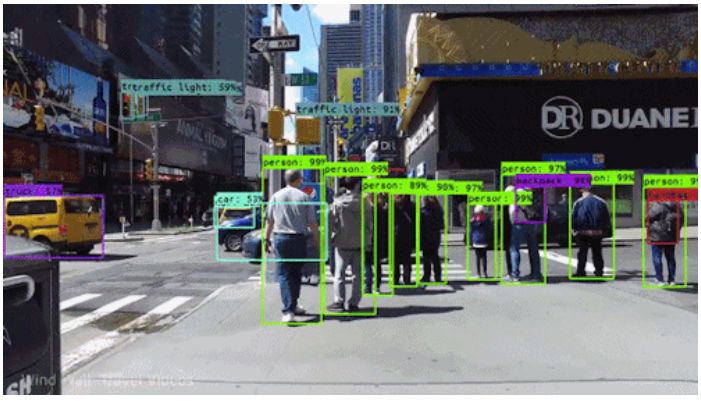


Figure: Classification versus Detection

CNN for Object Detection



Classification with Localization

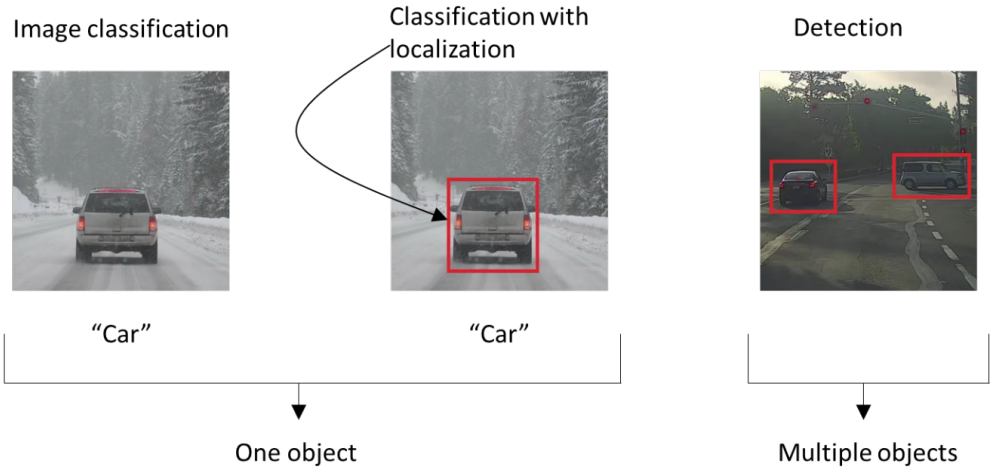
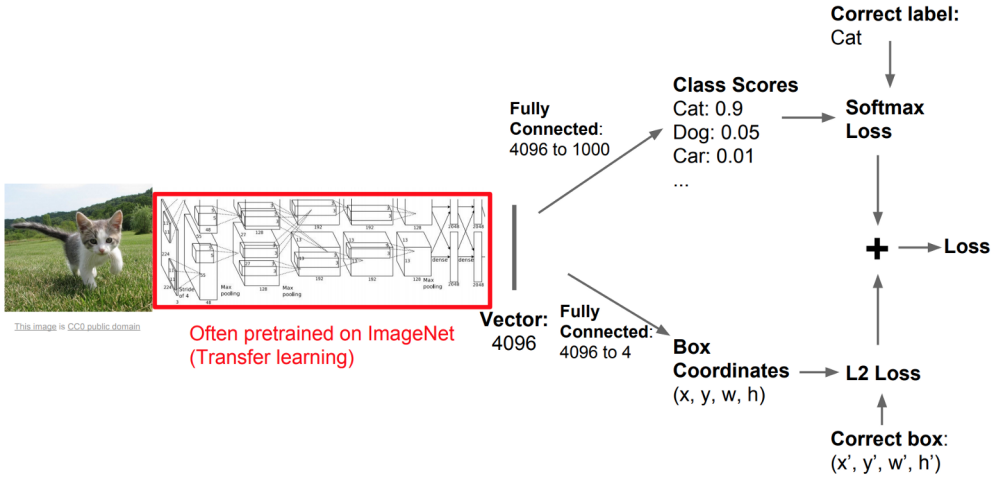


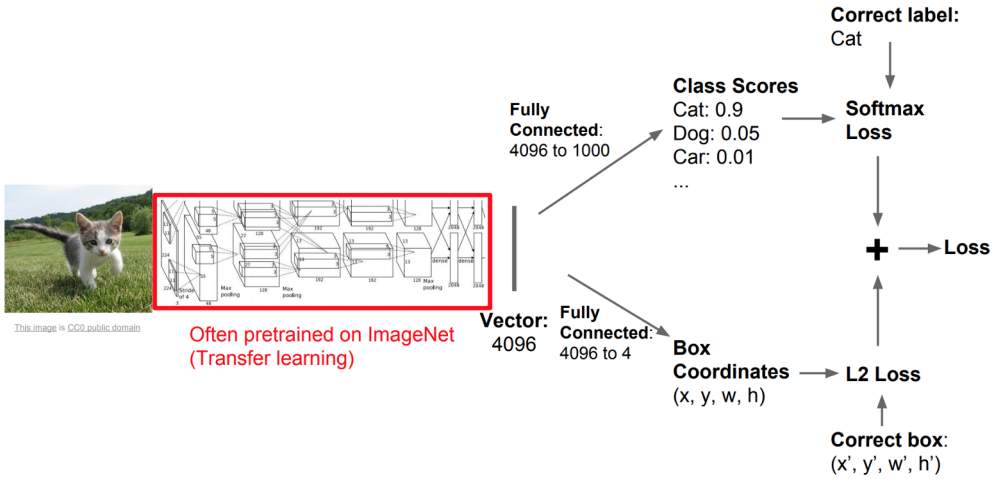
Figure: Classification with localization – Label the object as a car and put a bounding box around the position of the car in the image.

Classification with Localization



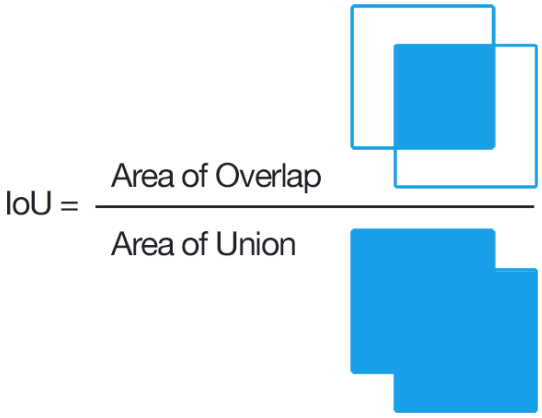
- A classification model (AlexNet, VGG, GoogLeNet) to predict the class label.
  Transfer learning to train the model for specific image dataset.
- Add fully-connected "regression output" to the network.

Classification with Localization



- Regression Output (x, y, w, h). x, y are the midpoint (or top-left) coordinates and w, h are the width and height of the bounding box.
- The Loss is a weighted sum of the softmax loss of the classification problem and the L_2 loss of the regression problem.

Metrics – Intersection over Union (IoU)



- Intersection over Union (IoU) is defined for the extend of overlap between two bounding boxes.
- It provides a score, between 0 and 1, representing the quality of overlap between the two bounding boxes.

Object Detection – Region-based Object Detectors

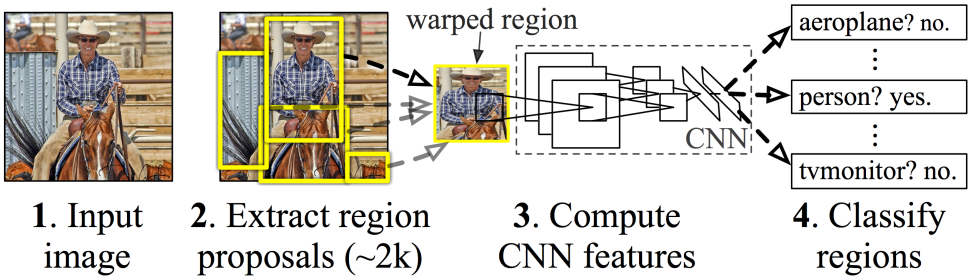
- The method of Regions with CNN features (R-CNN) for object detection uses region proposals.
- Region Proposals: Image regions that are likely to contain objects.
- Use Selective Search for region proposals.
- Selective search applies hierarchical grouping algorithm to merge most similar regions together based on similarity in color, texture, size and fill.

Uijlings, van de Sande, Gevers, and Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sept. 2013.

Selective Search

Figure: First row – bottom-up segmentation, merging regions at multiple scales. Second row – Convert regions to boxes. Blue rectangles are possible region proposals and green rectangles are the target objects that we want to detect.

Regions with CNN features (R-CNN)



- Use a region-extraction algorithm to propose about 2,000 objects' boundaries. (Selective search – high computation)
- Every region proposal is wrapped into a fixed-size 227×227 RGB image. It is processed by a CNN to extract a 4096-dimensional feature.
- A support vector machine (SVM) is applied to identify the object.

R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *arXiv*, 2013.

Regions with CNN features (R-CNN)

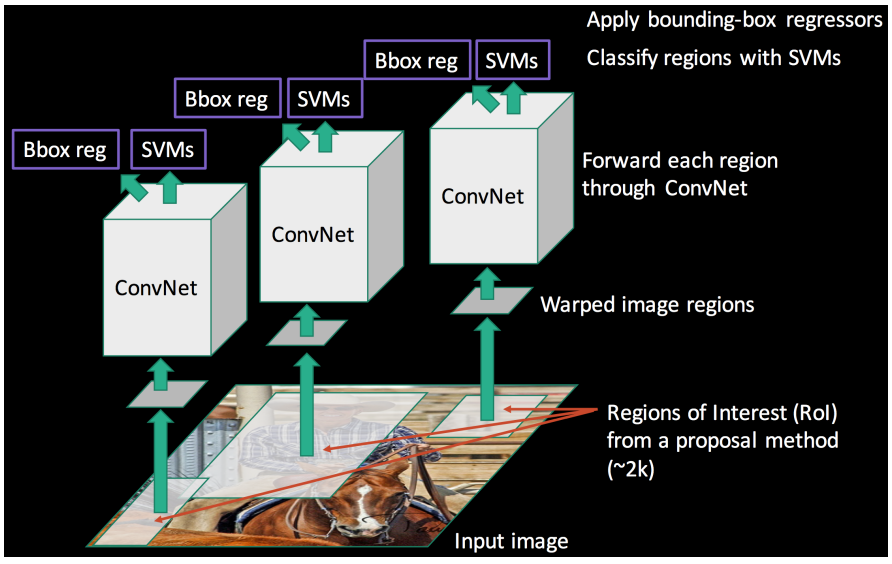


Figure: R-CNN classifies objects and produces the corresponding boundary box.

Regions with CNN features (R-CNN)

- At test time, we predict detection boxes using class-specific SVMs.
- Non-maximum suppression is used to deal with a lot of overlapping detection boxes at the time of testing.

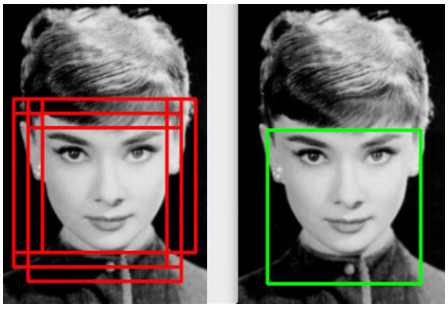


Figure: Sort all detection boxes on the basis of their scores. The detection box M with the maximum score is selected and all other detection boxes with a significant overlap (using a pre-defined threshold) with M are suppressed.

Fast R-CNN

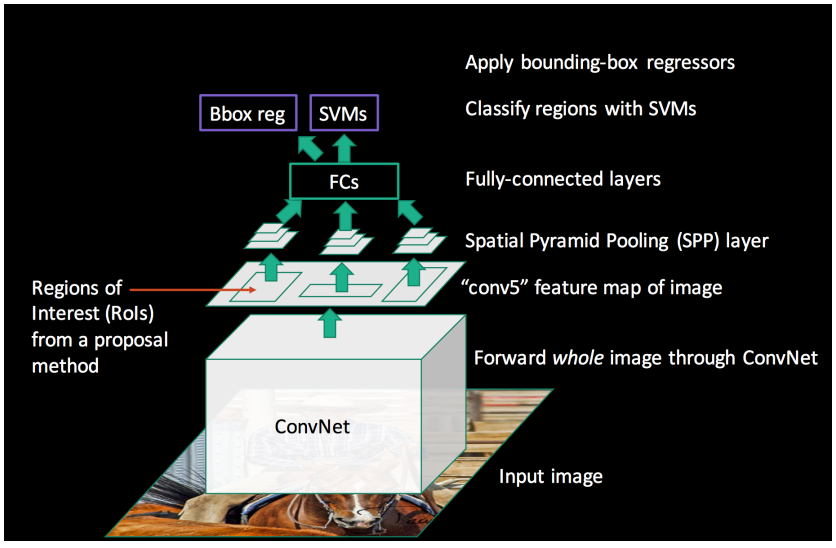


Figure: Run a single CNN on the input image and then apply region proposal crops on the features calculated by the CNN. Wrap region of interests (RoIs) into spatial pyramid pooling (SPP) layers.

Fast R-CNN

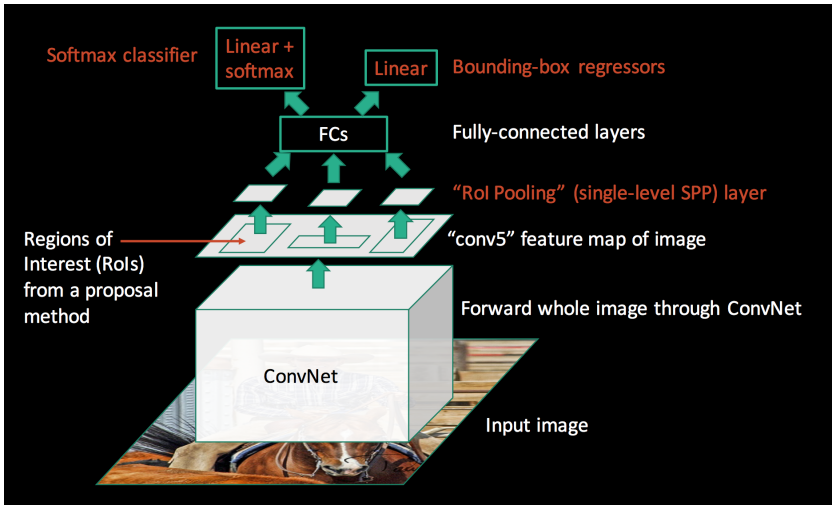
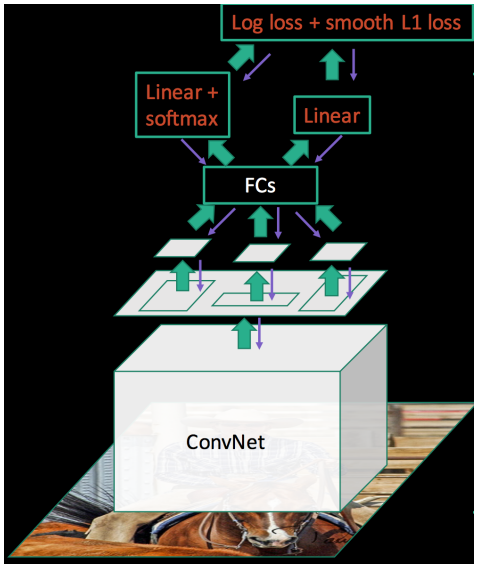


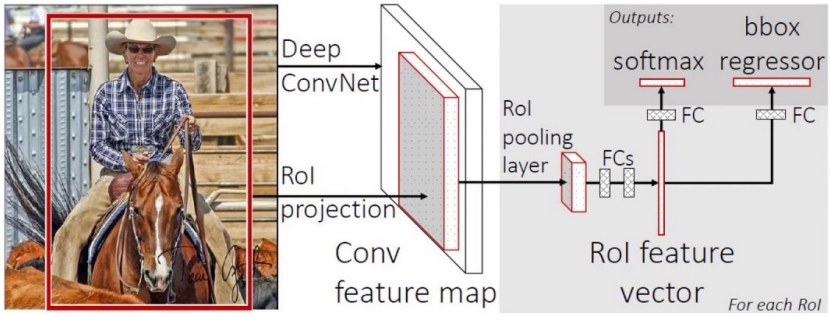
Figure: Instead of generating a pyramid of layers, Fast R-CNN wraps RoIs into one single layer using the RoI pooling. It is then fed into fully-connected layers for classification. The bounding box is further refined with linear regression.

Fast R-CNN



- The parameters including those of the CNN are trained together with a $\log$ loss function from the class classification and a L_1 loss function from the boundary box linear regression.

Fast R-CNN



Ross B. Girshick, "Fast R-CNN," *arXiv*, 2015.

Faster R-CNN

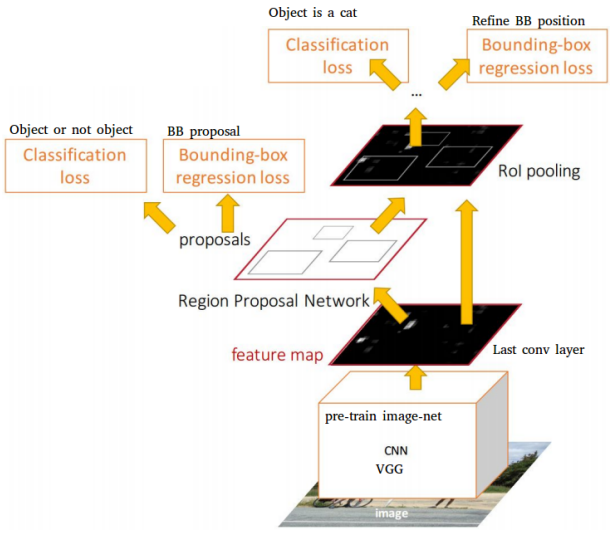


Figure: Faster R-CNN

- Can the network itself do region proposals?
- A region proposal network is trained that takes the feature map as input and outputs region proposals. These proposals are then fed into the RoI pooling layer in the Fast R-CNN.

S. Ren, K. He, and R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *arXiv*, 2015.

Region Proposal Network

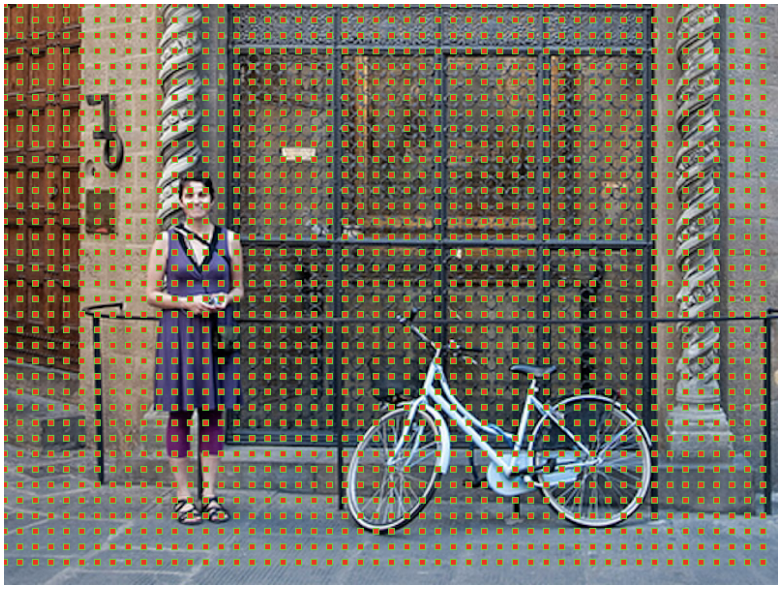
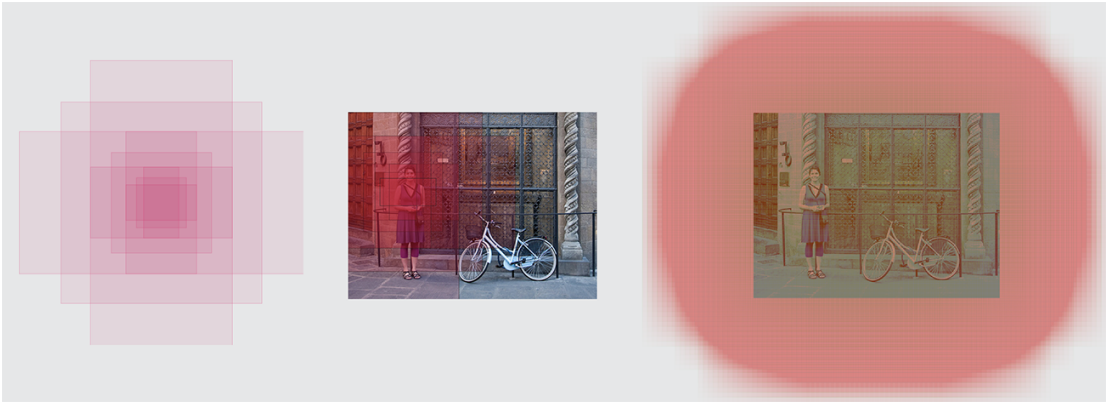


Figure: Anchors are fixed bounding boxes that are placed throughout the image to be used for predicting object locations. The anchor centers are evenly separated.

Region Proposal Network



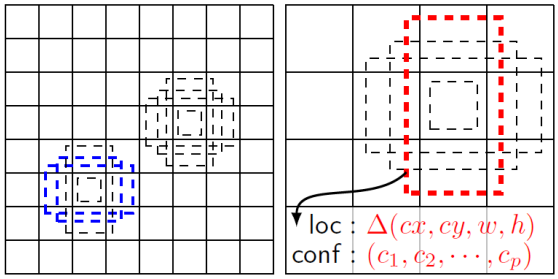
- In Faster R-CNN, 9 anchor boxes are generated per anchor center.
- The Region Proposal Network classifies which anchor boxes have objects and regresses the offsets of the bounding boxes.
- Non-maximum suppression is used to reduce region proposals.

Object Detection – Single-shot Object Detectors

- Single Shot Detector (SSD) takes one single shot to detect multiple objects within the image.
- *c.f.* Region-based detectors need two shots, one for generating region proposals, one for detecting the object of each proposal.

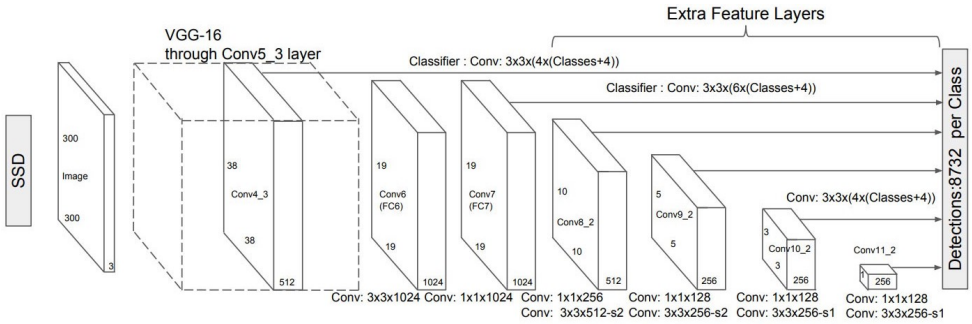
W. Liu, et al. "SSD: Single Shot MultiBox Detector," *Lecture Notes in Computer Science*, pp. 21–37, 2016.

Multibox Detector



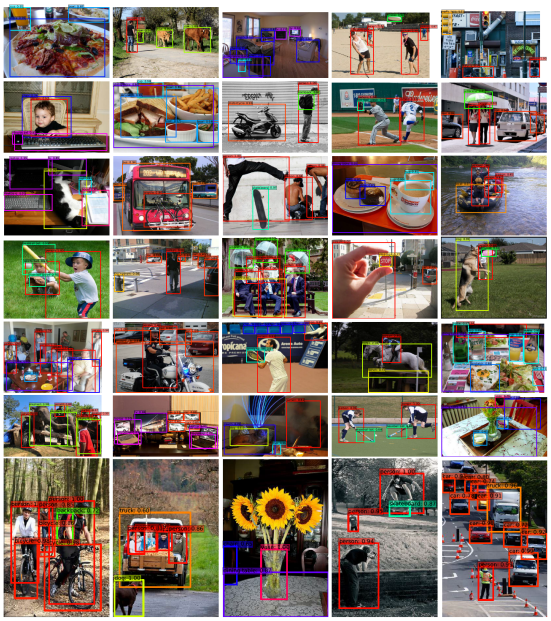
- A 3×3 conv is applied on the CNN feature layers. For example, the 8×8 and 4×4 feature maps in the figure with p channels each.
- There are k bounding boxes for each location. These k bounding boxes have different sizes and aspect ratios. (e.g., $k = 4$ above.)
- For each bounding box prior (anchor box), c class scores are computed with 4 offsets relative to the original default bounding box shape.

Single Shot Detector (SSD)



- SSD uses layers already deep down in the convolutional neural network to detect objects.
- For example, at Conv4_3 (size $38 \times 38 \times 512$), 3×3 conv is applied. There are 4 bounding boxes and each has (Classes + 4) outputs.
- Different layers of CNN feature maps go through a small 3×3 conv for multiple-object detection. Total 8732 bounding boxes.

Single Shot Detector (SSD)



Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

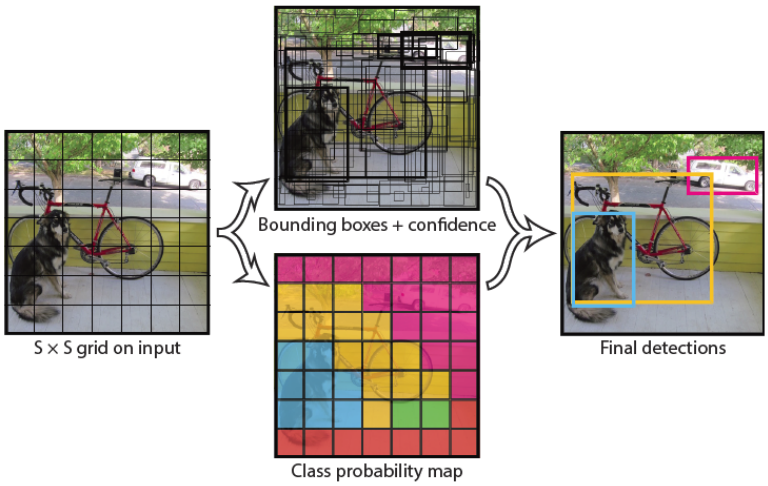
Figure: Accuracy and Inference Time.

Two models:

SSD300: 300×300 input image, lower resolution, faster.

SSD512: 512×512 input image, higher resolution, more accurate.

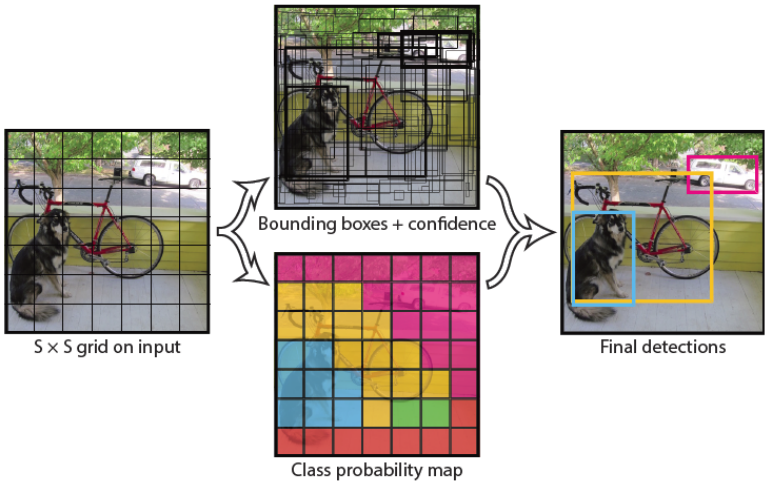
You Only Look Once (YOLO)



- YOLO only looks the image once to detect multiple objects.
- Detection speed is in real-time.

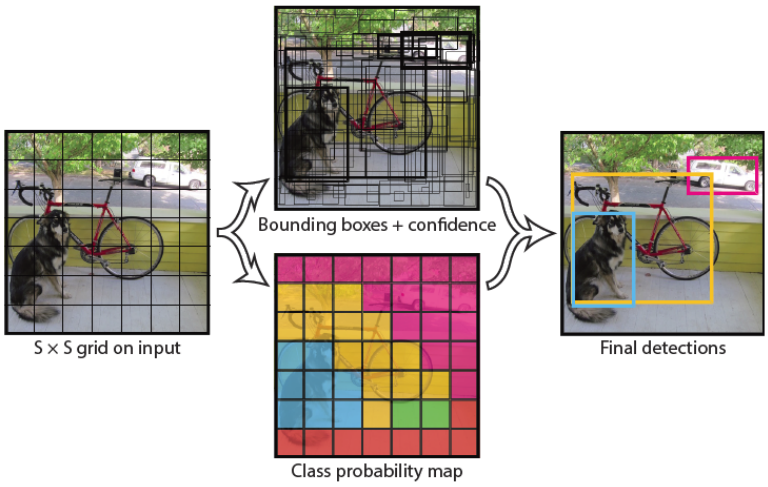
J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

You Only Look Once (YOLO)



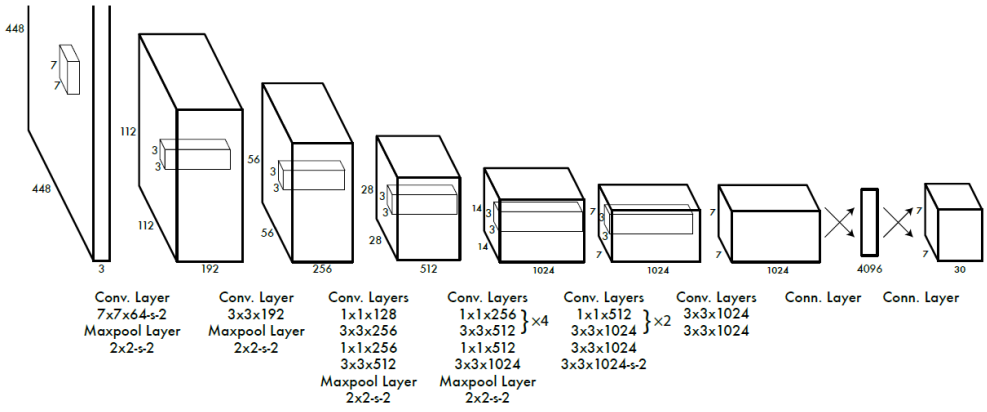
- The input image is divided into an $S \times S$ grid ($S = 7$).
- Each grid cell predicts $B = 2$ bounding boxes and confidence scores for those boxes. The confidence scores reflect how confident the model is that the box contains an object.

You Only Look Once (YOLO)



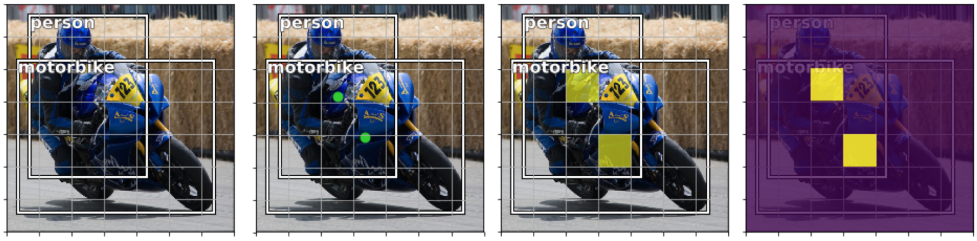
- Each bounding box consists of 5 predictions: x, y, w, h, *confidence*.
- Each grid cell also predicts conditional class probabilities $P(Class \mid Object)$. Total number of classes is 20.
- Therefore, the output size is $7 \times 7 \times (2 \times 5 + 20) = 1470$.

You Only Look Once (YOLO)



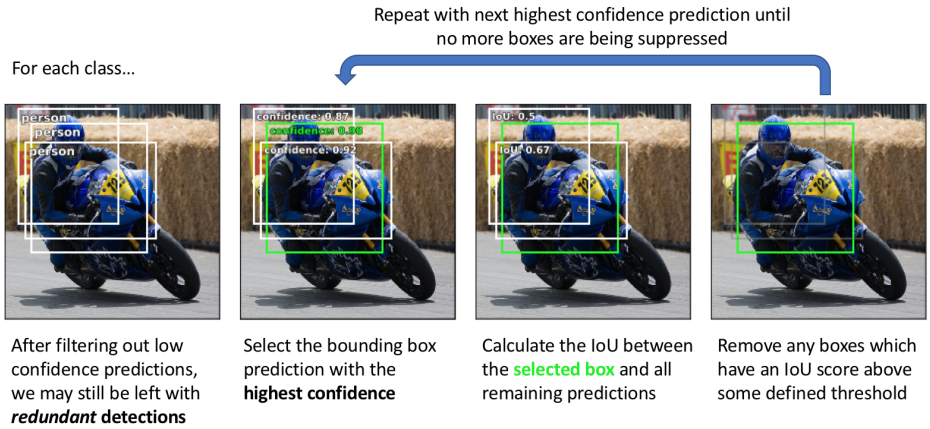
- The model consists of 24 convolutional layers followed by 2 fully connected layers.
- Alternating 1×1 convolutional layers reduce the features space from preceding layers.
- Except for the final layer, all other layers use leaky ReLU as activation function.

YOLO



- If a grid cell contains the center of the bounding box, this cell is "responsible" for detecting the specific object.

YOLO

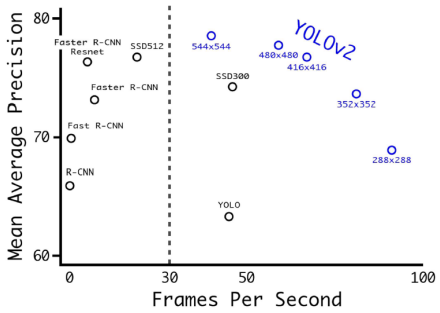


- Most bounding box predictions are filtered out if they are below a confidence threshold.
- Non-max suppression is used to remove redundant high-confidence predictions. It is performed on each class separately.

YOLO v2

Incremental Improvements of YOLO version 2.

	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**



- For high resolution YOLOv2, 78.6% mean Average Precision (mAP) is obtained at real-time speed.

Joseph Redmon and Ali Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv*, 2016.

YOLO v2

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure: Darknet-19 classification network is used in YOLOv2 for feature extraction.

YOLO v2

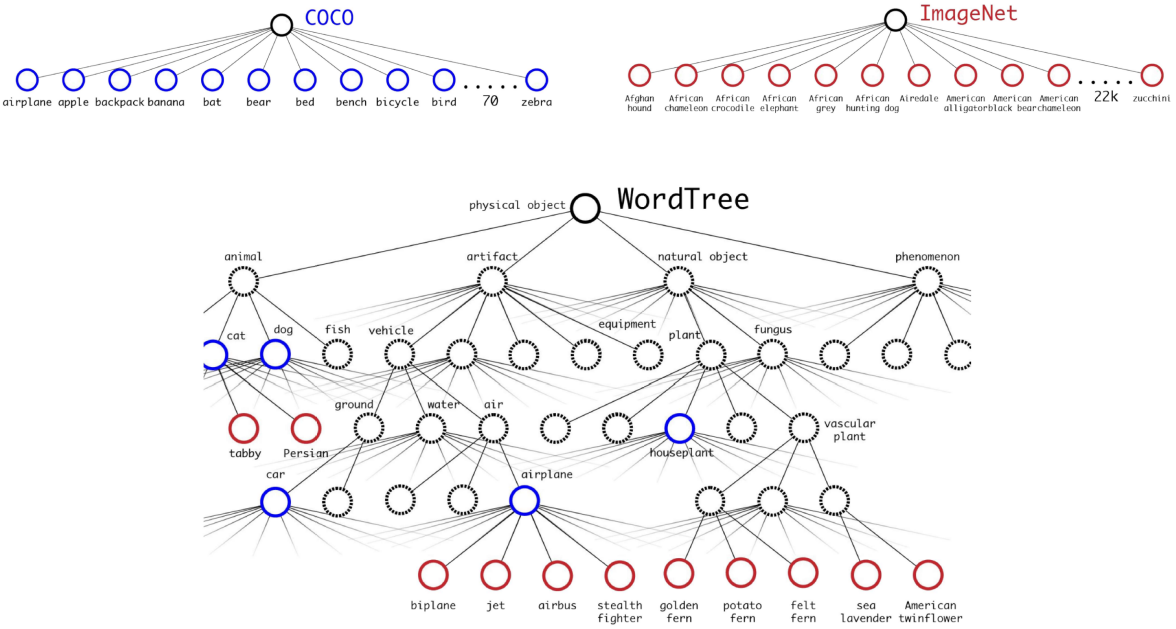


Figure: WordTree is used to combine multiple datasets for classification and detection. It is a hierarchical tree to relate the classes and subclasses together.

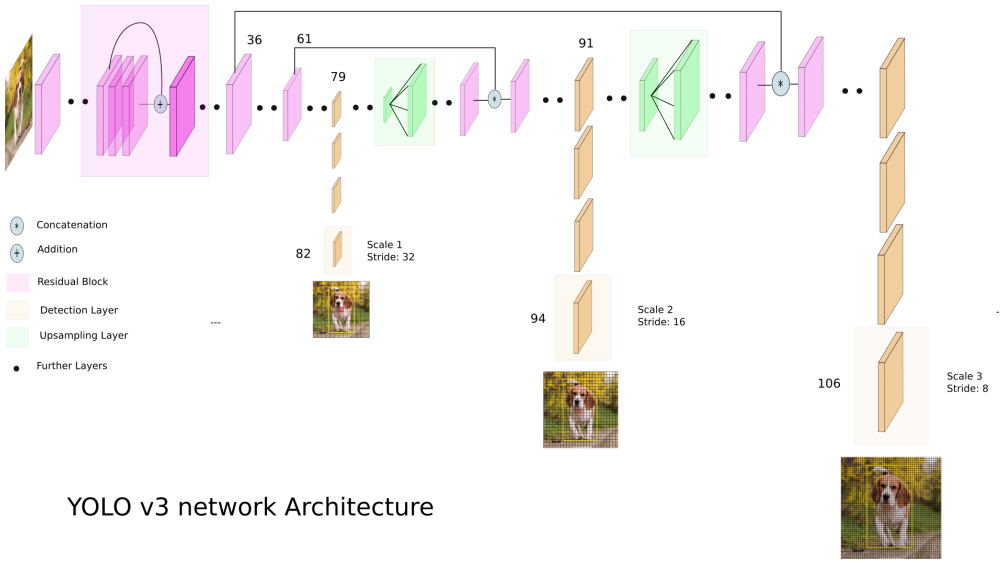
YOLO v3

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

- Darknet-53 is used in YOLOv3, i.e., much deeper with 53 convolutional layers.
- The architecture has residual skip connections and upsampling.

Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

YOLO v3

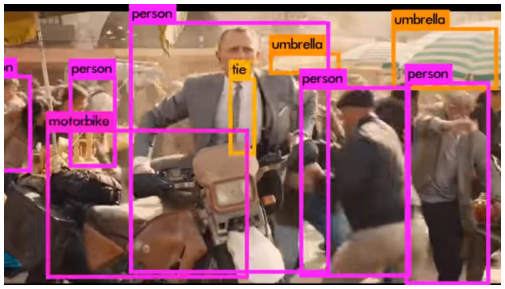


YOLO v3 network Architecture

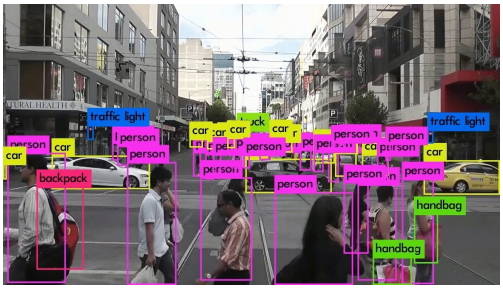
- YOLOv3 makes detection at three different scales. That helps detect small objects. On a grid, three anchor boxes for each scale.

YOLO Demo

YOLOv2 and YOLOv3 Demostration



YOLOv2 ← Click here



YOLOv3 ← Click here

RetinaNet

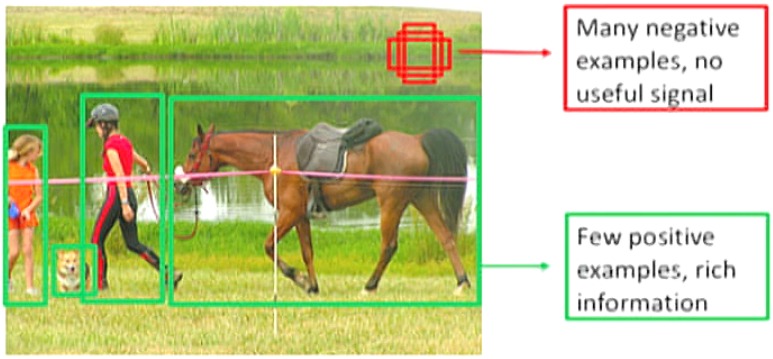


Figure: One-stage detector has a foreground-background class imbalance problem. No region proposal network to filter out many negative background samples.

- RetinaNet is an one-stage detector that improves prediction accuracy by using focal loss.

T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2999–3007.

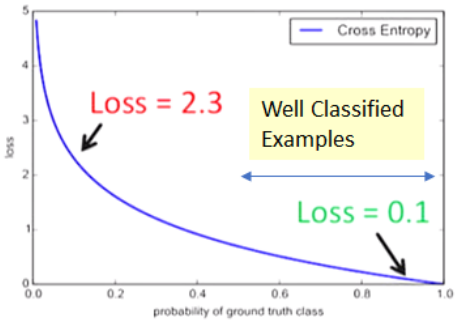
RetinaNet – Focal Loss

- Binary Cross-Entropy (CE) Loss

$$\mathrm{CE} = -\sum_{i=1}^{2} t_i \log(p_i) = -t_1 \log(p_1) - (1 - t_1) \log(1 - p_1)$$

where t_i and p_i are the groundtruth class (one-hot) and model predicted probability.

- 100000 easy : 100 hard examples
- **40x bigger loss** from easy examples



Class imbalance problem. e.g., 100,000 easy examples each with 0.1 loss and 100 hard examples each with 2.3 loss. The sum loss of easy examples is bigger.

RetinaNet – Focal Loss

- RetinaNet have ∼100,000 bounding boxes.
- By using focal loss, the total loss can be balanced adaptively between easy examples and hard examples.

$$\mathrm{FL} = -\sum_{i=1}^{2}(1-p_i)^{\gamma} t_i \log(p_i) = \begin{cases} -(1-p_1)^{\gamma}\log(p_1) & \text{if } t_1 = 1 \\ -p_1^{\gamma}\log(1-p_1) & \text{if } t_1 = 0 \end{cases}$$

  The focusing parameter $\gamma \in [0, 5]$ smoothly adjusts the rate at which easy examples are down-weighted.

- α-balanced variant of focal loss.

$$\mathrm{FL} = -\sum_{i=1}^{2}\alpha_i(1-p_i)^{\gamma} t_i \log(p_i)$$

  The weighting factor α can be set by inverse class frequency or treated as a hyperparameter.

RetinaNet

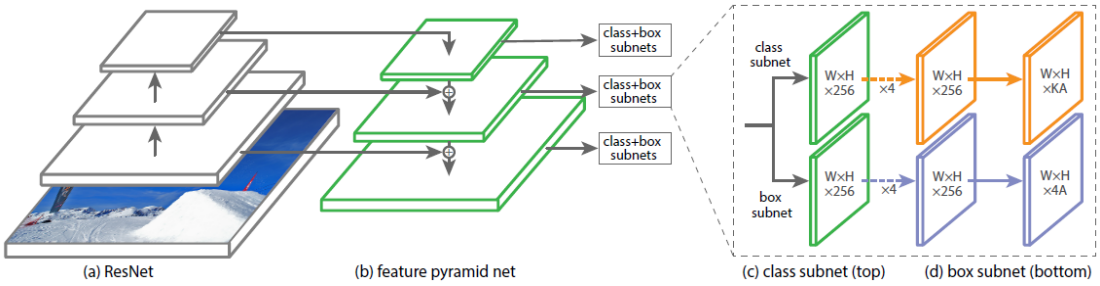


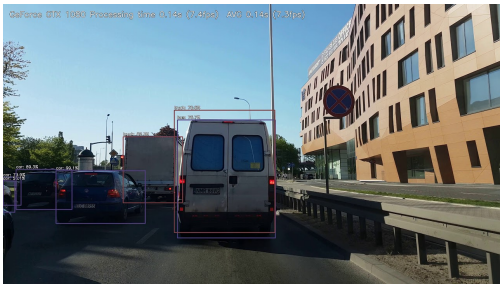
Figure: RetinaNet Architecture.

- RetinaNet uses ResNet and Feature Pyramid Network (FPN) for feature extraction and two task-specific subnetworks for classification and bounding box regression.
- FPN constructs a rich multi-scale feature pyramid from one single resolution input image.

RetinaNet Demo

RetinaNet Demostration

ResNet50 RetinaNet ← Click here



ResNet50 RetinaNet ← Click here