OverFeat

Integrated Recognition, Localization and Detection using Convolutional Networks

Sermanet et. al

Presentation by Eric Holmdahl

Roadmap

1. Goal
2. Background
3. Related Work
4. Algorithm Overview
5. Breakdown By Task
    1. Classification
    2. Localization
    3. Detection

Goal

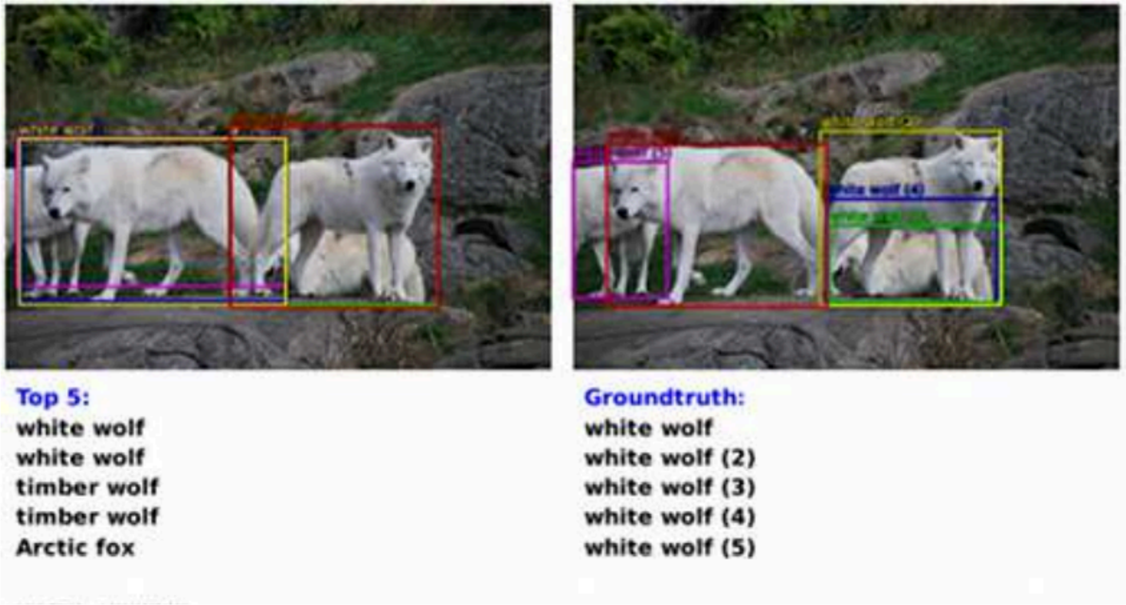
Perform classification, localization, and detection on the ImageNet Dataset

Classification

Determining what is the main object in an image

Localization

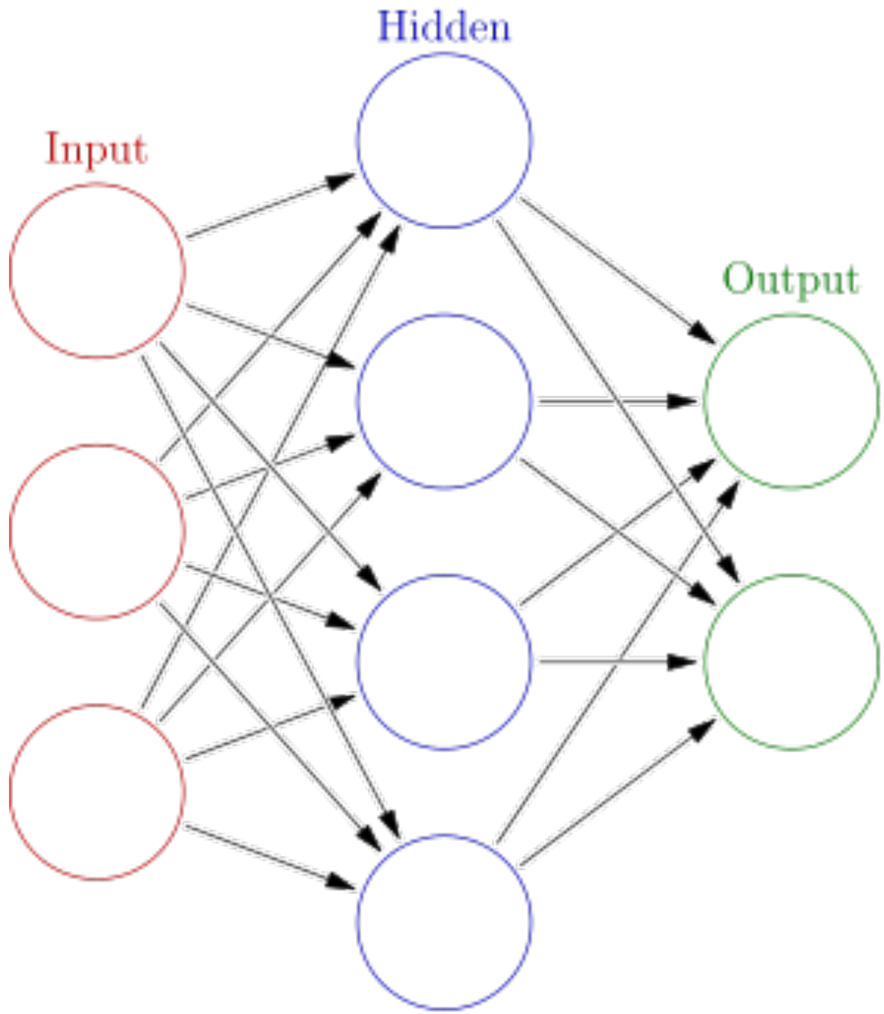
- Determining where an object is located in an image



Detection

- Performing localization for all objects present in an image

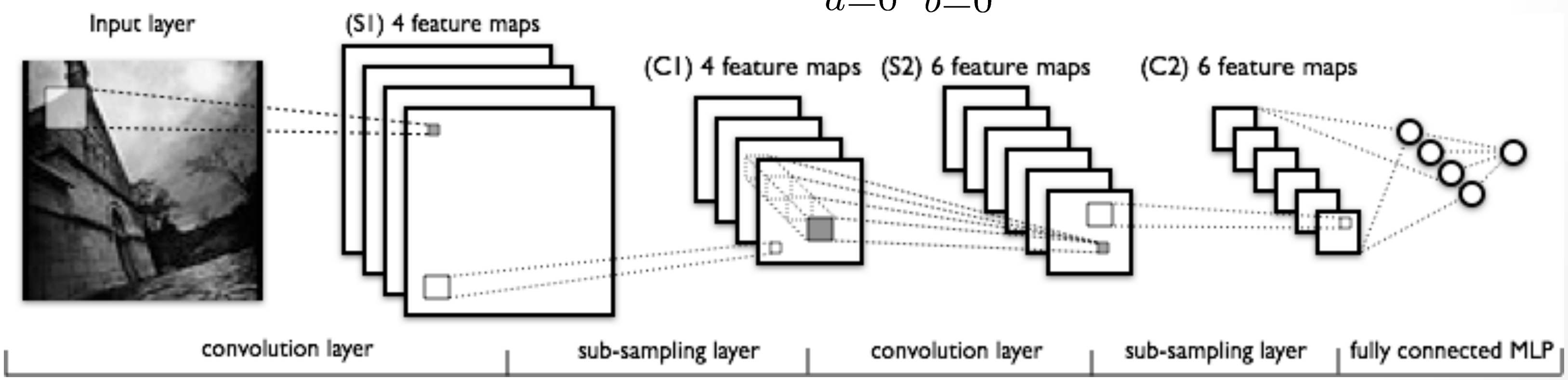
Background: Feed Forward Neural Networks



Background: Convolutional Nets

- Alternating convolution and max pooling layers feed into fully connected neural net
- Max pooling: with window size kxk, outputs highest intensity value in window size
- Convolution: Scanning window, shared weights within window

$$x_{ij}^{k} = \sum_{a=0}^{m-1} \sum_{b=0}^{m-1} \omega_{ab} y_{(i+a)(j+b)}^{k-1}$$

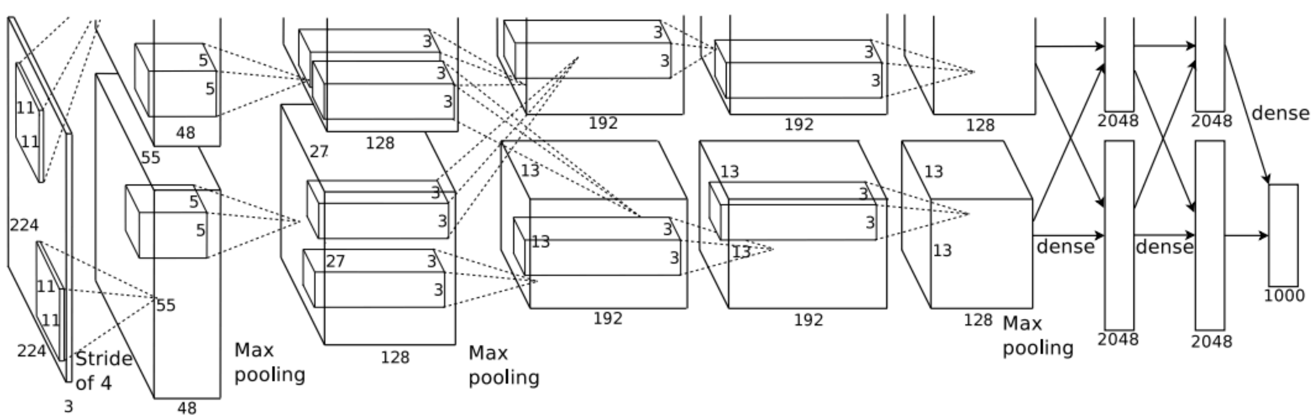


Related Work

Krizhevsky et. Al: *ImageNet Classification With Deep Convolutional Neural Networks*

Review: Krizhevsky Architecture

- Large CNNs used to densely process images with overlapping windows
- ReLU Nonlinear neuron output
- DropOut



Krizhevksy Results

- Brought CNNs to forefront of classification/localization/detection problem

Model	Top-1 (val)	Top-5 (val)	Top-5 (test)
SIFT + FVs [7]	—	—	*26.2%*
1 CNN	40.7%	18.2%	—
5 CNNs	38.1%	16.4%	**16.4%**
1 CNN*	39.0%	16.6%	—
7 CNNs*	36.7%	15.4%	**15.3%**

Giusti et. al: *Fast Image Scanning With Deep Convolutional Networks*

Giusti Fast Scanning

- Problem: CNNs perform a great deal of redundant computing of convolutions due to overlapping patches
- Solution: Apply convolution to entire image at once!

Giusti et. al: *Fast Image Scanning With Deep Convolutional Networks*

Convolutional Layer:

Apply convolutional kernel to each input map of each fragment, same number of fragments as previous level

Max Pooling Layer:

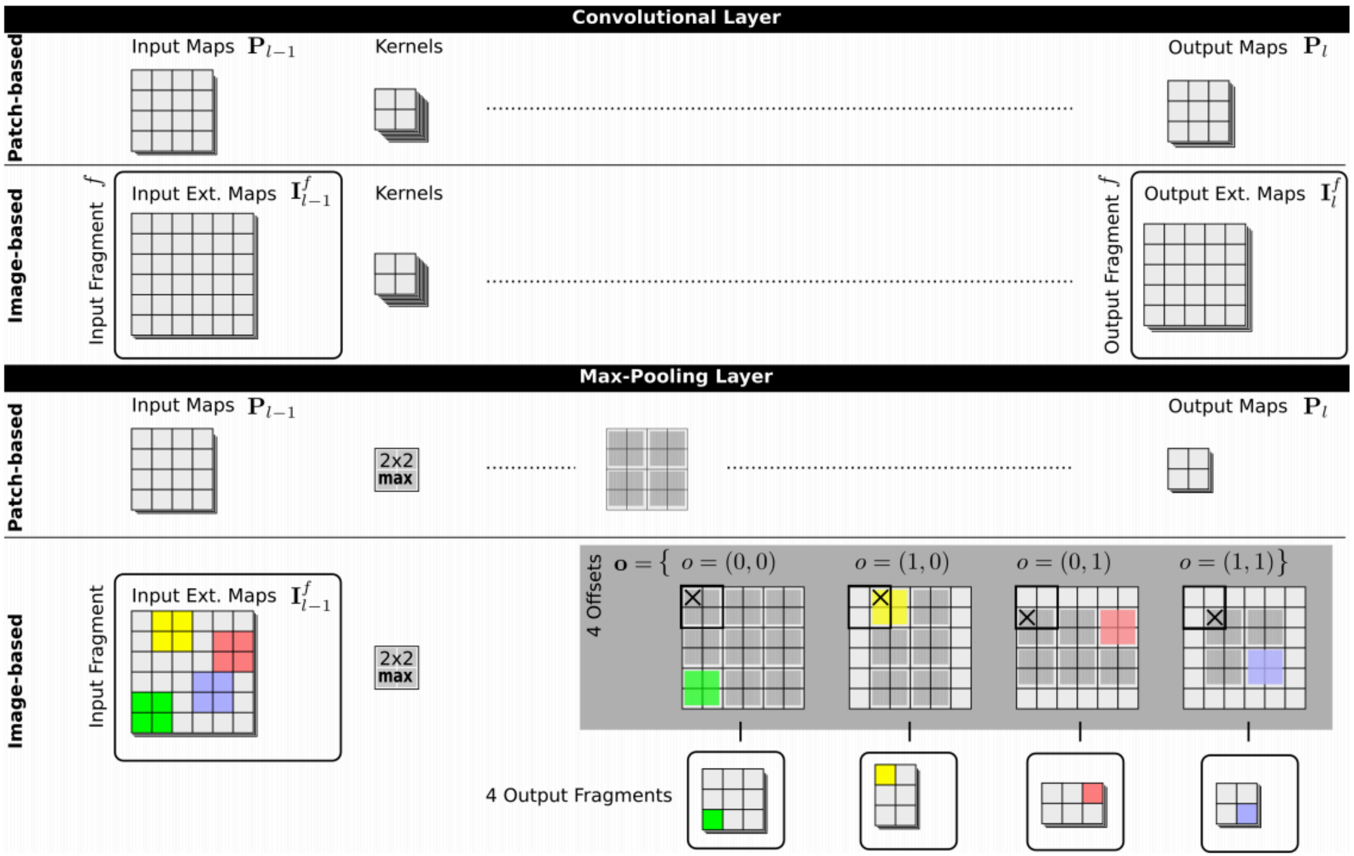
Pixel at $(\bar{x}, \bar{y})$in output map is max of all pixels from input map at (x, y) such that

$$o_x + k\bar{x} \leq x \leq o_x + k\bar{x} + k - 1$$

$$o_y + k\bar{y} \leq y \leq o_y + k\bar{y} + k - 1$$

Generates k^2 new fragments at each max pooling layer

Giusti et. al: *Fast Image Scanning With Deep Convolutional Networks*



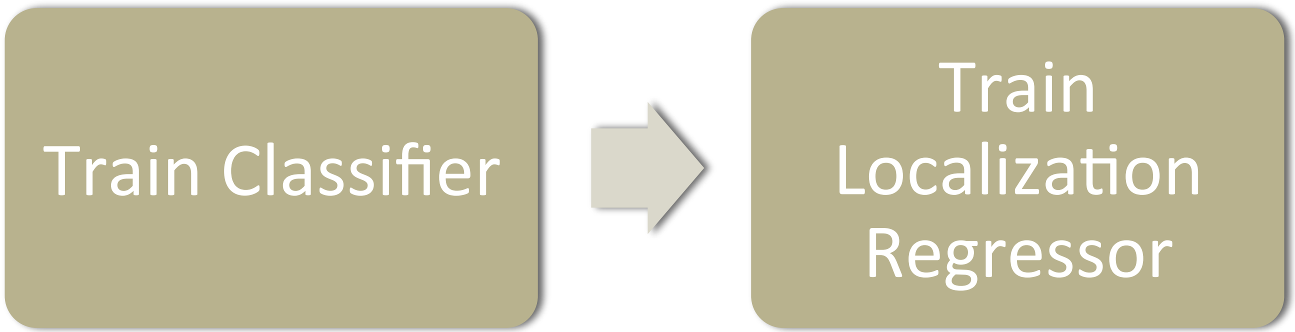
Giusti et. al: Results

Layer (l)	s	s_{l-1}	$\lvert\mathbf{P}_{l-1}\rvert$	$\lvert\mathbf{P}_l\rvert$	w_l	k_l	F_l	$\text{FLOPS}_l^{\text{patch}}[\cdot 10^9]$	$\text{FLOPS}_l^{\text{image}}[\cdot 10^9]$	speedup
1	512	559	1	48	92	4	1	3408	0.5	7114.8
3	512	279	48	48	42	5	4	53271	35.9	1485.1
5	512	139	48	48	18	4	16	6262	22.8	274.7
7	512	69	48	48	6	4	64	695	22.5	30.9
Total								63636	81.6	779.8

Provides massive improvements in speed for sliding window CNNs!

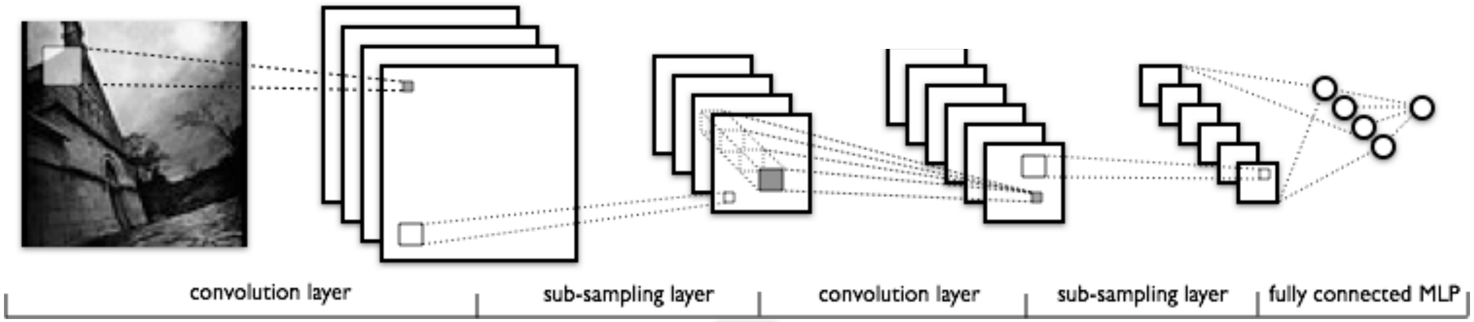
Algorithm Overview

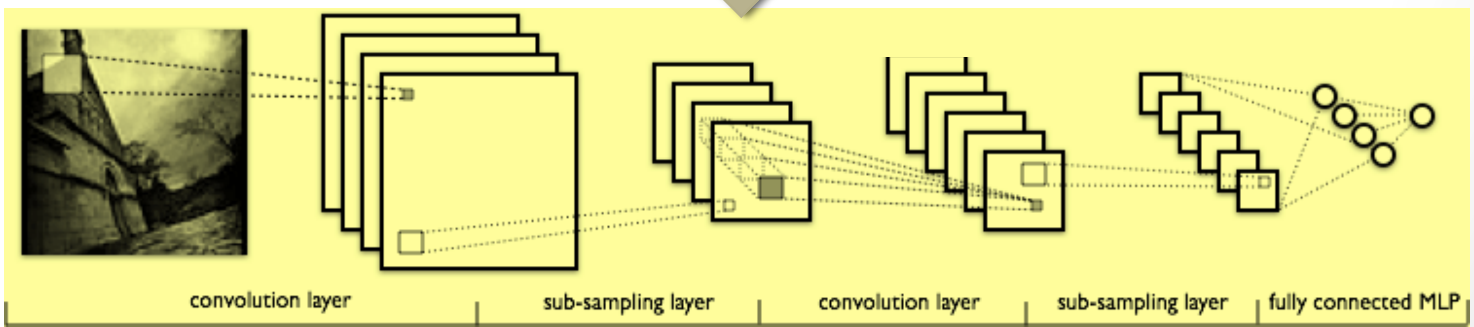
Algorithm Overview: Training



Algorithm Overview: Training

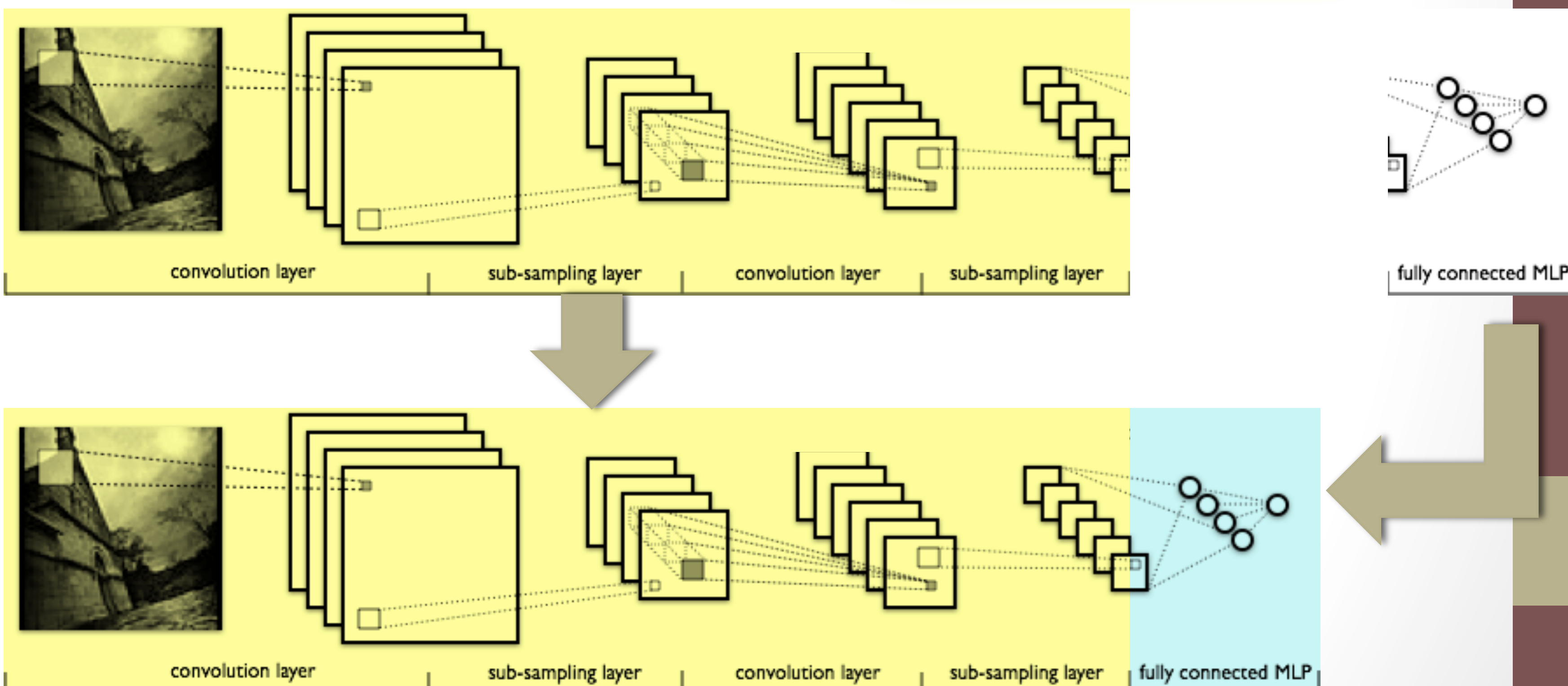
Train Classifier → Train Localization Regressor





Algorithm Overview: Training

Train Classifier → Train Localization Regressor



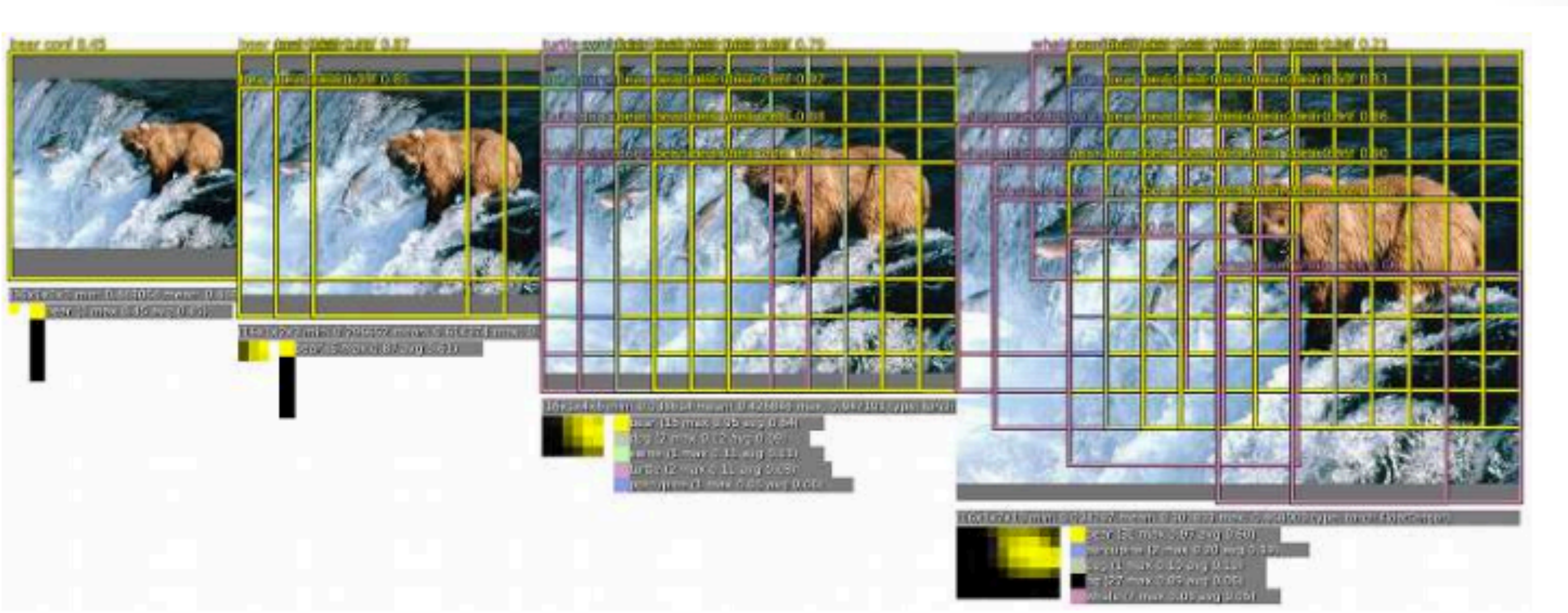
Algorithm Overview: Training

Train Classifier → Train Localization Regressor

- Input: Images with classification and bounding box
- Training objective: Minimize l2 norm between generated bounding box and ground truth
- One regressor generated for each possible image class
- Output: (x,y) coordinates of top left, top right corner of bounding box

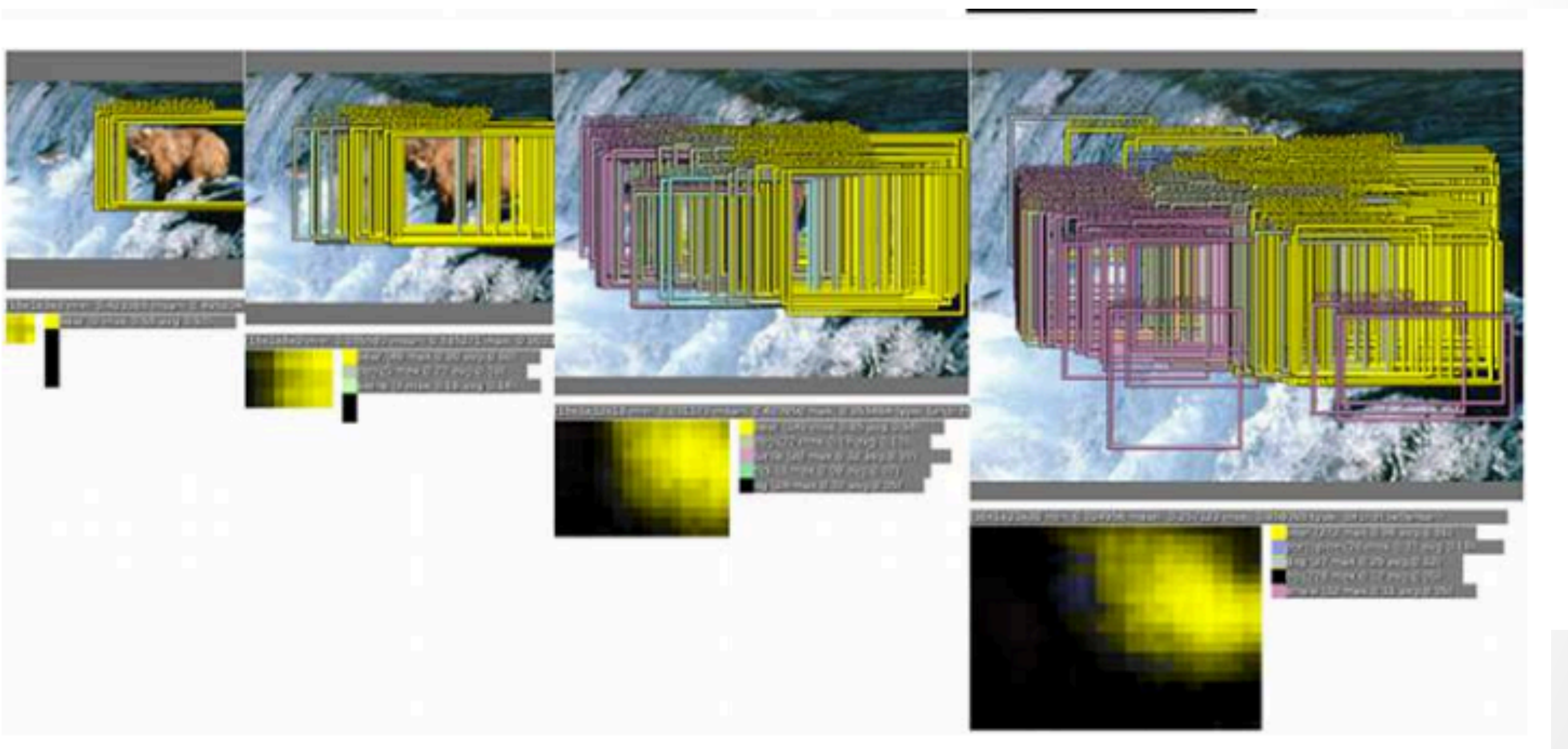
Algorithm Overview: Runtime

1. Perform classification at each location using trained CNN



Algorithm Overview: Runtime

2. Perform localization on all classified regions generated by classifier



Algorithm Overview: Runtime

3. Merge bounding boxes with sufficient overlap from localization and sufficient confidence of being same object from classifier

Breakdown By Task

Classification

OverFeat Feature Extraction

- First 5 layers of Deep Convolutional Neural Net: similar to Krizhevsky's
- Images downsampled to 256x256
- No contrast normalization, non-overlapping pooling

Layer	1	2	3	4	5	6	7	Output 8
Stage	conv + max	conv + max	conv	conv	conv + max	full	full	full
# channels	96	256	512	1024	1024	3072	4096	1000
Filter size	11x11	5x5	3x3	3x3	3x3	-	-	-
Conv. stride	4x4	1x1	1x1	1x1	1x1	-	-	-
Pooling size	2x2	2x2	-	-	2x2	-	-	-
Pooling stride	2x2	2x2	-	-	2x2	-	-	-
Zero-Padding size	-	-	1x1x1x1	1x1x1x1	1x1x1x1	-	-	-
Spatial input size	231x231	24x24	12x12	12x12	12x12	6x6	1x1	1x1

OverFeat Classification: Dense Sliding Window

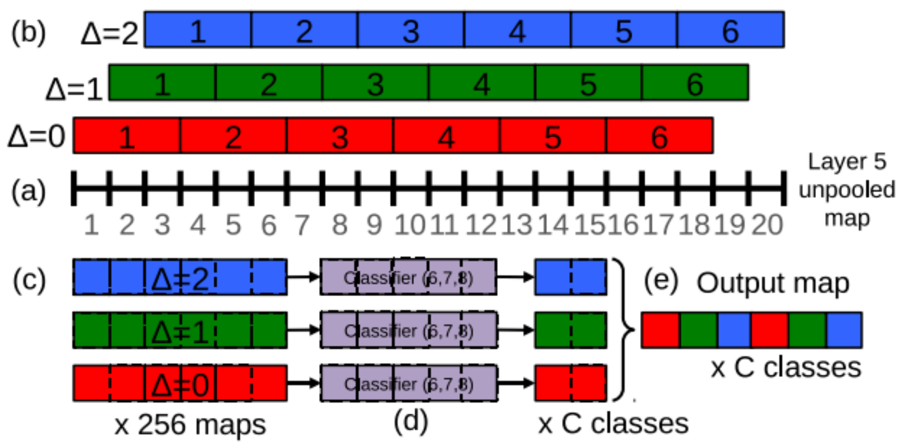


Figure 3: 1D illustration (to scale) of output map computation for classification, using y-dimension from scale 2 as an example (see Table 5). (a): 20 pixel unpooled layer 5 feature map. (b): max pooling over non-overlapping 3 pixel groups, using offsets of $\Delta = \{0, 1, 2\}$ pixels (red, green, blue respectively). (c): The resulting 6 pixel pooled maps, for different Δ. (d): 5 pixel classifier (layers 6,7) is applied in sliding window fashion to pooled maps, yielding 2 pixel by C maps for each Δ. (e): reshaped into 6 pixel by C output maps.

Multi-Scale Classification

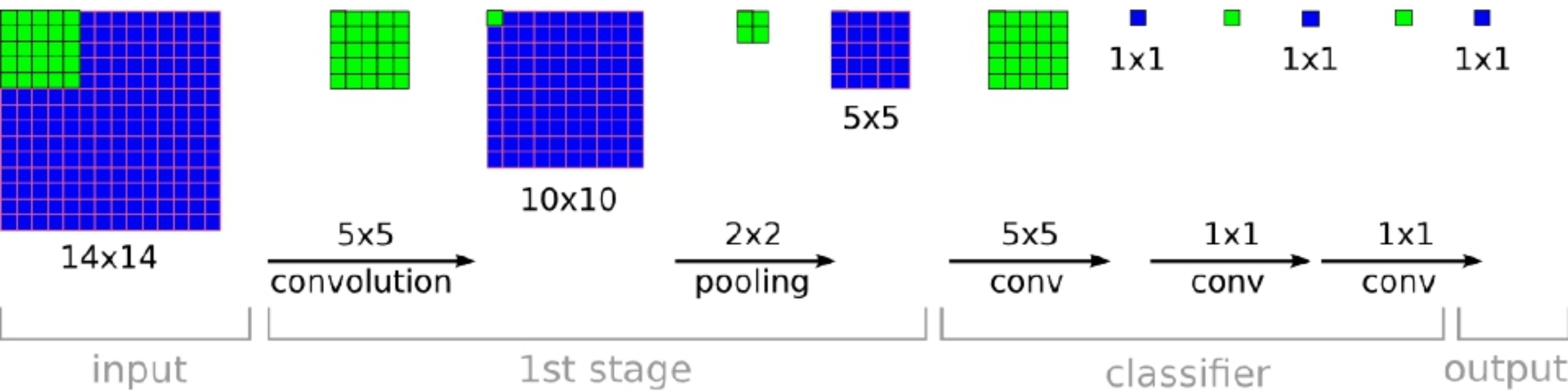
- Classification performed at 6 scales at test time, but only 1 scale at runtime
- Increases robustness of model

Scale	Input size	Layer 5 pre-pool	Layer 5 post-pool	Classifier map (pre-reshape)	Classifier map size
1	245x245	17x17	(5x5)x(3x3)	(1x1)x(3x3)xC	3x3xC
2	281x317	20x23	(6x7)x(3x3)	(2x3)x(3x3)xC	6x9xC
3	317x389	23x29	(7x9)x(3x3)	(3x5)x(3x3)xC	9x15xC
4	389x461	29x35	(9x11)x(3x3)	(5x7)x(3x3)xC	15x21xC
5	425x497	32x35	(10x11)x(3x3)	(6x7)x(3x3)xC	18x24xC
6	461x569	35x44	(11x14)x(3x3)	(7x10)x(3x3)xC	21x30xC

Table 5: **Spatial dimensions of our multi-scale approach**. 6 different sizes of input images are used, resulting in layer 5 unpooled feature maps of differing spatial resolution (although not indicated in the table, all have 256 feature channels). The (3x3) results from our dense pooling operation with $(\Delta_x, \Delta_y) = \{0, 1, 2\}$. See text and Fig. 3 for details for how these are converted into output maps.

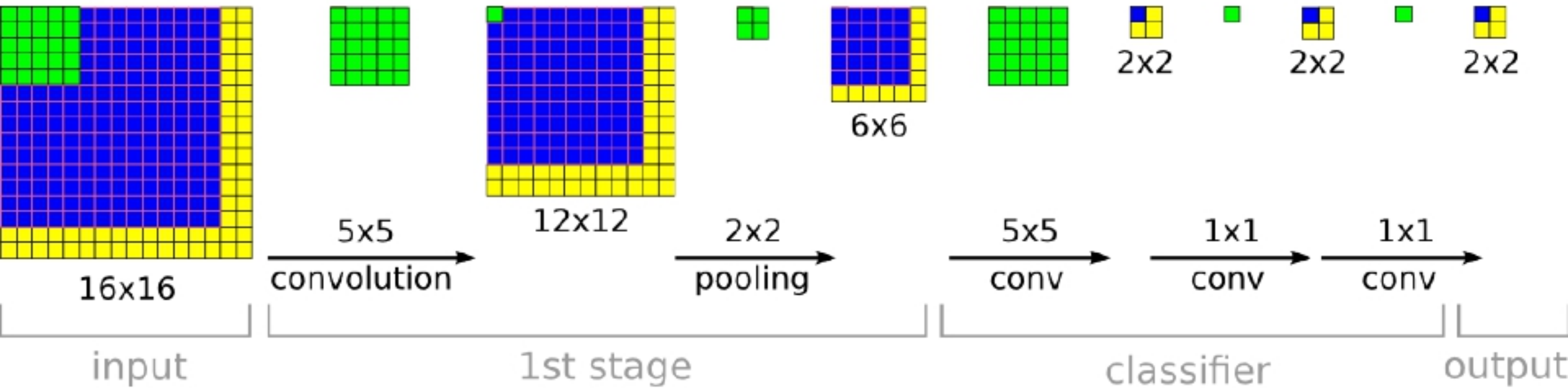
Classification: CNNs and Sliding Windows

- **Single output:**
  - 1x1 output
  - no feature space
  - blue: feature maps
  - green: operation kernel
  - typical training setup



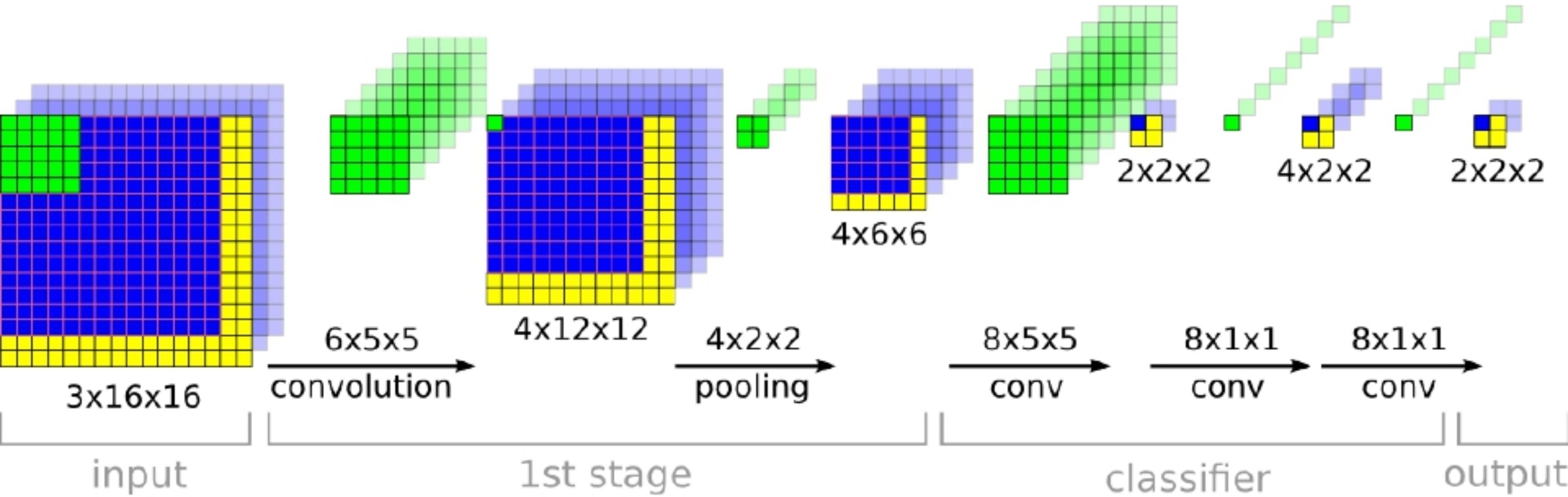
Classification: CNNs and Sliding Windows

- **Multiple outputs:**
  - 2x2 output
  - input stride 2x2
  - recompute only extra yellow areas



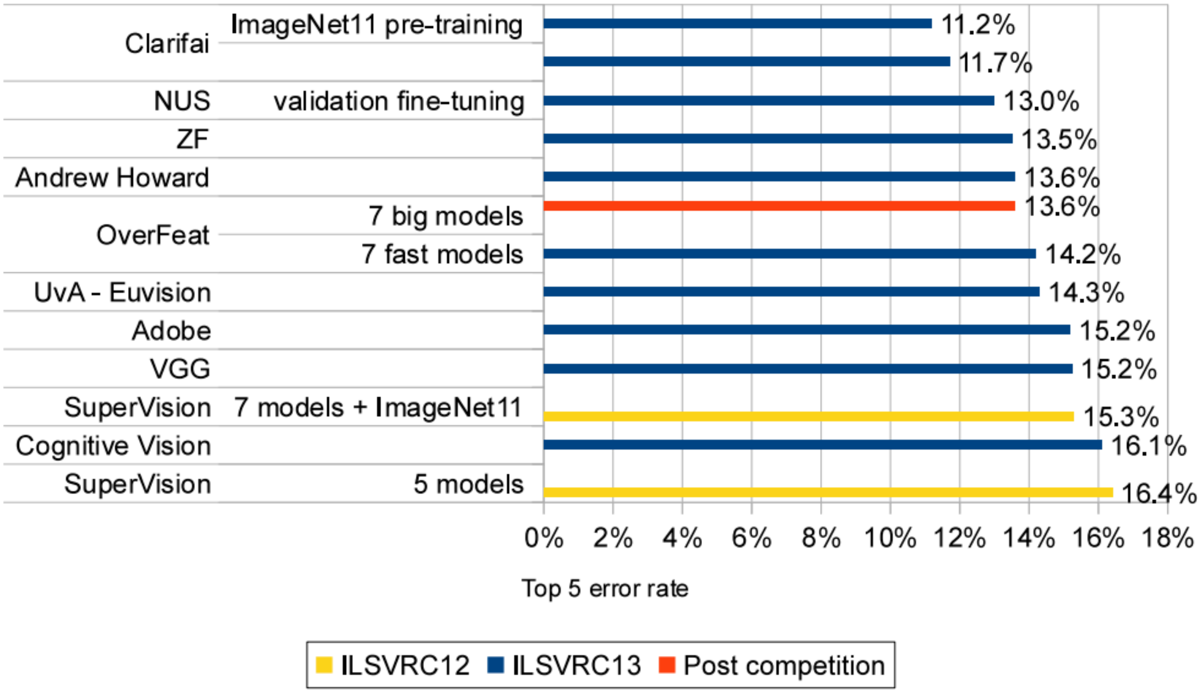
Classification: CNNs and Sliding Windows

- **With feature space**
  - 3 input channels
  - 4 feature maps
  - 2 feature maps
  - 4 feature maps
  - 2 outputs (e.g. 2-class classifier)



Classification: Results

Approach	Top-1 error %	Top-5 error %
Krizhevsky *et al.* [15]	40.7	18.2
OverFeat - 1 *fast* model, scale 1, coarse stride	39.28	17.12
OverFeat - 1 *fast* model, scale 1, fine stride	39.01	16.97
OverFeat - 1 *fast* model, 4 scales (1,2,4,6), fine stride	38.57	16.39
OverFeat - 1 *fast* model, 6 scales (1-6), fine stride	38.12	16.27
OverFeat - 1 *accurate* model, 4 corners + center + flip	35.60	14.71
OverFeat - 1 *accurate* model, 4 scales, fine stride	35.74	14.18
OverFeat - 7 *fast* models, 4 scales, fine stride	35.10	13.86
OverFeat - 7 *accurate* models, 4 scales, fine stride	33.96	13.24



Localization

Training Localizer

- Use same first 5 layers as trained classifier
- Remove fully connected layers, replace with regressor
- Train again on labeled input with bounding boxes

Localization: Fully Connected Layers

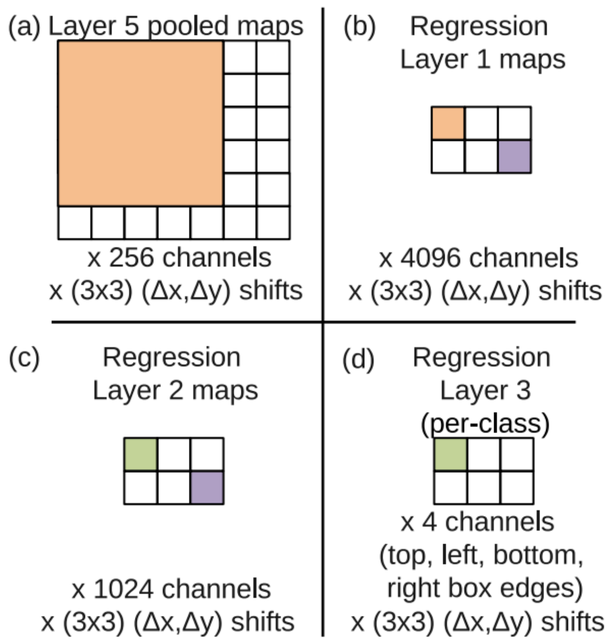


Figure 8: Application of the regression network to layer 5 features, at scale 2, for example. (a) The input to the regressor at this scale are 6x7 pixels spatially by 256 channels for each of the (3x3) Δ_x, Δ_y shifts. (b) Each unit in the 1st layer of the regression net is connected to a 5x5 spatial neighborhood in the layer 5 maps, as well as all 256 channels. Shifting the 5x5 neighborhood around results in a map of 2x3 spatial extent, for each of the 4096 channels in the layer, and for each of the (3x3) Δ_x, Δ_y shifts. (c) The 2nd regression layer has 1024 units and is fully connected (i.e. the purple element only connects to the purple element in (b), across all 4096 channels). (d) The output of the regression network is a 4-vector (specifying the edges of the bounding box) for each location in the 2x3 map, and for each of the (3x3) Δ_x, Δ_y shifts.

Localization: Bounding Boxes Produced By Regression

Localization: Combing Predictions

Algorithm:

(a) Assign to C_s the set of classes in the top k for each scale $s \in 1 \ldots 6$, found by taking the maximum detection class outputs across spatial locations for that scale.

(b) Assign to B_s the set of bounding boxes predicted by the regressor network for each class in C_s, across all spatial locations at scale s.

(c) Assign $B \leftarrow \bigcup_s B_s$

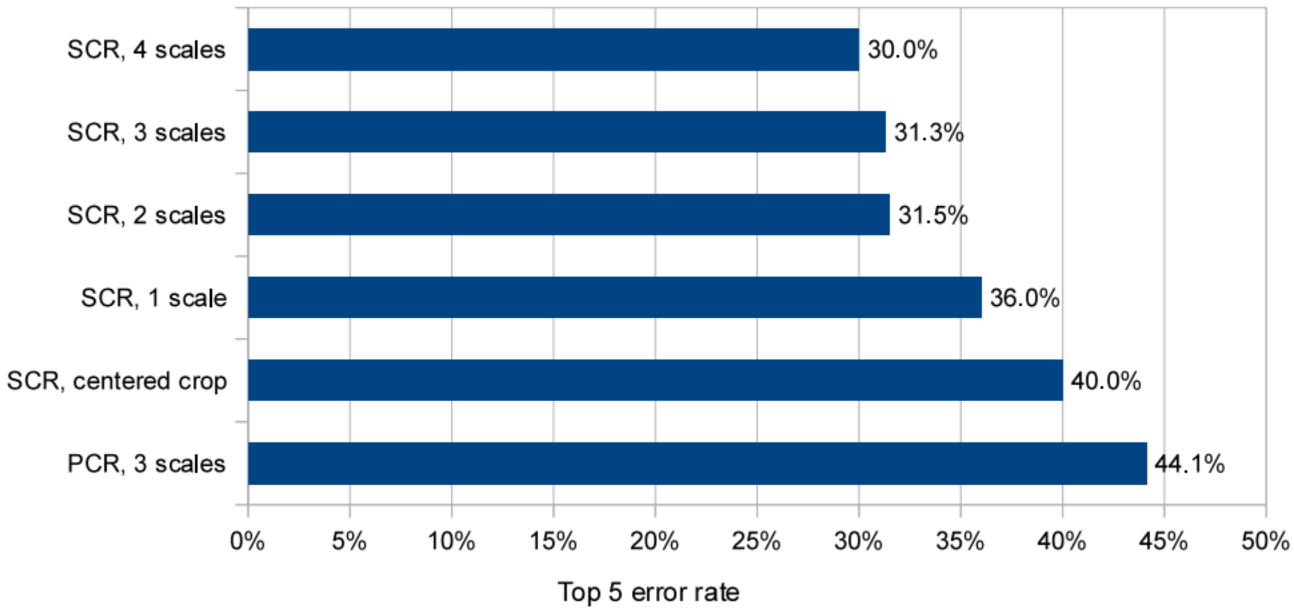
(d) Repeat merging until done:

(e) $(b_1^*, b_2^*) = \text{argmin}_{b_1 \neq b_2 \in B} \texttt{match\_score}(\mathtt{b_1}, \mathtt{b_2})$

(f) If $\texttt{match\_score}(b_1^*, b_2^*) > t$, stop.

(g) Otherwise, set $B \leftarrow B \backslash \{b_1^*, b_2^*\} \cup \texttt{box\_merge}(b_1^*, b_2^*)$

Localization: Results



Detection

- **Detection:**
  - 200 classes
  - Smaller objects than classification/localization
  - Any number of objects (including zero)
  - Penalty for false positives

Top predictions:
tv or monitor (confidence 11.5)
person (confidence 4.5)
miniskirt (confidence 3.1)

ILSVRC2012_val_00000119.JPEG

Groundtruth:
tv or monitor
tv or monitor (2)
tv or monitor (3)
person
remote control
remote control (2)

Differences Between Detection and Location

- Can now have many objects instead of just one
- Penalized for incorrect guesses
- Need to distinguish background from objects

Training Detector

- Almost identical to classification/localization training
- New class added – background
- Background class updated on the fly: extremely incorrect classifications are used to train background class

Detection Results

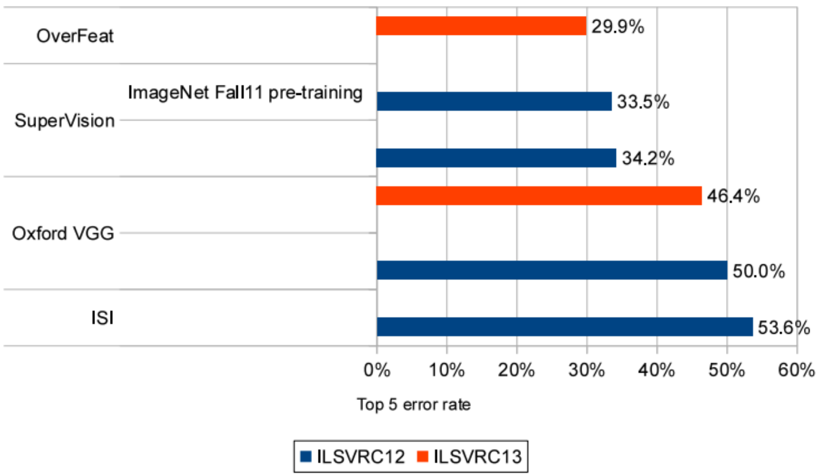
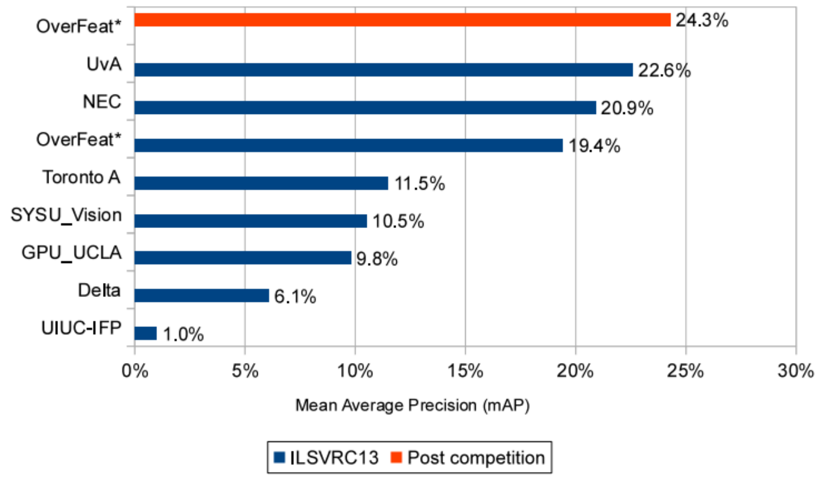


Figure 10: **ILSVRC12 and ILSVRC13 competitions results (test set).** Our entry is the winner of the ILSVRC13 localization competition with 29.9% error (top 5). Note that training and testing data is the same for both years. The OverFeat entry uses 4 scales and a single-class regression approach.



Conclusion

OverFeat provides a way to extract powerful CNN based features for image classification, localization and detection with high speed and precision

Thanks!