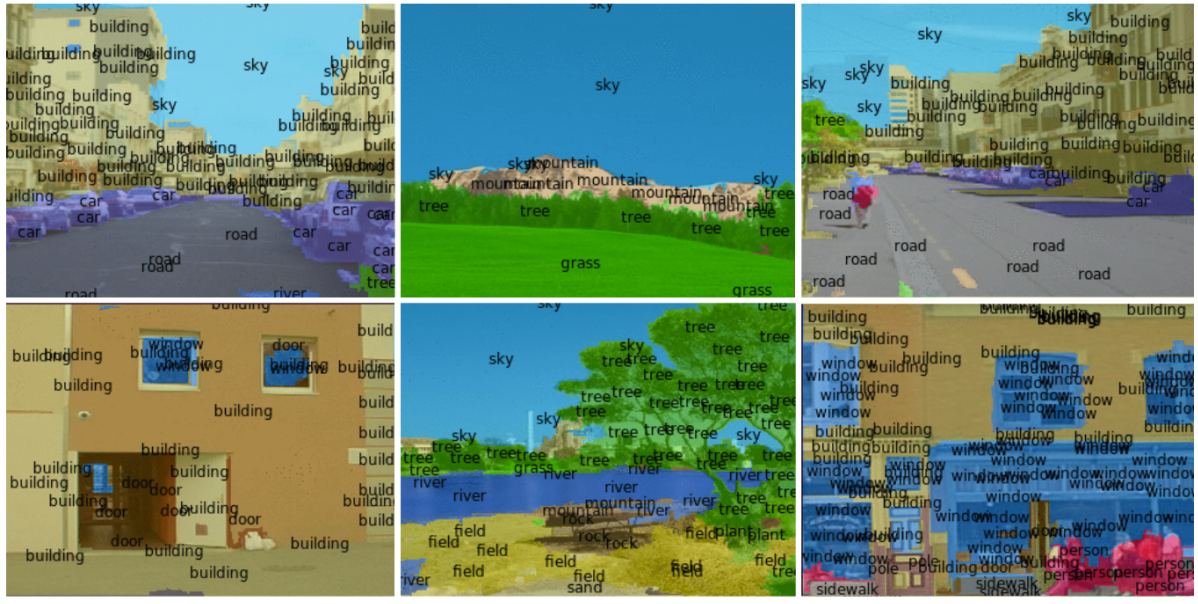
Recurrent Convolutional Neural Networks for Scene Labeling

Pedro O. Pinheiro, Ronan Collobert

Reviewed by Yizhe Zhang

August 14, 2015

Scene labeling task



- Scene labeling: assign a class label to each pixel in an image
- Involving detection, segmentation and multi-label recognition
- Traditional approaches: Graphical models (e.g. conditional random field)
- Limitation: computational cost at test time

Using convolutional network for scene labeling

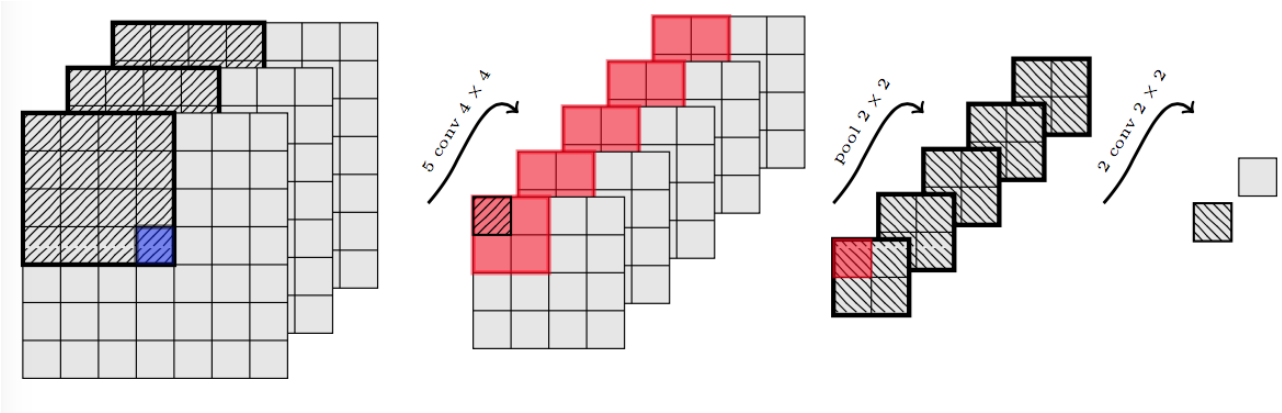


Figure: Convolutional network.

- Based on LeCun, Y. Learning hierarchical features for scene labeling. PAMI, 2013.
  - Uses a multiscale convolutional network to extract dense feature vectors that encode each patch
  - Average across superpixel/segmentation
  - Fast at test time
- Key point of this paper: using recurrent framework, directly processing the raw pixels, does not require any engineered features or segmentation

CNN for Scene Labeling

- Input: image patches $I_{i,j,k}$, centered at position (i,j) in the kth image
- Output: for each patch, obtain a vector of size N (number of total class) indicating the score for each class.
- For mth layer,
  $\boldsymbol{W}_m$: Toeplitz matrices as convolutional filters of this layer. $\boldsymbol{H}_m$: latent representation of original patch. $H_0 = I_{i,j,k}$ is the original patch

$$\boldsymbol{H}_m = tanh(pool(\boldsymbol{W}_m \boldsymbol{H}_{m-1} + b_m))$$

$$f(I_{i,j,k}) = \boldsymbol{W}_M \boldsymbol{H}_{M-1}$$

- Class probabilities are given by softmax function
- Model is trained by SGD (stochastic gradient descent) with a fixed learning rate

Recurrent Network Approach

- $\boldsymbol{F}^p = [f(\boldsymbol{F}^{p-1}), I^p_{i,j,k}]$ are $N+3$ feature maps of original image fed into next layer
- $f(\boldsymbol{F}^{p-1})$:N (number of class) planes formed by collecting prediction scores for all patch
- $I^p_{i,j,k}$: rescaled image (RGB) to match the size with $f(\boldsymbol{F}^{p-1})$

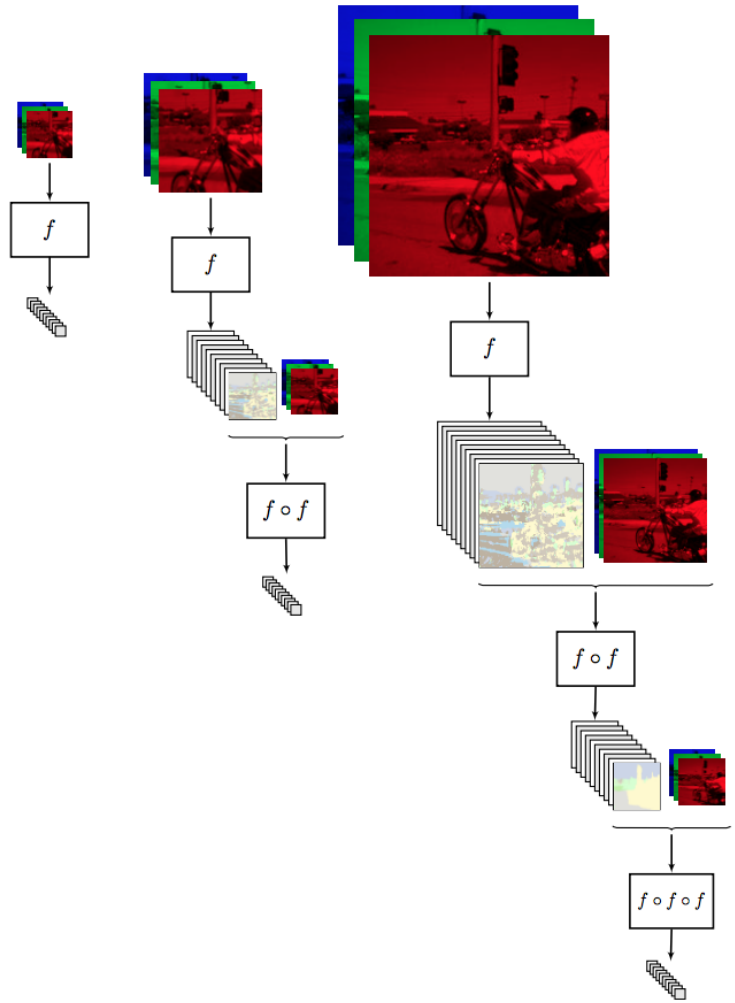


Figure: System architecture

Recurrent Network Approach (Cont'd)

- The system is trained by maximizing $L(f) + L(f \circ f) + \ldots + L(f \circ^P f)$
  - Can learn to correct its own mistakes made by previous layer
  - Can learn label dependencies (predictions for neighborhood patcheds are use for next layer)
- Inference detail: randomly alternating the maximization of each likelihood.
- Gradient is computed by BPTT (backpropagation through time)

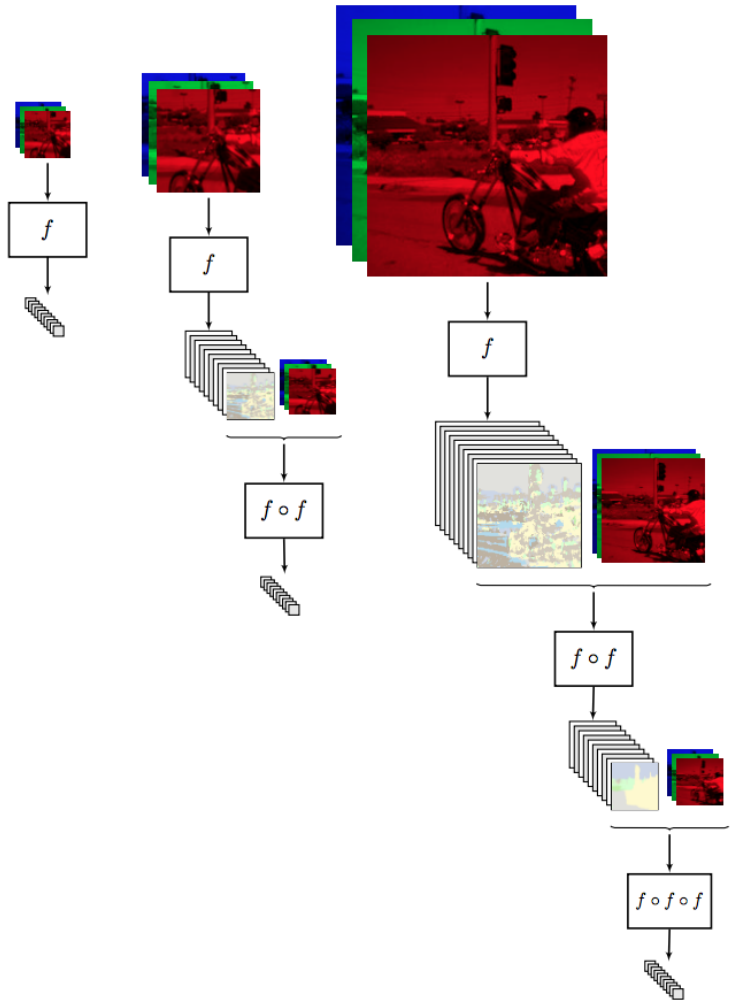


Figure: System architecture

Capacity control

MEANS	CAPACITY CONTROL	SPEED
GRAPHICAL MODEL	–	SLOW
MULTISCALE	SCALE DOWN INPUT IMAGE	FAST
LARGE INPUT PATCHES	INCREASE POOLING	SLOW
	RECURRENT ARCHITECTURE	FAST

- Avoid overfitting the data with too large model
- One possible way: increase the pooling size to reduce the overall number of parameters
  - Decrease the label output resolution
- Recurrent approach: shared parameters at various depths

Avoid downscaling label planes

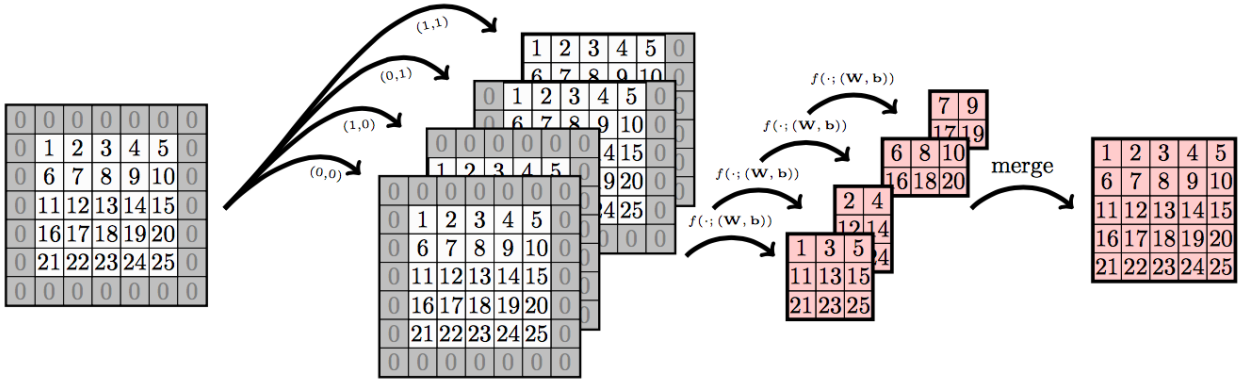


Figure: Example: a single 2×2 pooling layer with shifted pooling operation

- Pooling yielding low resolution. To achieve pixel level label, most CNN upscale the label plane to input size
- Approach: Feeding to the pooling layer with several versions of shifted input image
- Downscaled predicted label planes (red) are then merged to get back the full resolution label plane
- Improving classification performance, trade off between resolution and speed

Avoid downscaling label planes (Cont'd)

Output Resolution	Computing Time Per Image	Pixel Accuracy
1/8	0.20s	78.4%
1/4	0.70s	79.3%
1/2	2.15s	79.8%
1/1	10.68s	80.2%

Figure: trade off between resolution and speed

- Pooling yielding low resolution. To achieve pixel level label, most CNN upscale the label plane to input size
- Approach: Feeding to the pooling layer with several versions of shifted input image
- Downscaled predicted label planes (red) are then merged to get back the full resolution label plane
- Improving classification performance, trade off between resolution and speed

Experiments

- Two datasets: the Stanford Background and the SIFT Flow Dataset
- Stanford dataset: 715 images (320×240), 8 classes
- SIFT Flow Dataset: 2688 images (256×256), 33 classes
- 5-fold cross-validation

Comparison with other model

METHOD	PIXEL/CLASS ACCURACY (%)	COMPUTING TIME (S)
(GOULD ET AL., 2009)	76.4 / -	10 TO 600
(TIGHE & LAZEBNIK, 2010)	77.5 / -	10 TO 300
(MUNOZ ET AL., 2010)‡	76.9 / 66.2	12
(KUMAR & KOLLER, 2010)	79.4 / -	< 600
(SOCHER ET AL., 2011)	78.1 / -	?
(LEMPITSKY ET AL., 2011)	81.9 / 72.4	> 60
(FARABET ET AL., 2013)⋆	78.8 / 72.4	0.6
(FARABET ET AL., 2013)†	81.4 / 76.0	60.5
PLAIN CNN_1	79.4 / 69.5	15
CNN_2 (o^1)	67.9 / 58.0	0.2
$RCNN_2$ (o^2)	79.5 / 69.5	2.6
CNN_3 (o^1)	15.3 / 14.7	0.06
$RCNN_3$ (o^2)	76.2 / 67.2	1.1
$RCNN_3$ 1/2 RESOLUTION (o^3)	79.8 / 69.3	2.15
$RCNN_3$ 1/1 RESOLUTION (o^3)	80.2 / 69.9	10.7

Figure: Results for Stanford Background datasets

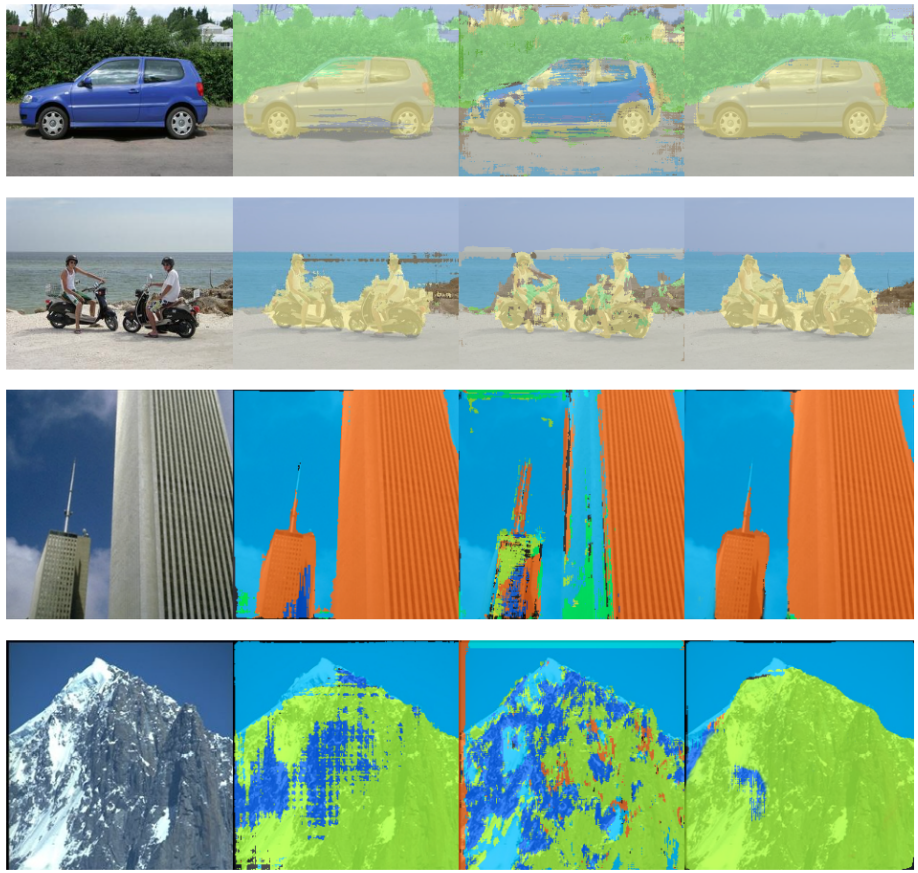
- Two measures: pixel level and class level
- Plain CNN_1: 133×133 input patches. 2 convolution&pooling layers
- $rCNN_2(o^2)$, $rCNN_3(o^2)$: 2 layers recurrent convolutional networks
- $rCNN_3(o^3)$: 3 layers recurrent convolutional networks

Comparison with other model (Cont'd)

Table 4. Pixel and averaged per class accuracy of other methods and our proposed approaches on the SIFT Flow Dataset. For recurrent networks, $\circ^n$ indicates the number of compositions.

METHOD	PIXEL/CLASS ACCURACY (%)
(LIU ET AL., 2011)	76.67 / -
(TIGHE & LAZEBNIK, 2013)	77.0 / 30.1
(FARABET ET AL., 2013)	78.5 / 29.6
PLAIN CNN_1	76.5 / 30.0
CNN_2 ($\circ^1$)	51.8 / 17.4
$RCNN_2$ ($\circ^2$)	76.2 / 29.2
$RCNN_3$ ($\circ^2$)	65.5 / 20.8
$RCNN_3$ ($\circ^3$)	77.7 / 29.8

Inference results



- The second column: output of the "plain CNN_1"
- Third column : results of rCNN_2 with one layer f
- Last column : result of rCNN_2 with the composition of two layers $f \circ f + f$
- Conclusion: the network learns itself how to correct its own label