Fourier-Based Fast Object Detection

François Fleuret

Joint work with Charles Dubout

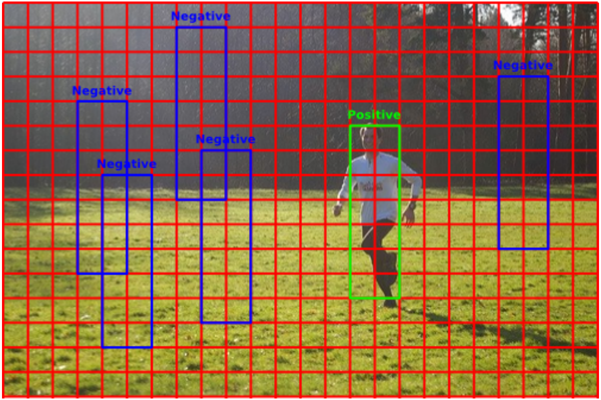
Computer Vision and Learning group

idiap RESEARCH INSTITUTE

EPFL ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Object detection
Sliding window

Object detection
Sliding window



- Applies a binary classifier at every image position and scale
- Detection transformed into an iterated binary classification

The core component of this approach is a binary classifier trained from data.

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

- Boosting

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

- Boosting
- Random forests (of decision trees)

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

- Boosting
- Random forests (of decision trees)
- Support-Vector machines (SVMs)

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

- Boosting
- Random forests (of decision trees)
- Support-Vector machines (SVMs)
- Multi-layer perceptrons and “deep” convolution networks

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

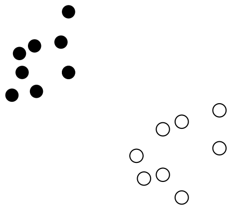
- Boosting
- Random forests (of decision trees)
- Support-Vector machines (SVMs)
- Multi-layer perceptrons and “deep” convolution networks

We will focuse on SVMs in the rest of this presentation.

The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

- Boosting
- Random forests (of decision trees)
- Support-Vector machines (SVMs)
- Multi-layer perceptrons and “deep” convolution networks

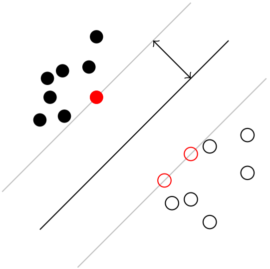
We will focuse on SVMs in the rest of this presentation.



The core component of this approach is a binary classifier trained from data. Many algorithms fit the bill

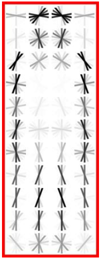
- Boosting
- Random forests (of decision trees)
- Support-Vector machines (SVMs)
- Multi-layer perceptrons and “deep” convolution networks

We will focuse on SVMs in the rest of this presentation.

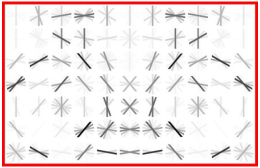


SVM, HOG, AND DPM

Pedestrian template



Bicycle template

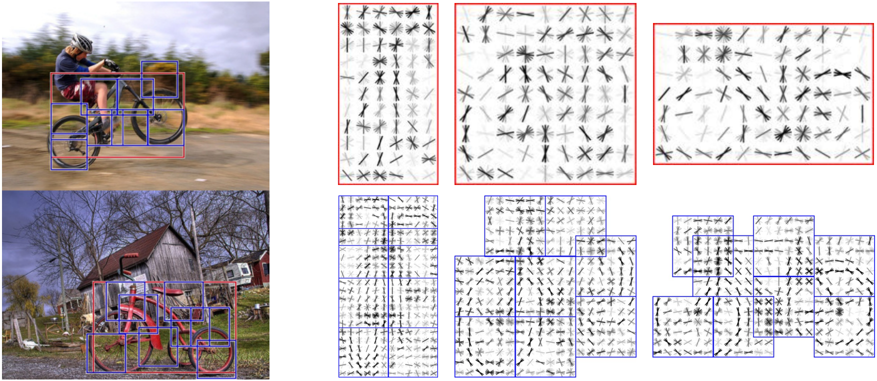


Objects are image positions on the HOG grid: $\mathit{score}_{\mathrm{w}}(\mathrm{x}) = \langle \mathrm{w}, \mathrm{x} \rangle$, where x is the vector of features extracted from the subwindow at the position of interest of size same as w.

The combination of HOG and a linear SVM is a powerful detector.

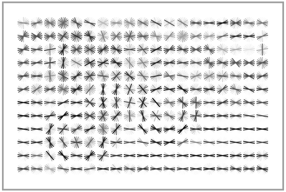
Deformable Part Model

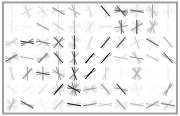
The combination of HOG and a linear SVM is a powerful detector.
It is also the building brick of the “Deformable part model”.

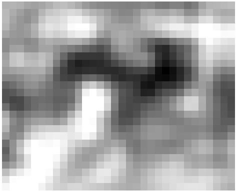


Deformable Part Model

Root detection

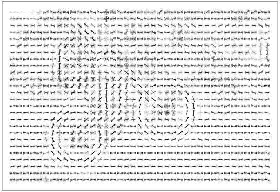


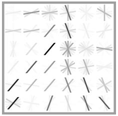


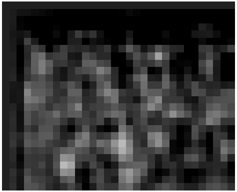
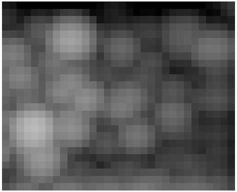
$S_0 =$ 

Deformable Part Model

Part detection

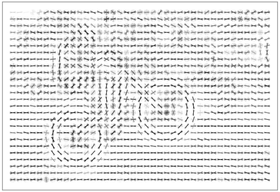


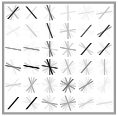


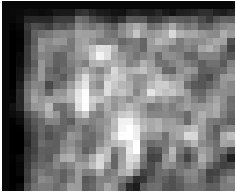
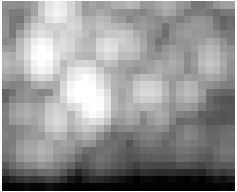
$S_1 =$  $T_1(S_1) =$ 

Deformable Part Model

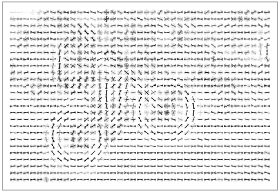
Part detection

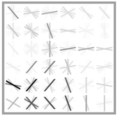



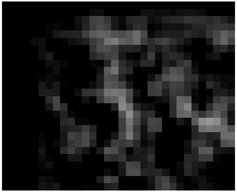
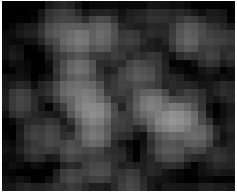


$S_2 =$  $T_2(S_2) =$ 

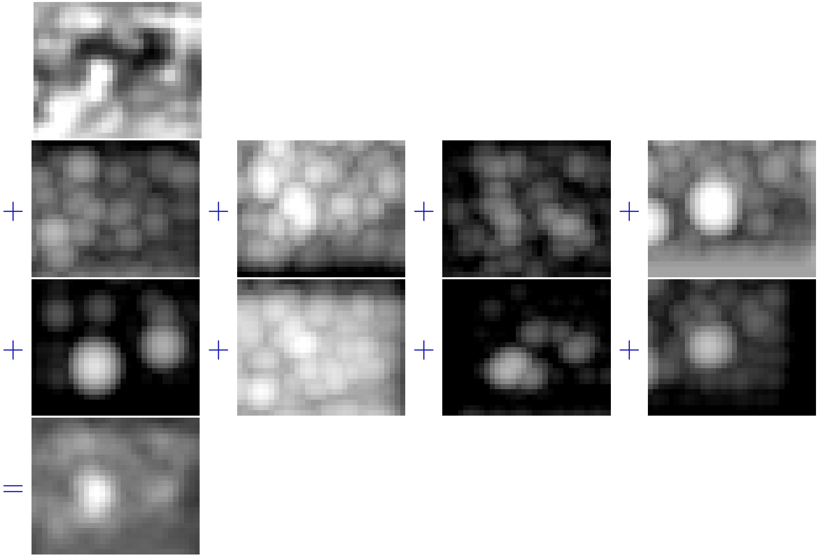
Deformable Part Model
Part detection



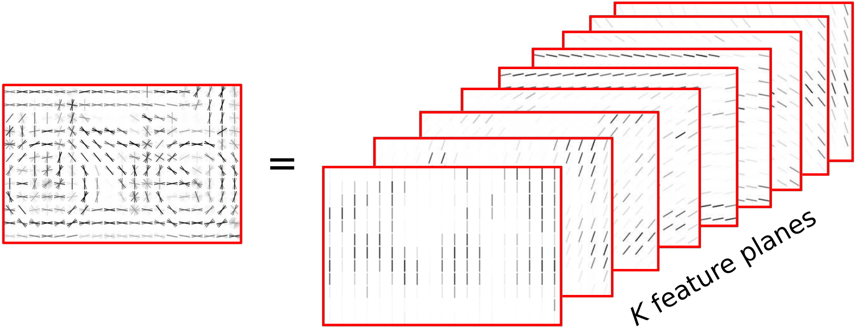
$S_3 =$  $T_3(S_3) =$ 

Deformable Part Model
Final score



Computational challenge

Cost of linear filters
Challenge



The HOG features can be seen as organized in planes, containing distinct features from each grid cell.

Typical settings uses 6 mixtures $\times$ 9 parts = 54 linear filters per object class.

Typical settings uses 6 mixtures $\times$ 9 parts = 54 linear filters per object class.

For the standard PASCAL data-set, we have to detect 20 different classes, which correspond to a reasonable practical situation (car, pedestrian, bicycle, dog, etc.)

Typical settings uses 6 mixtures $\times$ 9 parts = 54 linear filters per object class.

For the standard PASCAL data-set, we have to detect 20 different classes, which correspond to a reasonable practical situation (car, pedestrian, bicycle, dog, etc.)

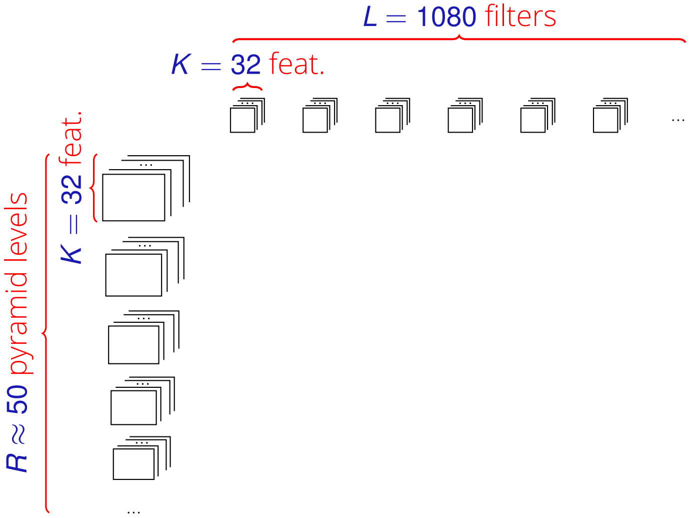
Total of 1080 filters

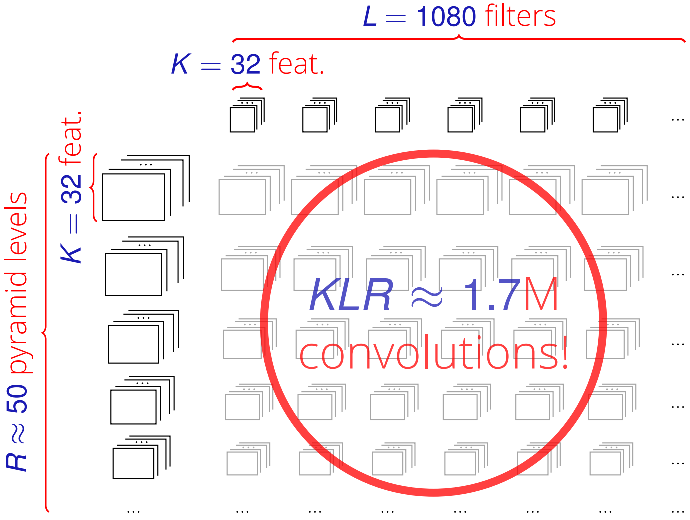
Typical settings uses 6 mixtures $\times$ 9 parts = 54 linear filters per object class.

For the standard PASCAL data-set, we have to detect 20 different classes, which correspond to a reasonable practical situation (car, pedestrian, bicycle, dog, etc.)

Total of 1080 filters and each filter is over 32 feature channels!

Cost of linear filters
Challenge

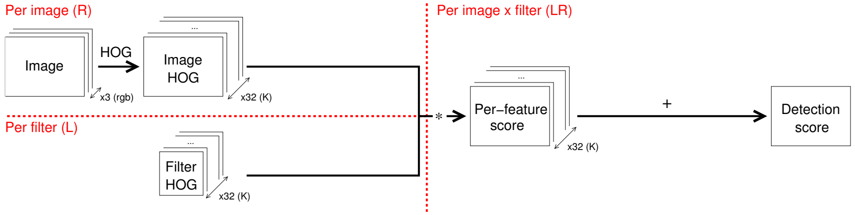



$L = 1080$ filters
$K = 32$ feat.
$K = 32$ feat.
$R \approx 50$ pyramid levels
$KLR \approx 1.7$M convolutions!

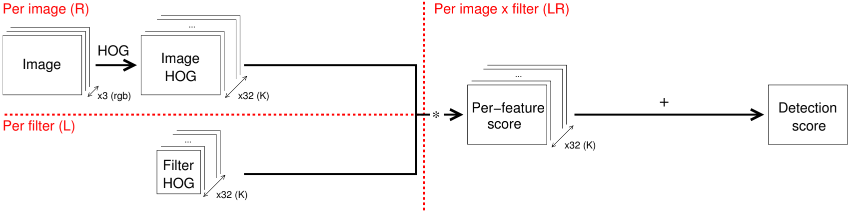
FFT AND MAKING THINGS WORK

Cost of linear filters

Standard convolution process



Cost of linear filters
Standard convolution process

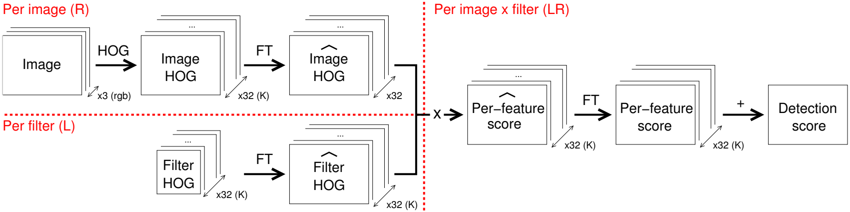


The computational cost to convolve a HOG image of size $M \times N$ with L filters of size $P \times Q$ across K features is:

$$C_{\text{std}} = \mathcal{O}(KLMNPQ)$$

Cost of linear filters

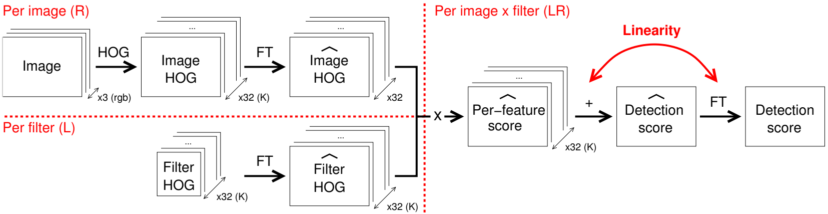
Fourier based convolutions



The computational cost to convolve a HOG image of size $M \times N$ with L filters of size $P \times Q$ across K features is:

$$C_{\text{FFT}} = \underbrace{\mathcal{O}(KMN \log MN)}_{\text{Forward FFTs}} + \underbrace{\mathcal{O}(KLMN)}_{\text{Multiplications}} + \underbrace{\mathcal{O}(KLMN \log MN)}_{\text{Inverse FFTs}}$$

Cost of linear filters
Fourier based convolutions



The computational cost to convolve a HOG image of size $M \times N$ with L filters of size $P \times Q$ across K features is:

$$C_{\text{opt}} = \underbrace{\mathcal{O}(KMN \log MN)}_{\text{Forward FFTs}} + \underbrace{\mathcal{O}(KLMN)}_{\text{Multiplications}} + \underbrace{\mathcal{O}(\cancel{K}LMN \log MN)}_{\text{Inverse FFTs}}$$
$$\approx \mathcal{O}(KLMN)$$

Cost of linear filters

What are typical numbers

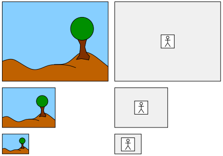
- $K = 32$ (number of HOG features)
- $L = 54$ (number of filters)
- $M \times N = 64 \times 64$ (size of the pyramid level)
- $P \times Q = 6 \times 6$ (size of the filters)

- $K = 32$ (number of HOG features)
- $L = 54$ (number of filters)
- $M \times N = 64 \times 64$ (size of the pyramid level)
- $P \times Q = 6 \times 6$ (size of the filters)

$$
\begin{aligned}
C_{\mathrm{std}} &\approx 2KLMNPQ && \approx 490 \text{ MFlop} \\
C_{\mathrm{FFT}} &\approx 3KLMN + 2.5(K + KL)MN \log_2 MN && \approx 230 \text{ MFlop} \\
C_{\mathrm{opt}} &\approx 4KLMN + 2.5(K + L)MN \log_2 MN && \approx 37 \text{ MFlop}
\end{aligned}
$$

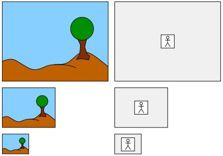
A gain by a factor 13 compared to the standard process, and 6 compared to the standard Fourier one!

To use the FFT the image and the filter need to be of the same size.

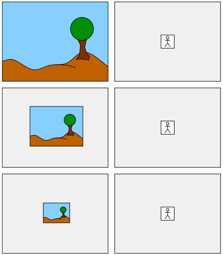


Memory inefficient

To use the FFT the image and the filter need to be of the same size.

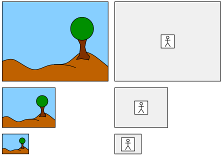


Memory inefficient

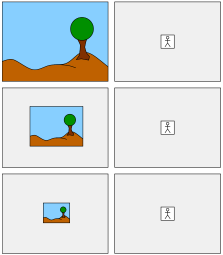


Computationally inefficient

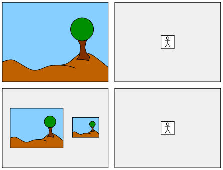
To use the FFT the image and the filter need to be of the same size.



Memory inefficient

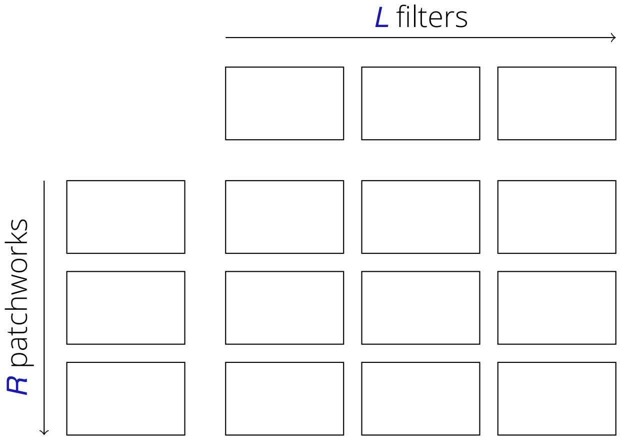


Computationally inefficient



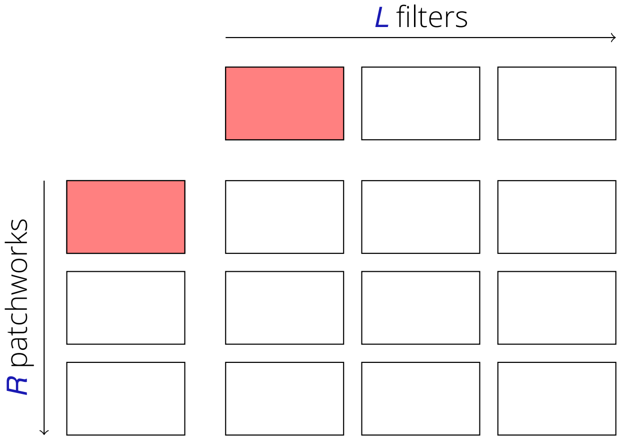
Best of both worlds

Cost of linear filters
Cache violations



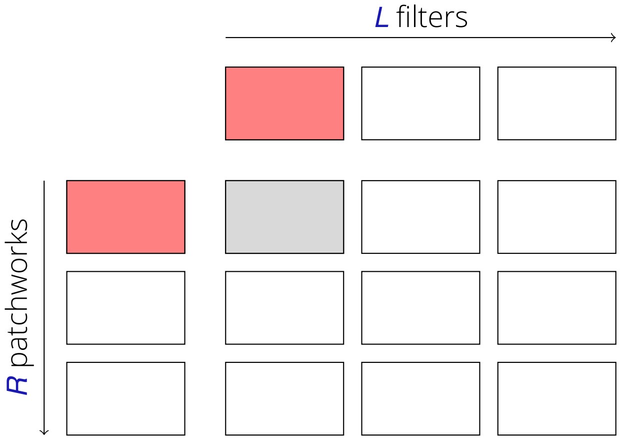
Cost of linear filters

Cache violations



Read 2 into cache

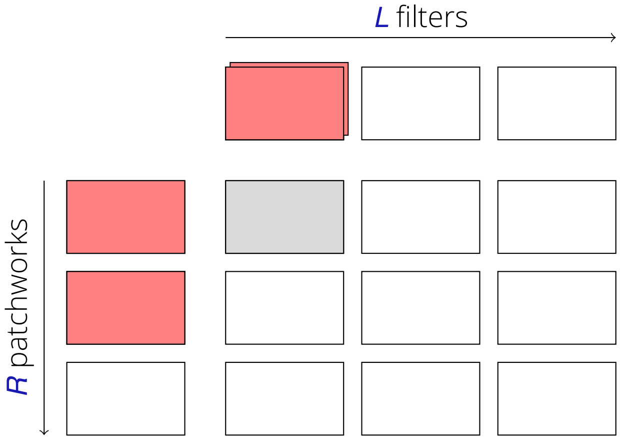
Cost of linear filters
Cache violations



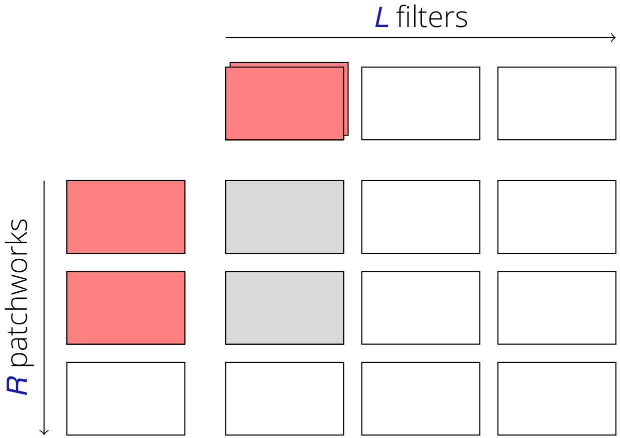
Read 2 into cache ⇒ compute 1.

Cost of linear filters

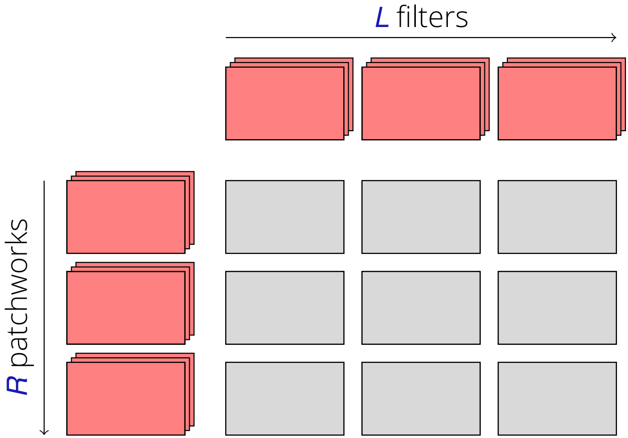
Cache violations



Read **2** into cache ⇒ compute **1**.



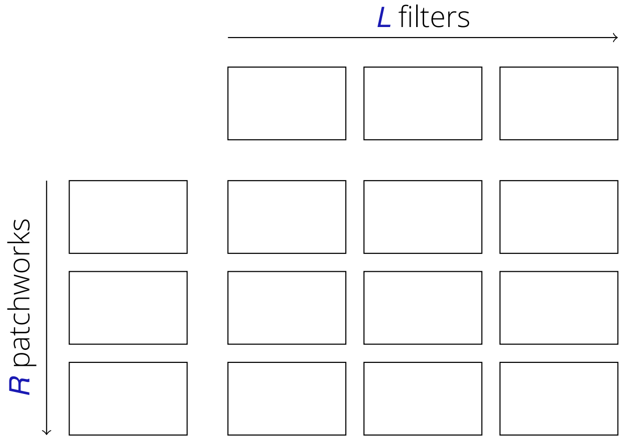
Read 2 into cache ⇒ compute 1.



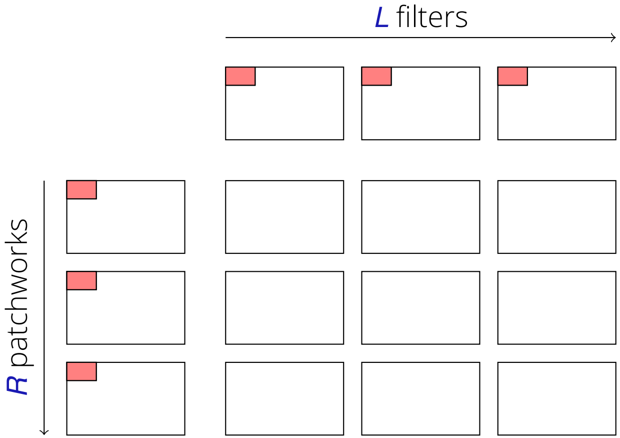
Read $2LR$ into cache $\Rightarrow$ compute LR.

Cost of linear filters

Cache violations

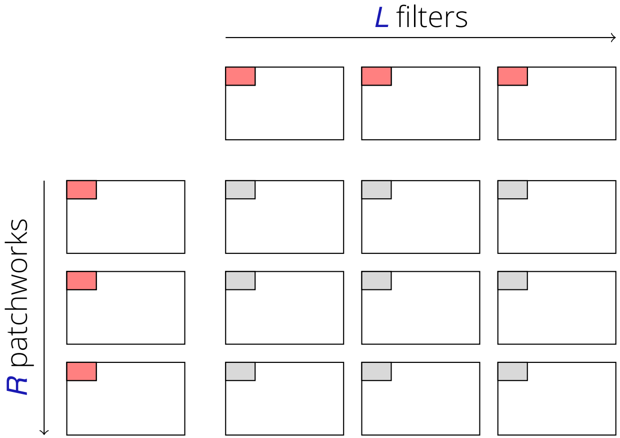


Cost of linear filters
Cache violations



Read $(L + R)\frac{\epsilon}{L+R} = \epsilon$ into cache

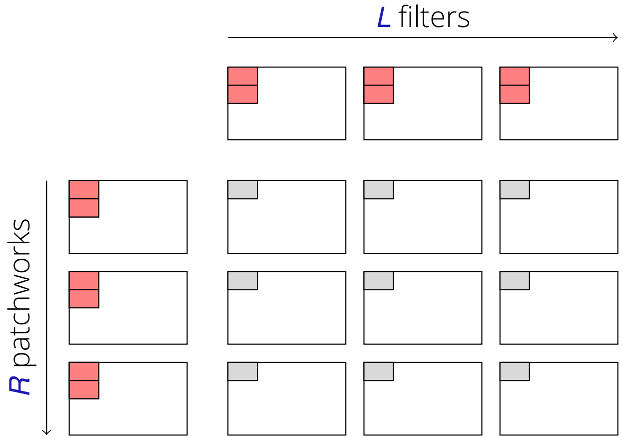
Cost of linear filters
Cache violations



Read $(L+R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

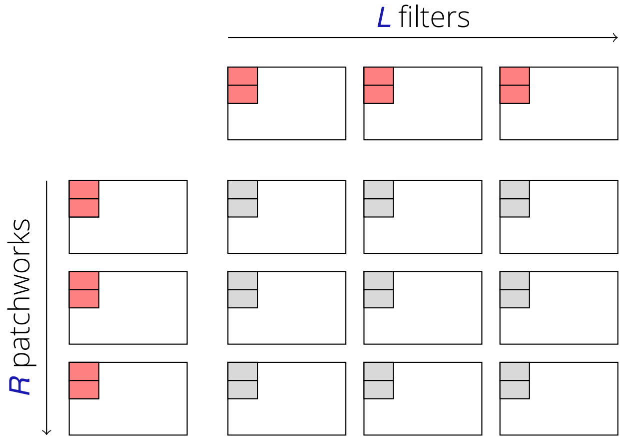
Cost of linear filters

Cache violations



Read $(L + R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

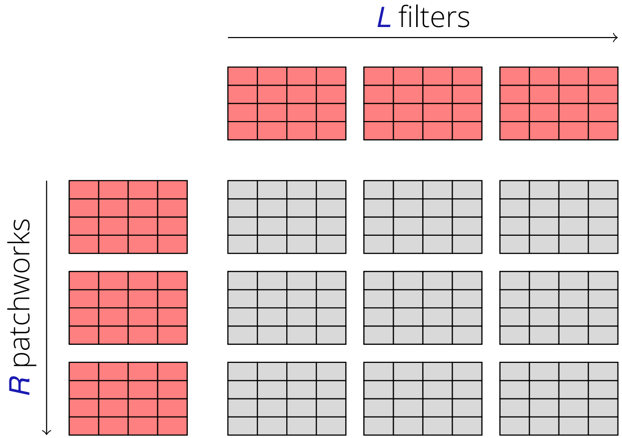
Cost of linear filters
Cache violations



Read $(L+R)\frac{\epsilon}{L+R} = \epsilon$ into cache $\Rightarrow$ compute $LR\frac{\epsilon}{L+R}$.

Cost of linear filters

Cache violations



Read $L + R$ into cache $\Rightarrow$ compute LR.

Table: Pascal VOC 2007 challenge convolution time and speedup

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
V4 (ms)	409	437	403	414	366	439	352	432	417	429	450
Ours (ms)	55	56	53	56	57	56	54	56	56	57	57
Speedup (x)	7.4	7.8	7.6	7.4	6.4	7.9	6.5	7.7	7.5	7.5	8.0

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
V4 (ms)	445	439	429	379	358	351	425	458	433	**413**
Ours (ms)	57	59	57	54	54	55	57	58	55	**56**
Speedup (x)	7.8	7.5	7.6	7.0	6.6	6.4	7.4	7.9	7.9	**7.4**

Table: Pascal VOC 2007 challenge convolution time and speedup

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
V4 (ms)	409	437	403	414	366	439	352	432	417	429	450
Ours (ms)	55	56	53	56	57	56	54	56	56	57	57
Speedup (x)	7.4	7.8	7.6	7.4	6.4	7.9	6.5	7.7	7.5	7.5	8.0

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
V4 (ms)	445	439	429	379	358	351	425	458	433	**413**
Ours (ms)	57	59	57	54	54	55	57	58	55	**56**
Speedup (x)	7.8	7.5	7.6	7.0	6.6	6.4	7.4	7.9	7.9	**7.4**

- Error rate: identical to the baseline (32.3% AP)

Table: Pascal VOC 2007 challenge convolution time and speedup

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table
V4 (ms)	409	437	403	414	366	439	352	432	417	429	450
Ours (ms)	55	56	53	56	57	56	54	56	56	57	57
Speedup (x)	7.4	7.8	7.6	7.4	6.4	7.9	6.5	7.7	7.5	7.5	8.0

	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mean
V4 (ms)	445	439	429	379	358	351	425	458	433	**413**
Ours (ms)	57	59	57	54	54	55	57	58	55	**56**
Speedup (x)	7.8	7.5	7.6	7.0	6.6	6.4	7.4	7.9	7.9	**7.4**

- Error rate: identical to the baseline (32.3% AP)
- Numerical accuracy: better than the baseline ($1.8 \cdot 10^{-8}$ vs. $2.4 \cdot 10^{-8}$ MAE)

- Part-based models obtain state-of-the-art performance at the price of a huge number of convolutions
- The FT is linear, enabling one to do the addition of the convolutions across feature planes in Fourier space
- The computational cost becomes invariant to the filters' sizes, resulting in a big speedup ($\times$7.4 in experiments)

The end

francois.fleuret@idiap.ch