Hashing Away Dot-products

Ishan Misra

Carnegie Mellon University

MOTIVATION

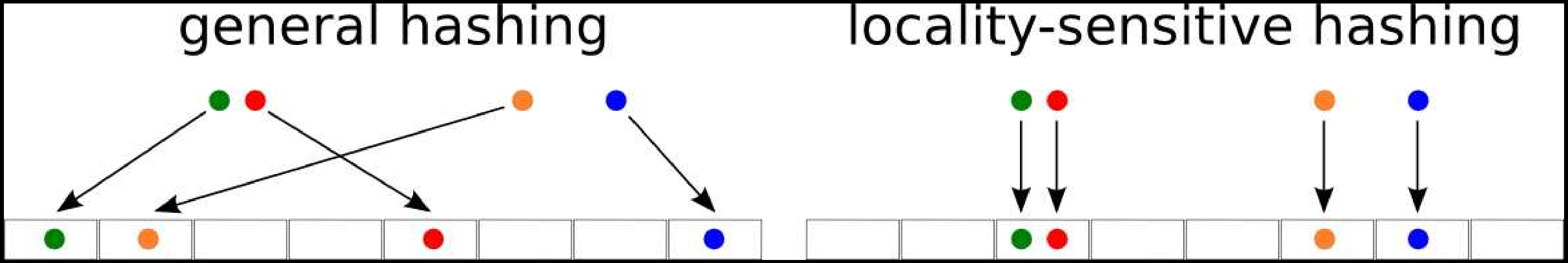
Dot products

- Used for computing similarity between vectors

- Object Detection in images
  - many dot products (~10^5) per image
  - High dimensional (3100) vectors
  - Bottleneck …

Revision: Locality Sensitive Hashing

- Similar vectors => Similar hash values



Paper - Winner Takes All (WTA) Hashing

The Power of Comparative Reasoning

Jay Yagnik, Dennis Strelow, David A. Ross, Ruei-sung Lin
{jyagnik, strelow, dross, rslin}@google.com

Comparative Reasoning

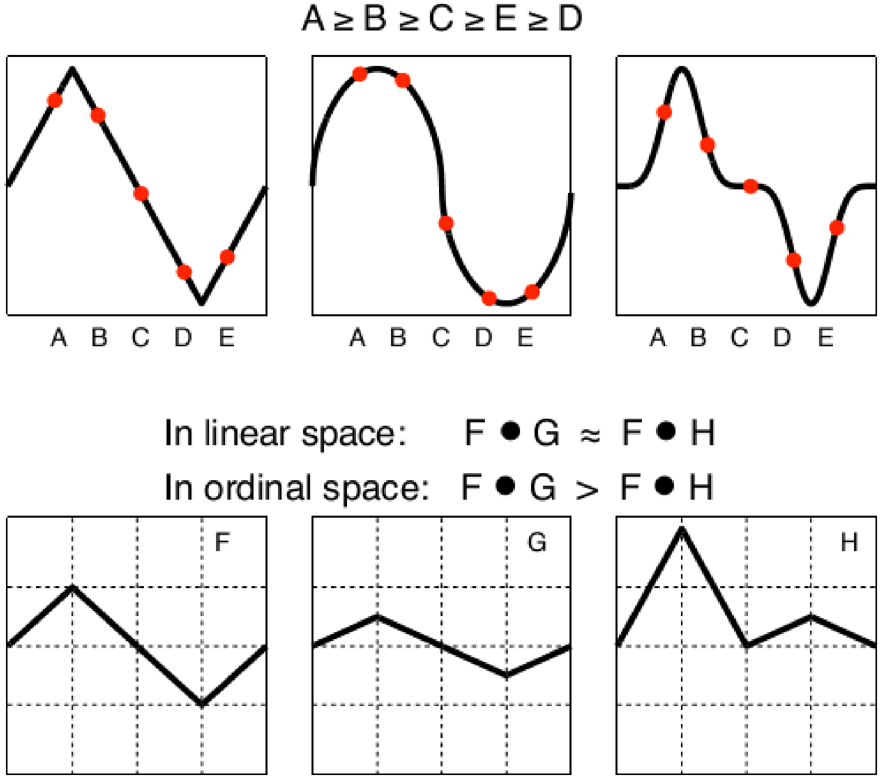
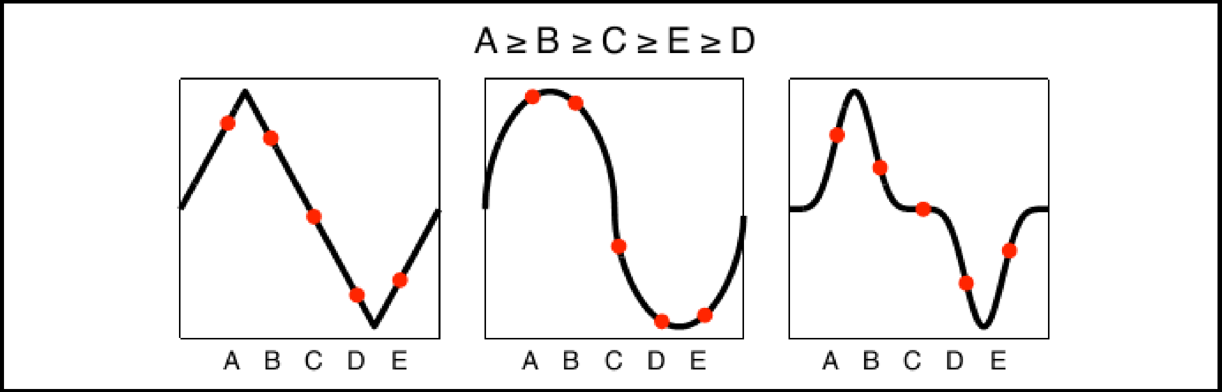


Figure 1. An ordinal measurement $A \geq B \geq C \geq E \geq D$ is robust to variations in individual filter values (top row). Ordinal measures of similarity capture filter response differences not reflected in linear measures (dot product) of similarity (bottom row).

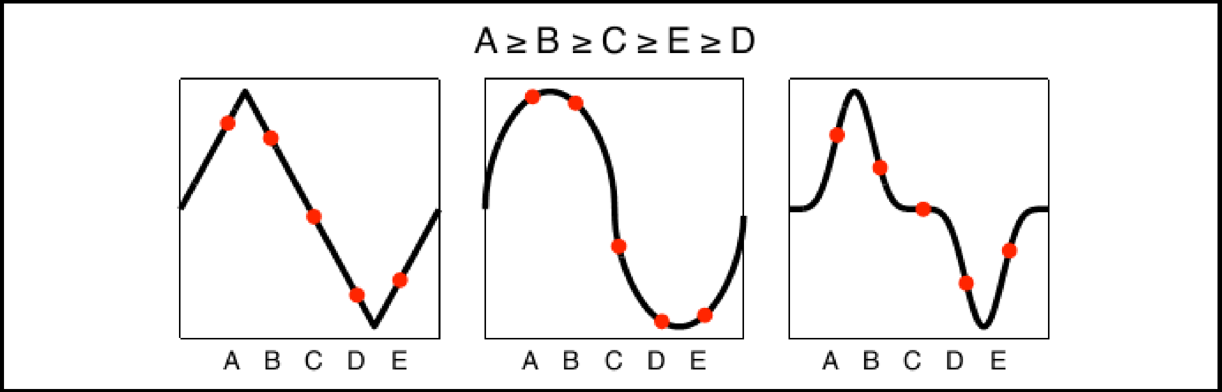
Intuition

- Small differences in absolute value of filter don’t matter as long as “nature of filter” is the same



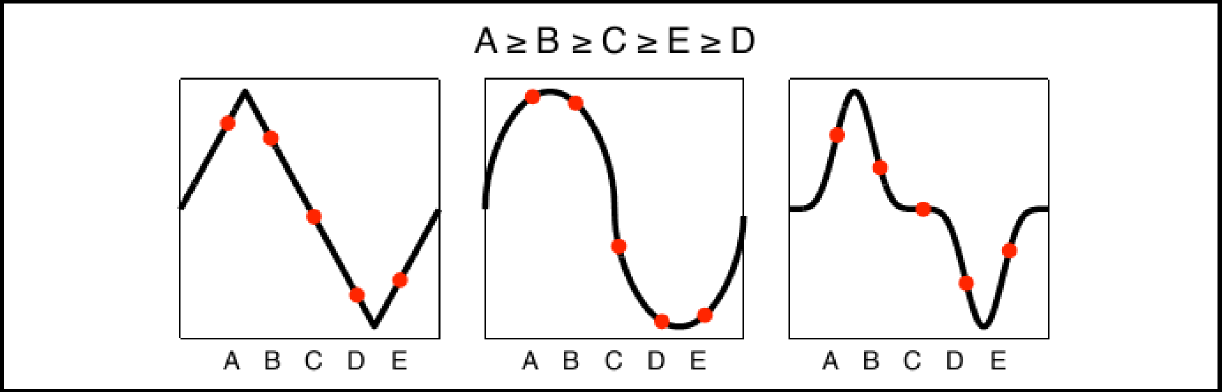
Intuition

- Capture “nature of the filter” using a set of inequalities
  - A>=B
  - B>=C



Intuition

- Comparing two signals is comparing their set of inequalities
  - Expensive



WTA Hashing

5	10	2	8	16	7	Input N – dim vector
7	16	5	10	8	2	Permute elements *P=[6,5,1,2,4,3]*
			10	8	2	Choose first $K=3$ elements
		Hash of vector	1	0	0	Set index of max=1, rest 0

WTA Hashing – another view

- Take random projection of the vector

- Compute index of the dimension with max magnitude

Window Parameter

- Effect of window parameter *K*
  - *K = length of signal*
  - We are just storing the index of global max
    - Exactly equal to “Max-hash”

- In general, smaller *K* captures more pair-wise ordering

- Larger *K* captures a more “global max”

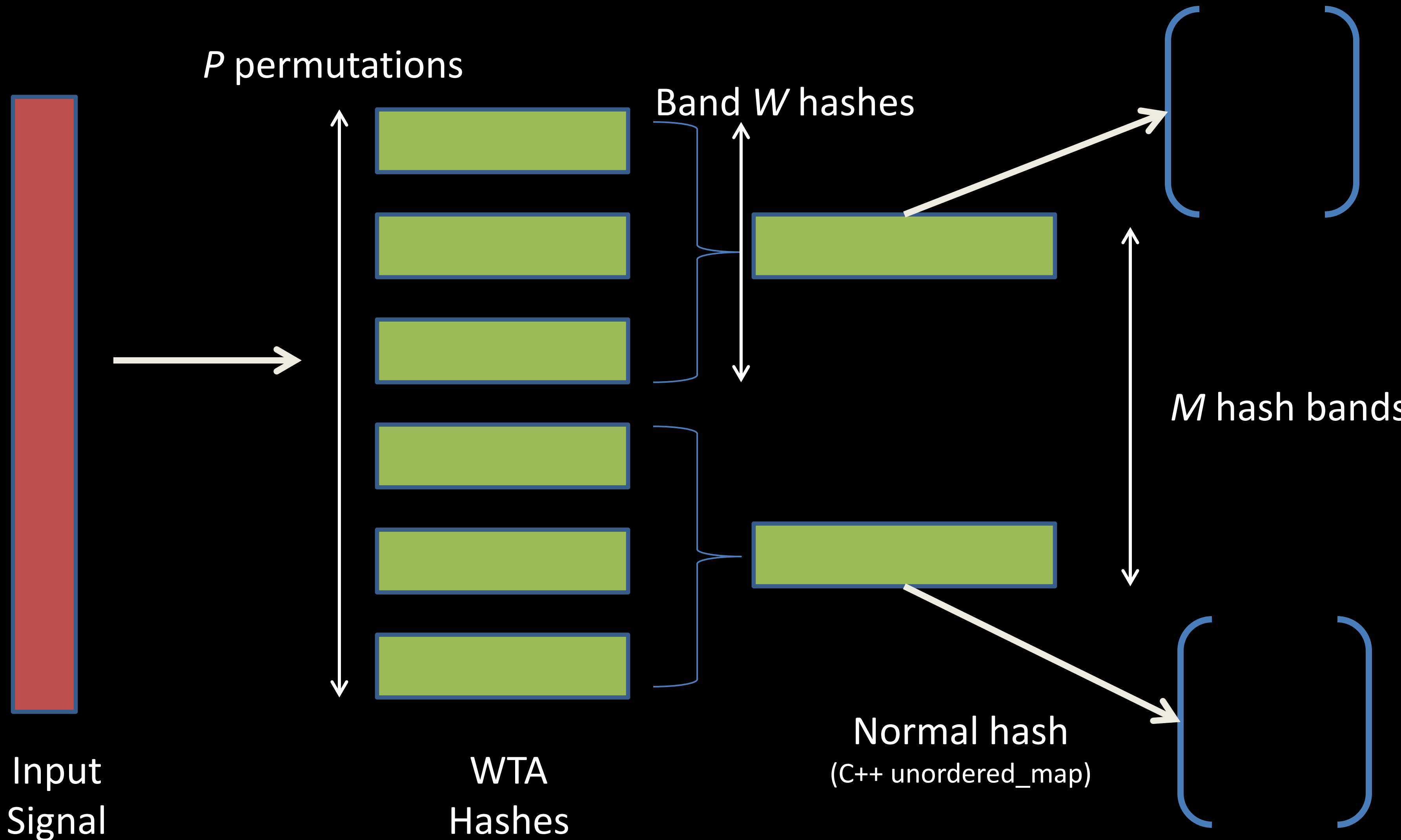
So if two hashes match?

- If two hashes of length K match
  – $K-1$ inequalities in both signals have matched
- By computing and matching hash in $O(K)$ we can determine $K-1$ inequalities
- Notice that hash is sparse and thus allows for efficient storage/computation

- One hash isn’t enough

- Need more “random projections”

Multi-band WTA



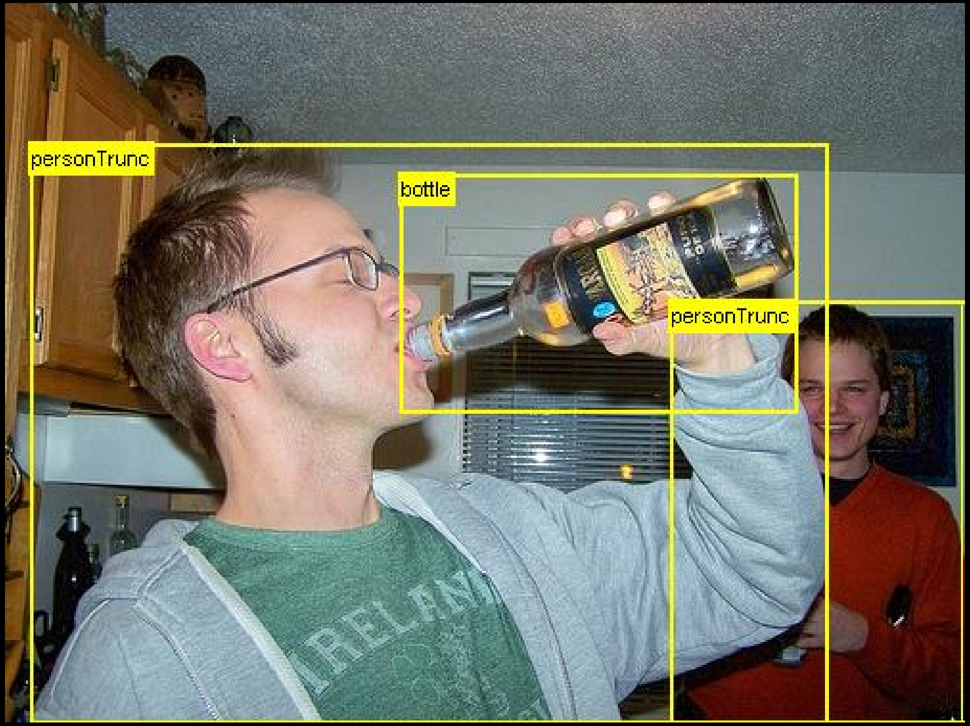
Multi-band WTA

- $P = 2400$ (number of permutations)
- $K = 16$ (window size)
- $M = 600$ (bands)
- $W = N/M = 4$ (number of WTA-hashes banded together)
- Each hash takes $W \log_2 K$ bits = 16 bits to store in memory => *C++ unordered_map*

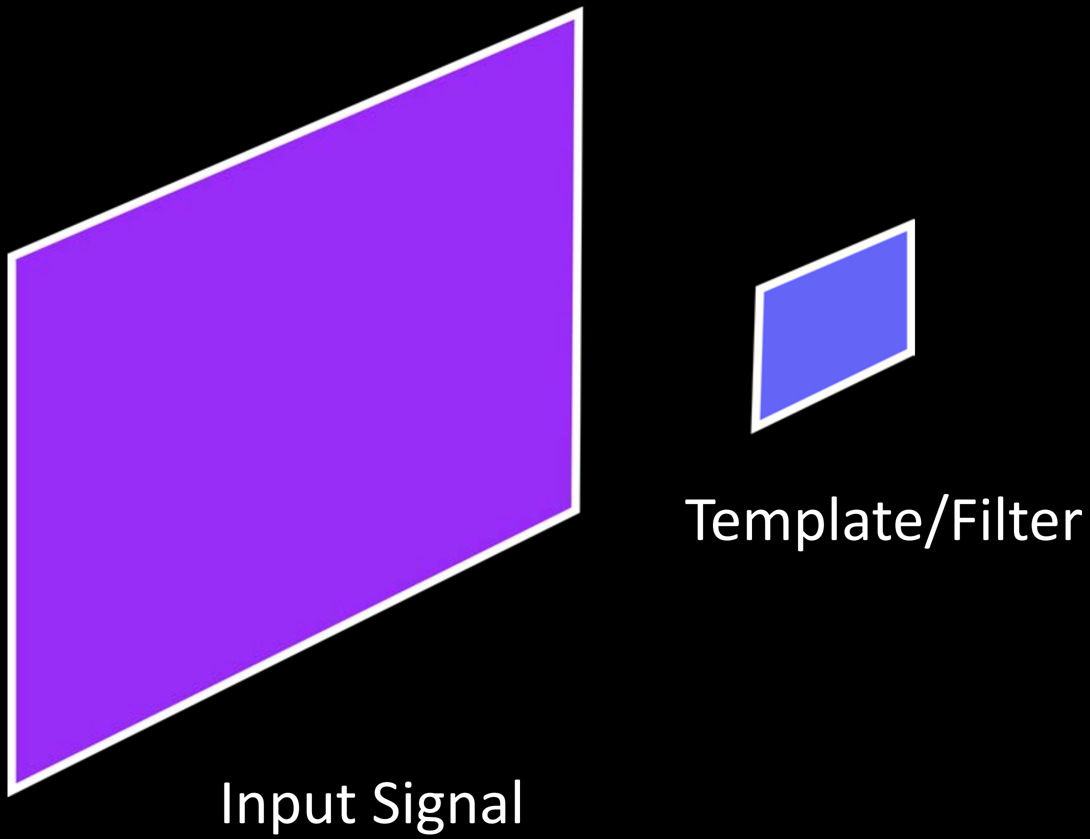
OBJECT DETECTION

Why Object Detection?

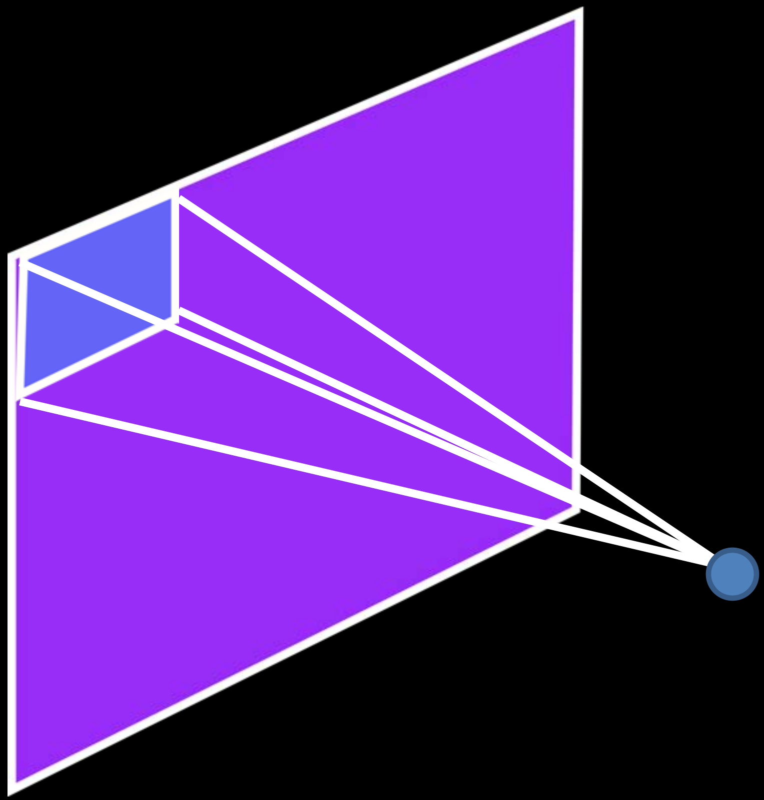
- Face recognition/detection
- Generic object (e.g. cat, dog, bottle)



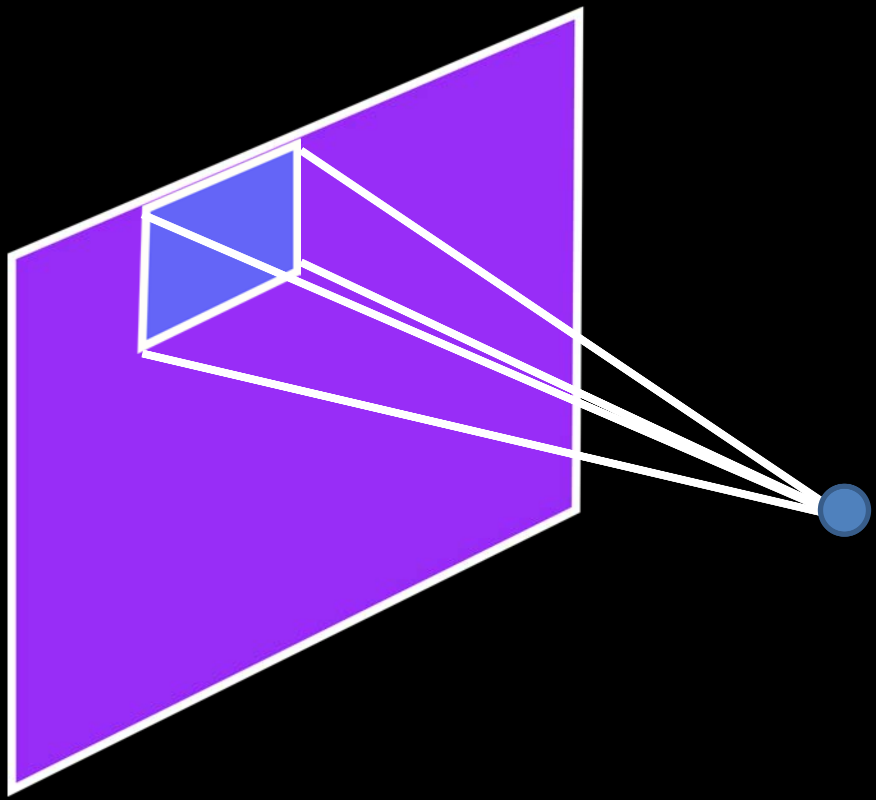
Revision: 2D Convolution



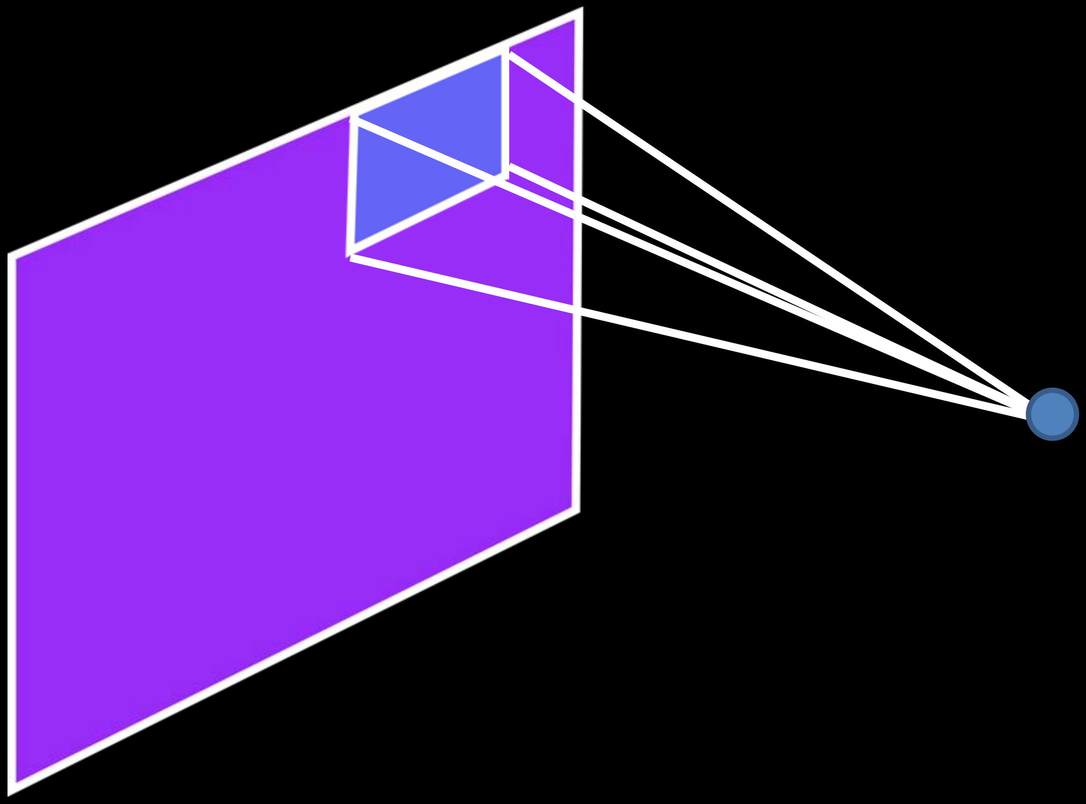
Revision: 2D Convolution



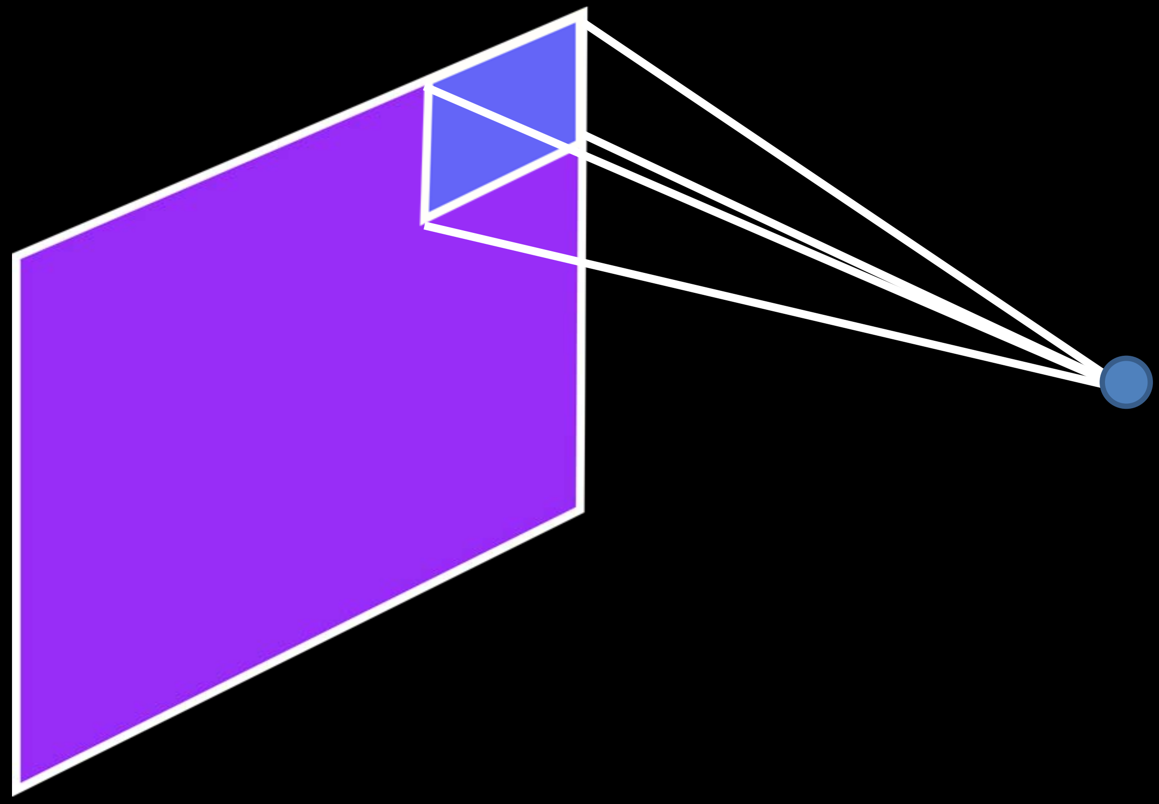
Revision: 2D Convolution



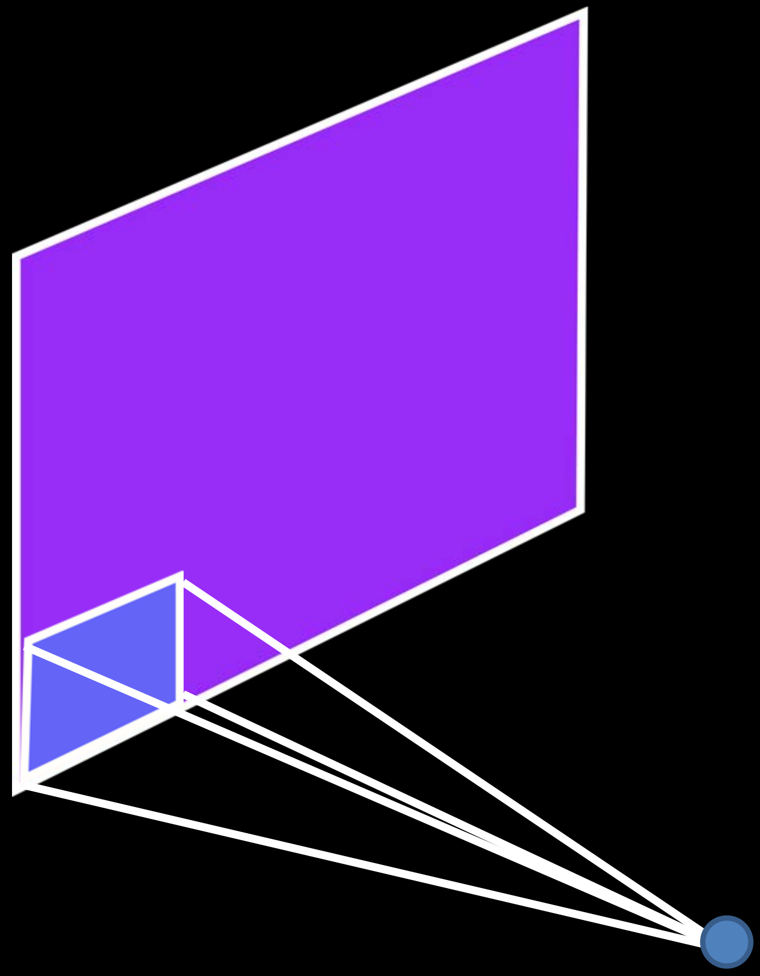
Revision: 2D Convolution



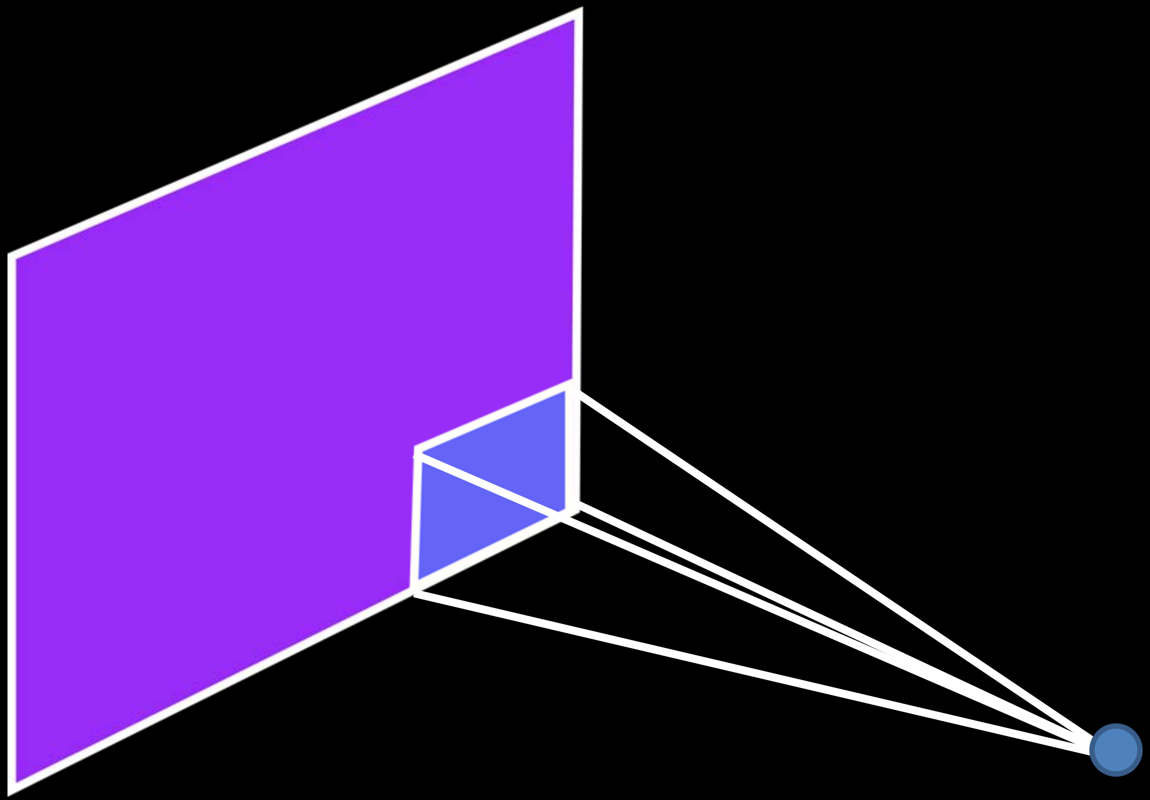
Revision: 2D Convolution



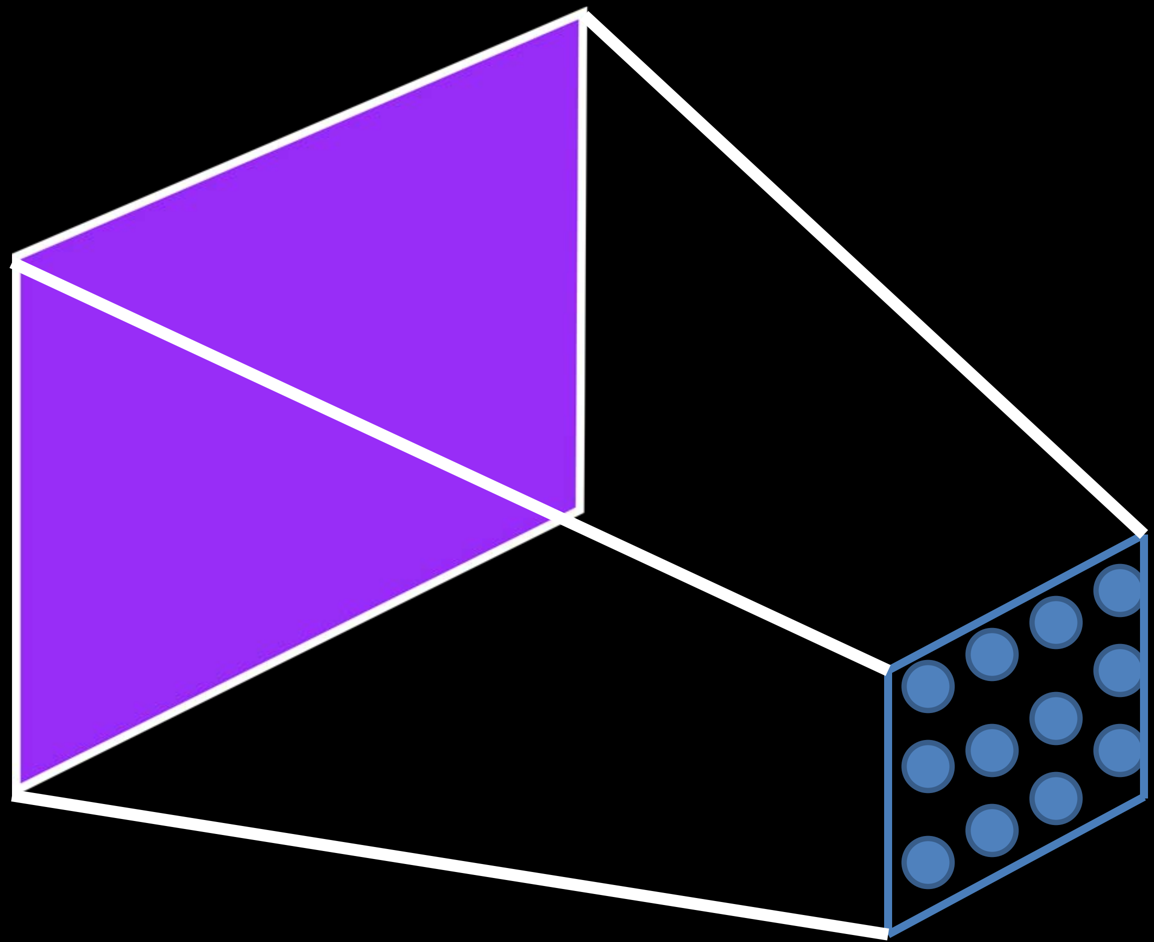
Revision: 2D Convolution



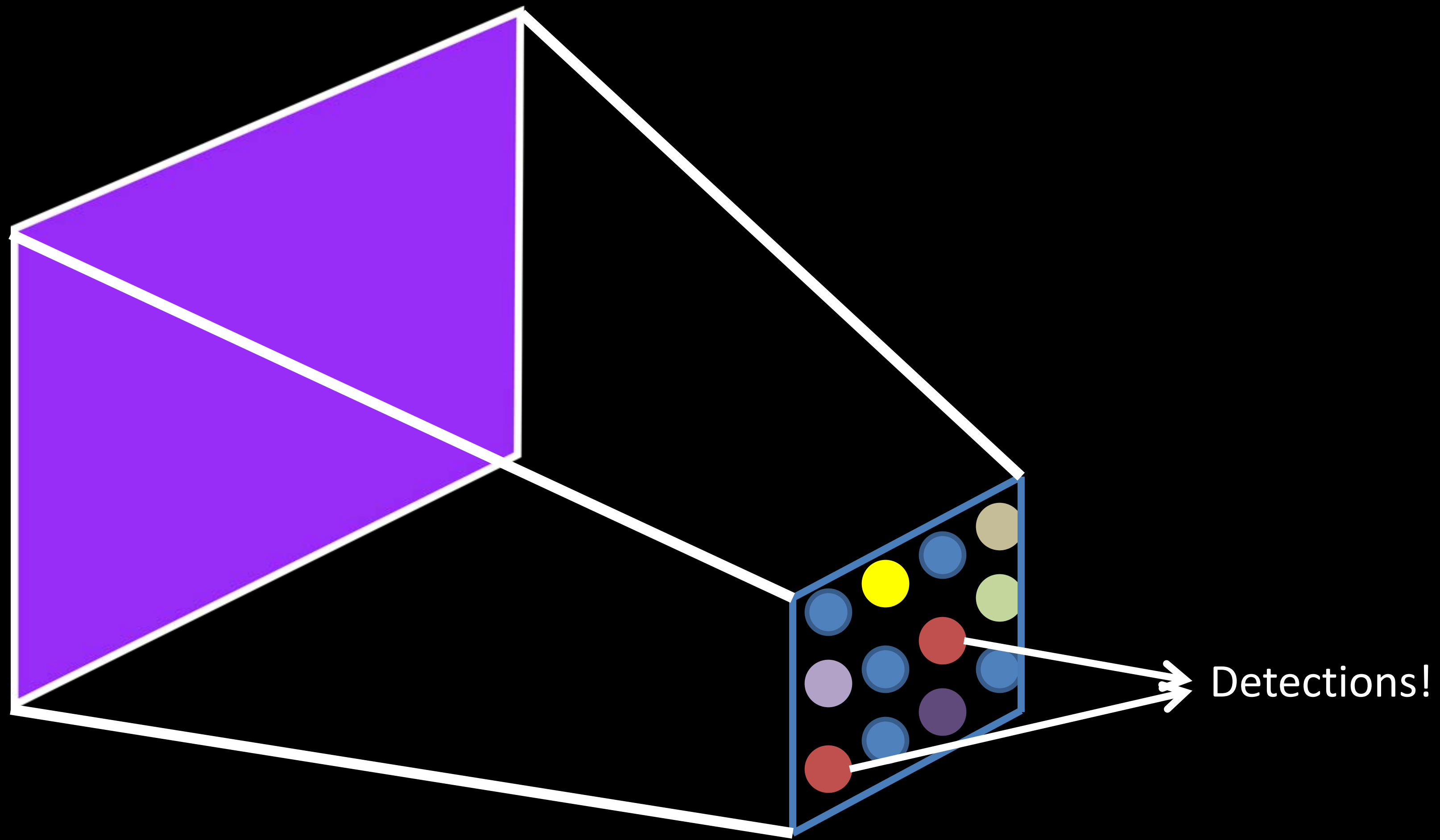
Revision: 2D Convolution



Revision: 2D Convolution



Revision: 2D Convolution

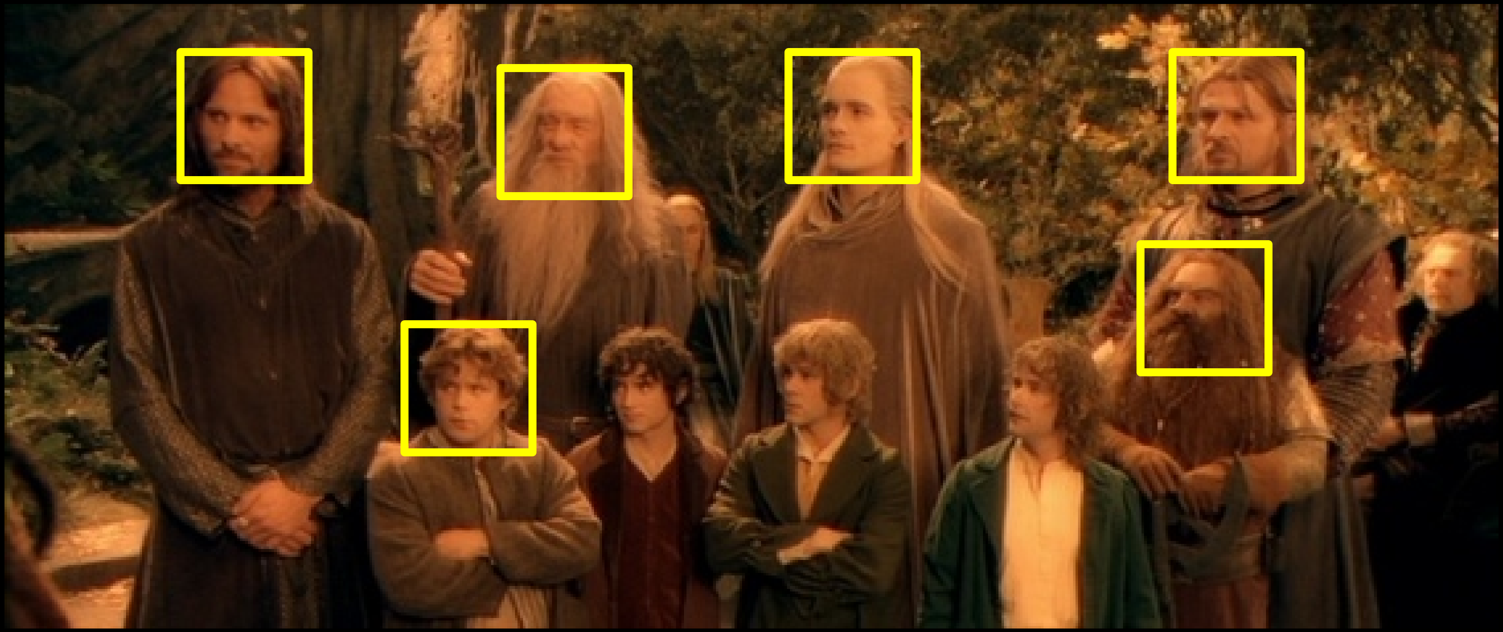


Template based object detection

Template based object detection

Template based object detection

- It works ... sort of



Template based object detection

- Number of templates >= number of classes
- Each template – ~10^5 dot products per image

Why can’t we scale object detection?

- Detecting one class for a regular (640x480 image) requires ~1-2 seconds
- 1000 classes => 1000x
  - ~1000 seconds per image
  - Imagine video, which has 30 frames per second
  - Not even close to real-time
- Even though embarrassingly parallel, latency is too high for real-time applications

Why can’t we scale object detection?

- Each class template is a high dimensional (~4000) vector of single/double precision values
- More classes, more memory to store templates
- Scarce main memory (on-board computers)

Paper – Scaling up object detection

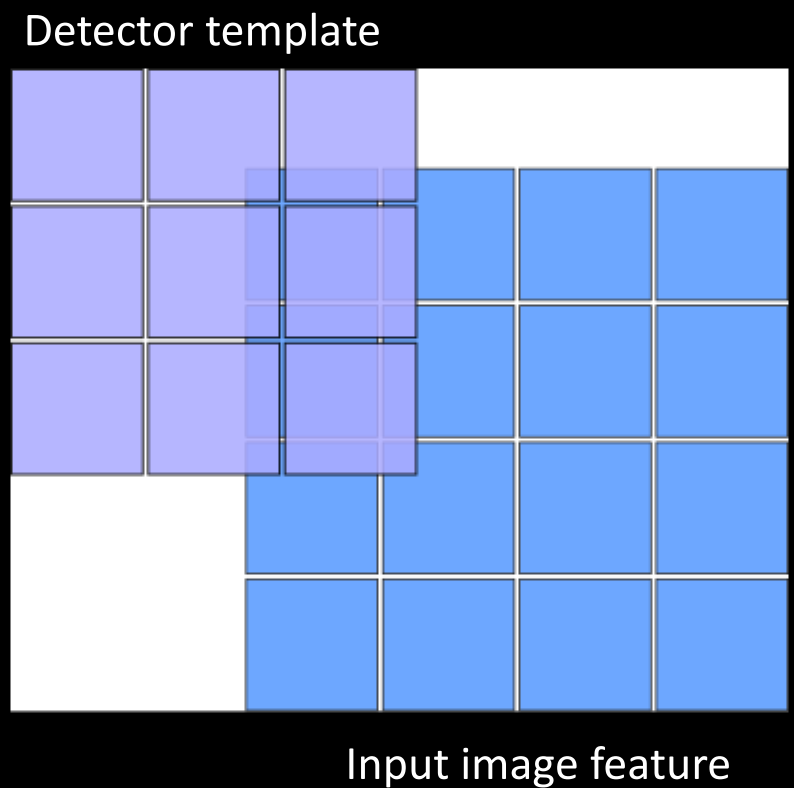
Fast, Accurate Detection of 100,000 Object Classes on a Single Machine

Thomas Dean Mark A. Ruzon Mark Segal
Jonathon Shlens Sudheendra Vijayanarasimhan Jay Yagnik[†]
Google, Mountain View, CA
{tld,ruzon,segal,shlens,svnaras,jyagnik}@google.com

Best Paper

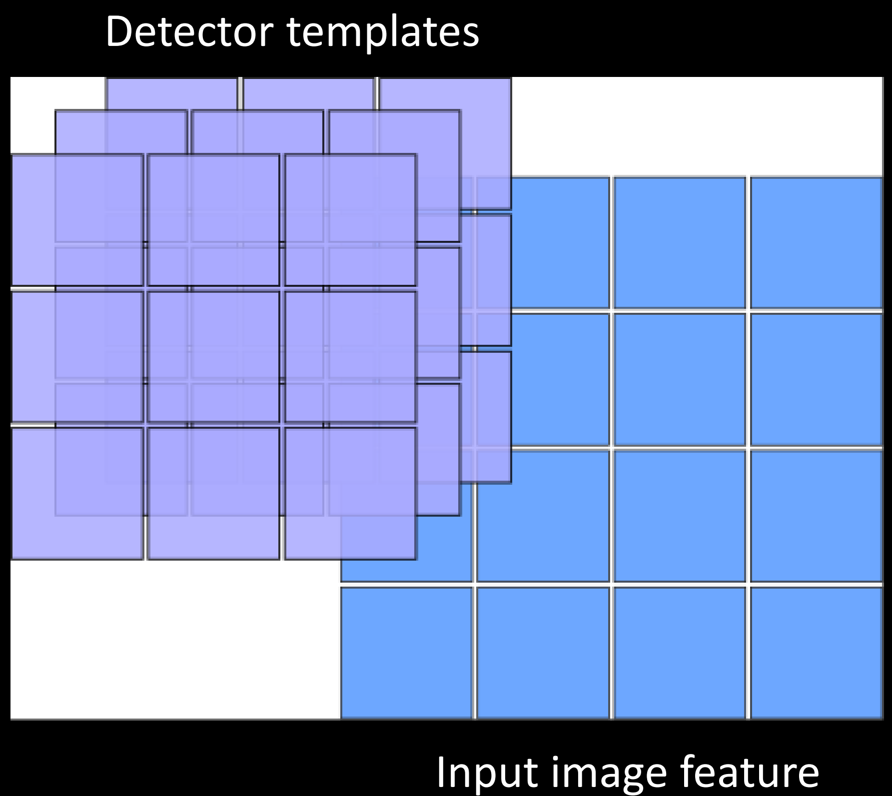
Follow up: Uses similar LSH based techniques to scale detection for CNN based detectors -Vijayanarasimhan et al., arXiv 2014

What’s the bottleneck?



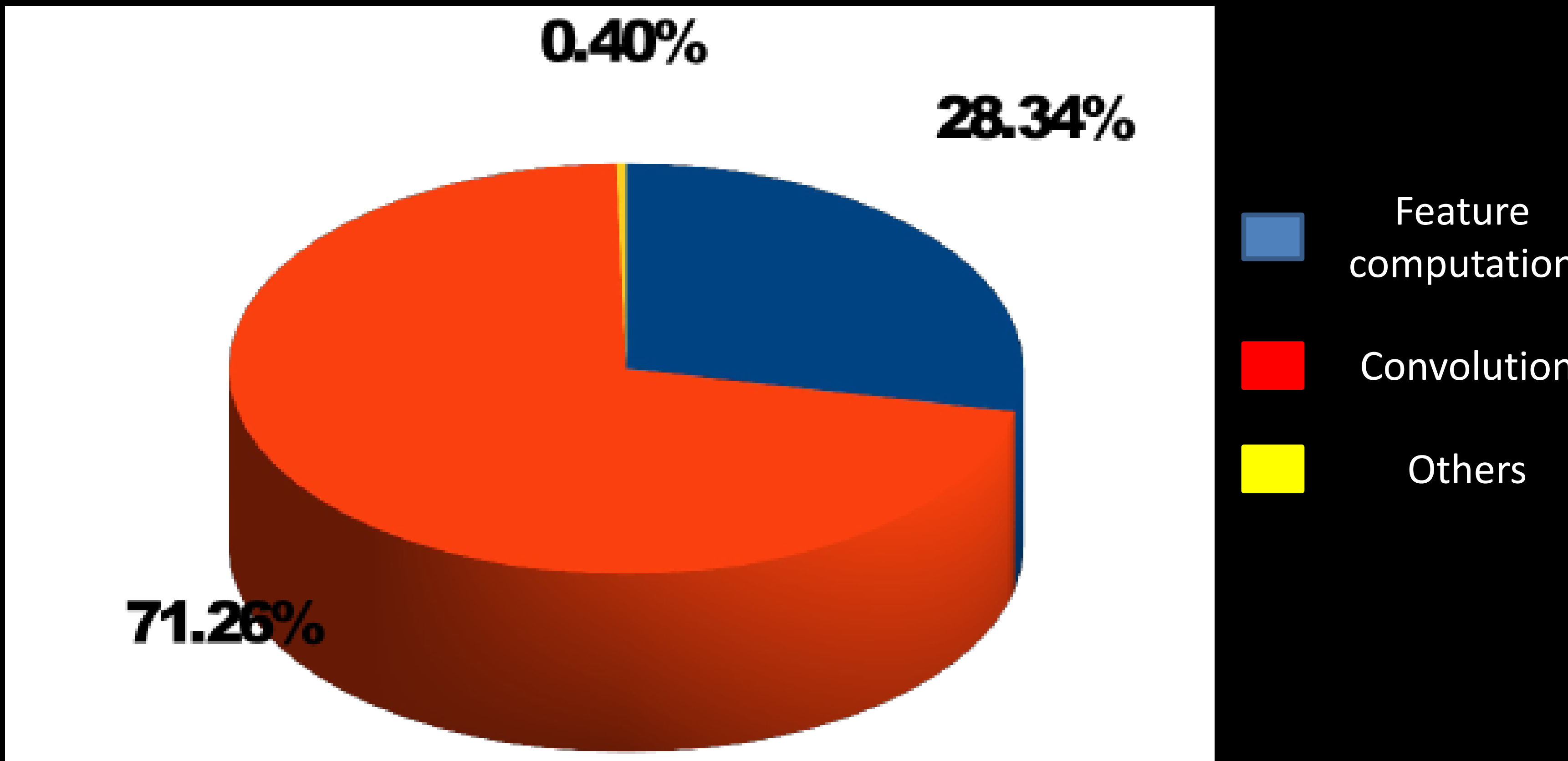
Each step is a dot product of two high dimensional (3100) vectors

What's the bottleneck?



Each step is a dot product of two high dimensional (3100) vectors
C detectors (one per class) => O(C) expensive convolutions

Computation bottleneck

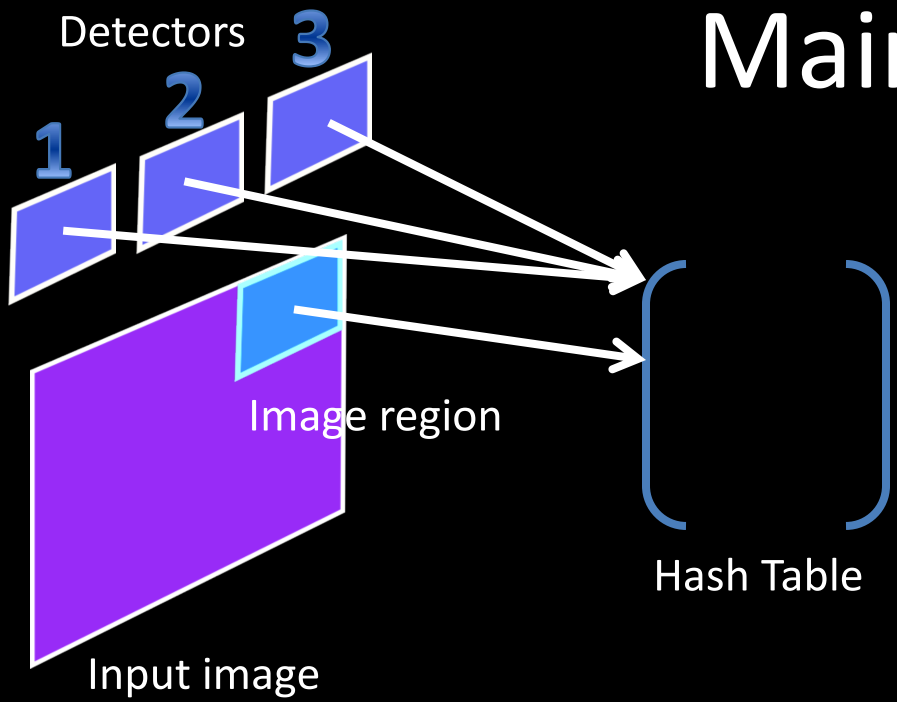


Misra et al., 2014

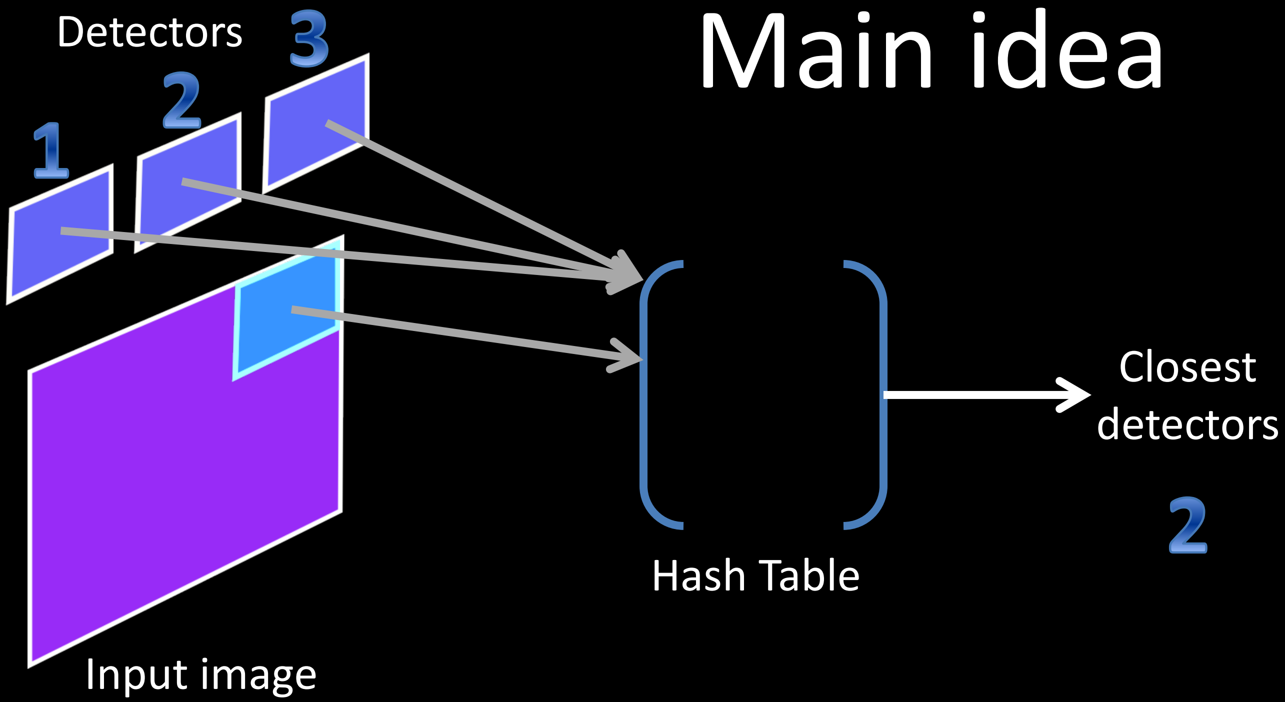
Main idea

- Reduce number of dot products
- Replace dot products by hash lookup
- All detectors will not give high “dot product” response
- Use hashing to find “high response” detectors. Dot product for only these detectors

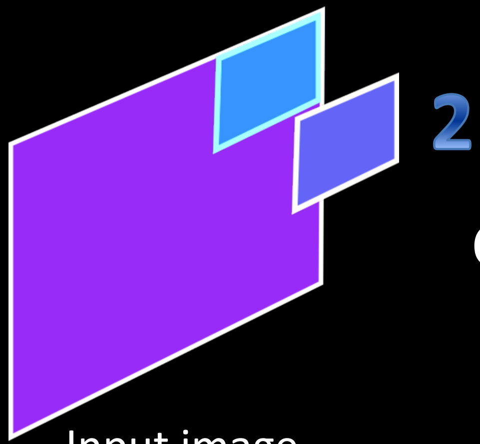
Main idea



Main idea



Main idea



Compute dot product only for closest detector

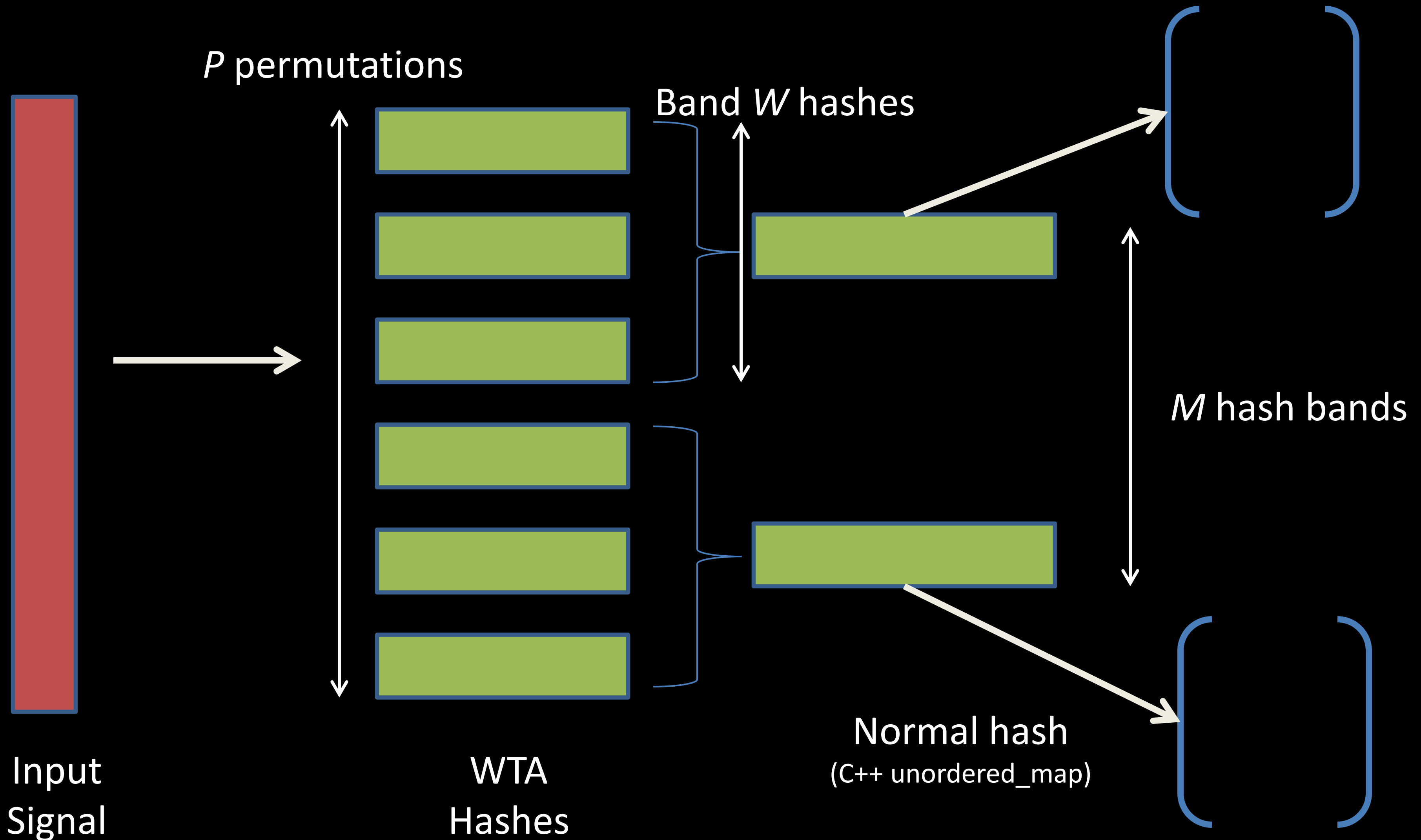
Input image

What should the hash do?

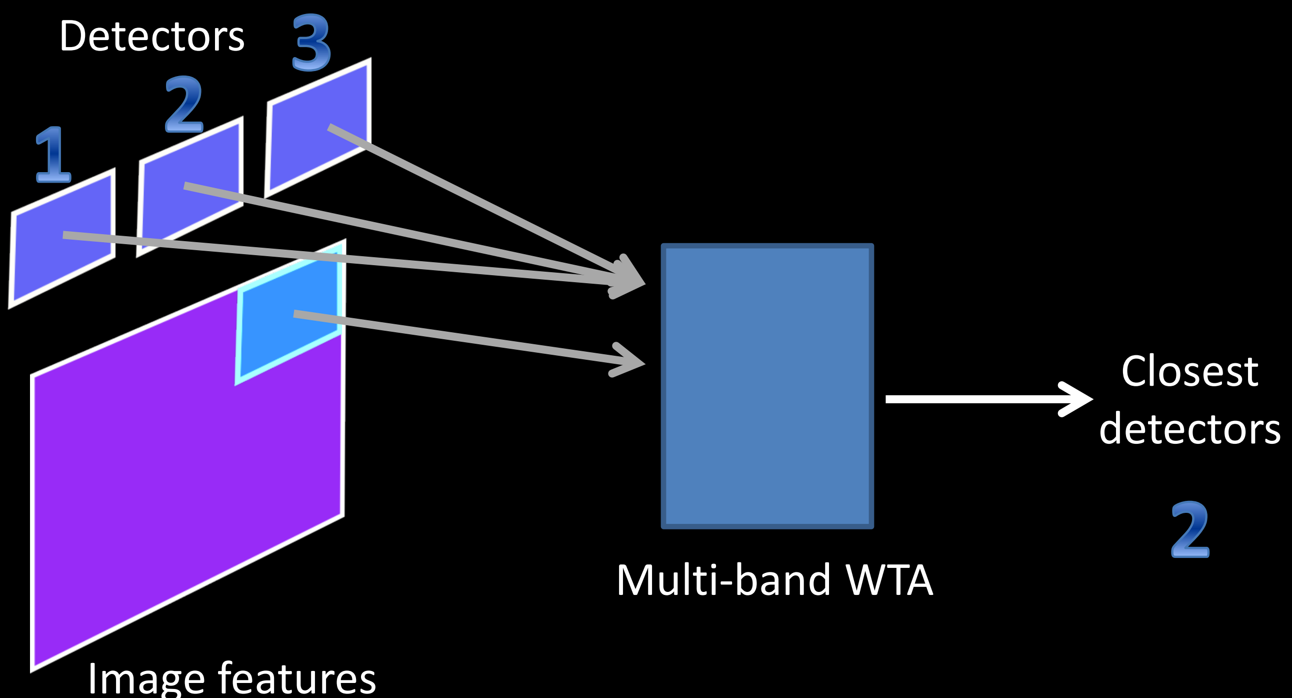
- The hash should make sure that two vectors with “similar filter response” should be close

- WTA Hash ☺

After Training, hash all templates



Test time

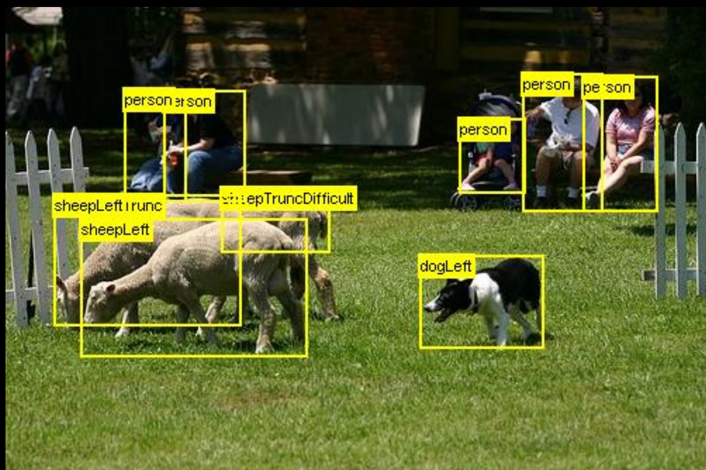

Detectors
3
2
1
Multi-band WTA
Closest
detectors
2
Image features

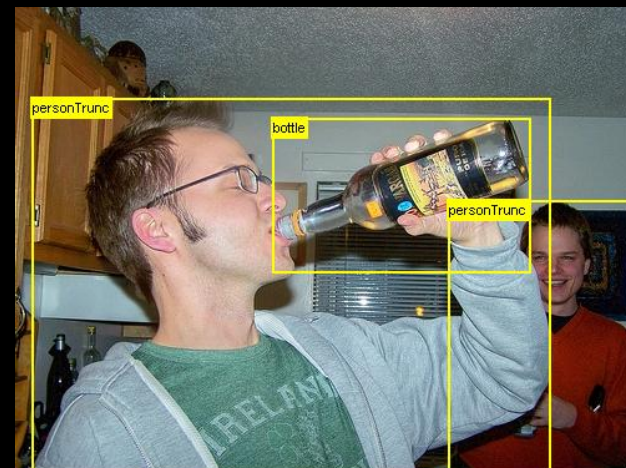
DATASETS

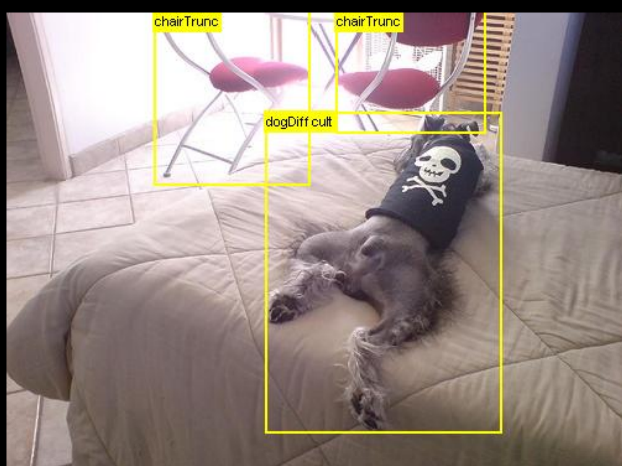
Pascal VOC2007

- Object detection benchmark
- Consists of images from Flickr
- 20 diverse object classes
  - Bottle, dog, human, car, aeroplane, bus, plant etc.

Pascal VOC 2007







RESULTS

Results (Accuracy vs. Speed)

	arp	bike	bird	boat	bttl	bus	car	cat	chr	cow	tbl	dog	hrs	mbke	prsn	plnt	shp	sofa	trn	tv	Mean
Ours	0.19	0.48	**0.03**	0.10	0.16	**0.41**	0.44	0.09	0.15	**0.19**	0.23	**0.10**	**0.52**	0.34	0.20	**0.10**	0.16	**0.28**	**0.34**	0.34	0.24
[6] (base)	**0.29**	**0.55**	0.01	**0.13**	**0.26**	0.39	**0.46**	**0.16**	**0.16**	0.17	**0.25**	0.05	0.44	**0.38**	**0.35**	0.09	**0.17**	0.22	0.34	**0.39**	**0.26**

Table 1. Comparison of the hashing-based and baseline algorithms on the PASAL VOC 2007 dataset

20 classes on PASCAL VOC 2007

5 seconds per image

Speed-up of 20x over standard DPM implementation (*not the cascade version*)

Paper: Fast Accurate Detection of 100,000 Object Classes on a Single Machine

Results – 100,000 detectors

time	t = 8	t = 28	t = 76
mAP	0.07	0.11	0.16
speed-up	$62,500\times$	$17,857\times$	$6,580\times$

(a)

Train them using 5000 machines
Test them on a single machine with 20GB RAM (*P=2400, K=16, M=600)*

Paper: Fast Accurate Detection of 100,000 Object Classes on a Single Machine, Dean et al., 2013

Bonus: Results

- Training a deep net with millions of classes as output

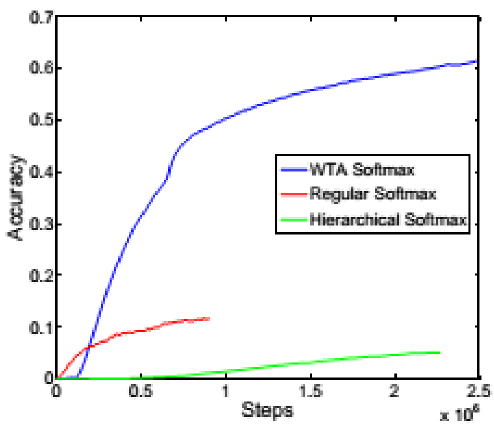


Figure 7: Accuracy of the various models on the Sports 1M evaluation set as a function of the number of training steps.

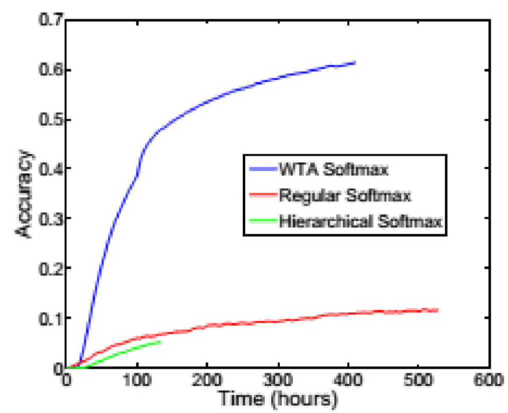
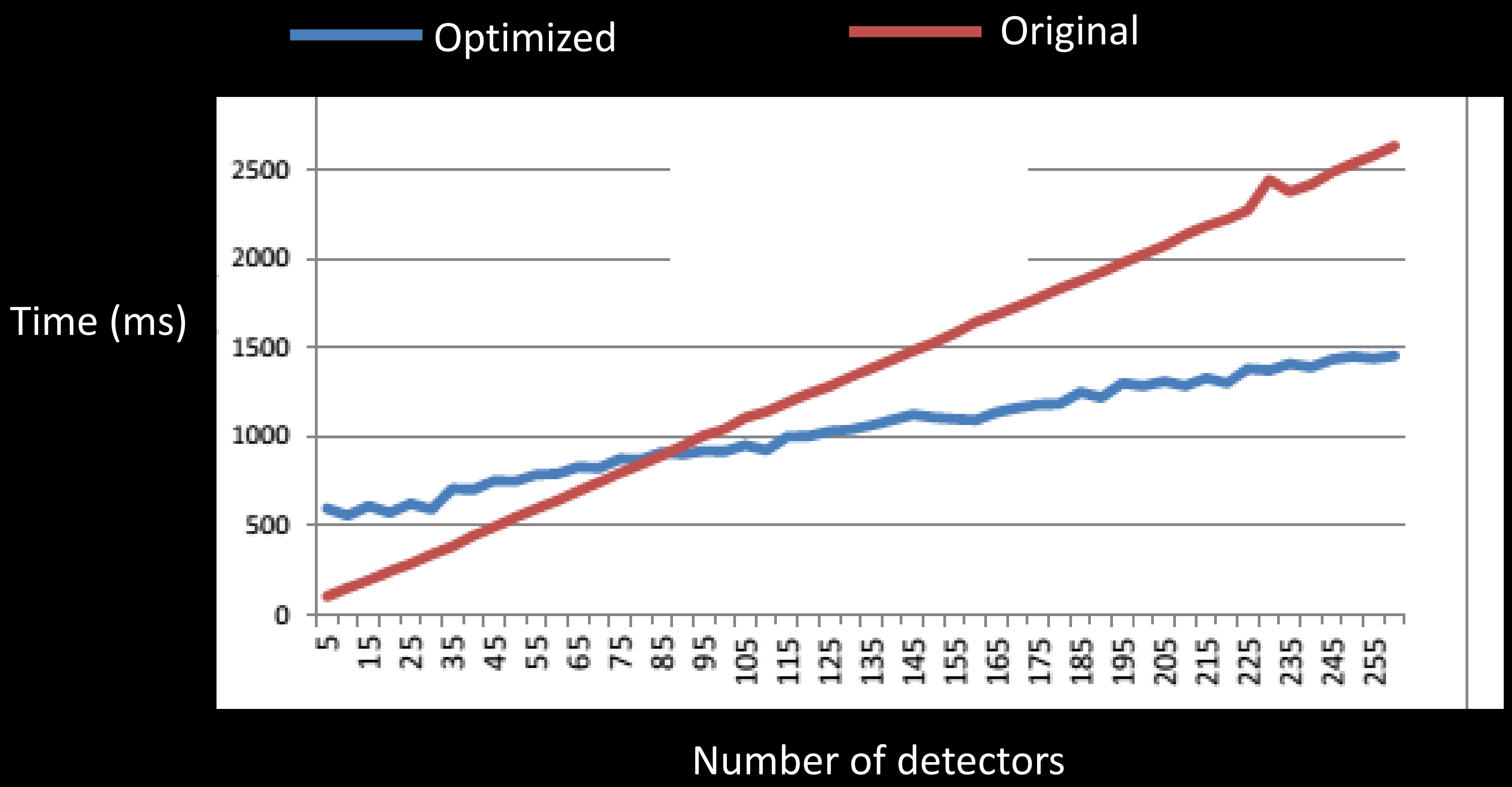


Figure 8: Accuracy of the various models on the Sports 1M evaluation set as a function of the total training time.

Paper: Deep Networks With Large Output Spaces, Vijayanarasimhan et al. 2014

THANK YOU!

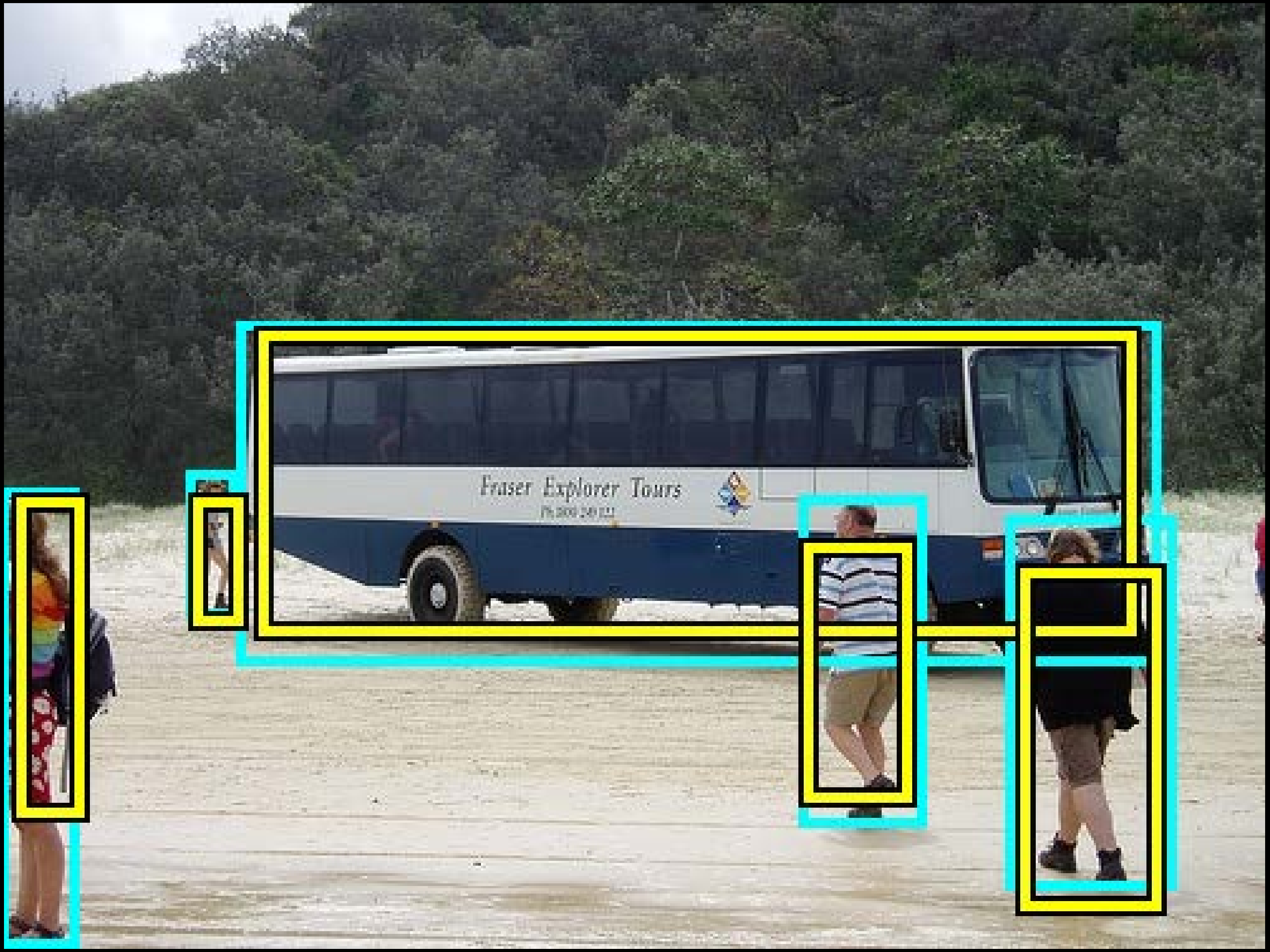
Optimized Implementation



Misra et al., 2014

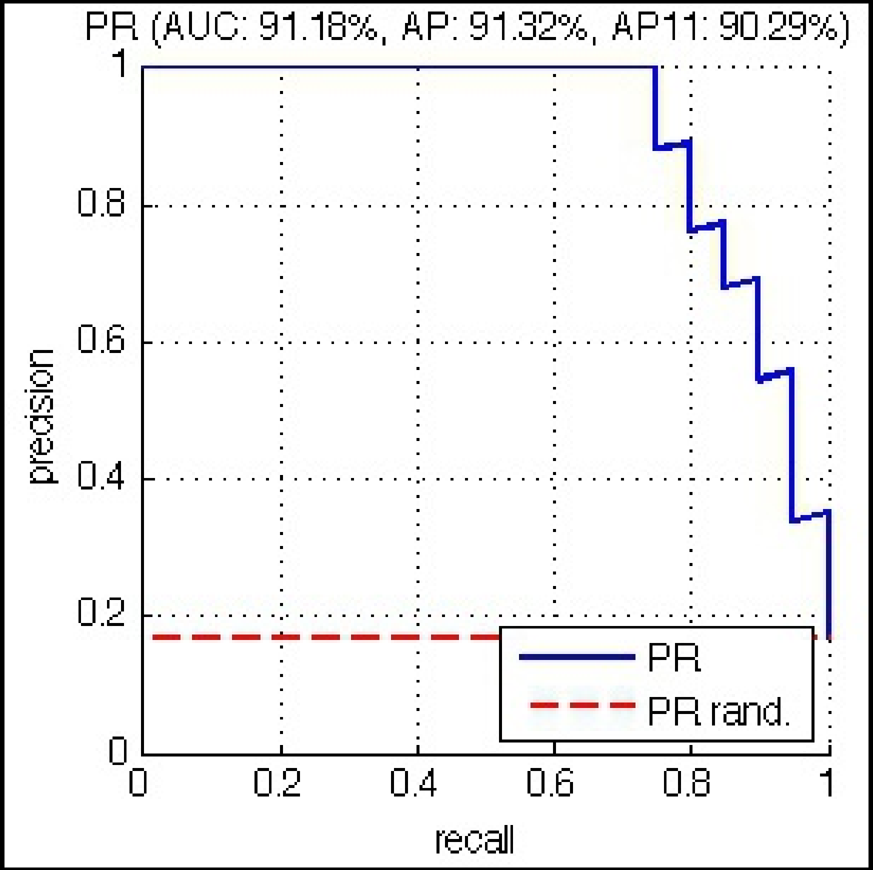
METRICS

Correct detection?



Average Precision (AP)

- Area under precision/recall curve



Results – Fixed compute budget

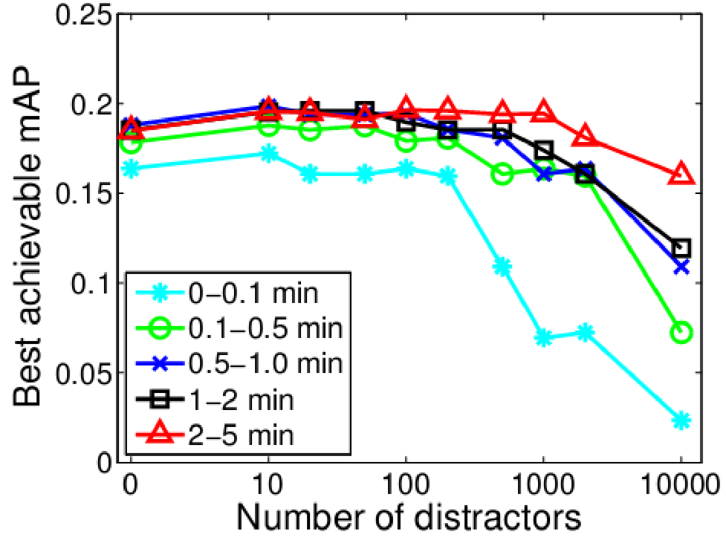
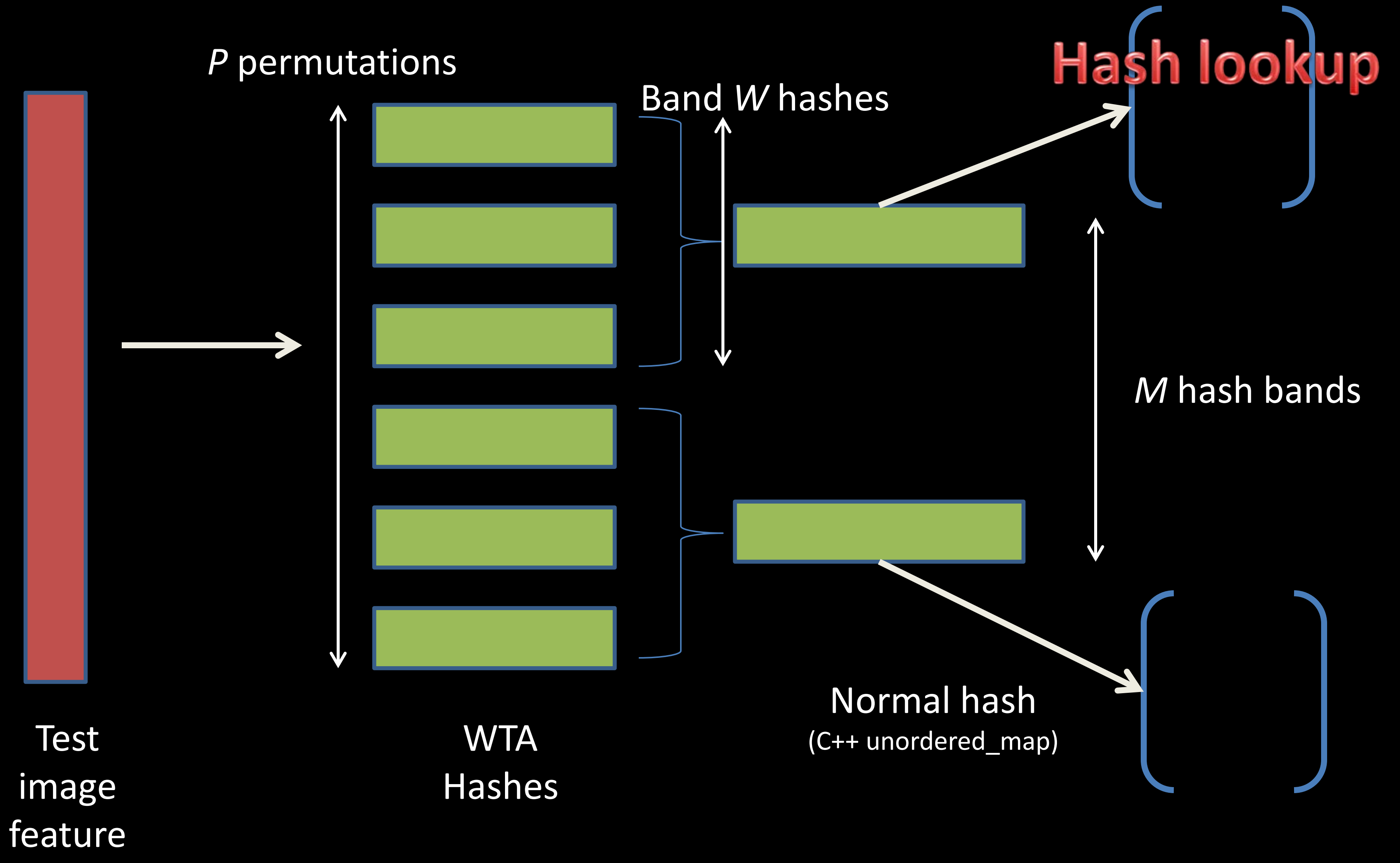


Figure 7. Increasing the number of objects gracefully degrades prediction accuracy on PASCAL VOC 2007 for a fixed computational budget.

Test time



P permutations
Band W hashes
Hash lookup
M hash bands
Normal hash
(C++ unordered_map)
Test image feature
WTA Hashes

WTA Hashing

- Winner Takes All Hashing
  - Input Signal of length L = [5,10, 2, 8]
  - *Parameter:* Window size K = 2
  - P is a permutation of indices *1 to N* = [4,3,2,1]
  - Permute signal according to P => [8,2,10,5]
  - Look at first K elements => [8,2]
  - Find index of the maximum element
  - Hash = all zeros except at max*; hash =* [1 0]

If K = N, then this is exactly Min-Hash

Paper – Scaling up object detection

Fast Accurate Detection of 100,000 Object Classes on a Single Machine – Thomas Dean et al., CVPR 2013

Best Paper

Fast, Accurate Detection of 100,000 Object Classes on a Single Machine

Thomas Dean   Mark A. Ruzon   Mark Segal
Jonathon Shlens   Sudheendra Vijayanarasimhan   Jay Yagnik†
Google, Mountain View, CA
{tld,ruzon,segal,shlens,svnaras,jyagnik}@google.com

Follow up: Uses similar LSH based techniques to scale detection for CNN based detectors - Vijayanarasimhan et al., arXiv 2014

Winner Takes All (WTA) Hashing

The Power of Comparative Reasoning – J. Yagnik, D. Strelow, D. Ross, R. Lin *in ICCV 2011*

The Power of Comparative Reasoning

Jay Yagnik, Dennis Strelow, David A. Ross, Ruei-sung Lin
{jyagnik, strelow, dross, rslin}@google.com

Abstract

Rank correlation measures are known for their resilience to perturbations in numeric values and are widely used in many evaluation metrics. Such ordinal measures have rarely been applied in treatment of numeric features as a representational transformation. We emphasize the benefits of ordinal representations of input features both theoretically and empirically. We present a family of algorithms for computing ordinal embeddings based on partial order statistics. Apart from having the stability benefits of ordinal measures, these embeddings are highly nonlinear, giving rise to sparse feature spaces highly favored by several machine learning methods. These embeddings are deterministic, data independent and by virtue of being based on partial order statistics, add another degree of resilience to noise. These machine-learning-free methods when applied to the task of fast similarity search outperform state-of-the-art machine learning methods with complex optimization setups. For solving classification problems, the embeddings provide a nonlinear transformation resulting in sparse binary codes that are well-suited for a large class of machine learning algorithms. These methods show significant improvement on VOC 2010 using simple linear classifiers which can be trained quickly. Our method can be extended to the case of polynomial kernels, while permitting very efficient computation. Further, since the popular MinHash algorithm is a special case of our method, we demonstrate an efficient scheme for computing MinHash on conjunctions of binary features. The actual method can be implemented in about 10 lines of code in most languages (2 lines in MATLAB), and does not require any data-driven optimization.

1. Introduction

Rank correlation measures have been well regarded as robust measures in many performance evaluation schemes. Since these methods rely on relative ordering of elements, they are very resilient to noise and variations that do not affect the implicit order. We make the case of applying this thought to feature representation. High-dimensional feature sets are quite common across all disciplines of signal analysis and many other domains. Precise values of each feature dimension in such high-dimensional spaces are often not important. We argue for creating representations that are based solely on the relative rank ordering of feature dimensions. Such representations would enjoy all the stability benefits of rank correlation measures while being useful to generate discriminative features. Further, we base our embeddings on multiple partial order statistics rather than total orderings, giving us another degree of resilience to noise and giving rise to representations that have local support on feature dimensions (useful for learning algorithms to distinguish "useless" dimensions if needed).

The main contribution of our paper is the Winner Take All (WTA) hash, a sparse embedding method that transforms the input feature space into binary codes such that Hamming distance in the resulting space closely correlates with rank similarity measures. Algorithm 1 gives the WTA hash and Figure 1 shows it operating on four example input vectors. In short, for each hash we permute the input feature vector with Θ, take the first K components from the permuted vector, and output the index of the maximum component. Many hashes corresponding to different Θ can be combined into an output hash vector.

This thought and simple methods that result from it are very widely applicable. Our embedding method requires no data-driven optimization. While being training-free and easy to compute it outperforms state-of-the-art methods on several tasks. Our method gives rise to a space of sparse binary vectors. This makes it readily amenable for many problems including:

- Using sparse binary vectors as tokens/hashes in Locality Sensitive Hashing schemes for fast similarity search. Here we show that our method outperforms several state-of-the-art machine learning methods [20, 19] for learning hash codes that are optimized for the specific problem of similarity search on LabelMe. The performance gap is significant, particularly when doing sub-linear (approximate) nearest neighbor search.

- Using our embeddings to induce a ranking metric on well known descriptors for similarity search. Here we show that our method improves on SIFT [21] and DAISY [26, 27] by about 11-12% (error rate for 95% recall) on standard benchmarks.
- Our embeddings act as a nonlinear feature-space transformation. When applied with linear classifiers they outperform linear and chi-square kernel classifiers on vocabulary histogram features.

Furthermore we make the following algorithmic contributions :

- We can compute rank embeddings of polynomial spaces of degree p in $O(p)$.
- Our method can bias the rank embeddings to be more sensitive to elements at the head of the rank list. This