Crosstalk Cascades for Frame-Rate Pedestrian Detection

Piotr Dollár[†] Ron Appel[‡] Wolf Kienzle[†]

[†]Microsoft Research Redmond [‡]California Institute of Technology
{pdollar,wkienzle}@microsoft.com appel@caltech.edu

Abstract. Cascades help make sliding window object detection fast, nevertheless, computational demands remain prohibitive for numerous applications. Currently, evaluation of adjacent windows proceeds independently; this is suboptimal as detector responses at nearby locations and scales are correlated. We propose to exploit these correlations by tightly coupling detector evaluation of nearby windows. We introduce two opposing mechanisms: detector *excitation* of promising neighbors and *inhibition* of inferior neighbors. By enabling neighboring detectors to communicate, *crosstalk cascades* achieve major gains (4-30× speedup) over cascades evaluated independently at each image location. Combined with recent advances in fast multi-scale feature computation, for which we provide an optimized implementation, our approach runs at 35-65 fps on 640 × 480 images while attaining state-of-the-art accuracy.

1 Introduction

In many applications, *fast detection* is as important as *accurate detection*. Notable recent efforts for increasing detection speed include work by Felzenszwalb et al. [1] and Pedersoli et al. [2] on cascaded and coarse-to-fine deformable part models, respectively, Lampert et al.'s [3] application of branch and bound search for detection, and Dollár et al.'s [4] and Benenson et al.'s [5] work on the theory and application of fast multi-scale features for detection.

Nevertheless, a majority of detectors remain slow. For most of the 15 pedestrian detectors surveyed in [6], detection time is best measured in seconds per frame (as opposed to frames per second). In this work our goal is to achieve *frame-rate* detection on 640 × 480 images, i.e. detection at 30 fps or higher. We explicitly avoid the ambiguous phrase *real-time* detection (e.g. a recent paper reported 'real-time' detection rates of under 4 fps on low resolution images).

Detector speed is determined by the speed of both the features and classifier. We provide an optimized implementation of the fast multi-scale features proposed in [4] and introduce a novel framework which couples cascade evaluations at nearby locations. By allowing neighboring detectors to communicate, computational cost is greatly reduced (see Figure 1). The resulting *crosstalk cascades* achieve 4-30 fold reduction in the number of evaluated weak classifiers.

Crosstalk cascades operate at 45 fps while matching state-of-the-art detection accuracy and 55-65 fps at slightly higher miss rates. These are 6× and greater

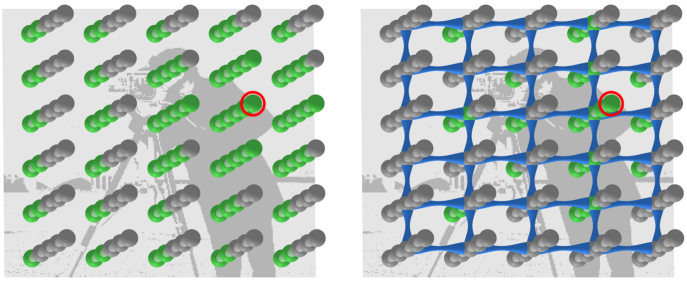

Fig. 1. Each column represents one cascade evaluation, green spheres represent evaluated cascade stages, and the red circles represent locally maximum detector responses. **Left:** In standard cascades, each location is evaluated in isolation. **Right:** Through a combination of *excitation* and *inhibition*, crosstalk cascades can significantly reduce computation, requiring fewer overall weak classifier evaluations.

speedups over [4], the fastest detector surveyed in [6]. Computation of the multi-scale features (with no detector evaluations) runs at 70 fps (~3× speedup over [4]). All reported runtimes are on a single core of a modern PC.

The most competitive approach to our own is [5], whose GPU implementation of [4], along with improved handling of scales, runs at 50 fps on monocular images. The improvements suggested in [5] are orthogonal to our own and could be combined. Other related work can be broken down as follows. A significant body of work exists on optimizing cascades [7–12]; however, in all existing work each image window is evaluated independently. Research on fast feature extraction [7, 13, 4] is complementary to our own. [14–16] propose to first compute a sparse set of detector responses and then sample more densely around promising locations; we use similar intuition for *excitation* of promising neighbors, but do so during every stage of detection for greater gains. Finally, research on optimizing other types of classifiers [1–3, 17–19] is of considerable interest, however, such methods have difficulty matching the speed achieved by boosted classifiers.

The rest of this paper is organized as follows. We describe our optimized feature implementation and baseline detector in §2. In §3 we explore lower and upper bounds on the performance of cascades. We describe crosstalk cascades and an unsupervised learning approach for tuning their speed in §4. Finally, in §5, we compare accuracy and speed to existing approaches.

2 Baseline Detector

Channel features [20, 4] have state-of-the-art performance and are among the fastest in the literature. Given an input image, several channels (e.g. gradient, color) with the same dimensions are computed. Sums over rectangular channel regions serve as features and can be computed efficiently using integral images [7]. In multi-scale detection, features are typically computed over a dense image pyramid. Instead, [4] showed how to leverage the observation that statistics of natural images follow a power law to accurately approximate features at nearby image scales using channels computed over a sparse image pyramid.

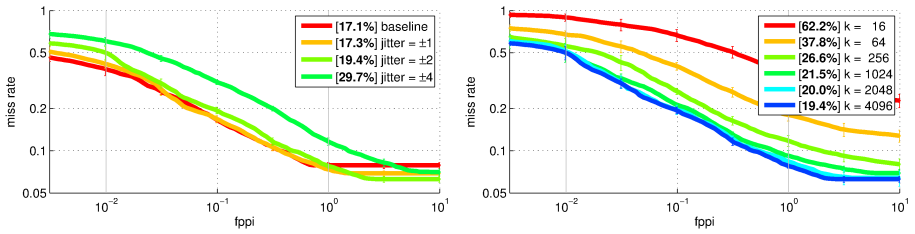

Fig. 2. Left: Our baseline detector (BASELINE) outperforms [20] and the best reported results on INRIA with a log-average miss rate of 17%. Slightly jittering the training data degrades performance slightly but increases detector correlations. **Right:** Increasing the number of weak classifiers k in the detector trained with jitter ± 2 improves performance and using $k = 4096$ is necessary to achieve state of the art accuracy.

We re-implement the channel features with a focus on low-level optimization. We use the same channels as [4]: gradient magnitude (1 channel), histogram of oriented gradients (6 channels), and LUV color channels (3 channels). For each scale, all 10 channels are downsampled by a factor of 2 for further speed improvements (for details see addendum to [20] available online). The functions make extensive use of SSE instructions but are implemented on a single CPU; further gains could be obtained using multiple cores or a GPU as in [5].

For 640×480 images, computation of sparse feature pyramids with one scale per octave runs at ~**70 fps** on a modern PC (a $4\times$ speedup from [4]). The sparse feature pyramid can be used to obtain detector responses at all scales by 'resampling' a trained detector by up to half an octave as described [4]. In contrast, the traditional approach of computing a dense image pyramid with 8-10 scales per octave is over $4\times$ slower. Benenson et al. [5] describe an extension of [4] that can be used to obtain detector responses at all scales using feature computed at just the original scale, doing so would give a ~25% speedup. Additional speedups of up to 50% are possible by removing gradient normalization, image smoothing, etc., but result in noticeably decreased performance. Our optimized implementation of channel features is available online.

For our baseline detector we use a similar setup to [20]. We apply AdaBoost [21] to train and combine 4096 depth-two trees using a pool of 30,000 random candidate features computed over the channels described above. Training with multiple rounds of bootstrapping takes ~10 minutes (a parallelized implementation of training takes ~3 minutes). The default step size used in the detector is 4 pixels and 8 scales per octave. We closely follow implementation details from [20] for bootstrapping, non-maximum suppression, etc.

Our baseline detector outperforms the best reported results on INRIA [22]. Results are shown in Figure 2 (left). As in [6], we plot miss rate against false positives per image (FPPI) and summarize performance using log-average miss rate (MR) between 10^{-2} and 10^{0} FPPI. The MR of our baseline, averaged over the eight random trials, is 17% (MRs for individual trials ranges from 16% to 19%). In comparison, the best reported results in [6] have a MR of 20% for Felzenszwalb et al. [23] and between 21%-22% for the detectors in [20, 4].

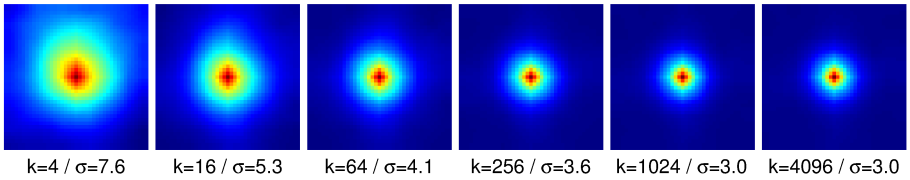

Fig. 3. Region of support (ROS) for our detector trained with jitter for various number of weak classifiers k. Key observations: (1) for every k the ROS has non-negligible extent σ and (2) σ decreases with increasing k. These observations inform our design of crosstalk cascades; in particular we exploit that correlations are strongest in early stages of detection but continue to have non-negligible support for all k.

2.1 Detector Correlations

Detector responses at nearby positions and scales are correlated. In the terminology of [15], every detector has a *region of support* (ROS) which is the neighborhood around a positive location in which the response remains positive. A detector's ROS is determined by the features, discriminability of the classifier, and alignment of the training data. We focus on the latter two factors.

First, we can increase detector correlations by 'jittering' the training data (we replace every positive sample with nine samples offset by $\pm j$ pixels). Performance for the baseline and various j (JITTER $\pm j$) are shown in Figure 2. Using $j = 2$ degrades performance slightly to 19.4% MR but results in stronger detector correlations. We therefore use JITTER ± 2 in all remaining experiments.

We compute the ROS by averaging detector responses across multiple windows whose center contains a locally maximum response and we record the ROS's standard deviation σ. In Figure 3 (top) we show the ROS for the detector for various number of weak classifiers k. For every k the ROS has a non-negligible extent ($\sigma \geq 3$). In previous work [14–16] a similar observation for the *complete* detector was used as a basis for fast detection schemes that compute a sparse set of responses and then sample more densely around promising locations.

In this work we exploit that correlations are present at *all* values of k. In fact, *the extent of the detector's ROS decreases with increasing* k. Intuitively this makes sense: we expect the extent of a detector's ROS to be inversely related to its discriminative power, which in this case is determined by k (see Figure 2 (right)). The strong correlation present for every k motivate our approach of allowing neighboring detectors to communicate during classification.

3 Bounds on Soft Cascades

The seminal work of Viola and Jones [7] popularized cascades for fast detection. A number of subsequent papers have addressed drawbacks of the original cascades [10–12]; however, perhaps the simplest and most elegant solution was proposed by Bourdev and Brandt in the form 'soft cascades' [8]. Instead of having multiple distinct cascade stages, a single boosted classifier is trained and only post-training are rejection thresholds set (with one threshold per weak classifier).

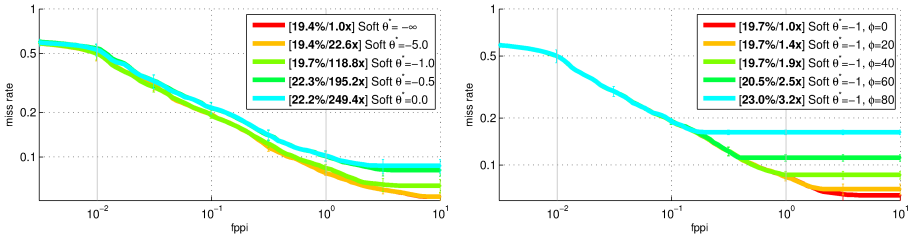

Fig. 4. Left: Soft cascades with thresholds $\theta_k = \theta^*$ for all k. Numbers in brackets indicate MR/speedup. Using $\theta^* = -1$ leaves detection accuracy virtually unchanged (all curves with $\theta^* \leq -1$ overlap nearly perfectly) but results in a ∼**120**× speedup. **Right:** Soft cascades with thresholds $\theta_k = \theta^* + \phi k/K$, which are equivalent to *recalibrating* a detector and using constant rejection thresholds (see text). Depending on the desired target recall, ϕ can be set appropriately for additional speedups relative to $\theta_k = \theta^*$.

A boosted classifier H consisting of K weak classifiers has the form:

$$H(x) = H_K(x) = \sum_{i=1}^{K} \alpha_i h_i(x), \tag{1}$$

where each h_i is a weak classifier (with output -1 or 1) and α_i is its associated weight. x is classified as positive if $H(x) > 0$ and $H(x)$ serves as the confidence. We can define a sequence of score functions $H_k(x)$ for $k < K$ in an analogous manner. During evaluation, a soft cascade tests each $H_k(x)$ against a rejection threshold θ_k, and if $H_k(x) < \theta_k$ computation stops. Various strategies for setting the θ_k have been proposed [8, 9], we postpone a discussion to §4.1.

3.1 Constant Rejection Thresholds

We begin by describing a simple heuristic for setting θ_k that will serve as a *lower bound* on the effectiveness of soft cascades: we simply set $\theta_k = \theta^*$ for all k for some $\theta^* \leq 0$. Resulting ROC curves for various choices of θ^* are shown in Figure 4 (left). In the plot legend we report '[MR/speedup]' on INRIA [22] for each method. Setting $\theta^* = -1$ leaves detection accuracy virtually unchanged but results in a ∼**120**× speedup. Specifically, on average only 35 weak classifiers are evaluated per classification as opposed to all $k = 4096$ for the original detector.

Why is setting $\theta_k = \theta^*$ effective? Boosting attempts to train a function H s.t. $H(x) > 0$ if and only if x is positive. However, this observation holds for all intermediate classifiers H_k for $k < K$ (which are trained using an identical procedure to H). In practice, it is rare that $H(x) >> 0$ while $H_k(x) << 0$ for any k. By setting $\theta_k = \theta^* \leq 0$, we are exploiting the fact that if $H(x) > 0$ it is unlikely that $H_k(x) < \theta^*$ (with the likelihood decreasing with decreasing θ^*).

Constant rejection thresholds can be extended to reject all detections such that $H(x) < \phi$ by *recalibrating* H. Let $H'_k(x) = H_k(x) - \phi k/K$ for all k so that $H'_k(x) > 0$ if and only if $H_k(x) > \phi k/K$. Using thresholds $\theta_k = \theta^*$ with H' is equivalent to using $\theta_k = \theta^* + \phi k/K$ with H; ϕ controls the tradeoff between recall and speed. Results for $\theta^* = -1$ and various ϕ are shown in Figure 4 (right). Constant rejection thresholds (with recalibration) will serve as a simple baseline.

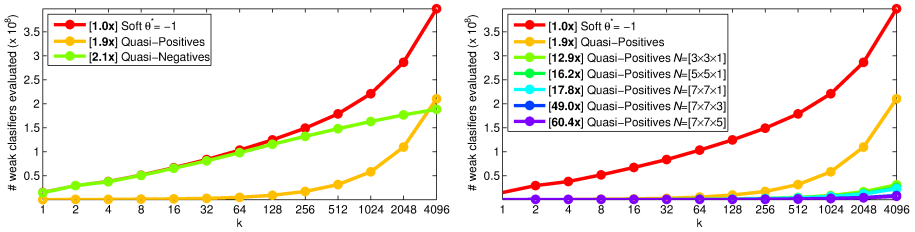

Fig. 5. Left: Cost of evaluating $H(x)$ (with $\theta^* = -1$) on all x and also separately on quasi-positives ($H(x) \geq 0$) and quasi-negatives ($H(x) < 0$). For small k the cost of quasi-negatives dominates; for large k the cost of quasi-positives increases. **Right:** Cost of evaluating $H(x)$ on all x with $z^{\mathcal{N}} = 1$. Evaluating only the x with locally maximum $H(x) > 0$ in fairly small neighborhoods $\mathcal{N}$ results in greatly reduced computation.

3.2 Optimal Soft Cascades

Given a trained classifier $H(x)$, how fast would a hypothetical *optimal* soft cascade be that rejected all $H(x) < 0$ without any computation? Answering this will provide us with intuition and an *upper bound* on soft cascade effectiveness.

We gather a set of detection windows $\mathcal{X}$ by densely sampling windows from 387 images from PASCAL VOC 2007 [24] that contain at least one person. This gives us a validation set with similar statistics to INRIA [22]. Using a step size of 4 pixels and 8 scales per octaves there are ∼15 million windows $x_i \in \mathcal{X}$. We assign a label $z_i \in \{\pm 1\}$ to each detection window x_i using our trained detector H according to the sign of $H(x_i)$: $z_i = 1$ if and only if $H(x_i) \geq 0$. We refer to x_i as *quasi-positive* if $z_i = 1$ and *quasi-negative* otherwise (while z_i is correlated with the ground truth label no supervision is needed to obtain z_i).

Over 99.5% of the 15 million $x_i \in \mathcal{X}$ are quasi-negatives. While we may thus expect quasi-negatives to dominate computation, in practice evaluating the small fraction of quasi-positives incurs roughly the same cost. If we compute $H(x_i)$ using rejection thresholds $\theta^* = -1$, evaluating $H(x_i)$ for $z_i = 1$ typically requires evaluating all 4096 weak classifiers but on average only ∼18 weak classifiers if $z_i = -1$. Figure 5 (left) shows the total number of weak classifiers that need to be evaluated in order to compute $H_k(x_i)$ over all x_i, x_i with $z_i = 1$, and x_i with $z_i = -1$. Results are averaged over eight trials (see §2).

Observe that even if there existed a cascade that could reject all x_i with $z_i = -1$ without any computation it would **only be ∼2×** faster than the soft cascade with $\theta^* = -1$ on this data. Thus, while tuning θ_k can increase speed (see §4.1), we need an alternate approach to achieve greater gains.

If instead of computing each $H(x_i)$ in isolation we consider neighboring x_i jointly, much greater gains are possible. We extend the definition of quasi-positives to include a neighborhood $\mathcal{N}$: $z_i^{\mathcal{N}} = 1$ if $H(x_i) \geq 0$ and $H(x_i) \geq H(x_i')$ $\forall x_i' \in \mathcal{N}(x_i)$. In other words $z_i^{\mathcal{N}} = 1$ if x_i has a locally maximum score $H(x_i) > 0$. Abusing notation we write $\mathcal{N} = [w \times h \times d]$ to denote neighborhoods of width w, height h, and depth d (number of scales) with $\sim whd$ neighbors (our detector uses a step size of 4 pixels so $\mathcal{N} = [w \times h \times d]$ covers a volume of $16whd$ pixels).

In Figure 5 (right) we show the cost of evaluating $H_k(x)$ for all x_i with $z_i^{\mathcal{N}} = 1$ for different neighborhoods sizes. Computing $H(x_i)$ only for x_i with $z_i^{\mathcal{N}} = 1$ for a moderately sized neighborhood $\mathcal{N} = [7 \times 7 \times 3]$ would result in ∼**50**× savings. Since typically non-maximum suppression (NMS) is applied to the output of H, returning only x_i that have locally maximum $H(x_i)$ should have little effect on accuracy (especially for small $\mathcal{N}$). We verify this empirically in §5.

A hypothetical soft cascade that could reject all x_i with $z_i^{\mathcal{N}} = -1$ with no computation would improve detection speed by a factor of 10-60× for modest neighborhood sizes $\mathcal{N}$. This gain is an order of magnitude larger than possible for optimal soft cascades that consider each x_i in isolation and motivates our approach of allowing neighboring detectors to communicate during classification.

4 Crosstalk Cascades

In this section we introduce algorithms for constructing four types of cascades:

- **Soft Cascades:** Introduced in §3, a soft cascade rejects a sample x if its score $H_k(x)$ falls below a per-stage *rejection* threshold $\theta_k^R \geq \theta^*$.
- **Excitatory Cascades:** Starting with evaluations over a sparse set of samples x, if for any k the score $H_k(x)$ rises above a per-stage *excitation* threshold $\theta_k^E \geq \theta^*$ evaluation begins on all $x' \in \mathcal{N}(x)$.
- **Inhibitory Cascades:** If the ratio $H_k(x)/H_k(x')$ falls below a per-stage *inhibition* threshold $\theta_k^I < 1$ for some $x' \in \mathcal{N}(x)$ then x is inhibited (rejected).
- **Crosstalk Cascades:** Combines soft, excitatory and inhibitory cascades in a straightforward manner with the goal of computing $H(x_i)$ if and only if $z_i^{\mathcal{N}} = 1$ and rejecting all other x_i with minimal additional computation.

We present an **unsupervised, data-driven framework** for learning the per-stage thresholds for each cascade type that has a **single free parameter** γ controlling tradeoff between speed and accuracy. No data annotation is necessary and only a single parameter γ needs to be selected to tune the cascades.

To keep overhead low we test thresholds only at stages $1 \leq k < K$ where k is a power of 2 (with $K = 4096$ weak classifiers there are 12 such stages). In all experiments we use windows $x_i \in \mathcal{X}$ sampled from the PASCAL images as described in §3.2. For learning we only require x_i for which $z_i^{\mathcal{N}} = 1$ (i.e. the quasi-positives) that survive a constant rejection cascade with $\theta^* = -1$. Based on experiments in §2.1 and §3.2, we use a neighborhood size of $\mathcal{N} = 7 \times 7 \times 3$.

We next describe each cascade type in detail and show how to learn respective thresholds given γ and a set of quasi-positives. For each cascade type we show two plots (Figures 6-9). **Left**: We plot the breakdown of computational effort versus k using the plot type introduced in §3.2 with overall speedup relative to constant soft cascades given in legend brackets. **Right**: We define the *quasi miss rate* (QMR) as the fraction of quasi-positives with $H_k(x) > \theta^*$ rejected by a cascade. We plot QMR for all x with $H(x) > H_0$ as a function of H_0 and in legend brackets list QMR averaged over $0 \leq H_0 \leq 200$. The QMR is computed in a fully unsupervised manner and serves as a useful estimate of the true MR.

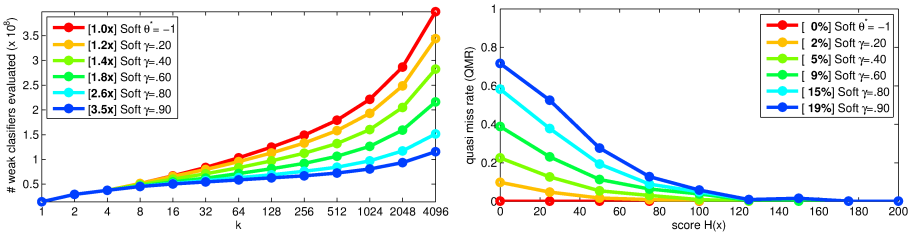

Fig. 6. Soft cascades with per-stage rejection thresholds for various γ. See text.

4.1 Soft Cascades

Details for how soft cascades are applied at runtime were given in §3. We now describe our unsupervised approach for setting the rejection thresholds θ_k^R given the boosted classifier H, quasi-positives x_i, and target QMR γ. In practice we only test at rejection stages $k_j = 2^{j-1}$ for $1 \leq j \leq 12$ to keep overhead low, but for notational simplicity we assume testing occurs for every $k \leq K$.

Let $\gamma' = 1-(1-\gamma)^{1/K}$. If at each rejection stage the QMR is $\leq \gamma'$ the overall QMR of the soft cascade will be $\leq \gamma$. Let $\mathcal{X}_1$ denote the set of quasi-positives and let $\mathcal{H}_1 = \{H_1(x_i)|x_i \in \mathcal{X}_1\}$. We set the first rejection threshold θ_1^R via

$$\theta_1^R = [\mathcal{H}_1]_r - \epsilon \quad \text{where } r = \lfloor \gamma' \cdot |\mathcal{H}_1| \rfloor. \tag{2}$$

Here $[\mathcal{H}]_r$ denotes the r^{th} order statistic of $\mathcal{H}$ (i.e. the r^{th} smallest value in $\mathcal{H}$) and $\epsilon = 10^{-5}$. For each remaining stage $1 < k \leq K$ we let $\mathcal{X}_k = \{x_i \in \mathcal{X}_{k-1}|H_{k-1}(x_i) > \theta_{k-1}^R\}$ and $\mathcal{H}_k = \{H_k(x_i)|x_i \in \mathcal{X}_k\}$ and compute:

$$\theta_k^R = [\mathcal{H}_k]_r - \epsilon \quad \text{where } r = \lfloor \gamma' \cdot |\mathcal{H}_k| \rfloor. \tag{3}$$

Claim: the soft cascade with θ_k^R as defined above has QMR of at most γ. Proof: by construction $|\mathcal{X}_k| \geq |\mathcal{X}_{k-1}| \cdot (1-\gamma')$; therefore $|\mathcal{X}_{K+1}| \geq |\mathcal{X}_1| \cdot (1-\gamma')^K$. The fraction of rejected quasi-positives is therefore at most $1-(1-\gamma')^K = \gamma$. ■

We can further optimize θ_k^R by performing a second pass with a target QMR $\gamma = 0$ and $\mathcal{X}_{K+1}$ as the initial set of quasi-positives, but doing so has little effect in practice. Results for various choices of γ are shown in Figure 6. A value of $\gamma = .60$ results in a 1.8× speedup over using a constant soft cascade, more aggressive settings yield up to a 3.5× speedup but with larger QMR. Note that error decreases for x_i with larger scores $H(x_i)$, meaning high scoring detections are less likely to be rejected.

How does our approach for computing θ_k^R compare with existing methods? Zhang and Viola [9] improved upon the original *fully supervised* method of Bourdev and Brandt [8] by proposing a simple *semi-supervised* approach: their key observation was that if a single sample survives in the neighborhood of a true positive no loss is incurred. Our approach is a natural generalization to the *unsupervised* case, and indeed, the above mechanism shares similarities with [9]; the primary advantage being that no additional labeled data is needed.

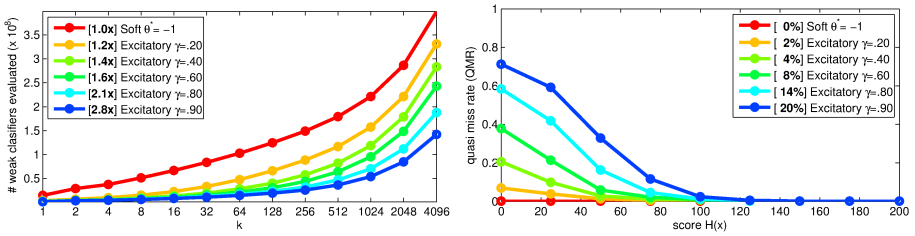

Fig. 7. Excitatory cascades effectively reduce computation for *small* k. See text.

4.2 Excitatory Cascades

The goal of excitatory cascades is to generate a set of candidates $\mathcal{X}_c$ that contains every quasi-positive. Let $\mathcal{X}_g$ denote all x sampled in a grid with a step size half the size of $\mathcal{N}$. For $\mathcal{N} = 7 \times 7 \times 3$, the step size is $(3, 3, 1)$, meaning that one in nine x is in $\mathcal{X}_g$. Initially $\mathcal{X}_c = \emptyset$. Now, for each $x \in \mathcal{X}_g$, computation of $H(x)$ continues until: (1) $H_k(x) < \theta^*$, (2) the maximum excitation stage $k > k^*$ is reached, or (3) $H_k(x) > \theta_k^E$ in which case x and all $x' \in \mathcal{N}(x)$ are added to $\mathcal{X}_c$.

We now describe the unsupervised approach for learning θ_k^E and maximum excitation stage k^* given H, x_i, and target QMR γ. We describe only the 2D case (single scale) and begin with learning θ_k^E for a single excitation stage $k^* = k$. Let $\mathcal{X}_g^o$ denote a sampling grid offset by o relative to x_i. Given a step size (s_x, s_y), there are $s_x s_y$ possible offsets o for $\mathcal{X}_g^o$ (including one case where $x_i \in \mathcal{X}_g^o$). We treat every sampling grid $\mathcal{X}_g^o$ as a separate, equally likely possibility. Let h_i^o be the maximum score $H_k(x_i')$ of all the $x_i' \in \mathcal{N}(x_i) \cap \mathcal{X}_g^o$. Because the step size is half the size of $\mathcal{N}$, at least $n \geq 4$ neighbors of x_i will be in $\mathcal{X}_g^o$ regardless of the offset o, however, the number of neighbors of x_i that survive the soft cascade may be fewer. If all the $x_i' \in \mathcal{N}(x_i) \cap \mathcal{X}_g^o$ were rejected by the soft cascade set $h_i^o = -\infty$, otherwise set $h_i^o = \max(H_k(x_i'))$. Next, compute:

$$\theta_k^E = [\mathcal{H}_k]_r - \epsilon \quad \text{where } r = \lfloor \gamma \cdot |\mathcal{H}_k| \rfloor \text{ and } \mathcal{H}_k = \{h_i^o\}. \tag{4}$$

If $\theta_k^E \neq -\infty$, then by construction on average $1 - \gamma$ quasi-positives will end up in $\mathcal{X}_c$. Unfortunately $\theta_k^E = -\infty$ implies no such threshold exists.

The above procedure fails if the excitation stage is too late. Recall from §2.1 that a detector's region of support (ROS) decreases with increasing k while its discriminative power increases. This causes a tension between performing excitation early, when discriminability is low, and performing excitation late when the ROS is small. We can find the *last* valid excitation stage by starting with $k = K$ and working backwards until θ_k^E is valid. Finally, θ_k^E for earlier stages are set to the smallest value such that $|\mathcal{X}_c|$ does not increase.

To measure error, we compute the full soft cascade (with $\theta^* = -1$) for all $x \in \mathcal{X}_c$. Results are shown in Figure 7. While overall speedup is modest, computation at early stages k is greatly reduced (a $9\times$ speedup is possible with the given step size). The excitatory cascade effectively deals with quasi-negatives (see Figure 4); to reduce computation for larger k we turn to inhibitory cascades next.

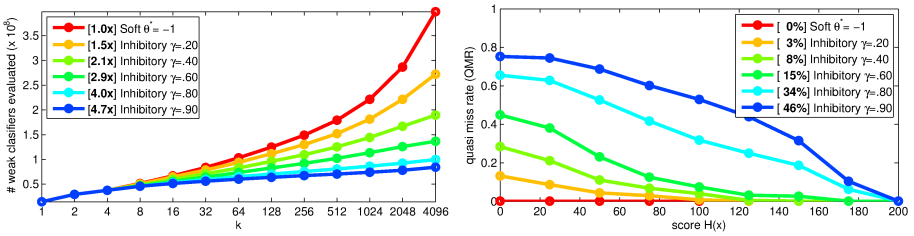

Fig. 8. Inhibitory cascades effectively reduce computation for *large* k. See text.

4.3 Inhibitory Cascades

Inhibitory cascades operate on the set of all candidates $x \in \mathcal{X}_1$ in an image. At each stage k we construct the set $\mathcal{X}_{k+1}$ by including $x \in \mathcal{X}_k$ in $\mathcal{X}_{k+1}$ only if $H_k(x) > \theta^*$ and x is not inhibited by any of its neighbors that are still in $\mathcal{X}_k$. Specifically, $x \in \mathcal{X}_k$ is inhibited (not added to $\mathcal{X}_{k+1}$) if for any $x' \in \mathcal{N}(x) \cap \mathcal{X}_k$ with $H_k(x') > \theta^*$ the ratio $H_k(x)/H_k(x') < \theta^I_k$. Observe that previously inhibited x' (i.e. $x' \notin \mathcal{X}_k$) cannot inhibit x (although x' inhibited in the current stage can). To protect against $H_k(x) \leq 0$ we add the constant $\epsilon - \theta^*$ to both $H_k(x)$ and $H_k(x')$ prior to computing $H_k(x)/H_k(x')$. After the last stage $k = K$, the set $\mathcal{X}_{K+1}$ contains all x that survive the inhibitory cascade.

We now describe the approach for learning θ^I_k given H, x_i, and target QMR γ. The procedure bears resemblance to learning soft cascade rejection thresholds (see §4.1). We begin by defining $R_k(x) = \min_{x' \in \mathcal{N}(x)}\{H_k(x)/H_k(x')\}$. Let $\gamma' = 1 - (1-\gamma)^{1/K}$, $\mathcal{X}'_k$ denote the set of surviving quasi-positives in stage k, and $\mathcal{H}_k = \{R_k(x_i) | x_i \in \mathcal{X}'_k\}$. We set θ^I_k via:

$$\theta^I_k = \min\left\{1, [\mathcal{H}_k]_r/(1+\epsilon)\right\} \quad \text{where } r = \lfloor \gamma' \cdot |\mathcal{H}_k| \rfloor. \tag{5}$$

$\mathcal{X}'_1$ contains all quasi-positives and we set $\mathcal{X}'_k = \{x_i \in \mathcal{X}'_{k-1} | R_{k-1}(x_i) > \theta^I_{k-1}\}$.

Claim: the inhibitory cascade with θ^I_k as defined above has QMR of at most γ. Proof: it is not hard to see that $\mathcal{X}'_k \subseteq \mathcal{X}_k$ if we run the inhibitory cascade with thresholds θ^I_k. However, by construction $|\mathcal{X}'_k| \geq |\mathcal{X}'_{k-1}| \cdot (1-\gamma')$. Following the proof in §4.1, this implies the QMR is at most γ. ■

Results for various choices of γ are shown in Figure 8. A value of $\gamma = .60$ results in a 2.9× speedup over using a constant soft cascade; larger γ yield bigger speedups but the QMR starts to rise considerably even at higher scores. Nearly all computational savings for inhibitory cascades occur at later stages k.

4.4 Crosstalk Cascades

We now describe the details of how to combine soft, excitatory and inhibitory cascades into *crosstalk cascades*. Excitatory cascades reduce computation for *small* k, inhibitory cascades reduce computation for *large* k, and soft cascades reduce computation across all k. Their combination proves very effective.

Incorporating per-stage rejection thresholds θ^R_k into both excitatory and inhibitory cascades is simple. Both cascade types use θ^* as a rejection criteria,

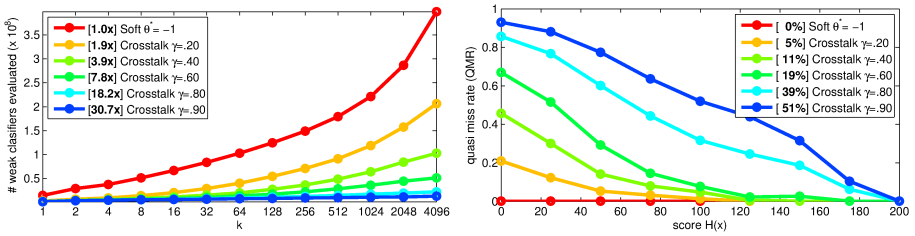

Fig. 9. Crosstalk cascades effectively reduce computation for *all* k by combining soft, excitatory and inhibitory cascades and can achieve dramatic speedups. See text.

replacing θ^* with θ_k^R in the algorithm definitions is valid and achieves a reduction in computation (but also an increase in error). Combining excitatory and inhibitory cascades is straightforward as well. Excitatory cascades generate a sparse set of candidates $\mathcal{X}_c$ while inhibitory cascades operate on an initial candidate set $\mathcal{X}_1$. We can therefore use the output of excitatory cascades as input to the inhibitory cascades and set $\mathcal{X}_1 = \mathcal{X}_c$. Additionally, evaluations of $H(x)$ used to compute $\mathcal{X}_c$ can be cached and re-used in the inhibitory stage.

To adjust the crosstalk cascade we use a single γ for setting the rejection, excitation, and inhibition thresholds separately. We leave joint optimization of the thresholds for future work.

Results for various choices of γ are shown in Figure 9. Crosstalk cascades achieve dramatic speedups, e.g. for $\gamma = .8$ computation time is reduced by $\sim$**20**$\times$ fold (compared to about 2.5$\times$, 2$\times$, and 4$\times$ for the individual cascade types). The speedups for crosstalk cascades are close to the product of the individual speedups, especially for lower γ. The QMR is higher, but not drastically so. We examine the true detection accuracy of crosstalk cascades next.

5 Evaluation

We begin by evaluating strategies for setting per-stage rejection thresholds θ_k^R for soft cascades, including our unsupervised approach. We compare three strategies. (1) First, we simply *recalibrate* our trained detector as described in §3.1 by setting $\theta_k^R = \theta^* + \phi k/K$ for various ϕ; this strategy has no data-driven component. (2) Next we utilize a *semi-supervised* approach inspired by the work of Zhang and Viola [9]. For each positive training example x_i, we select the $x_i' \in \mathcal{N}(x_i)$ with maximum scoring response $H(x_i')$, giving us a new set of examples $\mathcal{X}_{semi}$; to set θ_k^R we simply utilize the algorithm in §4.1 but with $\mathcal{X}_{semi}$ in place of the quasi-positives. While not identical to [9], this approach is similar in spirit. (3) Finally we utilize our *unsupervised* approach described in §4.1, varying γ.

Results for soft cascades on INRIA [22] are shown in Figure 10 (left). For each strategy we sweep over γ (or ϕ) and plot the resulting log-average miss rate (MR) versus speedup relative to soft cascades with $\theta_k^R = \theta^*$. Legend brackets list MR at 4$\times$ speedup. All approaches achieve a relative speedup of around 2$\times$ with no loss in accuracy, but for higher speedups the proposed unsupervised approach to soft cascades has significantly lower error due to its data-driven nature and availability of a large amount of unsupervised data.

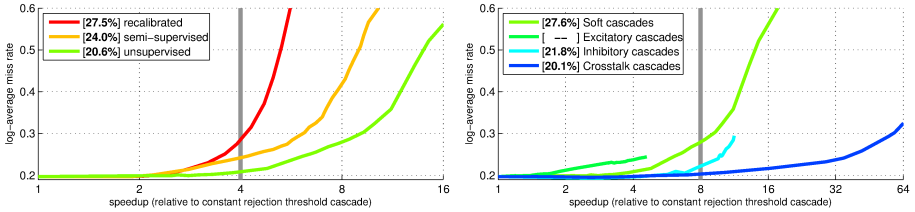

Fig. 10. **Left**: Effectiveness of strategies for setting per-stage rejection thresholds for soft cascades. For each strategy we sweep over γ (or ϕ) and plot the resulting MR versus relative speedup (legend brackets list MR at 4× speedup). Our *unsupervised* approach to learning soft cascades outperforms other baselines. **Right**: MR versus relative speedup for soft, excitatory, inhibitory, and crosstalk cascades as γ varies (legend lists MR at 8× speedup). In isolation, excitatory and inhibitory cascades provide little benefit, but when coupled the resulting crosstalk cascade achieves dramatic speedups.

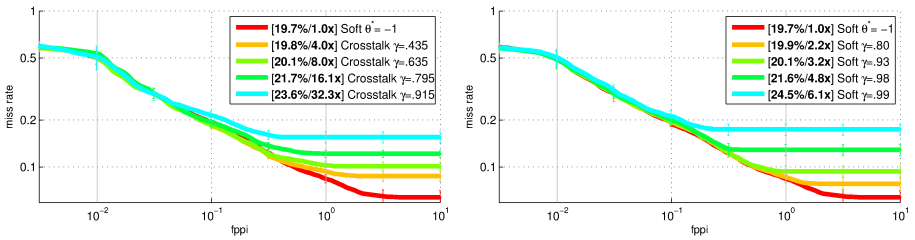

Fig. 11. **Left**: Crosstalk cascades evaluated on INRIA for multiple settings of γ (MR and speedup given in brackets). Crosstalk cascades achieve a speedup of **4**× with no loss in performance, **8**× speedup for a loss under .5% MR, and speedups between $\mathbf{16-32}\times$ if larger errors (2-4%) are acceptable. **Right**: For each soft cascade of approximately equal accuracy (matched by color), the corresponding crosstalk cascade is much faster.

Results on INRIA for crosstalk cascades are shown in Figure 10 (right). *Crosstalk cascade achieve large gains, giving a speedup of* **4**× *over the baseline with no loss in accuracy,* **8**× *speedup for a loss under .5% MR, and speedups between* $\mathbf{16-32}\times$ *for somewhat larger errors of 2-4%.* Figure 11 shows ROC curves for crosstalk cascades with γ's set to achieve 4-32× speedups and corresponding soft cascades with γ's set to match the error of the crosstalk cascades. The relative speedup of the matching crosstalk cascades is 2-5 times higher.

Using a larger spatial step size can provide additional speedups for soft cascades. We evaluate soft cascades with step sizes that range from 4-12 pixels (the default is 4 pixels). As before, for each variant we sweep over γ and plot MR versus relative speedup, see Figure 12. Crosstalk cascades, using the default 4 pixel step size, outperform soft cascade regardless of their spatial step size.

Crosstalk cascades operate at **45 fps** while matching state-of-the-art detection accuracy and **55-65 fps** at slightly higher MR. Complete results on INRIA along with a comparison to the state of the art are shown in Figure 13. Crosstalk cascades are over 5 times faster than any previously published results. Results on additional pedestrian datasets are shown in Figure 14; on all datasets crosstalk cascade match or outperform the state-of-the-art while running much faster.

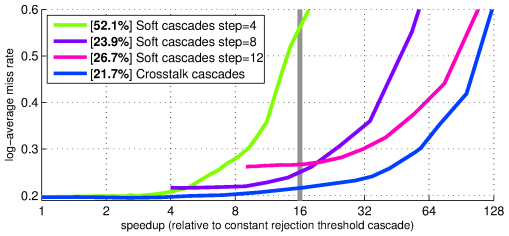

Fig. 12. No setting of the spatial step size and γ for soft cascades can simultaneously match the speed and accuracy of crosstalk cascades (legend lists MR at 16× speedup).

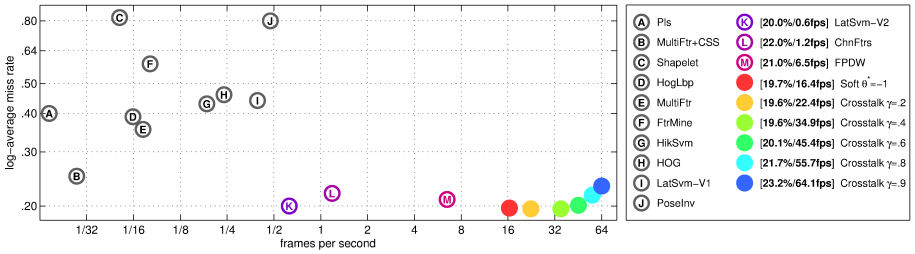

Fig. 13. Log-average miss rate (MR) versus speed (measured in fps) for various detectors on INRIA. Method runtimes were obtained from [6], see also [6] for detector citations. Legend brackets show MR/speedup for select methods. Crosstalk cascades for all setting of γ are much faster than any competing approach. At $\gamma = .6$ **crosstalk cascades have state of the art accuracy while operating at 45 fps.**

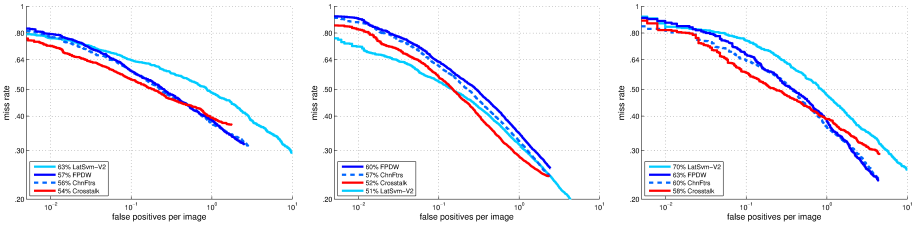

Fig. 14. From left to right: results on the Caltech, ETH, and TUD-Brussels pedestrian datasets (see [6]). Shown competing approaches are ChnFtrs [20], FPDW [4] and LatSvm [23]. On all datasets, crosstalk cascades (with $\gamma = .6$) match or outperform the state-of-the-art (except at high false positives) while running at much higher speeds. Complete results along with comparisons to numerous additional algorithms are available at `www.vision.caltech.edu/Image_Datasets/CaltechPedestrians`.

6 Discussion

In this paper we: (1) analyzed cascades and experimentally demonstrated lower and upper bounds on their performance, (2) proposed a novel approach to more effectively learn soft cascade rejection thresholds using an unsupervised approach, and (3) introduced crosstalk cascade that enable neighboring detectors to communicate and thereby achieve major computational gains. Our approach is simple and effective and achieves faster than frame-rate detection.

References

1. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade object detection with deformable part models. In: CVPR. (2010)
2. Pedersoli, M., Vedaldi, A., Gonzalez, J.: A coarse-to-fine approach for fast deformable object detection. In: CVPR. (2011)
3. Lampert, C.H., Blaschko, M.B., Hofmann, T.: Efficient subwindow search: A branch and bound framework for object localization. PAMI **31** (2009) 2129–2142
4. Dollár, P., Belongie, S., Perona, P.: The fastest pedestrian detector in the west. In: BMVC. (2010)
5. Benenson, R., Mathias, M., Timofte, R., Van Gool, L.: Pedestrian detection at 100 frames per second. In: CVPR. (2012)
6. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian detection: An evaluation of the state of the art. PAMI **99** (2011)
7. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: CVPR. (2001)
8. Bourdev, L., Brandt, J.: Robust object detection via soft cascade. In: CVPR. (2005)
9. Zhang, C., Viola, P.: Multiple-instance pruning for learning efficient cascade detectors. In: NIPS. (2007)
10. Xiao, R., Zhu, L., Zhang, H.: Boosting chain learning for object detection. In: ICCV. (2003)
11. Šochman, J., Matas, J.: Waldboost - learning for time constrained sequential detection. In: CVPR. (2005)
12. Masnadi-Shirazi, H., Vasconcelos, N.: High detection-rate cascades for real-time object detection. In: ICCV. (2007)
13. Zhu, Q., Avidan, S., Yeh, M., Cheng, K.: Fast human detection using a cascade of histograms of oriented gradients. In: CVPR. (2006)
14. Butko, N., Movellan, J.: Optimal scanning for faster object detection. In: CVPR. (2009)
15. Gualdi, G., Prati, A., Cucchiara, R.: Multi-stage sampling with boosting cascades for pedestrian detection in images and videos. In: ECCV. (2010)
16. Gualdi, G., Prati, A., Cucchiara, R.: A multi-stage pedestrian detection using monolithic classifiers. In: AVSS. (2011)
17. Felzenszwalb, P., Huttenlocher, D.: Efficient matching of pictorial structures. In: CVPR. (2000)
18. Fleuret, F., Geman, D.: Coarse-to-fine face detection. IJCV **41** (2001) 85–107
19. Vempati, S., Vedaldi, A., Zisserman, A., Jawahar, C.V.: Generalized RBF feature maps for efficient detection. In: BMVC. (2010)
20. Dollár, P., Tu, Z., Perona, P., Belongie, S.: Integral channel features. In: BMVC. (2009)
21. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. The Annals of Statistics **38** (2000) 337–374
22. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
23. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part based models. PAMI **99** (2009)
24. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL visual object classes (VOC) challenge. IJCV **88** (2010) 303–338