Computer Vision

Lecture 10 – Recognition

Prof. Dr.-Ing. Andreas Geiger

Autonomous Vision Group

University of Tübingen / MPI-IS

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

ellis
European Laboratory for Learning and Intelligent Systems

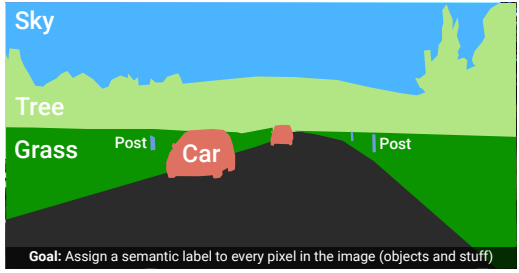
Agenda

10.1 Image Classification

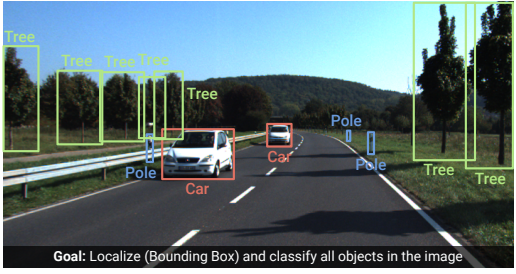
10.2 Semantic Segmentation

10.3 Object Detection and Segmentation

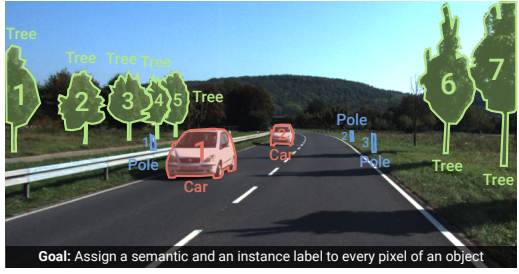
Image Classification



Semantic Segmentation



Object Detection



Instance Segmentation

10.1
Image Classification

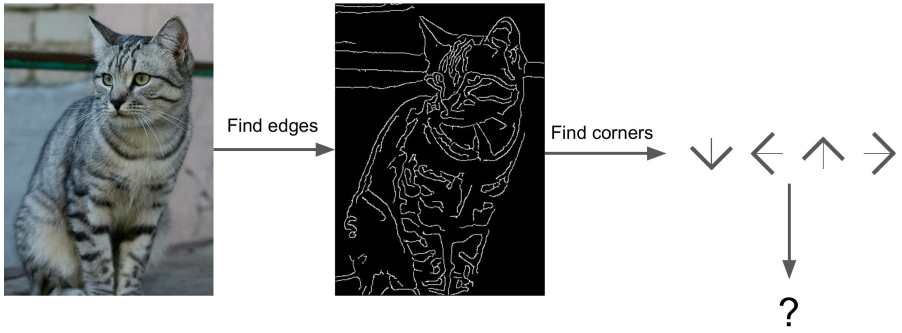
Image Classification

Image Classification

Why Image Classification?

- Useful task on its own (e.g., handwritten digits / image search)
- Simple input/output specification, standard loss functions
- For long time, one of the difficult gold-standard tasks in computer vision
- Useful high-level proxy-task to learn good representations
  which transfer to new tasks and domains

Early Attempts

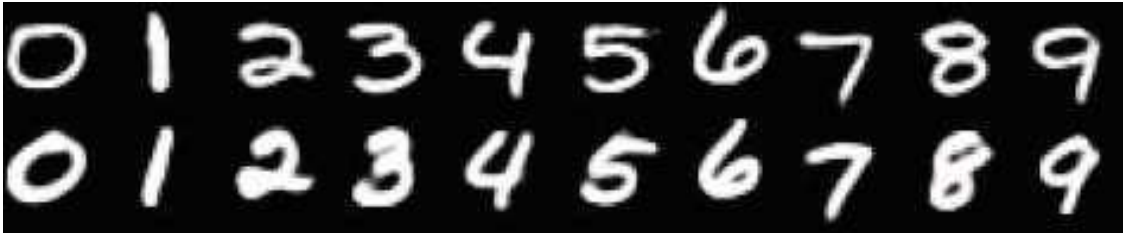


- Early attempts of **hand-engineering** object detection algorithms **failed** as object shapes and appearances are simply to hard to describe
- We need **machine learning** and labeled datasets to learn these relationships!

Slide Credits: Li Fei-Fei.

Datasets

MNIST



MNIST Handwritten Digits:

- One of the most popular datasets in ML (many variants, still in use today)
- Based on a data from the National Institute of Standards and Technology
- Hand written by Census Bureau employees and high-school children
- Resolution: 28 x 28 pixels, 60k training samples with labels, 10k test samples

LeCun, Bottou, Bengio and Haffner. Gradient-based learning applied to document recognition. IEEE, 1998.

Caltech101

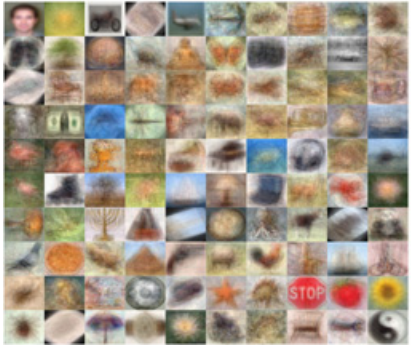






Caltech101:

- Caltech101 was the first major object recognition dataset, collected in 2003
- 101 object categories
- Hand-curated from Google Image Search
- Biased: canonical size and location



Fei-Fei, Fergus and Perona: Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories. CVPR, 2004.

ImageNet

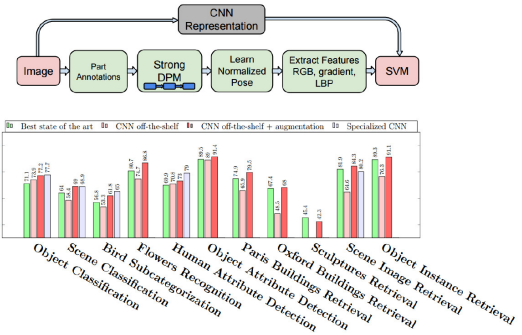
- AlexNet results on ImageNet challenge started the **deep learning revolution**
- Still one of the primary datasets for **pre-training** generic representations today

Deng, Dong, Socher, Li, Li, Fei-Fei: ImageNet: A large-scale hierarchical image database. CVPR, 2009.

ImageNet Pre-Training

Transfer Learning:

- Deep representations generalize well despite large number of parameters
- Pre-train CNN on large amounts of data on generic task (e.g., ImageNet)
- Fine-tune (re-train) only last layers or all layers on little data of a new task leads to state-of-the-art performance
- Standard paradigm today



Challenges

Challenges: Large Number of Image Categories

Slide Credits: Koenderink, Lazebnik, Fei-Fei.

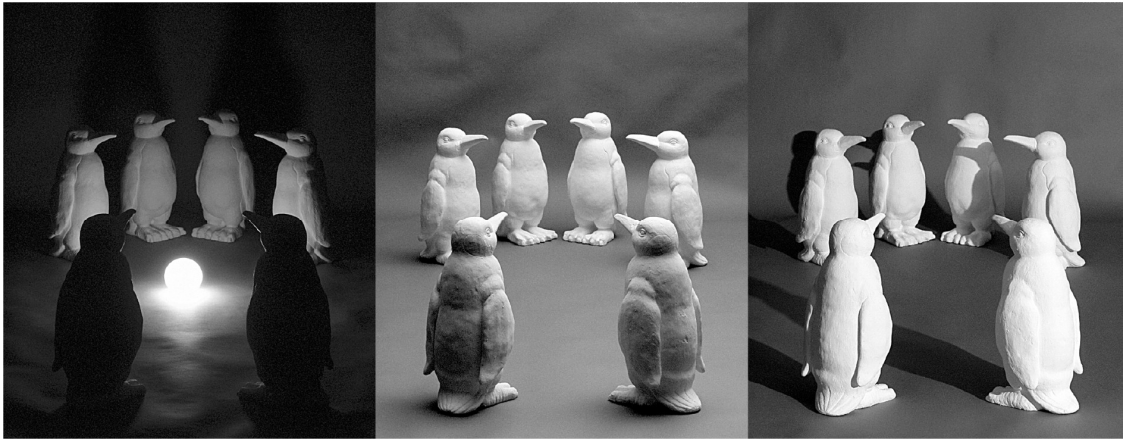
Challenges: Intra-class Variation

Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Challenges: Viewpoint Variation

Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Challenges: Illumination Changes

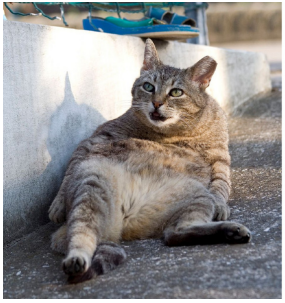
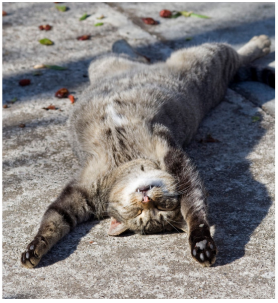


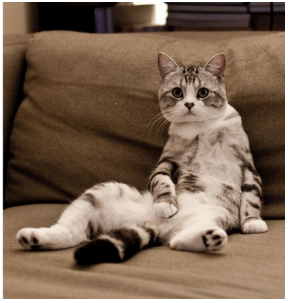
Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Challenges: Background Clutter

Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Challenges: Deformation



Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Challenges: Occlusion



Slide Credits: Koenderink, Lazebnik, Fei-Fei.

Simple Models

Nearest Neighbor

Nearest Neighbor Classifier:

- Choose image distance:

$$d(\mathbf{I}_1, \mathbf{I}_2) = \sum_{\mathbf{p}} |I_1(\mathbf{p}) - I_2(\mathbf{p})|_1$$

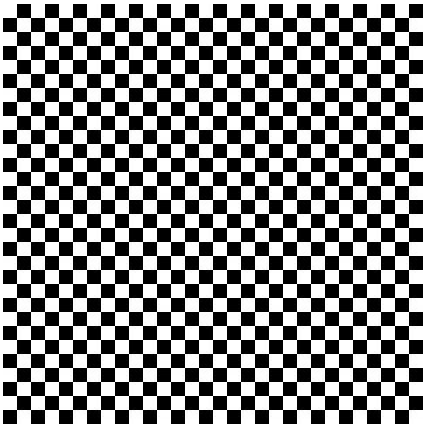
- Given $\mathbf{I}$, find nearest neighbor:

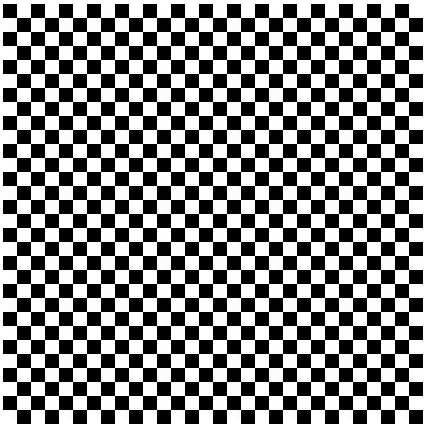
$$\mathbf{I}^* = \underset{\mathbf{I}' \in \mathcal{D}}{\operatorname{argmin}}\ d(\mathbf{I}, \mathbf{I}')$$

- Return class of $\mathbf{I}^*$

- This is a **slow** (NN at test time) and **bad** (pixel distances uninformative) **classifier**

Nearest Neighbor





Thought Experiment:

- Two checkerboards, horizontally displaced by one field $\Rightarrow$ What is $d(\mathbf{I}_1, \mathbf{I}_2)$?

Bag-of-Words (BoW)

- Idea: Obtain spatial invariance by comparing **histograms of local features**

Fei-Fei and Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. CVPR, 2005.

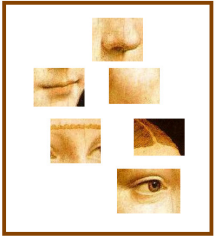
Bag-of-Words (BoW)

- The idea of bag-of-word models originates from natural language processing
- **Orderless** document representation: **frequencies of words** from a dictionary

Fei-Fei and Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. CVPR, 2005.

Bag-of-Words (BoW)

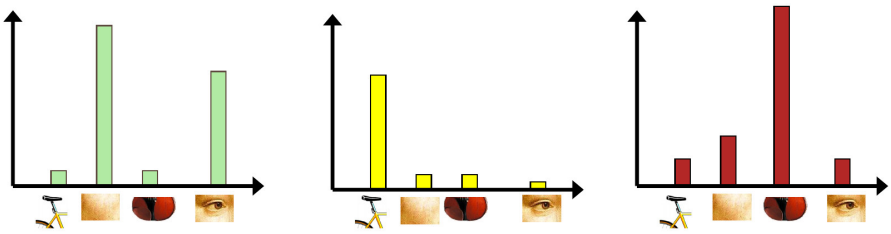
1. **Extract features** (e.g., SIFT detector)
2. Learn **visual vocabulary** (e.g., k-means on SIFT feature vectors)
3. **Quantize features** into visual words using vocabulary (nearest neighbors)
4. Represent images by **histograms** of visual word frequencies
5. **Train classifier** (e.g., k-NN, SVM, random forest, neural network)



Fei-Fei and Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. CVPR, 2005.

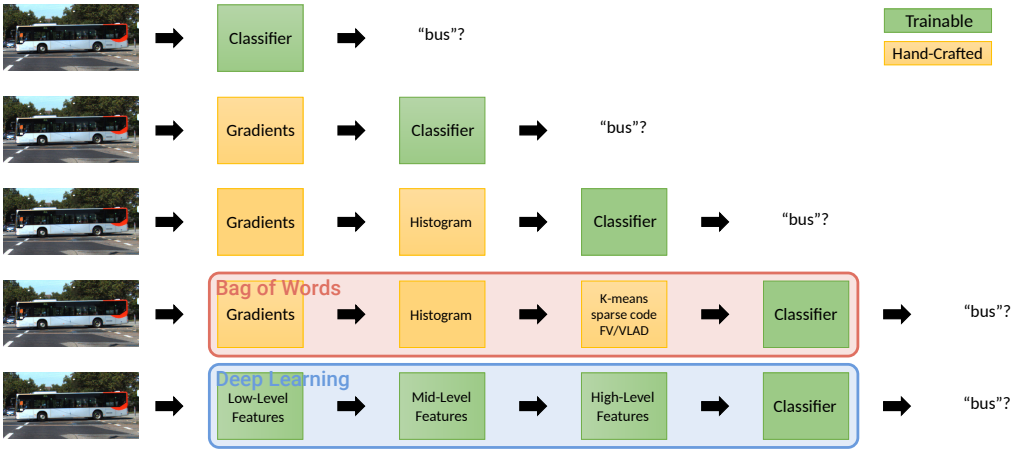
Bag-of-Words (BoW)

1. **Extract features** (e.g., SIFT detector)
2. Learn **visual vocabulary** (e.g., k-means on SIFT feature vectors)
3. **Quantize features** into visual words using vocabulary (nearest neighbors)
4. Represent images by **histograms** of visual word frequencies
5. **Train classifier** (e.g., k-NN, SVM, random forest, neural network)



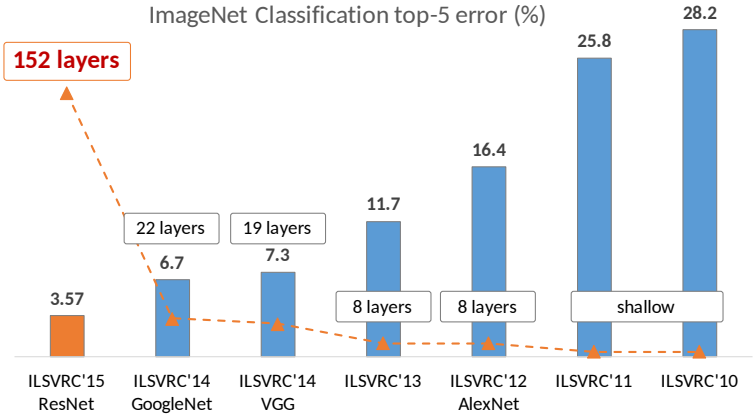
Fei-Fei and Perona. A Bayesian Hierarchical Model for Learning Natural Scene Categories. CVPR, 2005.

Representation Learning



- Problem with BoW: Still **many hand-designed components**, little learning
- Solution: Learn entire model (from image to decision) **end-to-end** from data

ImageNet Large Scale Visual Recognition Challenge

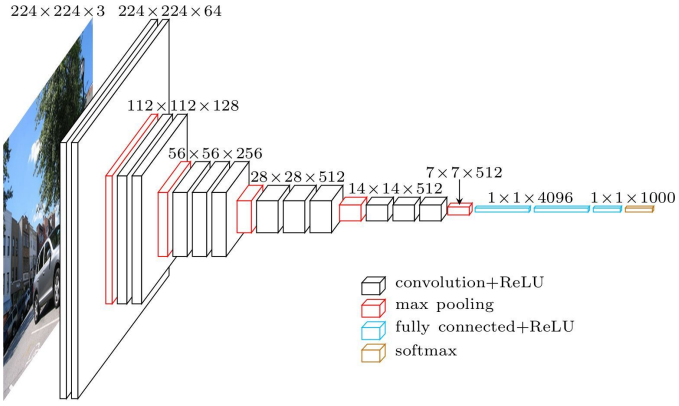


- Classification into 1000 object categories. Current state-of-the-art: 1.3 %

Russakovsky et al.: ImageNet Large Scale Visual Recognition Challenge. IJCV, 2015.

Convolutional Neural Networks

VGG Network

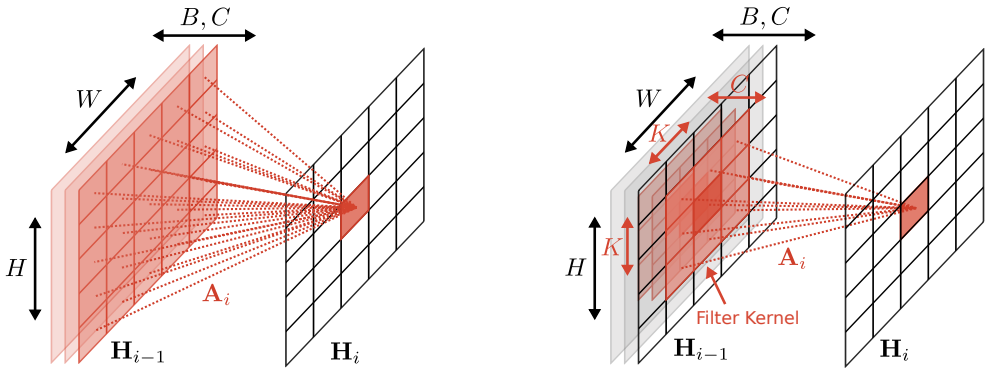


3 Types of Layers:

- Convolution layers, downsampling layers and fully connected layers

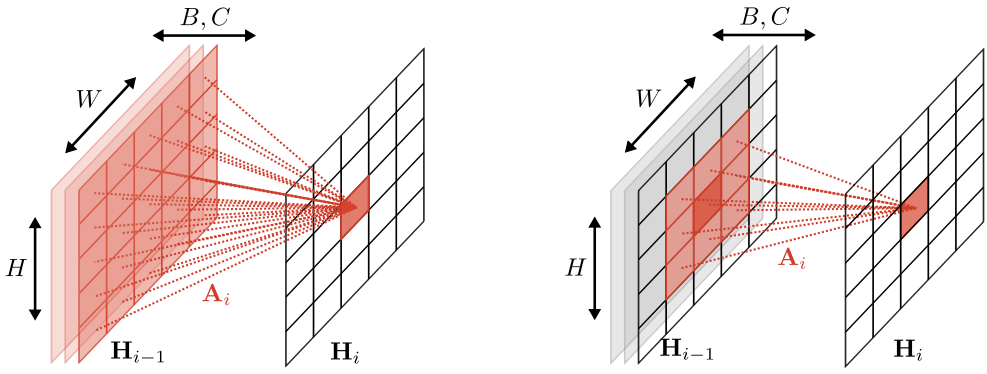
Simonyan and Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

Convolutional Layers



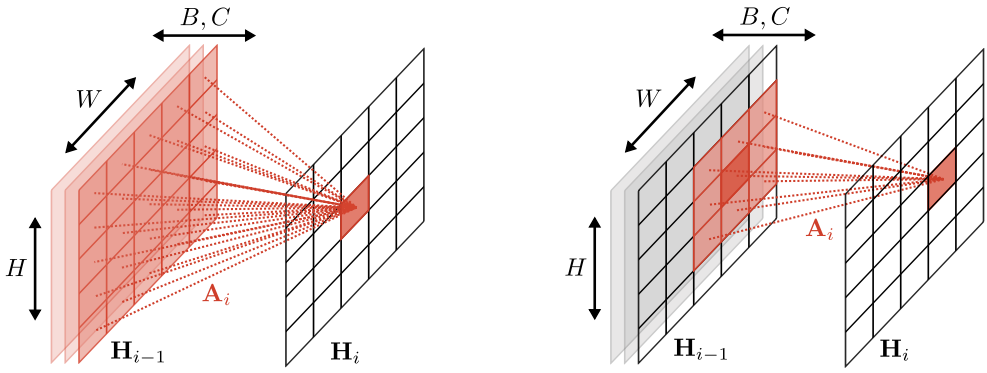
- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

Convolutional Layers



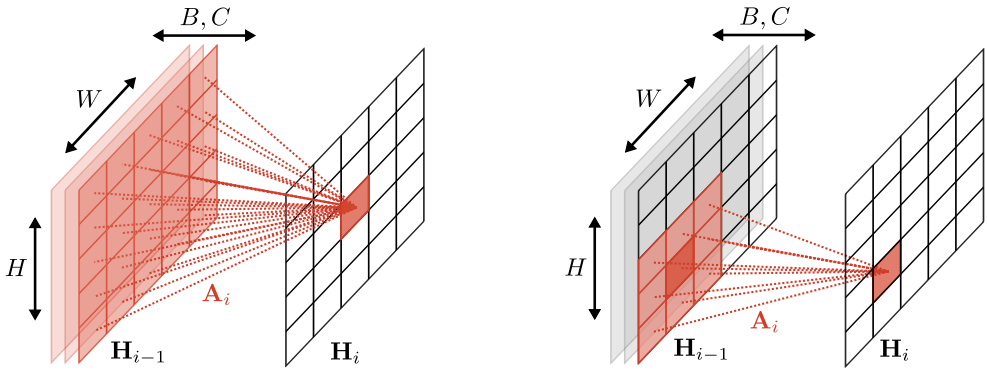
- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

Convolutional Layers



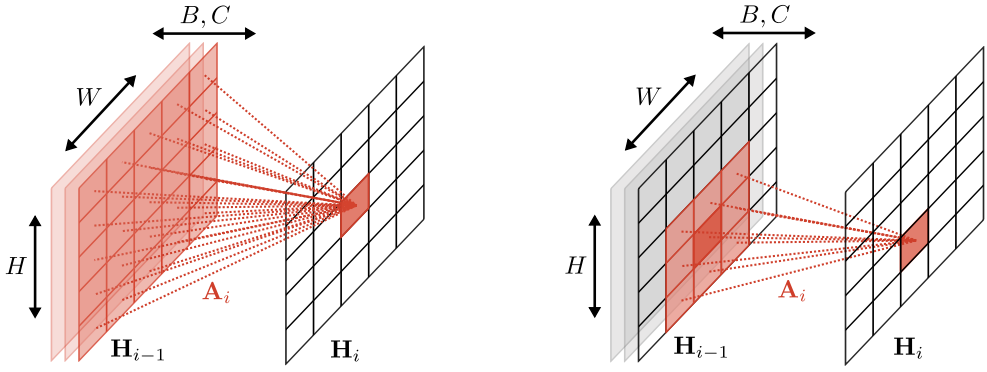
- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

Convolutional Layers



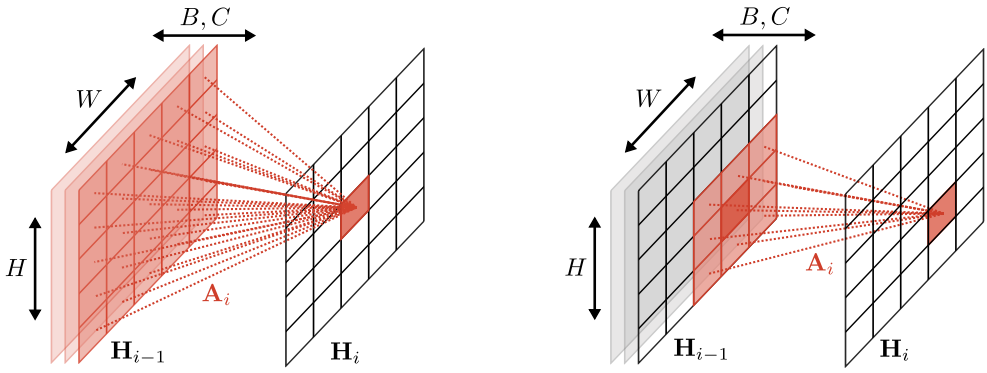
- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

Convolutional Layers



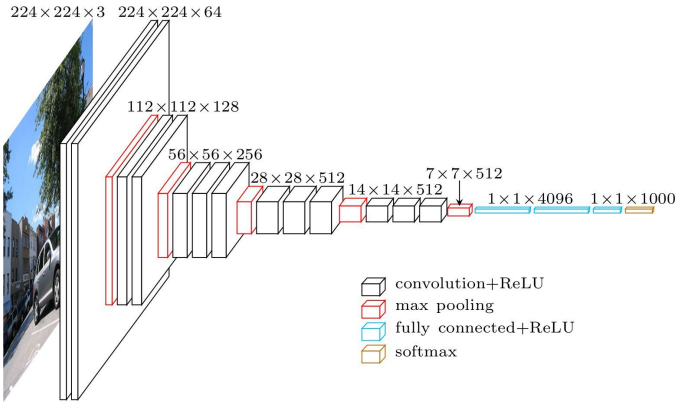
- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

Convolutional Layers



- Compared to fully conn. layers (left), convolutional layers (right) **share weights**
- Convolutional layers are **translation equivariant:** $\mathcal{T}_\theta[f](\mathbf{H}) = f(\mathcal{T}_\theta[\mathbf{H}])$

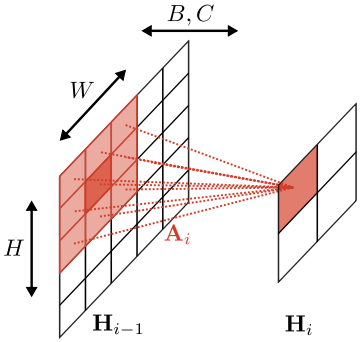
Downsampling



- Downsampling **reduces the spatial resolution** (e.g., for image level predictions)
- Downsampling **increases the receptive field** (which pixels influence a neuron)

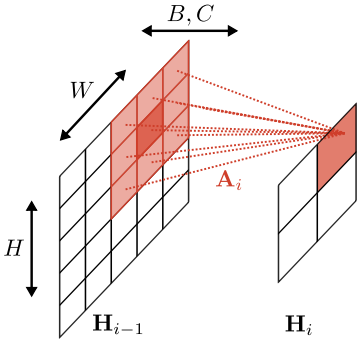
Simonyan and Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

Pooling



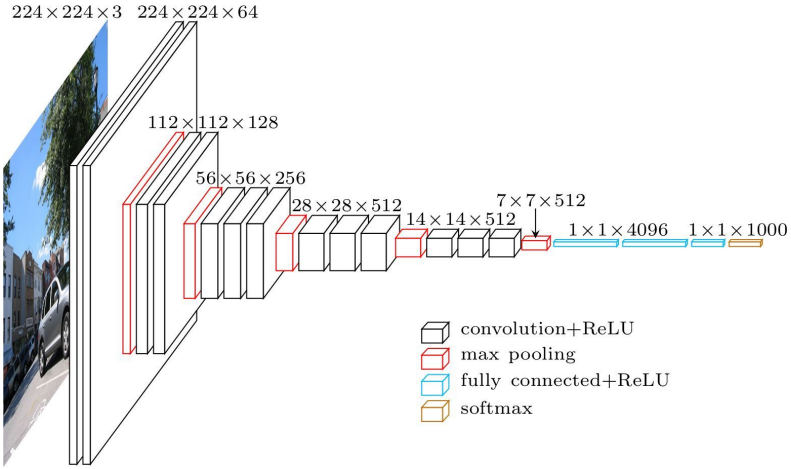
- Typically, stride $s = 2$ and kernel size $2 \times 2 \Rightarrow$ **reduces spatial dimensions** by 2
- Pooling has **no parameters** (typical pooling operations: max, min, mean)
- Pooling is **applied to each channel separately** $\Rightarrow$ keeps number of channels

Pooling



- Typically, stride $s = 2$ and kernel size $2 \times 2 \Rightarrow$ **reduces spatial dimensions** by 2
- Pooling has **no parameters** (typical pooling operations: max, min, mean)
- Pooling is **applied to each channel separately** $\Rightarrow$ keeps number of channels

Fully Connected Layers

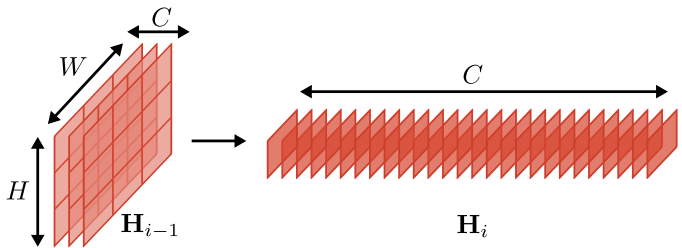


- Fully connected layers are most **memory intensive** part of VGG architecture

Simonyan and Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

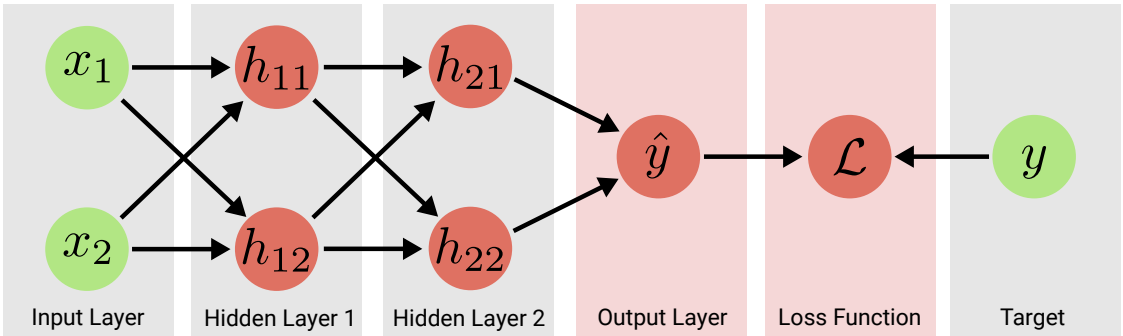
Fully Connected Layers

- Reshape $H_{i-1}[B, X, Y, C]$ into $H_i[B, C]$



- Now X and Y are **reshaped** into the feature channel dimension C

Output and Loss Functions



- The **output layer** is the last layer in a neural network which computes the output
- The **loss function** compares the result of the output layer to the target value(s)
- In image classification, we use a **softmax** output layer and a **cross entropy** loss

Categorical Distribution

Categorical distribution:

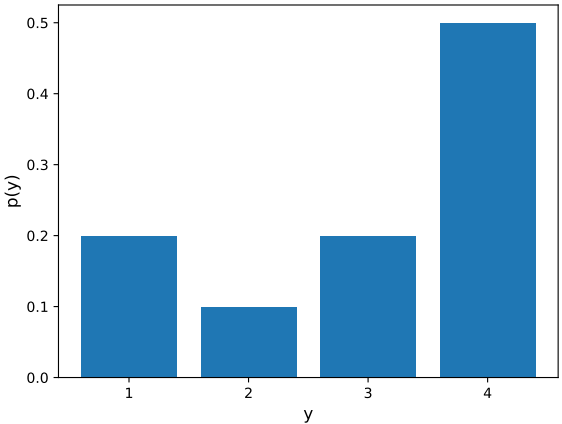
$$p(y = c) = \mu_c$$

- μ_c: probability for class c

Alternative notation:

$$p(\mathbf{y}) = \prod_{c=1}^{C} \mu_c^{y_c}$$

- $\mathbf{y}$: "one-hot" vector with $y_c \in \{0, 1\}$
- $\mathbf{y} = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ with all zeros except for one (the true class)



One-Hot Vector Representation

class	y	$\mathbf{y}$
	1	$(1,0,0,0)^\top$
	2	$(0,1,0,0)^\top$
	3	$(0,0,1,0)^\top$
	4	$(0,0,0,1)^\top$

- One-hot vector $\mathbf{y}$ with binary elements $y_c \in \{0,1\}$
- Index c where $y_c = 1$ determines the correct class
- Often used in ML as it can make formalism more convenient

Categorical Distribution / CE Loss

Let $p_{model}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \prod_{c=1}^{C} f_{\mathbf{w}}^{(c)}(\mathbf{x})^{y_c}$ be a **Categorical distribution.**
Maximizing the (log-)likelihood leads to the **cross-entropy (CE)** loss:

$$
\begin{aligned}
\hat{\mathbf{w}}_{ML} &= \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{N} \log p_{model}(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}) \\
&= \operatorname*{argmax}_{\mathbf{w}} \sum_{i=1}^{N} \log \prod_{c=1}^{C} f_{\mathbf{w}}^{(c)}(\mathbf{x}_i)^{y_{i,c}} \\
&= \operatorname*{argmin}_{\mathbf{w}} \sum_{i=1}^{N} \underbrace{\sum_{c=1}^{C} -y_{i,c} \log f_{\mathbf{w}}^{(c)}(\mathbf{x}_i)}_{\text{CE Loss}}
\end{aligned}
$$

The target $\mathbf{y} = (0, \ldots, 0, 1, 0, \ldots, 0)^\top$ is a “one-hot” vector with y_c its c'th element.

Softmax

How can we ensure that $f_{\mathbf{w}}^{(c)}(\mathbf{x})$ predicts a **valid Categorical distribution?**

- We must guarantee (1) $f_{\mathbf{w}}^{(c)}(\mathbf{x}) \in [0, 1]$ and (2) $\sum_{c=1}^{C} f_{\mathbf{w}}^{(c)}(\mathbf{x}) = 1$
- The **softmax function** guarantees both (1) and (2):

$$\text{softmax}(\mathbf{x}) = \left(\frac{\exp(x_1)}{\sum_{k=1}^{C} \exp(x_k)}, \dots, \frac{\exp(x_C)}{\sum_{k=1}^{C} \exp(x_k)} \right)$$

- Let the score vector $\mathbf{s}$ denote the network output after the last affine layer. Then:

$$f_{\mathbf{w}}^{(c)}(\mathbf{x}) = \frac{\exp(s_c)}{\sum_{k=1}^{C} \exp(s_k)} \quad \Rightarrow \quad \log f_{\mathbf{w}}^{(c)}(\mathbf{x}) = s_c - \log \sum_{k=1}^{C} \exp(s_k)$$

Log Softmax

Intuition: Assume c is the correct class. Our goal is to maximize the log softmax:

$$\log f_{\mathbf{w}}^{(c)}(\mathbf{x}) = s_c - \log \sum_{k=1}^{C} \exp(s_k)$$

- The first term encourages the score s_c for the correct class c to increase
- The second term encourages all scores in $\mathbf{s}$ to decrease (negative sign)
- The second term can be approximated by: $\log \sum_{k=1}^{C} \exp(s_k) \approx \max_k s_k$ as $\exp(s_k)$ is insignificant for all $s_k < \max_k s_k$
- Thus, the loss always strongly penalizes the most active incorrect prediction
- If the correct class already has the largest score (i.e., $s_c = \max_k s_k$), both terms roughly cancel and the example will contribute little to the overall training cost

Softmax

- Softmax responds to differences between inputs
- It is invariant to adding the same scalar to all its inputs:

$$\mathrm{softmax}(\mathbf{x}) = \mathrm{softmax}(\mathbf{x} + c)$$

- We can therefore derive a numerically more stable variant:

$$\mathrm{softmax}(\mathbf{x}) = \mathrm{softmax}(\mathbf{x} - \max_{k=1..L} x_k)$$

- Allows accurate computation even when $\mathbf{x}$ is large

Cross Entropy Loss Example

Putting it together: Cross Entropy Loss for a single training sample $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}$:

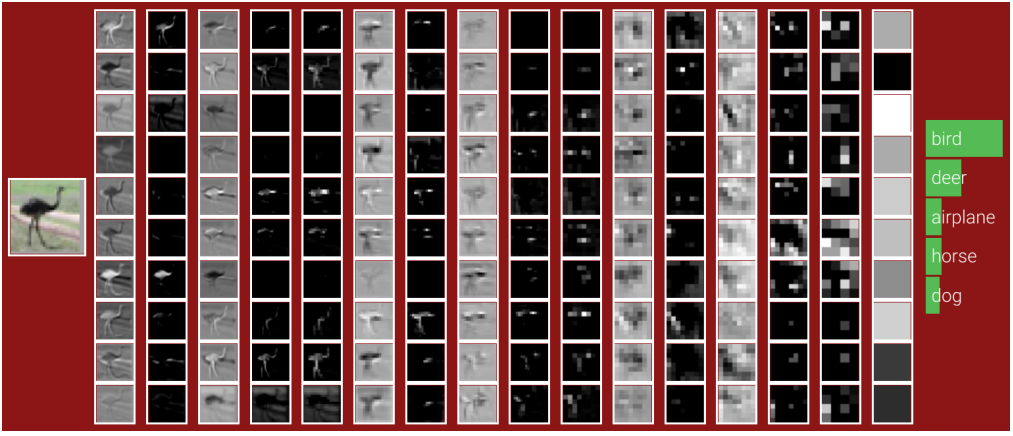
$$\text{CE Loss: } \sum_{c=1}^{C} -y_c \log f_{\mathbf{w}}^{(c)}(\mathbf{x})$$

Example: Suppose 4 classes and 4 training samples:

Input $\mathbf{x}$	Label $\mathbf{y}$	Predicted scores $\mathbf{s}$	softmax($\mathbf{s}$)	CE Loss
	$(1,0,0,0)^\top$	$(+3,+1,-1,-1)^\top$	$(0.85,0.12,0.02,0.02)^\top$	0.16
	$(0,1,0,0)^\top$	$(+3,+3,+1,+0)^\top$	$(0.46,0.46,0.06,0.02)^\top$	0.78
	$(0,0,1,0)^\top$	$(+1,+1,+1,+1)^\top$	$(0.25,0.25,0.25,0.25)^\top$	1.38
	$(0,0,0,1)^\top$	$(+3,+2,+3,-1)^\top$	$(0.42,0.16,0.42,0.01)^\top$	4.87

We observe that sample 4 contributes most strongly to the loss function!

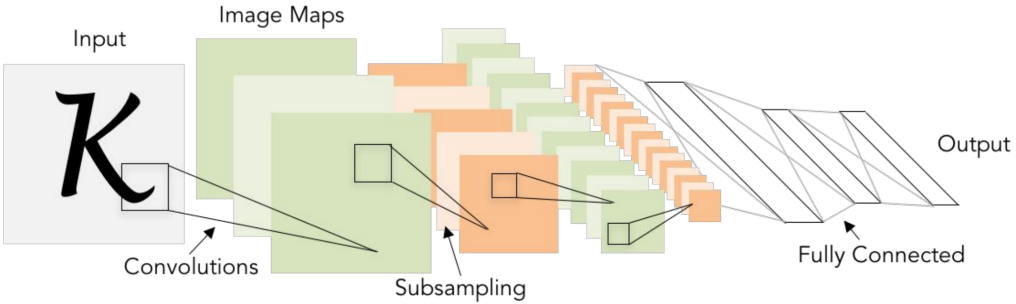
Convolutional Neural Network Example



http://cs231n.stanford.edu/2017/index.html

Network Architectures

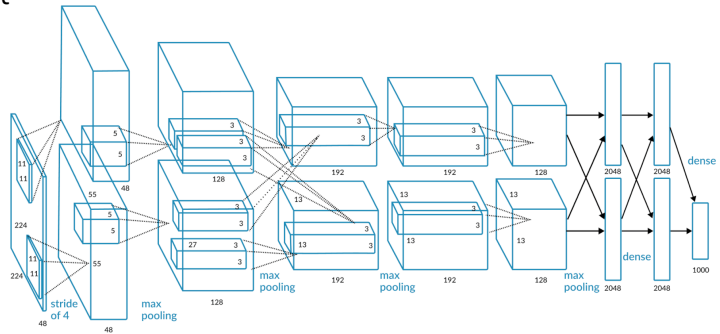
1998: LeNet-5



- 2 convolution layers (5×5), 2 pooling layers (2×2), 2 fully connected layers
- Achieved state-of-the-art accuracy on MNIST (prior to ImageNet)

LeCun et al. Gradient-based learning applied to document recognition. IEEE, 1998.

2012: AlexNet

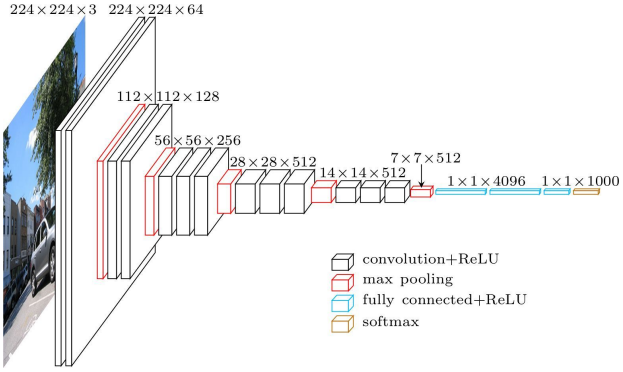


AlexNet:

- 8 layers, ReLUs, dropout, data augmentation, trained on 2 GTX 580 GPUs
- Number of feature channels increase with depth, spatial resolution decreases
- Triggered deep learning revolution, showed that CNNs work well in practice

Krizhevsky, Sutskever and Hinton: ImageNet Classification with Deep Convolutional Neural Networks. NIPS, 2012.

2015: VGG

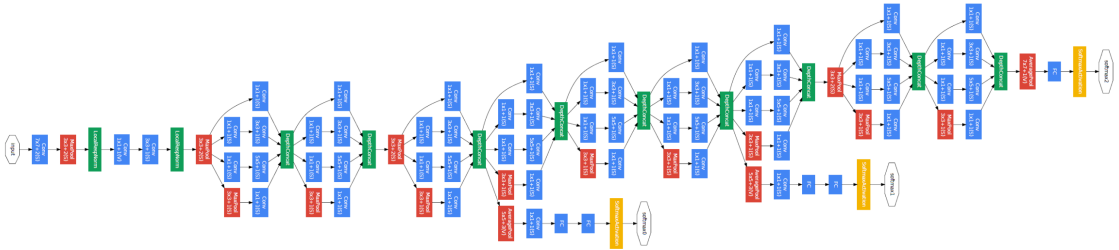


VGG:

- Uses 3×3 convolutions everywhere (same expressiveness, fewer parameters)
- Three 3×3 layers: same receptive field as one 7×7 layer, but less parameters
- 2 Variants: 16 and 19 Layers. Showed that deeper networks are better.

Simonyan and Zisserman: Very Deep Convolutional Networks for Large-Scale Image Recognition. ICLR, 2015.

2015: Inception / GoogLeNet

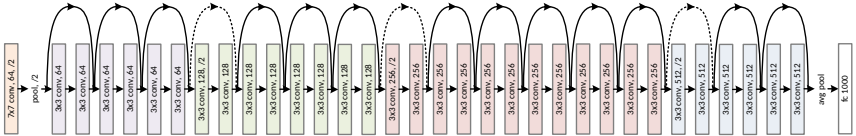


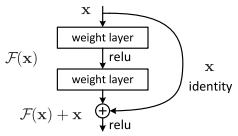
Inception:

- 22 layers. Inception modules utilize conv/pool operations with varying filter size
- Multiple intermediate classification heads to improve gradient flow
- Global avarage pooling (no FC layers), 27x less parameters (5 million) than VGG-16

Szegedy et al.: Going deeper with convolutions. CVPR, 2015.

2016: ResNet



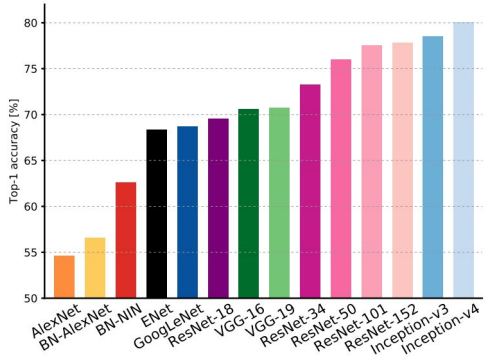


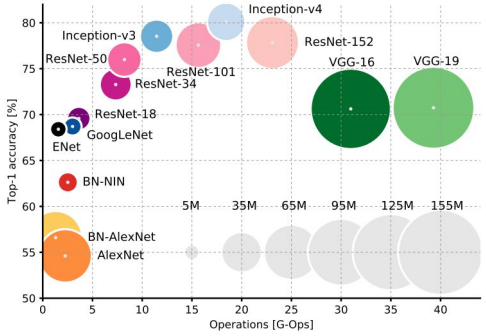
ResNet:

- Residual connections allow for training deeper networks (up to 152 layers)
- Very simple and regular network structure with 3×3 convolutions
- Uses strided convolutions for downsampling
- ResNet and ResNet-like architectures are dominating today

He, Zhang, Ren and Sun: Deep Residual Learning for Image Recognition. CVPR, 2016.

Accuracy vs. Complexity

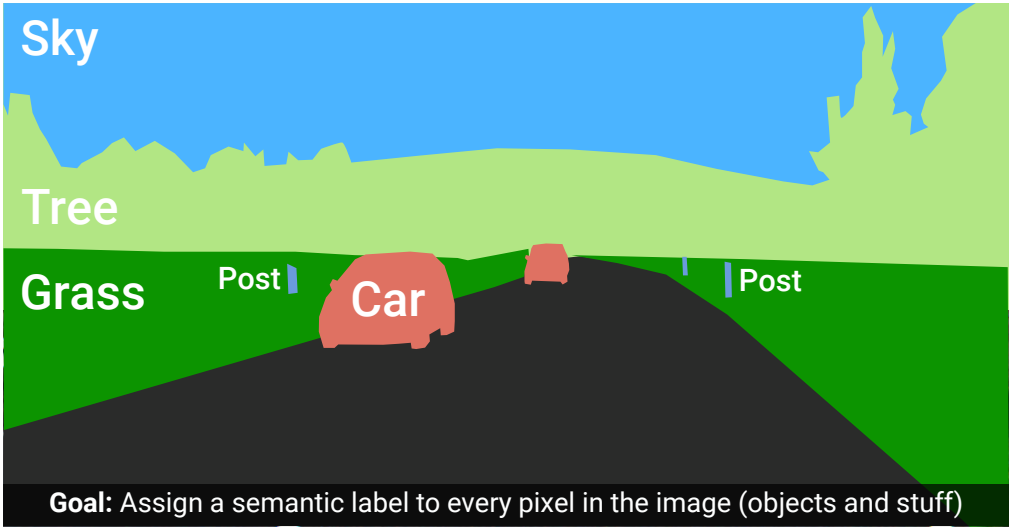




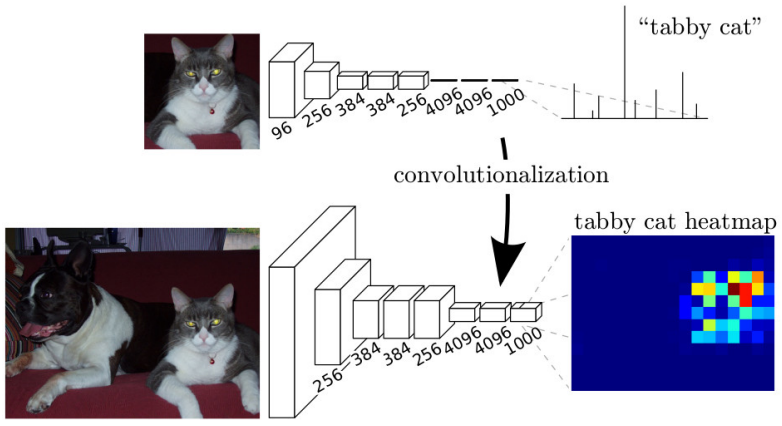
- Performance: Top-1 or Top-5 accuracy (target label in top N predictions)
- VGG has most parameters and is slowest
- ResNet/Inception/GoogLeNet are faster and have fewer parameters

10.2
Semantic Segmentation

Semantic Segmentation



Fully Convolutional Networks



- Apply classification network in convolutional fashion to obtain class heatmaps
- Use cross entropy loss at every pixel of the output (sum over classification losses)

Long, Shelhamer and Darrell: Fully convolutional networks for semantic segmentation. CVPR, 2015.

Fully Convolutional Networks

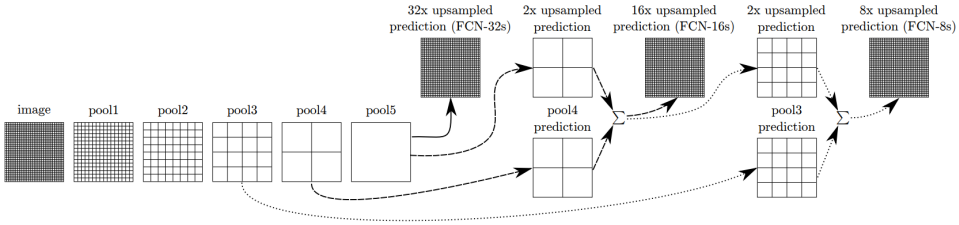
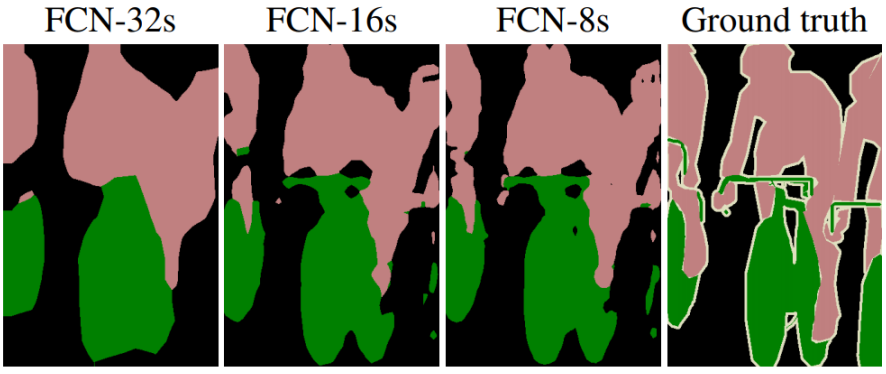


Figure 3. Our DAG nets learn to combine coarse, high layer information with fine, low layer information. Layers are shown as grids that reveal relative spatial coarseness. Only pooling and prediction layers are shown; intermediate convolution layers (including our converted fully connected layers) are omitted. Solid line (FCN-32s): Our single-stream net, described in Section 4.1, upsamples stride 32 predictions back to pixels in a single step. Dashed line (FCN-16s): Combining predictions from both the final layer and the `pool4` layer, at stride 16, lets our net predict finer details, while retaining high-level semantic information. Dotted line (FCN-8s): Additional predictions from `pool3`, at stride 8, provide further precision.

- Problem: Output heatmaps are **low resolution** due to downsampling operations
- But using only convolutional filters does not work due to small receptive field
- Idea: **Learn upsampling** operation and combine **low- and high-level** information

Long, Shelhamer and Darrell: Fully convolutional networks for semantic segmentation. CVPR, 2015.

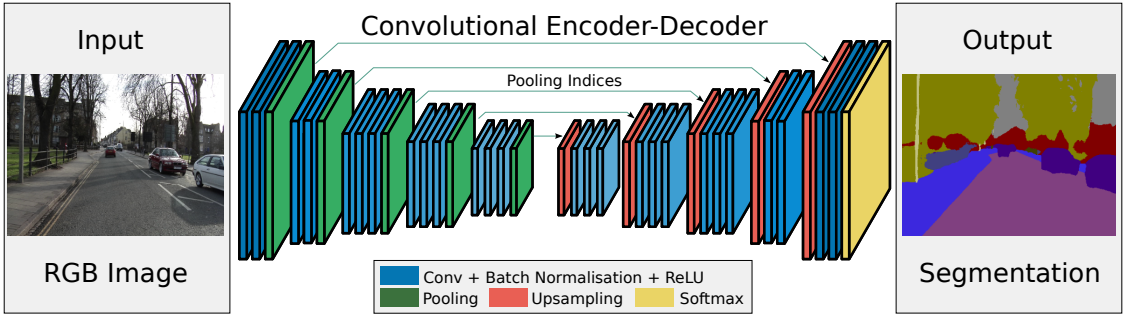
Fully Convolutional Networks



- Fusing information from layers with different strides (FCN-8s) improves detail
- However, output of early 2015 results still relatively coarse

Long, Shelhamer and Darrell: Fully convolutional networks for semantic segmentation. CVPR, 2015.

SegNet

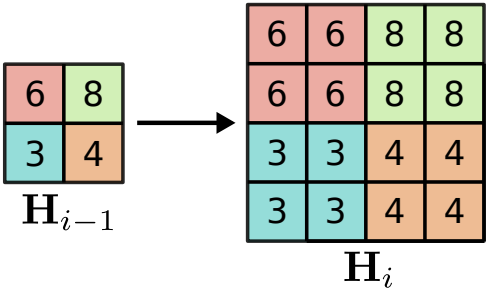


- SegNet is another model that uses the idea of down- and upsampling
- Downsampling provides strong features with large receptive field
- Upsampling yields output at the same resolution as input

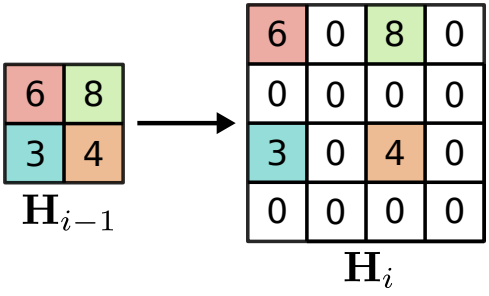
Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. TPAMI, 2017.

Upsampling

Nearest Neighbor Upsampling

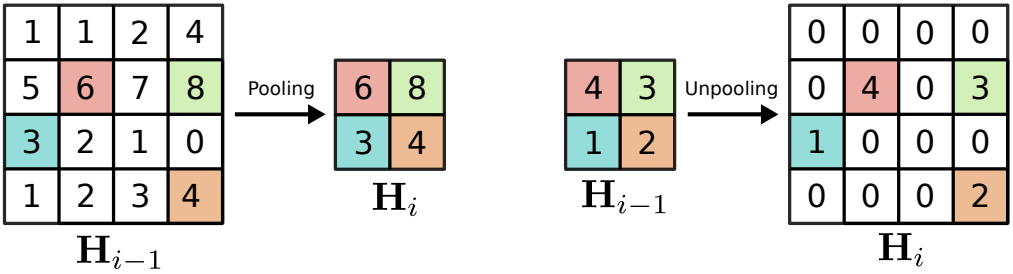


Bed of Nails Upsampling



- **Nearest neighbor:** scale each channel using nearest neighbor interpolation
- **Bilinear:** scale each channel using bilinear neighbor interpolation
- **Bed of nails:** insert elements at sparse location (followed by convolution)

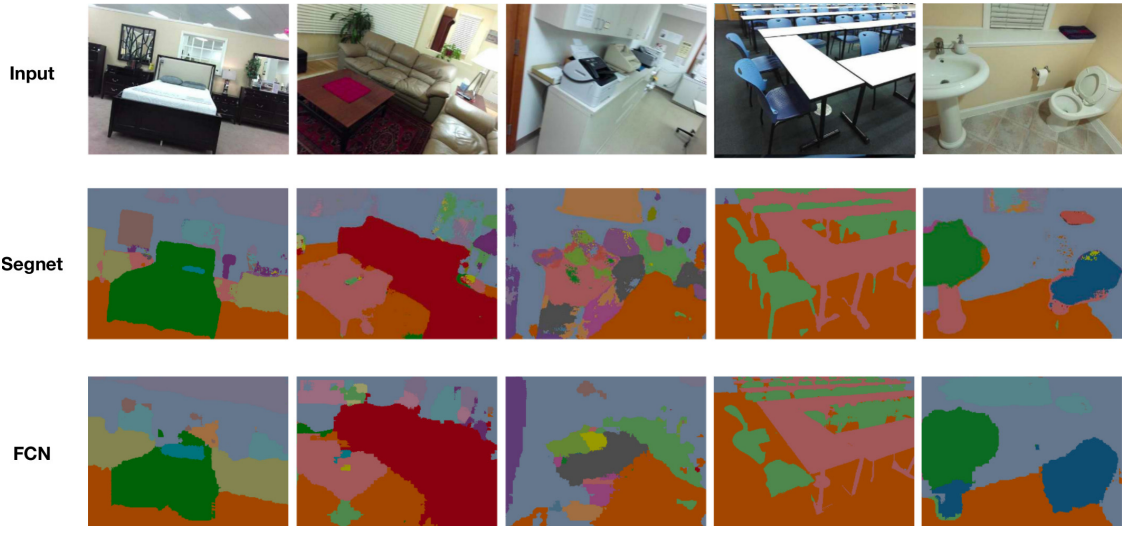
Max Unpooling



- For unpooling, **remember** which element was **maximum** during pooling
- Requires corresponding pairs of downsampling and upsampling layers
- This approach has been used in SegNet

Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. TPAMI, 2017.

SegNet Results



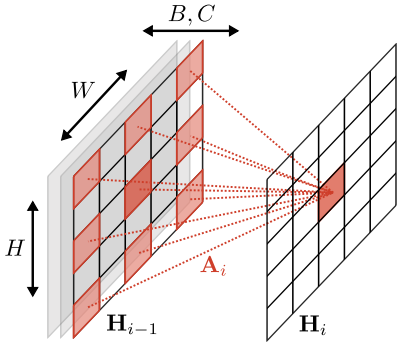
Badrinarayanan, Kendall and Cipolla: SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. TPAMI, 2017.

Dilated Convolution

- Dilated convolutions are an alternative to combining downsampling and upsampling opterations in order to reach a large receptive field size quickly
- Dilated convolutions increase the receptive field of standard convolutions without increasing the number of parameters
- Thus, a network with dilated convolutions is able to perform image-level predictions (e.g., semantic segmentation) without upsampling and downsampling
- However, in practice, dilated convolutions combined with standard backbone

Yu and Koltun: Multi-Scale Context Aggregation by Dilated Convolutions. ICLR, 2016.

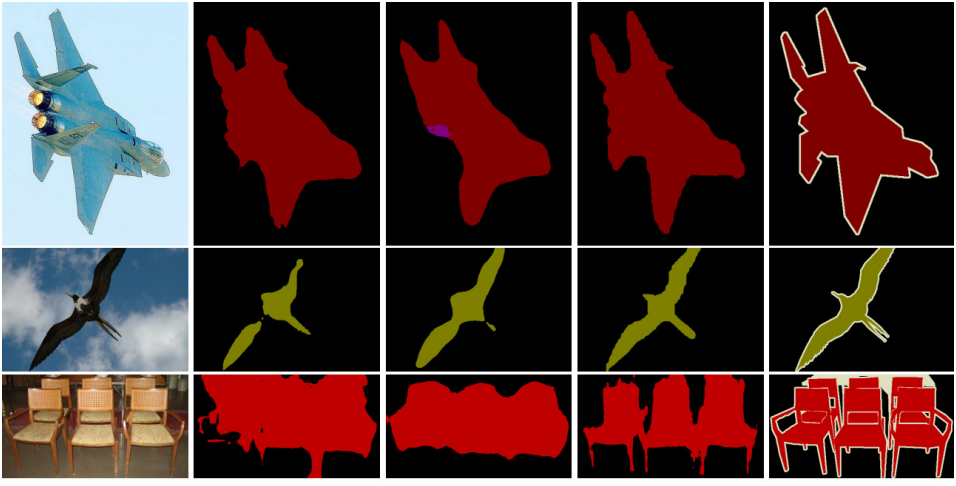
Dilated Convolution



- Dilation factor d (here $d = 2$) leads to more “distributed” sampling of the input
- **Increases receptive field** without increasing parameters or reducing spatial dim.

Yu and Koltun: Multi-Scale Context Aggregation by Dilated Convolutions. ICLR, 2016.

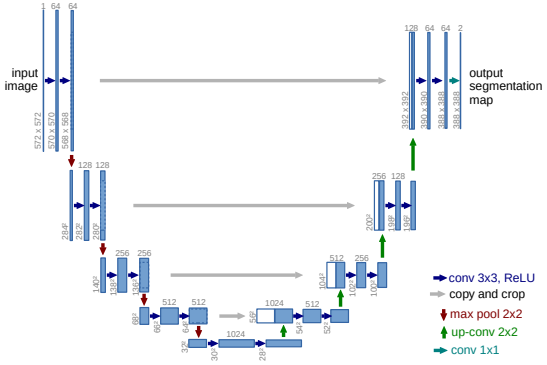
Dilated Convolution



(a) Image (b) FCN-8s (c) DeepLab (d) Our front end (e) Ground truth

Yu and Koltun: Multi-Scale Context Aggregation by Dilated Convolutions. ICLR, 2016.

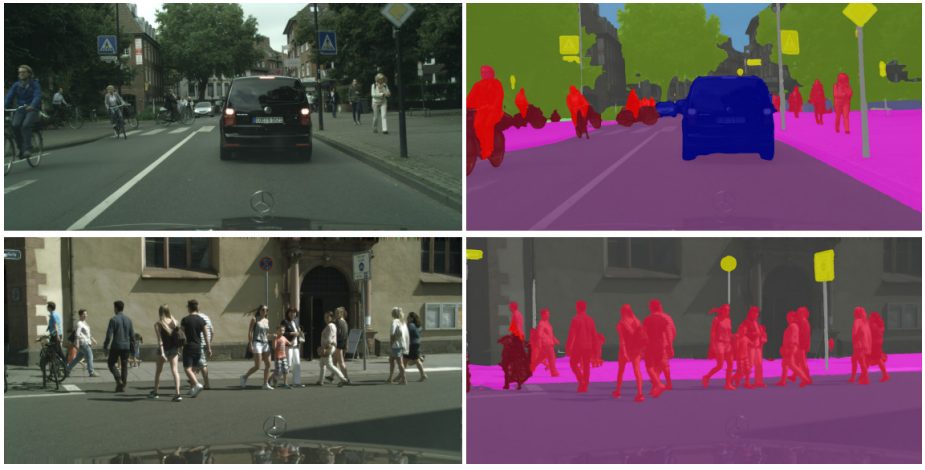
U-Networks



- U-Nets use max-pooling, up-convolutions and skip-connections (concatenation)
- Defacto standard for many tasks with image output (e.g., depth, segmentation)

Ronneberger, Fischer and Brox: U-Net: Convolutional Networks for Biomedical Image Segmentation. MICCAI, 2015.

Feature Space Optimization



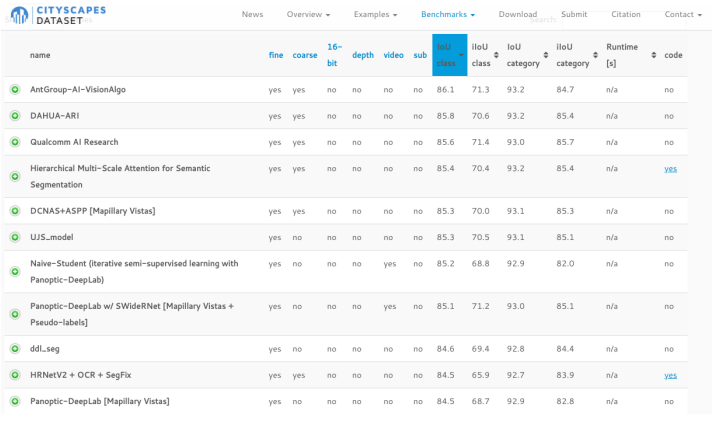
- Temporally consistent semantic segmentation in video sequences

Kundu, Vineet and Koltun: Feature Space Optimization for Semantic Video Segmentation. CVPR, 2016.

Cityscapes Leaderboard

Evaluation Metrics:

- Per-pixel accuracy
- Per-class Jaccard Index / Intersection-over-Union
- IoU weighted by inv. instance size



CITYSCAPES DATASET

News Overview Examples Benchmarks Download Submit Citation Contact

name	fine	coarse	16-bit	depth	video	sub	IoU class	iIoU class	IoU category	iIoU category	Runtime [s]	code
AntGroup-AI-VisionAlgo	yes	yes	no	no	no	no	86.1	71.3	93.2	84.7	n/a	no
DAHUA-ARI	yes	yes	no	no	no	no	85.8	70.6	93.2	85.4	n/a	no
Qualcomm AI Research	yes	yes	no	no	no	no	85.6	71.4	93.0	85.7	n/a	no
Hierarchical Multi-Scale Attention for Semantic Segmentation	yes	yes	no	no	no	no	85.4	70.4	93.2	85.4	n/a	yes
DCNAS+ASPP [Mapillary Vistas]	yes	yes	no	no	no	no	85.3	70.0	93.1	85.3	n/a	no
UJS_model	yes	no	no	no	no	no	85.3	70.5	93.1	85.1	n/a	no
Naive-Student (iterative semi-supervised learning with Panoptic-DeepLab)	yes	no	no	no	yes	no	85.2	68.8	92.9	82.0	n/a	no
Panoptic-DeepLab w/ SWideRNet [Mapillary Vistas + Pseudo-labels]	yes	no	no	no	yes	no	85.1	71.2	93.0	85.1	n/a	no
ddl_seg	yes	no	no	no	no	no	84.6	69.4	92.8	84.4	n/a	no
HRNetV2 + OCR + SegFix	yes	yes	no	no	no	no	84.5	65.9	92.7	83.9	n/a	yes
Panoptic-DeepLab [Mapillary Vistas]	yes	no	no	no	no	no	84.5	68.7	92.9	82.8	n/a	no

www.cityscapes-dataset.com

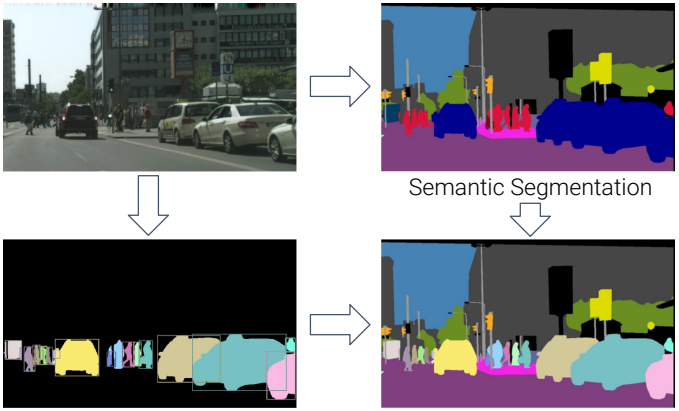
Cordts et al.: The Cityscapes Dataset for Semantic Urban Scene Understanding. CVPR, 2016.

Hierarchical Multi-Scale Attention

- State-of-the-art multi-scale approach, currently ranked fourth on Cityscapes

Tao, Sapra and Catanzaro: Hierarchical Multi-Scale Attention for Semantic Segmentation. Arxiv, 2020.

Panoptic Segmentation

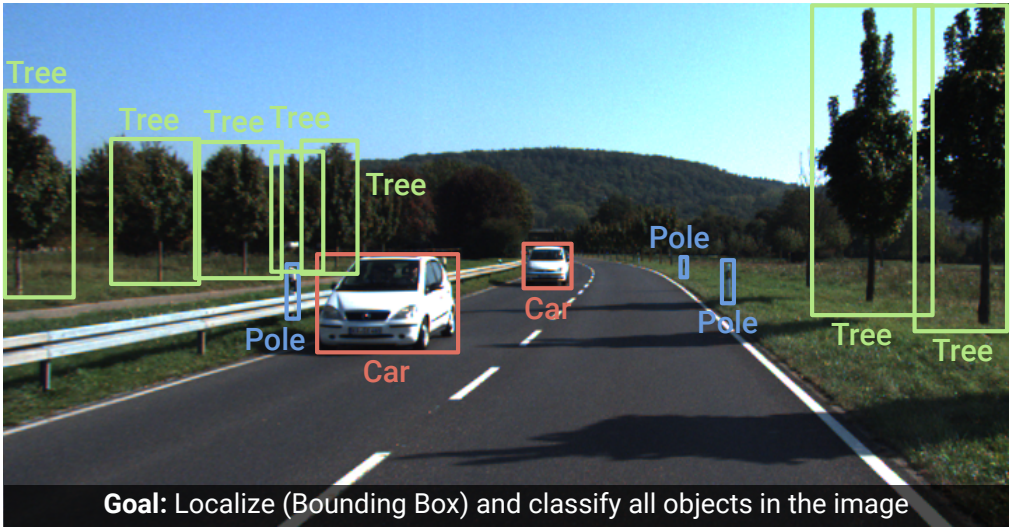


- Combines semantic and instance segmentation: predict semantic category for both stuff and objects, as well as instance label for each object

Kirillov, He, Girshick, Rother and Dollár: Panoptic Segmentation. CVPR, 2019.

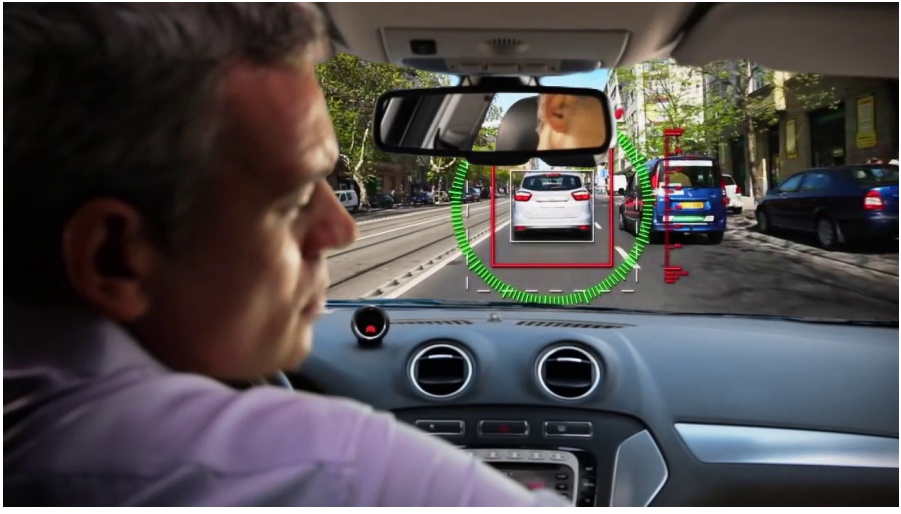
10.3
Object Detection and Segmentation

Object Detection



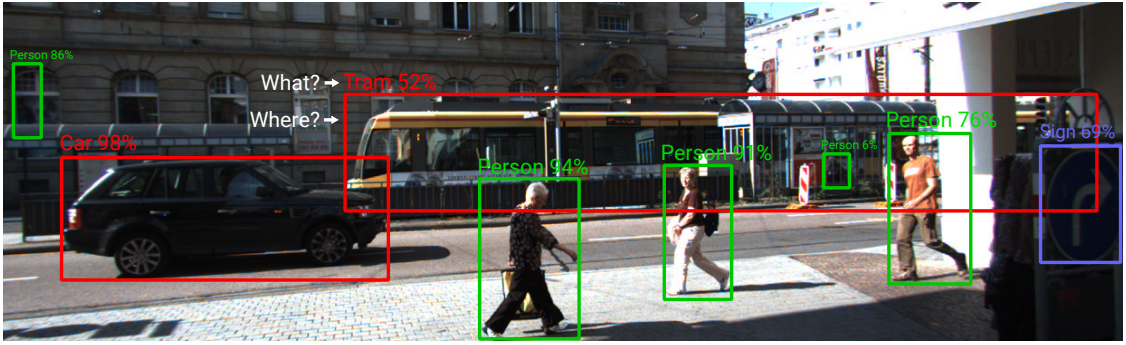
Goal: Localize (Bounding Box) and classify all objects in the image

Motivation



Source: Mobileye.

Problem Setting



Problem Setting:

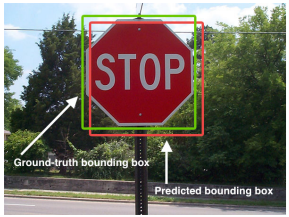
- **Input:** RGB Image or laser range scan.
- **Output:** Set of 2D/3D bounding boxes with category label and confidence
- Note that the number of objects and the object size not known a priori

Problem Setting

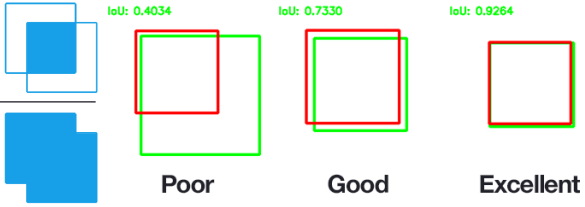
Problem Setting:

- **Input:** RGB Image or laser range scan.
- **Output:** Set of 2D/3D bounding boxes with category label and confidence
- Note that the number of objects and the object size not known a priori

Performance Evaluation



$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



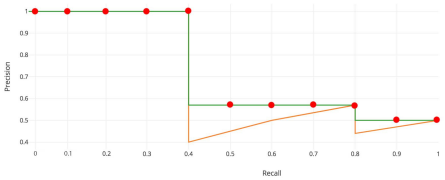
Performance Evaluation:

- IoU = Intersection-over-Union (predicted bbox vs. true bbox)
- Detection considered successful if IoU ≥ 0.5
- How to fairly measure detection performance in case of multiple objects? (Number of detections depends on detector threshold and is detector specific!)

Performance Evaluation

Average Precision Metric:

1. Run detector with varying thresholds
2. Assign detections to ground truth objects (max. 1 per object, IoU$\geq$0.5)
3. Count TP, FP, FN
4. Compute **Average Precision (AP)**



True Positives TP: Number of objects correctly detected (IoU$\geq$0.5)

False Negatives FN: Number of objects not detected (IoU$<$0.5)

False Positives FP: Wrong detections

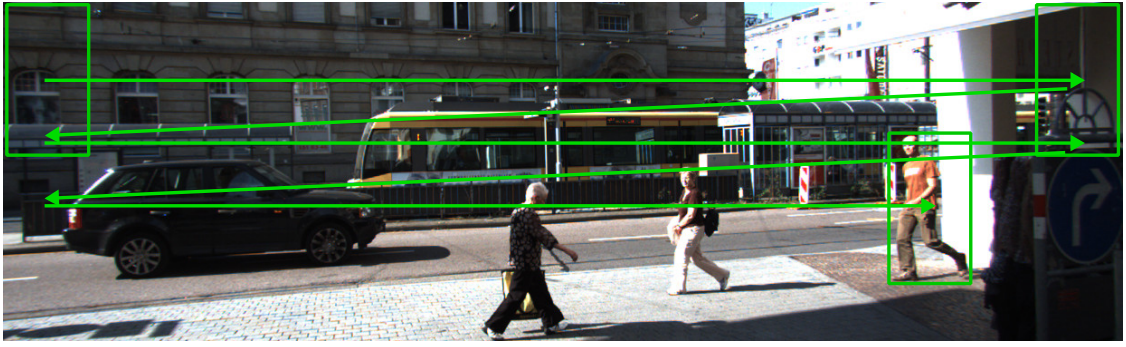
$$\text{Precision } P = \frac{TP}{TP + FP}$$

$$\text{Recall } R = \frac{TP}{TP + FN}$$

$$\text{Avg. Prec. } AP = \frac{1}{11} \sum_{R \in \{0,\ldots,1\}} \max_{R' \geq R} P(R')$$

Sliding-Window Object Detection

Sliding Window Object Detection



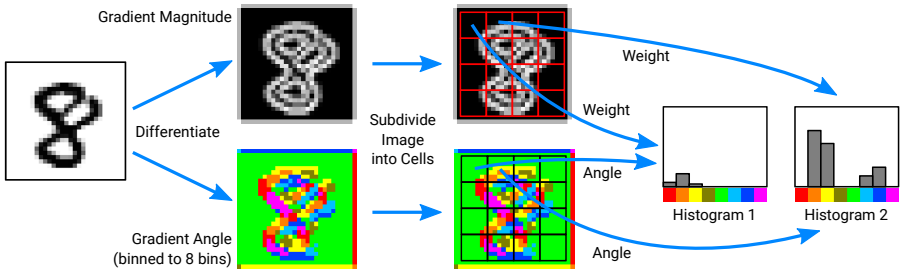
Sliding Window Detection:

- Run **sliding window** of fixed size over the image; extract features for each window
- **Classify each crop** (object vs. background) using SVM, random forest, boosting, ..
- Iterate over multiple aspect ratios/scales and perform **non-maxima suppression**

Histograms of Oriented Gradients

Which feature space to use?

- Simplest: RGB pixel space $\Rightarrow$ neither viewpoint nor illumination invariant
- Better: Histograms of Oriented Gradients (HoG) (defacto standard for $>$10 years)
  - Idea: represent patches with histograms (gradient angles weighted by magnitude)



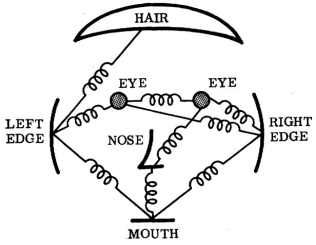
- HoG is invariant to small deformations (translation, scale, rotation, perspective)

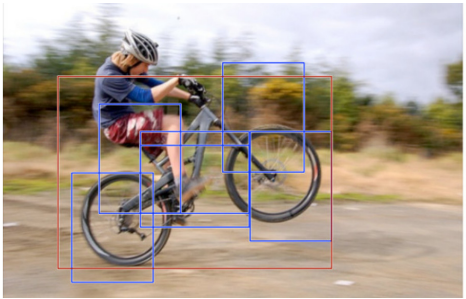
Dalal and Triggs: Histograms of Oriented Gradients for Human Detection. CVPR, 2005.

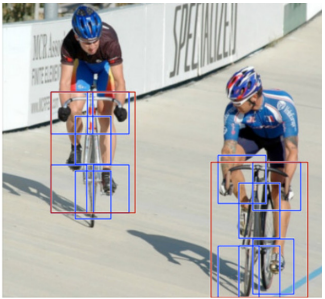
Part Based Models

Part-based Models:

- Idea: Model object based on its parts, **model distribution of part configurations**
- Allows for even more invariance (non-rigid deformations etc.)
- However: **slower inference** and not much gain wrt. multi-view HoG models







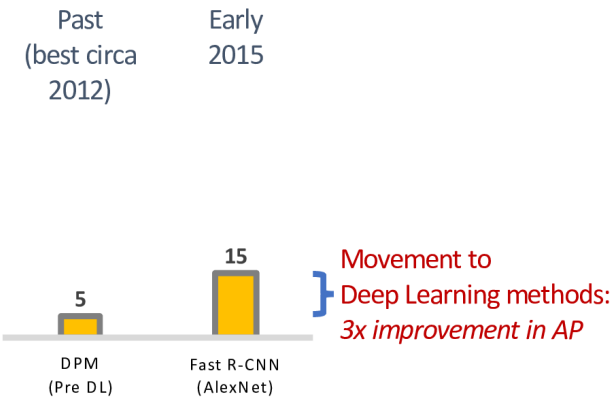
Object Detection with Discriminatively Trained Part Based Models. Felzenszwalb, Girshick, McAllester and Ramanan. PAMI, 2010.

Recognition - A Success Story?

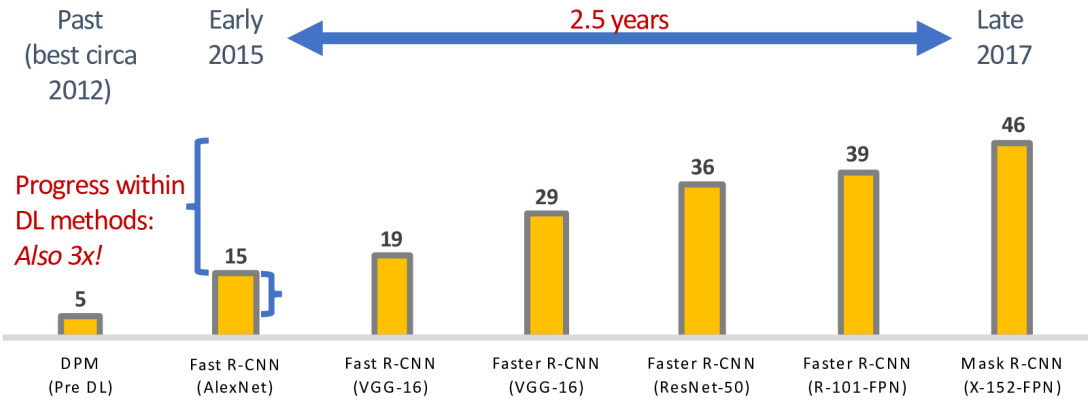
Classical approaches **haven't lead to major breakthroughs:** only for specific non-safety-critical tasks (e.g., face detection), they have seen great success. **Why?**

- Computer vision features are very difficult to hand-engineer
- It is unclear how a good representation should look like
- Furthermore, sliding window approaches are slow in practice
- Inference in complicated part-based models is even slower
- Computation is often not efficiently re-used
- But: deep learning has changed this ..

Hand-crafted Representations vs. Learned Representations



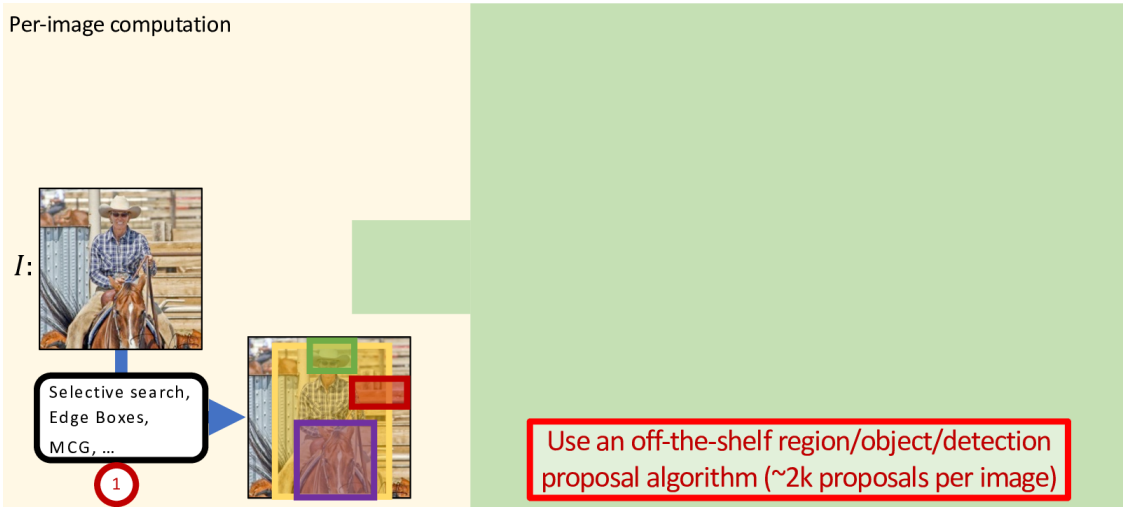
Hand-crafted Representations vs. Learned Representations



Object Detection with Deep Neural Networks

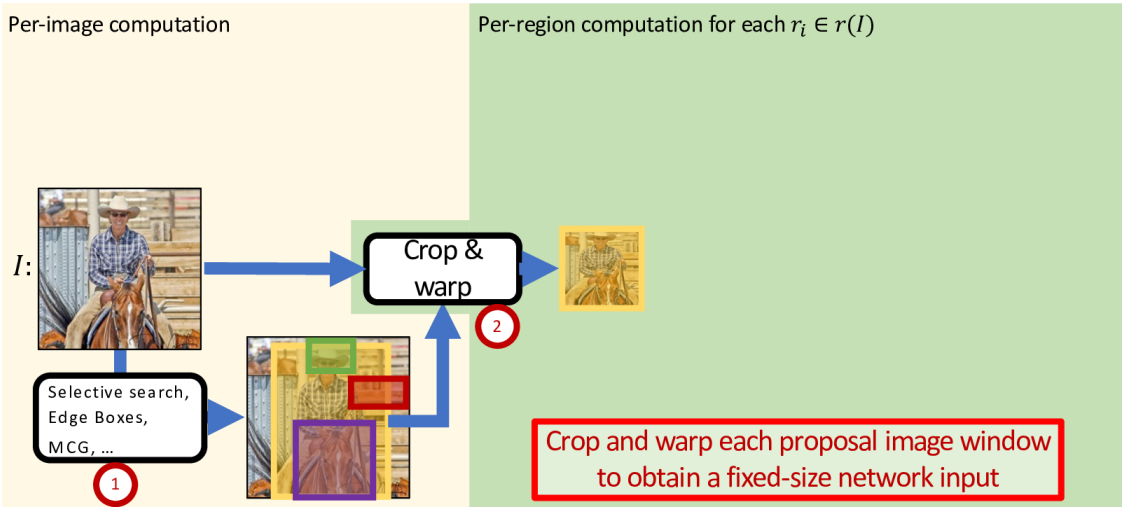
Slide Credits: Ross Girshick

R-CNN: Region-based Convolutional Neural Network

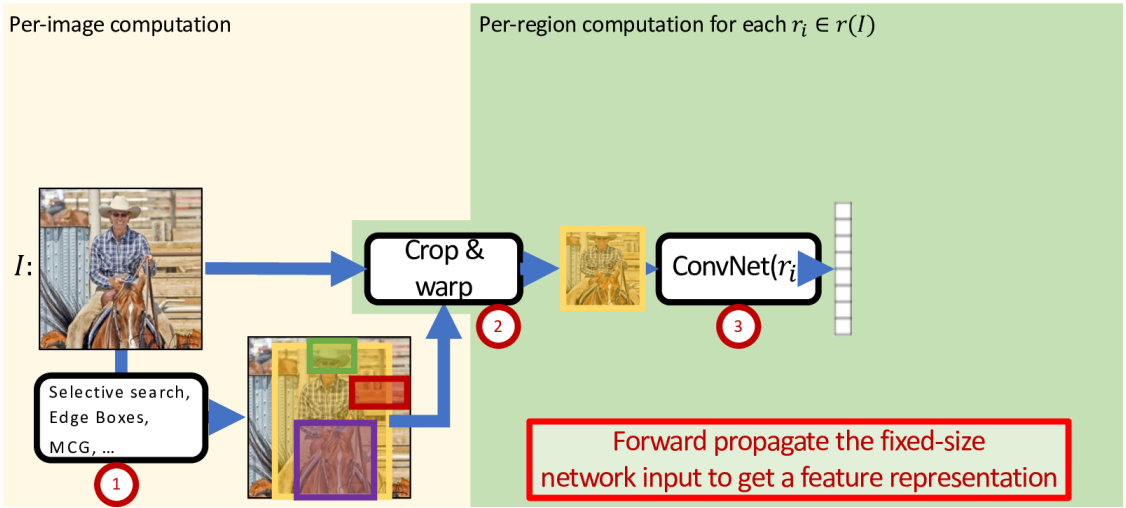


Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Region-based Convolutional Neural Network

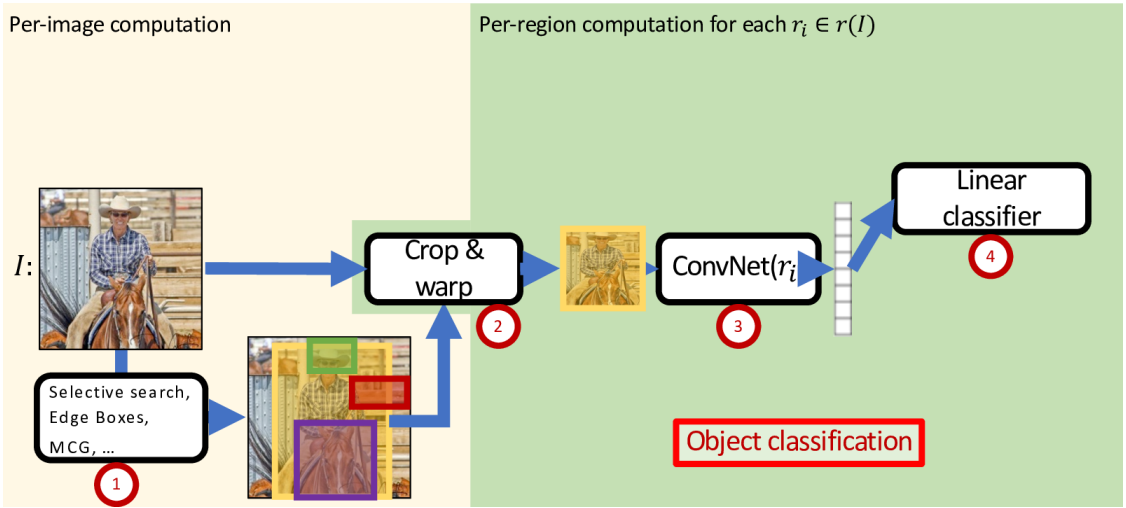


R-CNN: Region-based Convolutional Neural Network



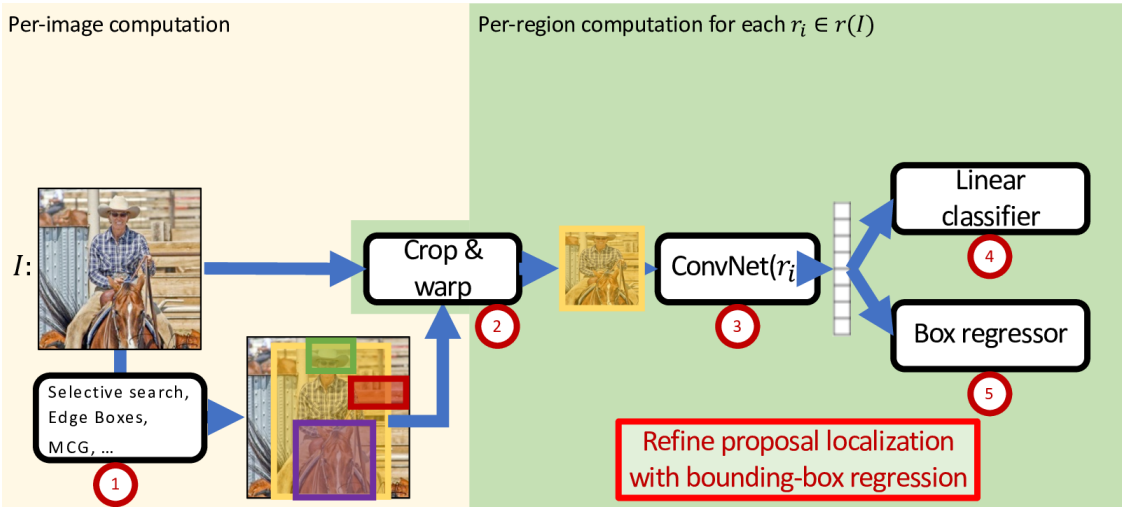
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Region-based Convolutional Neural Network



Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN: Region-based Convolutional Neural Network



Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework

Per-image computation

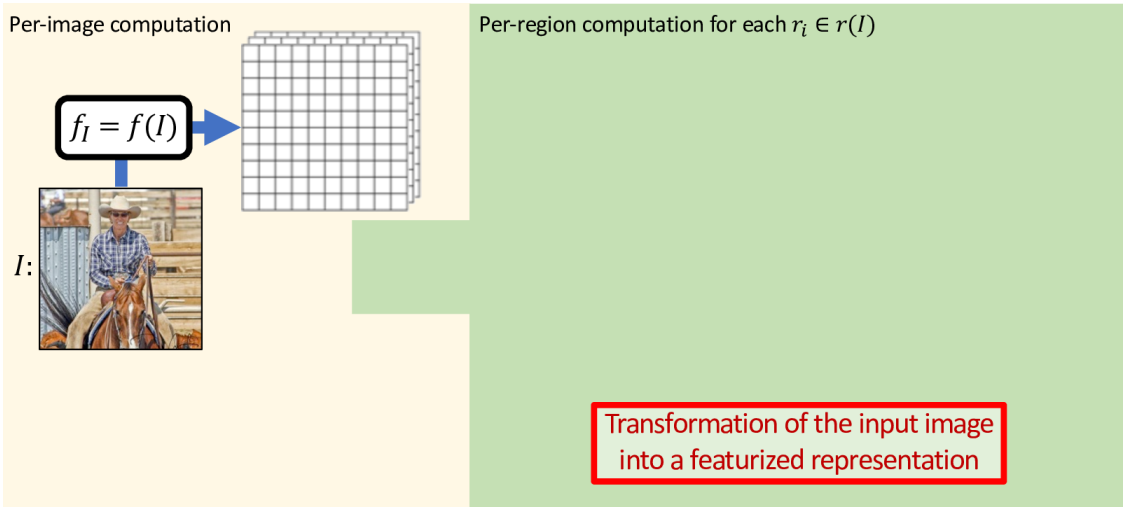
Per-region computation for each $r_i \in r(I)$

I:

Input image
per-image operations | per-region operations

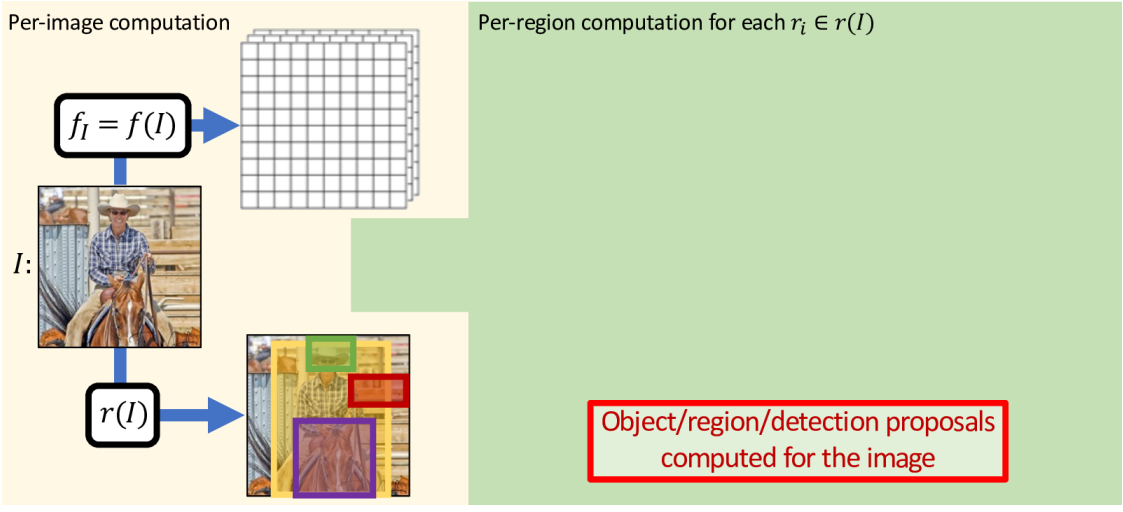
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework



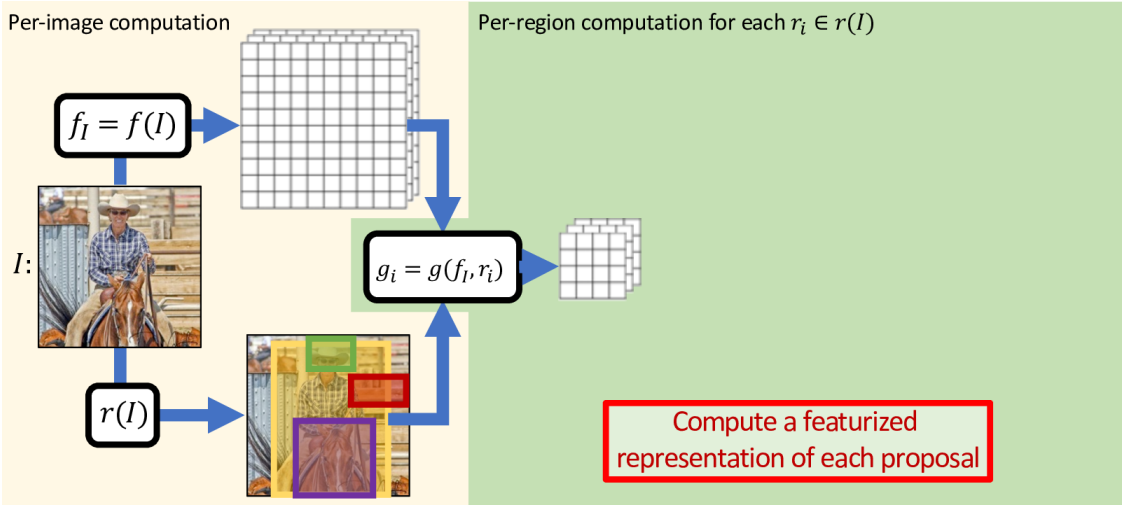
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework



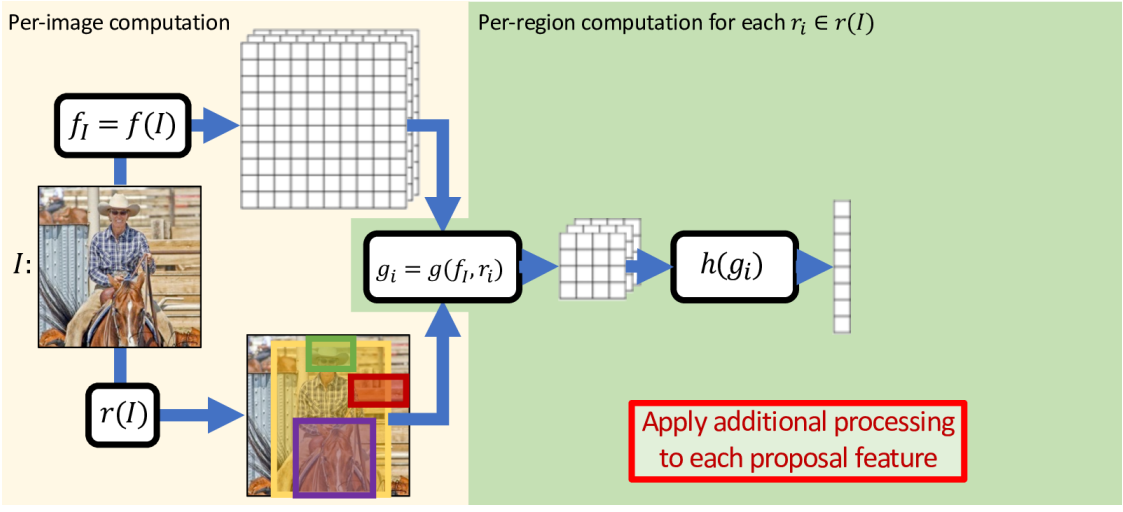
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework



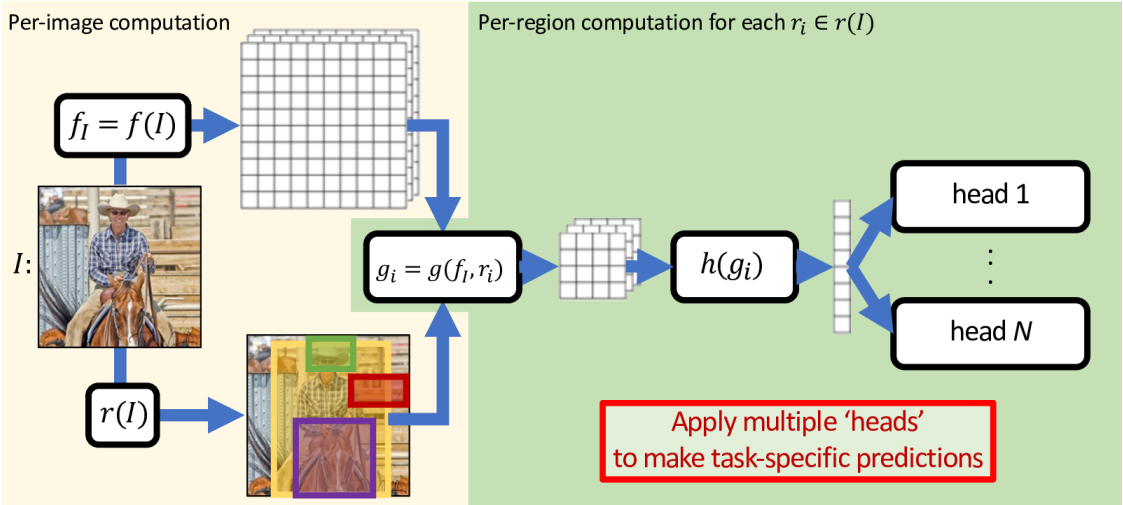
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework



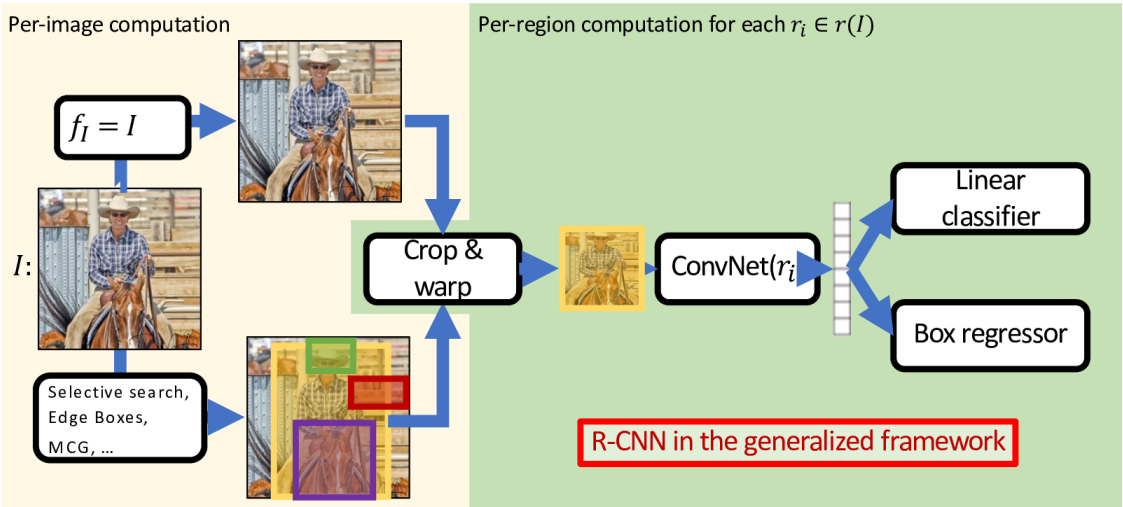
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

Generalized Framework



Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN in Generalized Framework



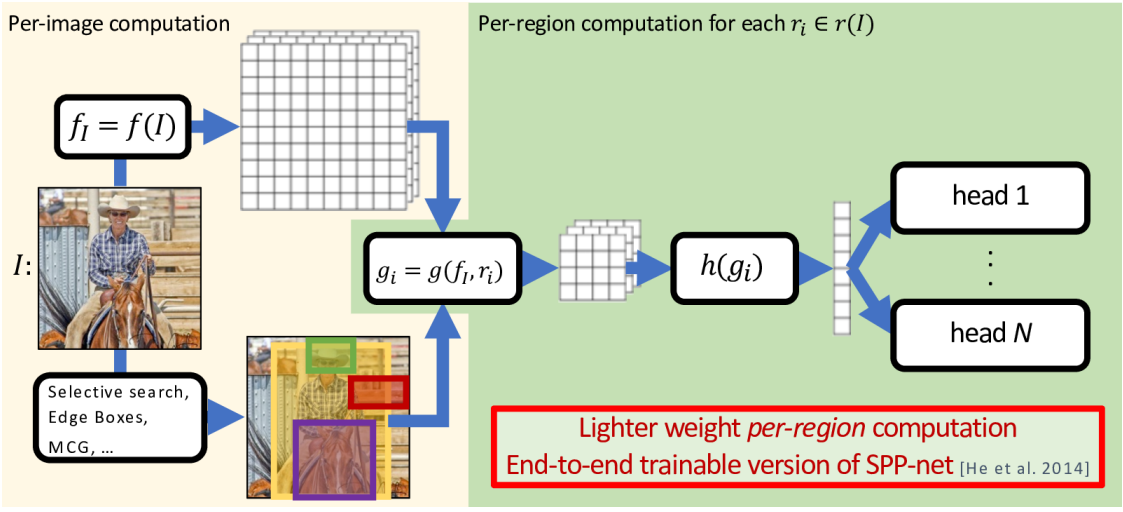
Girshick, Donahue, Darrell and Malik: Rich feature hierarchies for accurate object detection and semantic segmentation. CVPR, 2014.

R-CNN in Generalized Framework

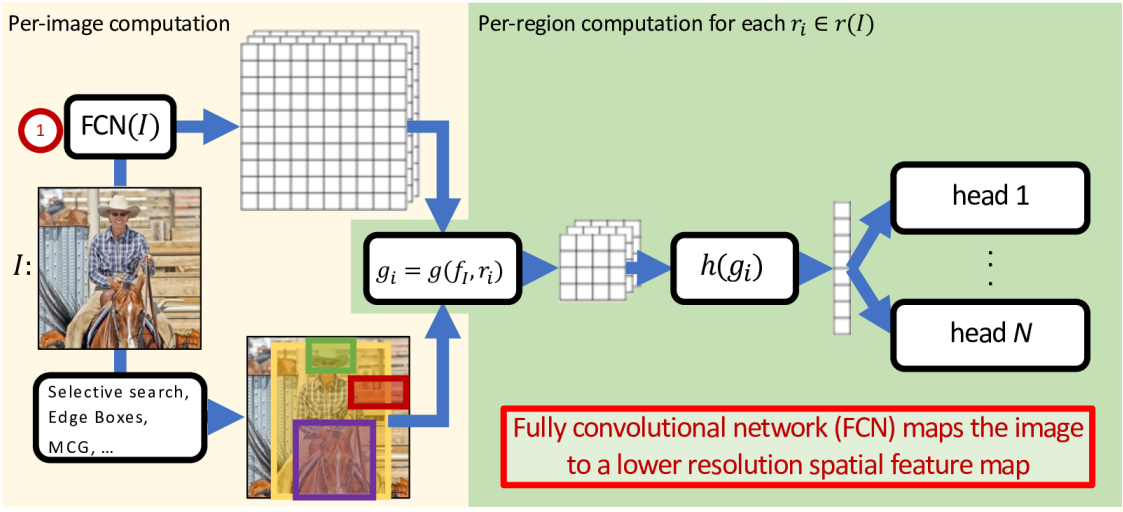
What is the problem with R-CNN?

- Heavy per-region computation (e.g., 2000 full network evaluations)
- No computation/feature sharing
- Slow region proposal method adds to runtime
- Generic region proposal techniques have limited recall

Fast R-CNN

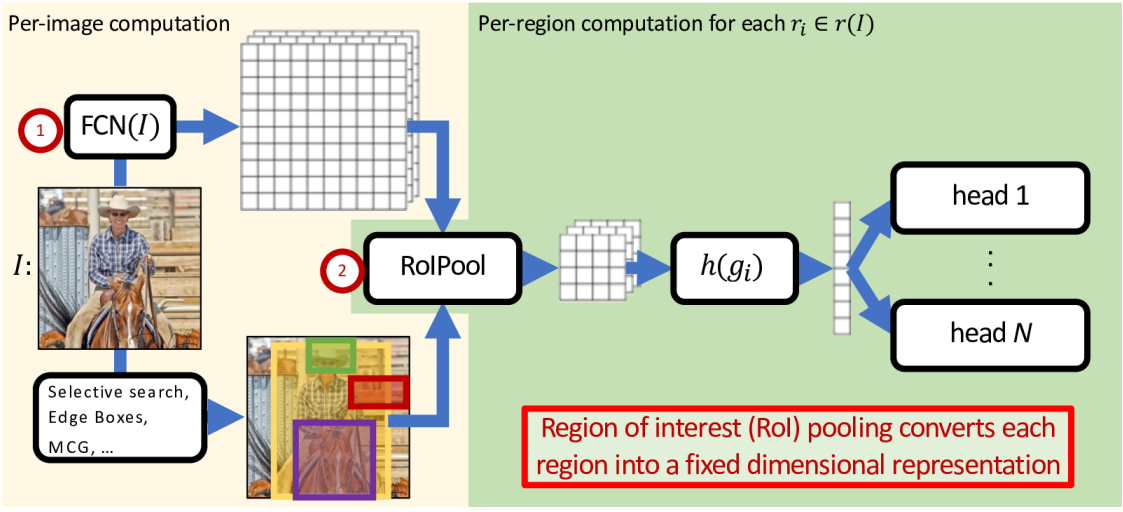

Per-image computation
Per-region computation for each $r_i \in r(I)$
$f_I = f(I)$
I:
Selective search,
Edge Boxes,
MCG, …
$g_i = g(f_I, r_i)$
$h(g_i)$
head 1
head N
Lighter weight *per-region* computation
End-to-end trainable version of SPP-net [He et al. 2014]

Fast R-CNN



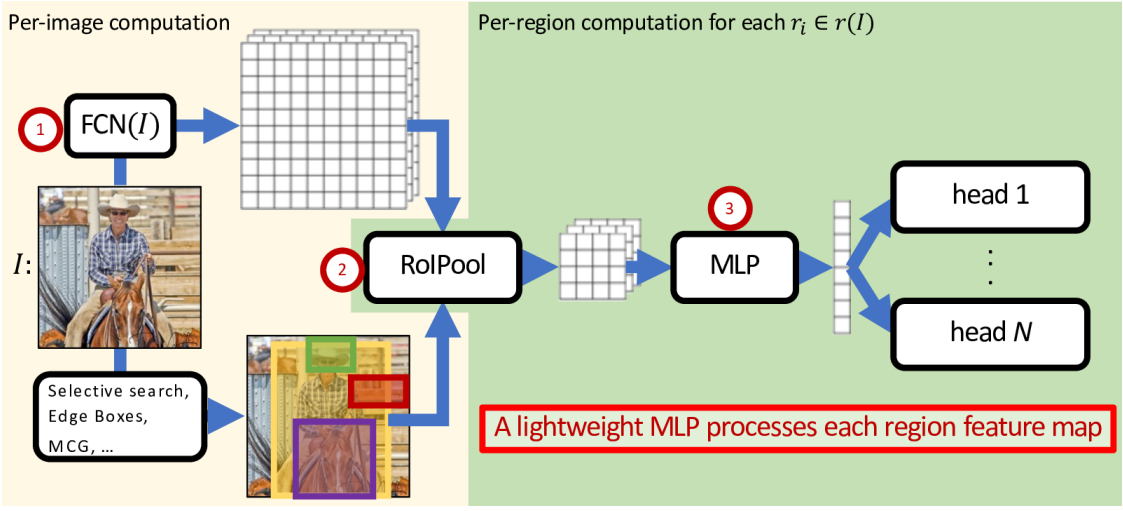
Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN



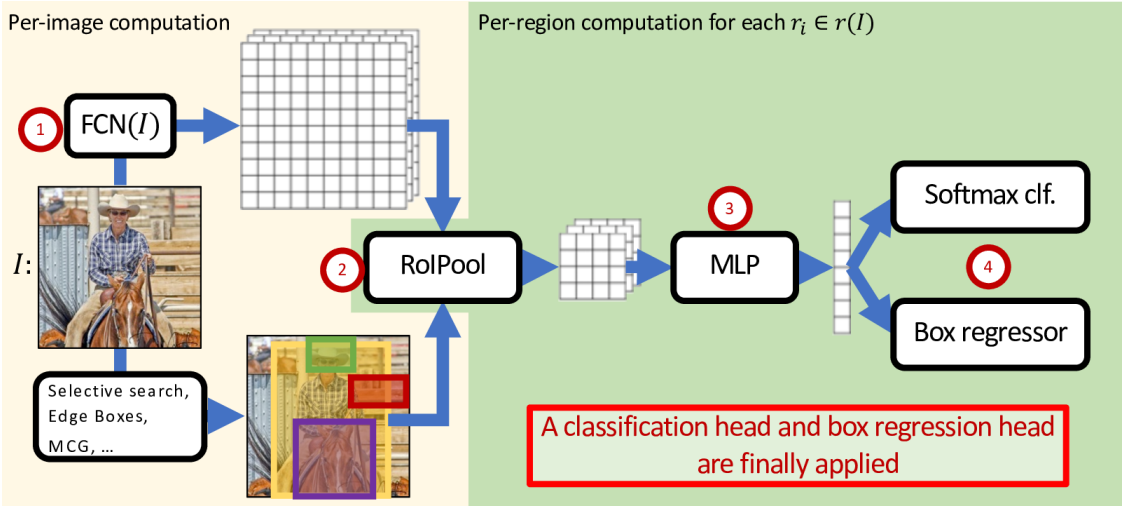
Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

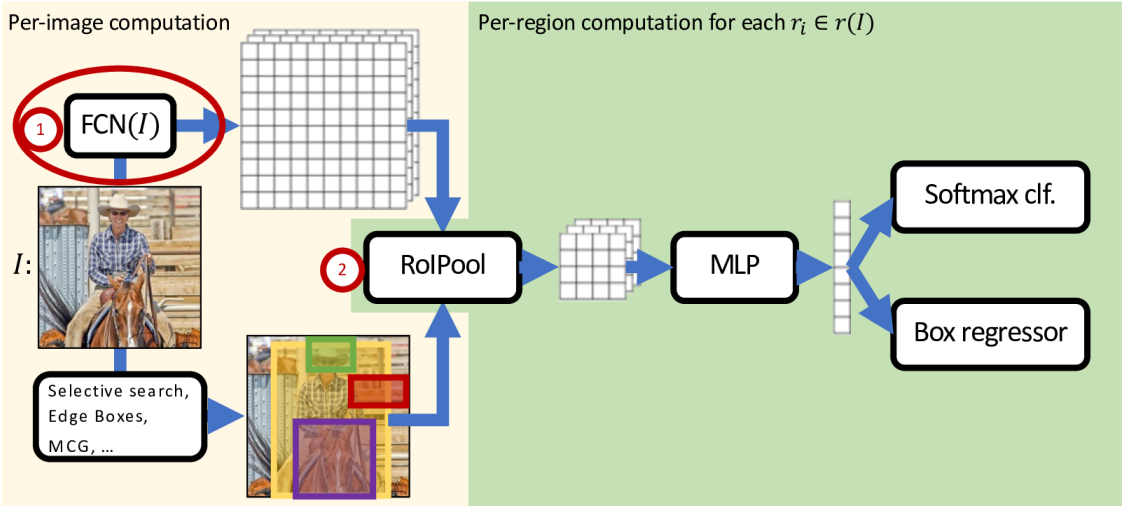


Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN


Per-image computation
Per-region computation for each $r_i \in r(I)$
1
FCN(I)
I:
Selective search,
Edge Boxes,
MCG, ...
2
RoIPool
3
MLP
Softmax clf.
4
Box regressor
A classification head and box regression head
are finally applied

Fast R-CNN

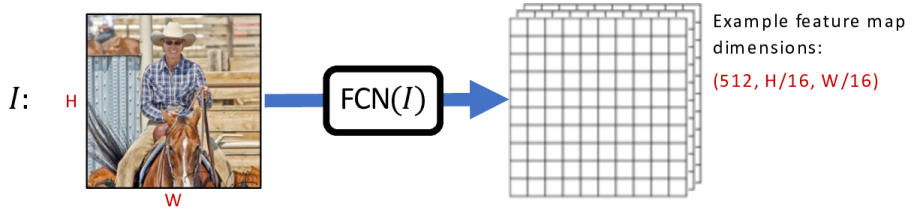


Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

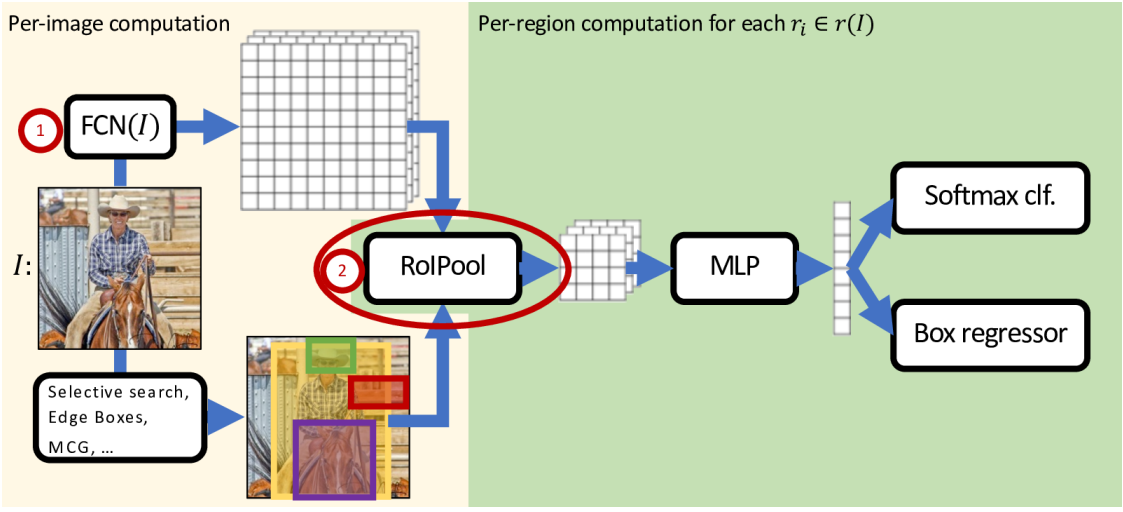
Backbone:

- Use any standard convolutional network as “backbone” architecture
  - E.g.: AlexNet, VGG, ResNet, Inception, ResNeXt, DenseNet, ...
- **Remove global pooling** $\Rightarrow$ output spatial dims proportional to input spatial dims
- A good network exploits the strongest recognition backbone (features matter!)



Girshick: Fast R-CNN. ICCV, 2015.

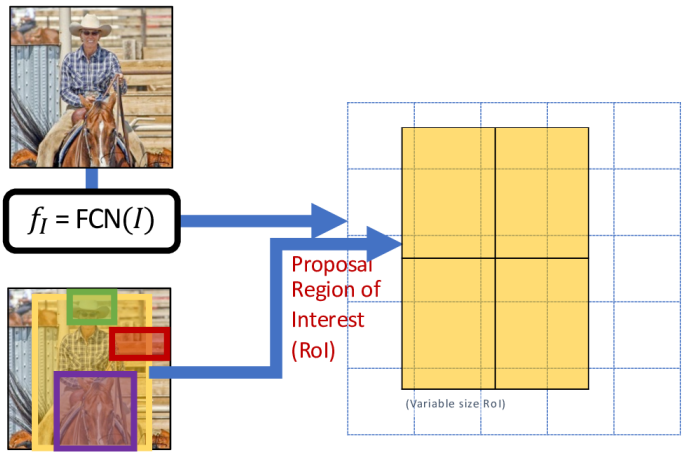
Fast R-CNN



Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

Region-of-Interest (RoI) Pooling on each Proposal:

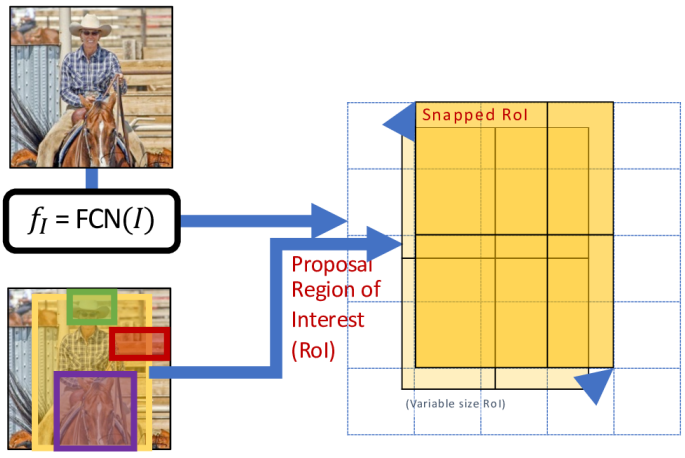


Key innovation in SPP-net
[He et al. 2014]

Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

Region-of-Interest (RoI) Pooling on each Proposal:

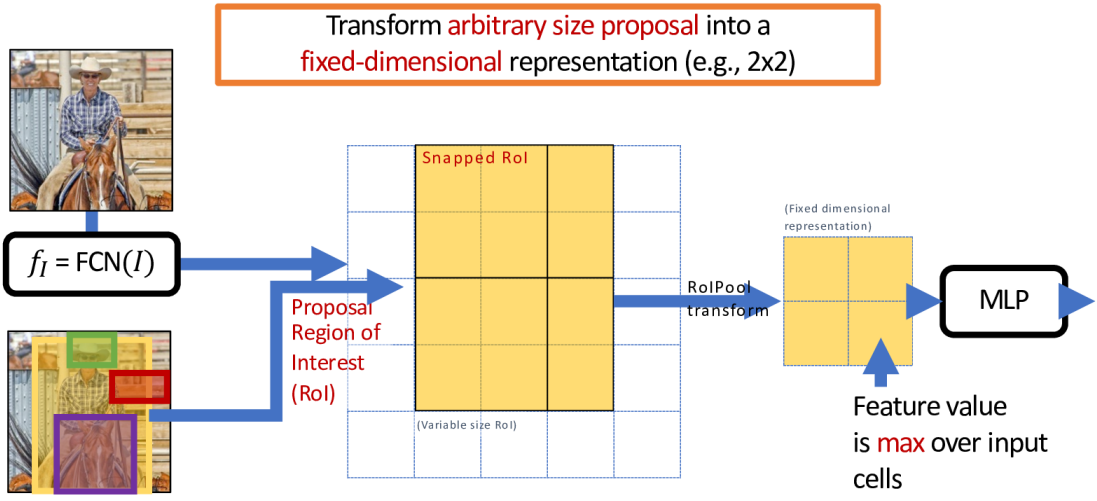


Key innovation in SPP-net [He et al. 2014]

Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

Region-of-Interest (RoI) Pooling on each Proposal:



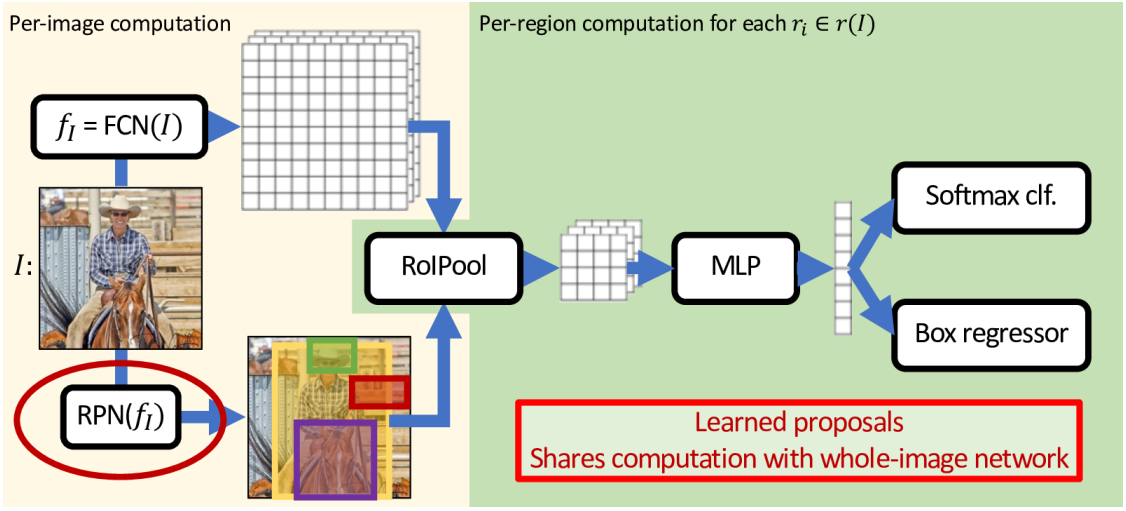
Girshick: Fast R-CNN. ICCV, 2015.

Fast R-CNN

What is the problem with Fast R-CNN?

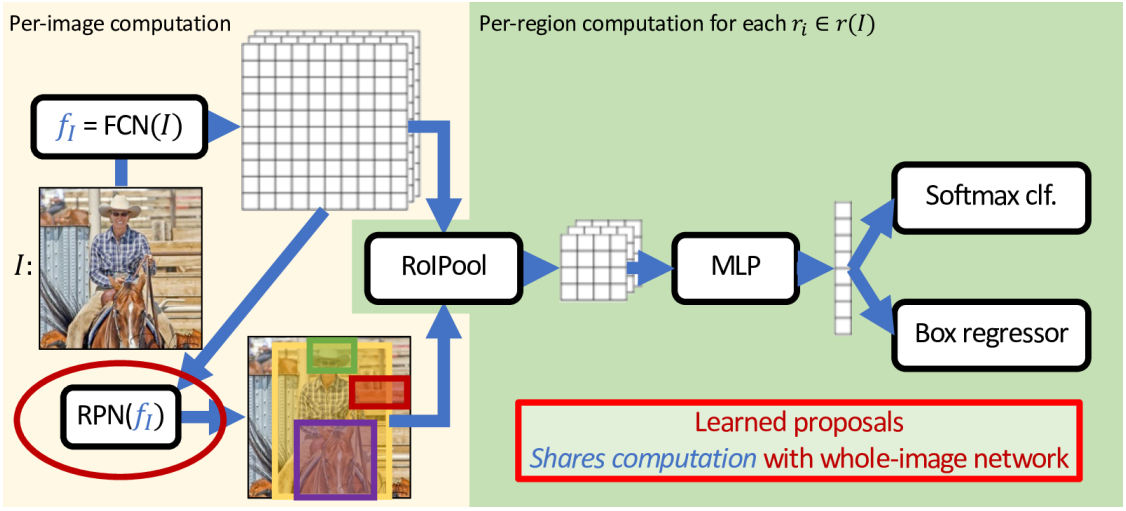
- ~~Heavy per-region computation (e.g., 2000 full network evaluations)~~
- ~~No computation/feature sharing~~
- Slow region proposal method adds to runtime
- Generic region proposal techniques have low recall

Faster R-CNN



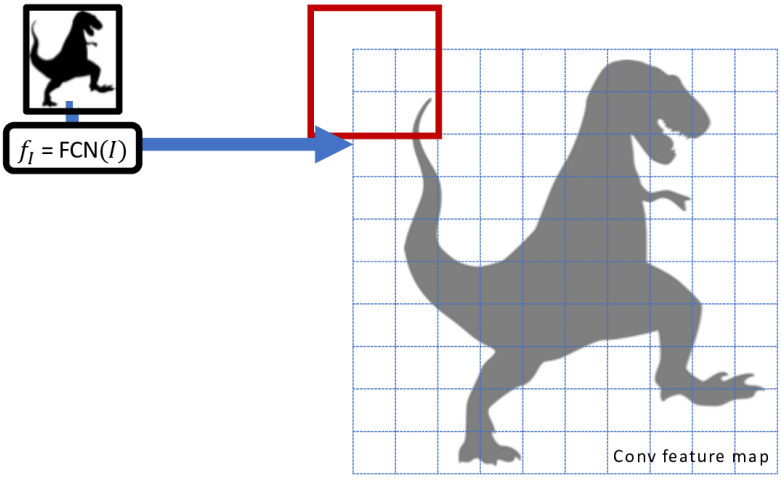
Ren, He, Girshick and Sun et al.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN



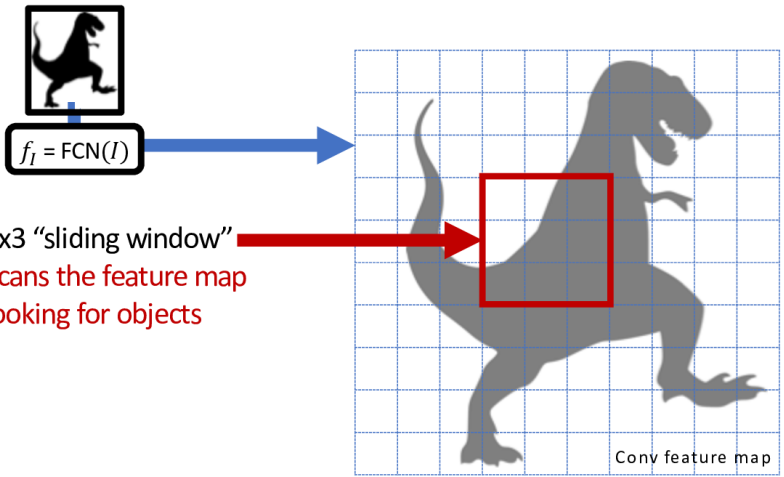
Ren, He, Girshick and Sun et al.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)

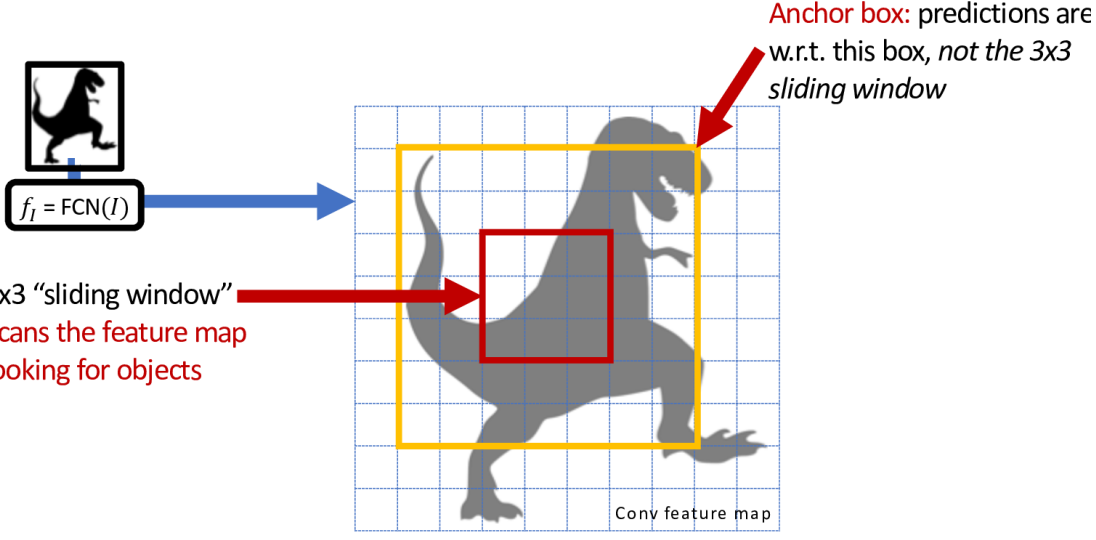


Ren, He, Girshick and Sun et al.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

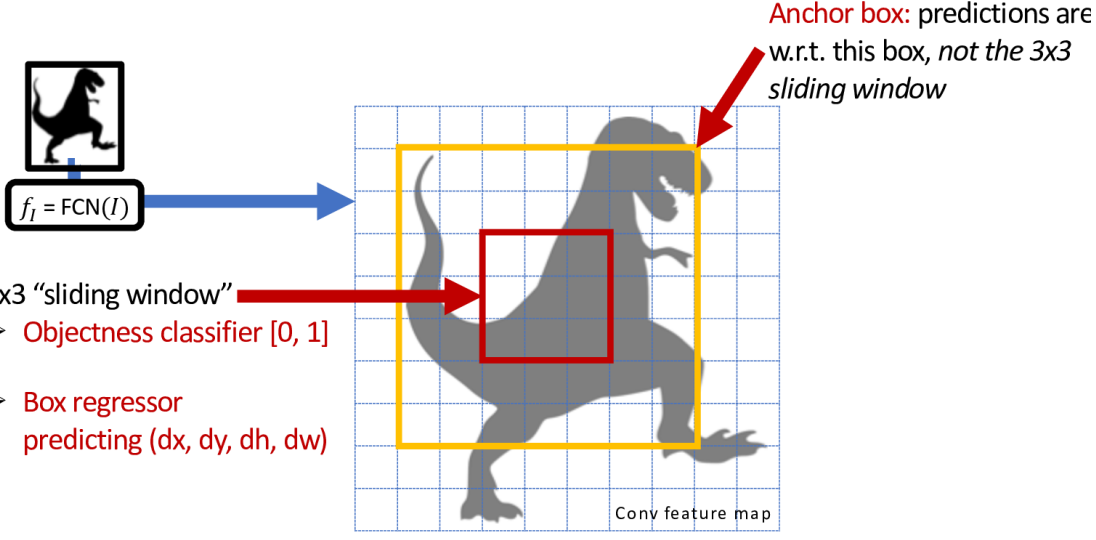
Faster R-CNN: Region Proposal Network (RPN)



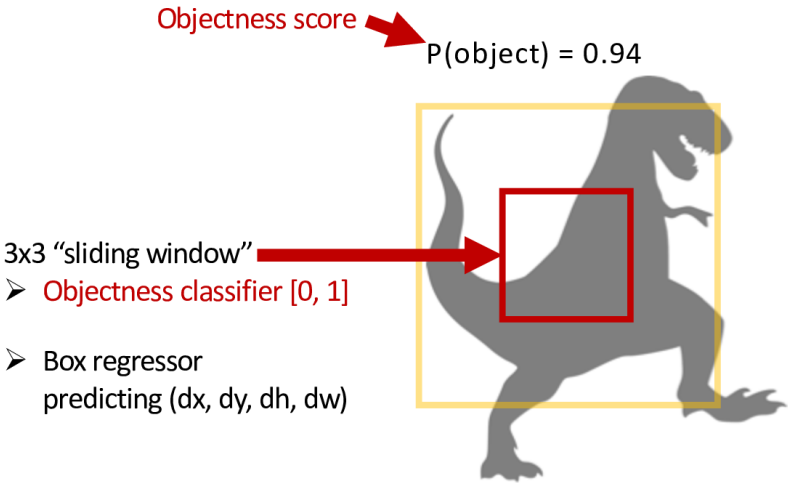
Faster R-CNN: Region Proposal Network (RPN)



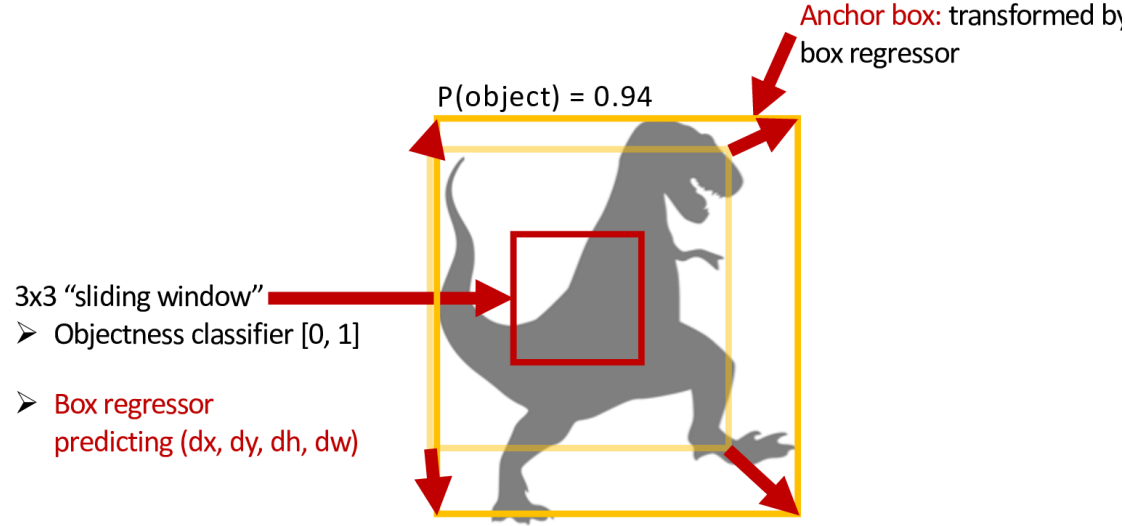
Faster R-CNN: Region Proposal Network (RPN)



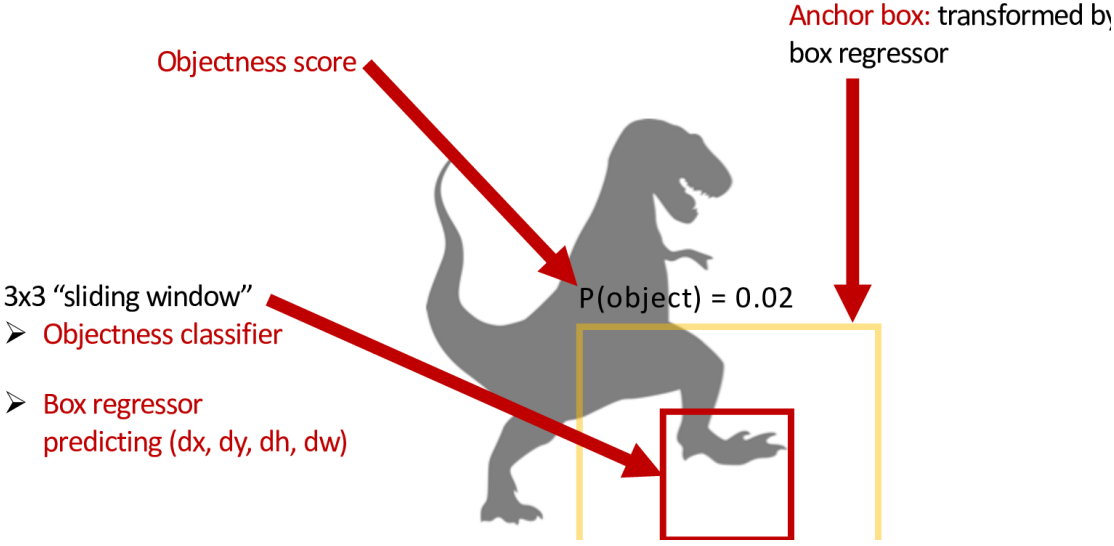
Faster R-CNN: Region Proposal Network (RPN)



Faster R-CNN: Region Proposal Network (RPN)



Faster R-CNN: Region Proposal Network (RPN)



Ren, He, Girshick and Sun et al.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015.

Faster R-CNN: Region Proposal Network (RPN)

f_I = FCN(I)

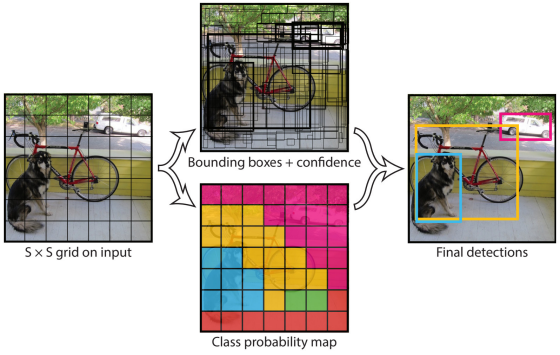
Anchor boxes: *K* anchors per location with different scales and aspect ratios

3x3 "sliding window"

- *K* objectness classifiers
- *K* box regressors

Conv feature map

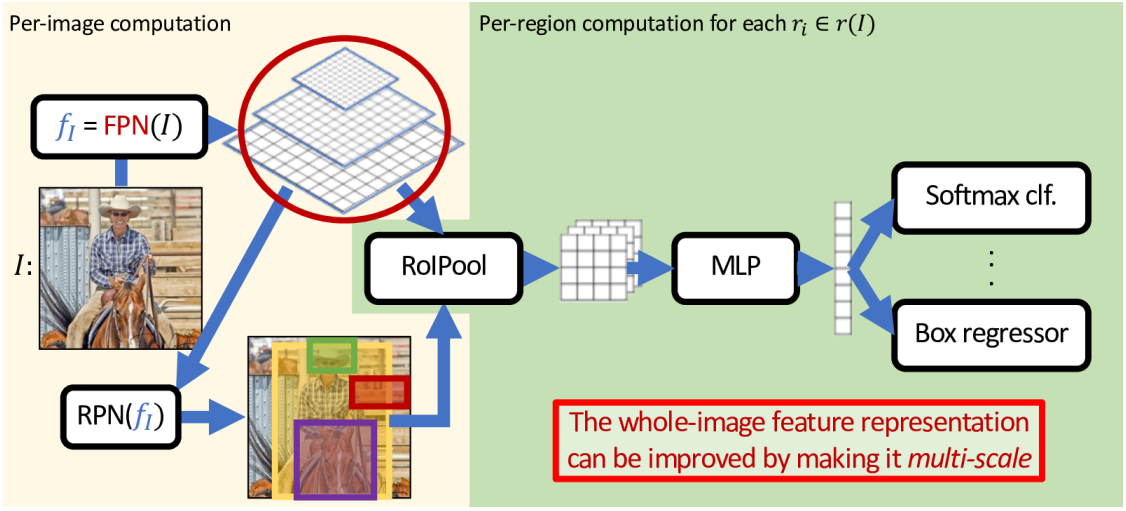
Single Stage Detection



- YOLO and SSD: Alternative to proposal based (two stage) methods
- Predict 2D bounding boxes in a single stage
- Faster, but often worse performance in practice

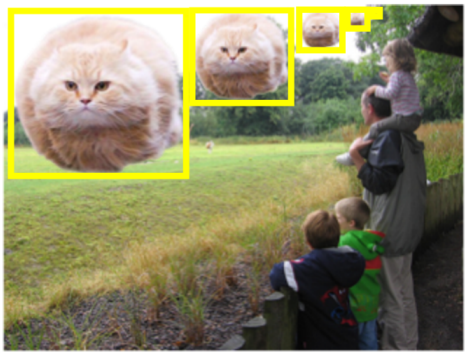
Joseph, Divvala, Girshick and Farhadi: You Only Look Once: Unified, Real-Time Object Detection. CVPR, 2016.

Feature Pyramid Network



Feature Pyramid Network

Goal of Feature Pyramid Network: Improve Scale Equivariance



Detectors need to

1. classify and
2. localize

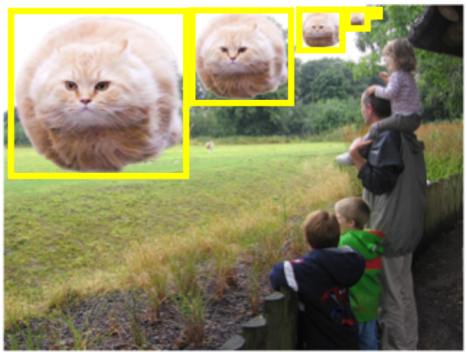
objects over a wide range of scales

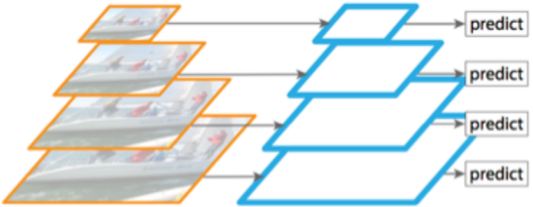
FPN improves this ability

Lin, Dollár, Girshick, He, Hariharan and Belongie: Feature Pyramid Networks for Object Detection. CVPR, 2017.

Feature Pyramid Network

Strategy 1: Image Pyramid





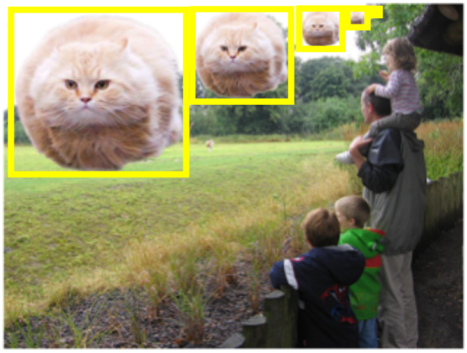
(a) Featurized image pyramid

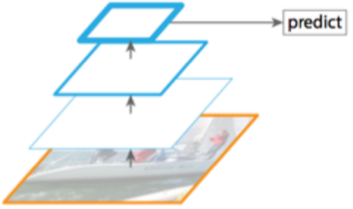
Standard solution – *slow!*

(E.g., Viola & Jones, HOG, DPM, SPP-net,

Feature Pyramid Network

Strategy 2: Multi-scale Features (Single-scale Map)





(b) Single feature map

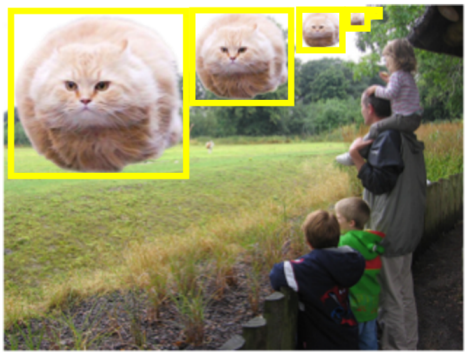
Leave it all to the features – *fast, suboptimal*

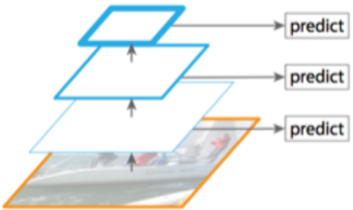
(E.g., Fast/er R-CNN, YOLO, …)

Lin, Dollár, Girshick, He, Hariharan and Belongie: Feature Pyramid Networks for Object Detection. CVPR, 2017.

Feature Pyramid Network

Strategy 3: Naïve In-network Pyramid





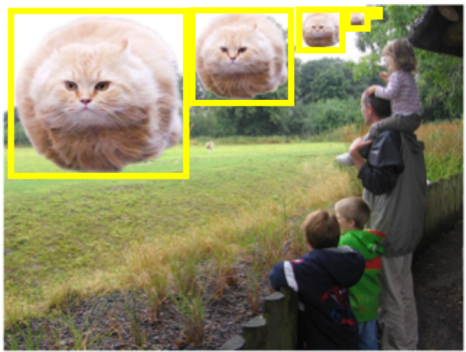
(c) Pyramidal feature hierarchy

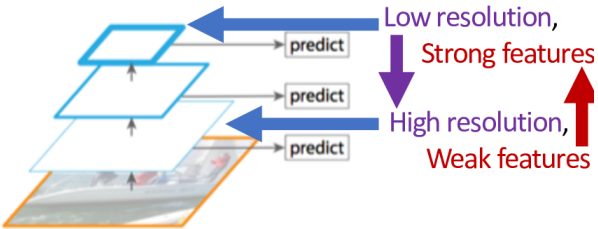
Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Feature Pyramid Network

Strategy 3: Naïve In-network Pyramid





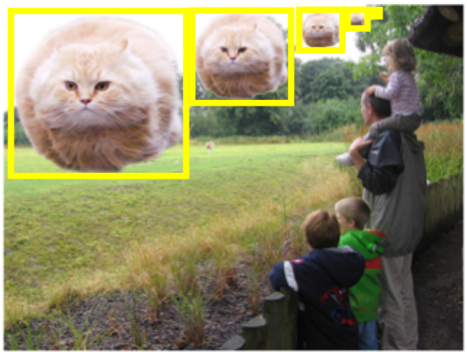
(c) Pyramidal feature hierarchy

Use the internal pyramid – *fast, suboptimal*

(E.g., ≈ SSD, …)

Feature Pyramid Network

Strategy 4: Feature Pyramid Network



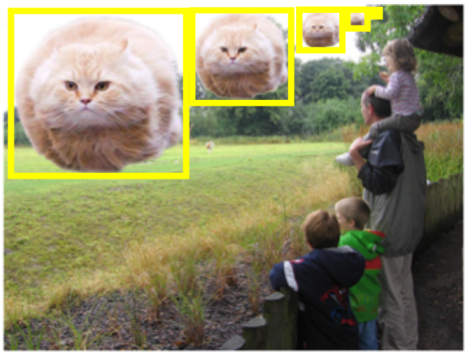
(d) Feature Pyramid Network

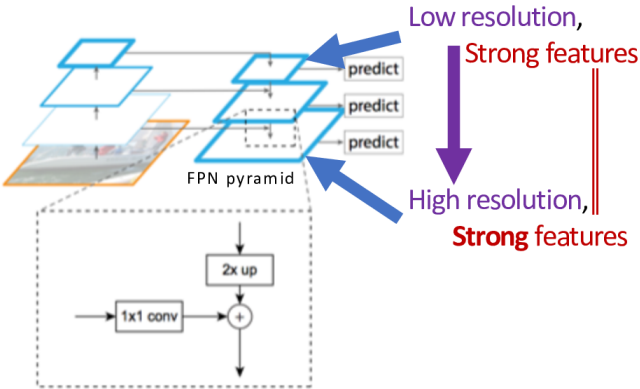
Top-down enrichment of high-res features – *fast, less suboptimal*

Lin, Dollár, Girshick, He, Hariharan and Belongie: Feature Pyramid Networks for Object Detection. CVPR, 2017.

Feature Pyramid Network

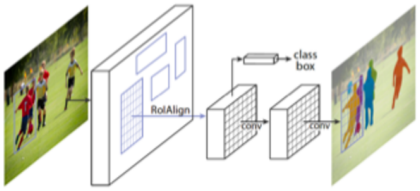
Strategy 4: Feature Pyramid Network



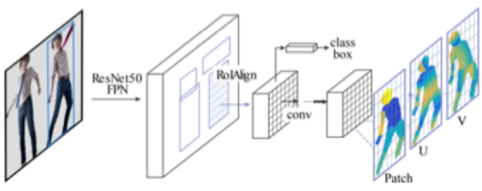


Lin, Dollár, Girshick, He, Hariharan and Belongie: Feature Pyramid Networks for Object Detection. CVPR, 2017.

Generalization to other Output Modalities



Mask R-CNN
[He, Gkioxari, Dollár, Girshick]

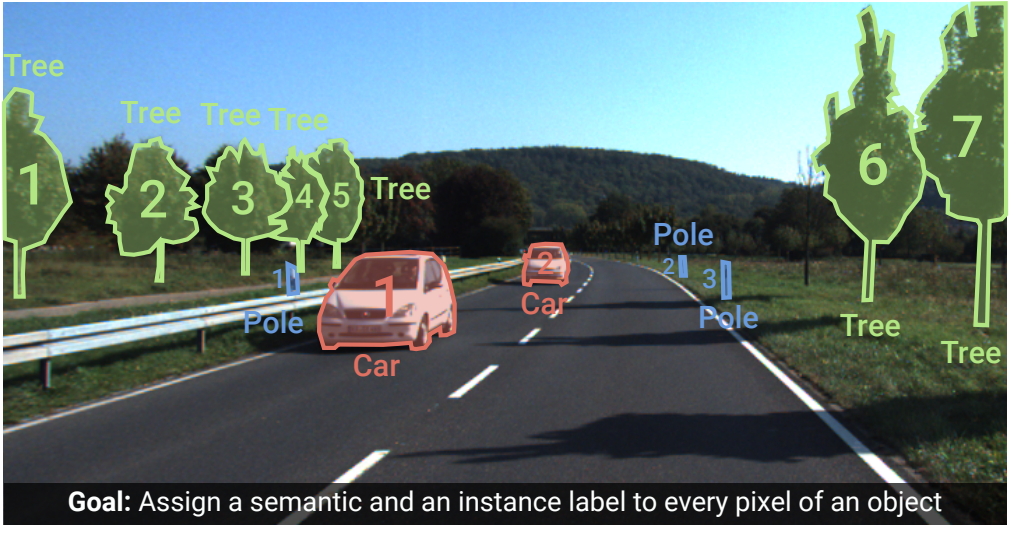


DensePose
[Güler, Neverova, Kokkinos]

It is easy to add **additional network heads** to this framework:

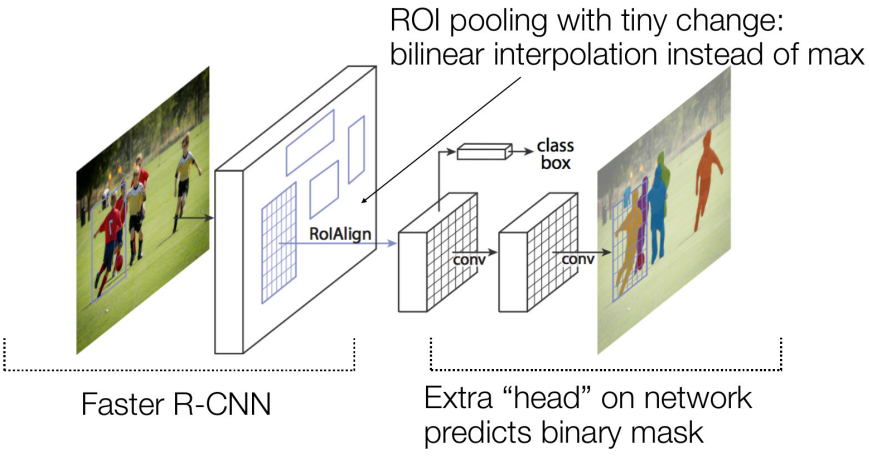
- **Mask R-CNN:** Predicting an instance segmentation mask per detection
- **DensePose:** Predict object coordinates (texture map coordinates)

Instance Segmentation



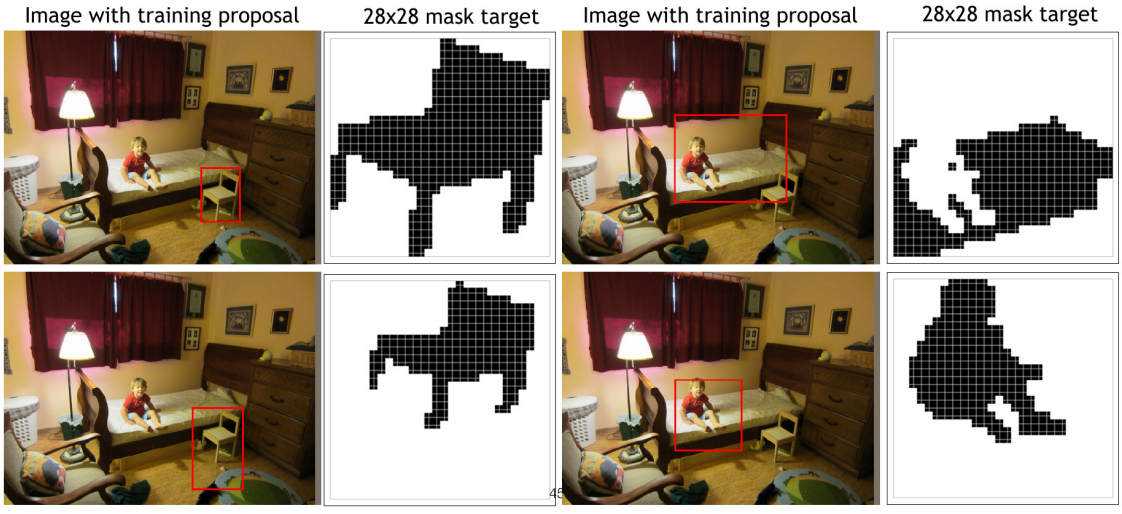
Goal: Assign a semantic and an instance label to every pixel of an object

Mask R-CNN: Faster R-CNN for Instance Segmentation



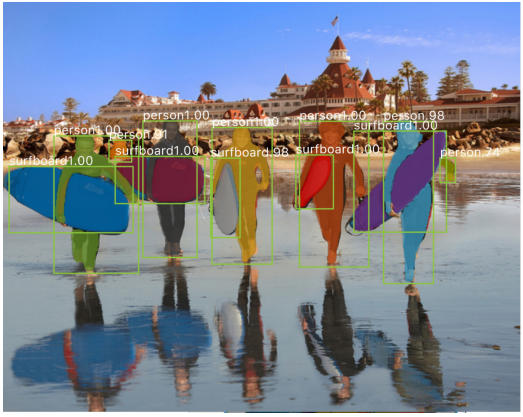
- Predict binary instance segmentation (foreground vs. background) per detection

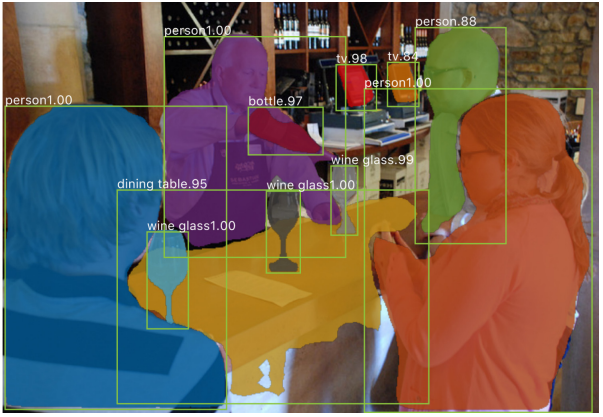
Mask R-CNN: Faster R-CNN for Instance Segmentation



He, Gkioxari, Dollár and Girshick: Mask R-CNN. ICCV, 2017.

Mask R-CNN: Faster R-CNN for Instance Segmentation

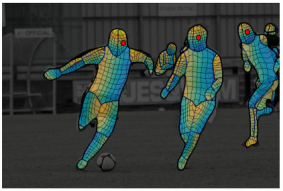




- Evaluation: Intersection-over-Union at mask level (not bounding box level)

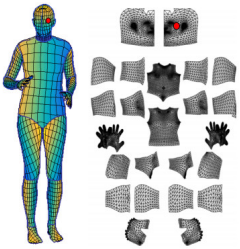
He, Gkioxari, Dollár and Girshick: Mask R-CNN. ICCV, 2017.

DensePose: Faster R-CNN for Dense Human Pose Estimation



DensePose-RCNN Results

DensePose COCO Dataset



- Map all human pixels of an RGB image to the 3D surface of the human body
- Densely regress part-specific UV coordinates within every human region
- Main contribution: DensePose-COCO, a large-scale dataset with manually annotated image-to-surface correspondences

Mesh R-CNN: Faster R-CNN for Meshes

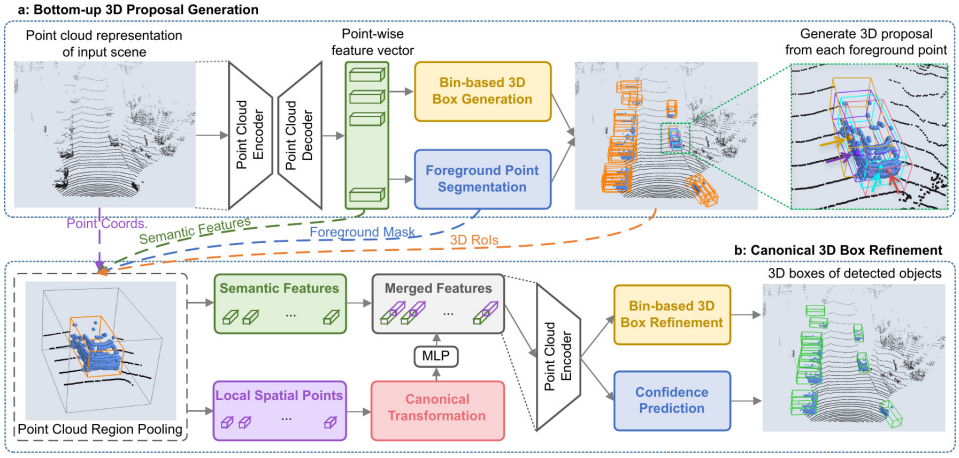
Box & Mask Predictions

Mesh Predictions

- Predict vertices and faces of a 3D mesh for each 2D bounding box

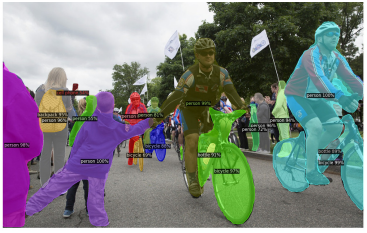
Gkioxari, Malik and Johnson: Mesh R-CNN. ICCV, 2019.

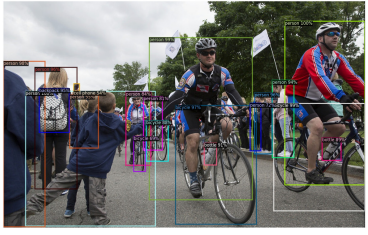
PointRCNN: Faster R-CNN for Point Clouds



- Predict 3D bounding boxes from point cloud (e.g., Velodyne HDL-64 laser scan)

Detectron2 for PyTorch





https://github.com/facebookresearch/detectron2