Timely R-CNN Detection

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a novel cascaded approach for speeding up convolutional neural networks and evaluate it on the recently introduced R-CNN object detection system. R-CNN has excellent detection accuracy, but is slow, running at 10 seconds per image on a GPU. Most of R-CNN's time is spent classifying roughly 2k regions of interest per image without sharing computation between them. Our approach introduces a *reject* option between layers in the CNN, allowing the network to terminate computation early, as in a traditional cascade. We also investigate strong baselines for speeding up R-CNN, including a novel approximation to R-CNN that uses a feature pyramid to amortize region computation. These baselines validate the efficacy of the Cascaded CNN. We achieve a 8x speed-up while losing no more than 10% of the top R-CNN detection performance – or a 16x speed-up while losing no more than 20%.

1 Introduction

Multi-layered convolutional models (CNNs) have recently demonstrated impressive levels of performance on visual detection tasks, by applying deep CNNs trained on object recognition tasks to multiple candidate image windows in a scene [1, 2]. While convolutional models are inherently efficient to implement, as convolution itself is a parallelizable and long studied computational primitive, the amount of work to perform complete inference at all candidate region windows is considerable.

The standard method for quickly applying a CNN to a large number of candidate regions is to restrict the regions to lie in a regular scale-space pyramid. With this restriction, the CNN can efficiently compute features for all regions using convolution. This approach was used in classical CNN-based detection systems, as well as recent approaches such as OverFeat [1]. The recently proposed Regions with CNN features (R-CNN), in contrast, allows for more general region proposals guided by low-level segmentation cues ([3]), but repeats the computation of convolutional features across numerous overlapping windows without sharing computation. While R-CNN has leading performance on PASCAL and ImageNet detection as of the date of this writing, it is rather slow, at 10s per image on a GPU (not counting region generation).

In this paper, we investigate a novel approach for speeding up R-CNN by proposing a general technique for accelerating CNNs applied to class imbalanced data (such as in object detection). We bring the classic idea of the cascade to CNNs by inserting a *reject* option between CNN layers. When the CNN processes batches of images, which is standard for many applications, the reject layers allows the CNN to "thin" the batch as it progresses through the network, thus saving processing time.

We also consider a variety of strong baselines for speeding up R-CNN. One of these baselines is a novel approach that approximates R-CNN by first building a feature pyramid, which is inherently efficient, and then using the pyramid to share features between overlapping region proposals. While this methods performs quite well on its own, we show that our Cascaded CNN provides a better speed-up with a simpler approach.

We discuss related work in section 2, cover all parts of our proposed method in section 3, and present evaluation results on the PASCAL VOC in section 4.

2 Related Work

Object recognition with CNN The recent success of deep convolutional neural networks (CNN) such as Alexnet [4] on image classification tasks such as ImageNet [5] has prompted many attempts to apply these computationally expensive methods to detection [2, 6, 7, 1] .

These previous attempts aim at reducing computational cost by reducing the search space of sliding window search by a bottom up window proposal scheme [2], reducing computational overhead by sharing computation between windows to be evaluated [6, 1], or deriving a CNN-based window sampling scheme [7]. All these method have a fixed strategy and are unable to adapt their computational scheme based on intermediate results, while we propose a dynamic scheme that steers computation based on intermediate outcomes in order to spend computation time most effectively.

Cascaded detection Detection cascades [8, 9] are a well established techniques to condition the computational scheme on intermediate outcomes based on a fixed evaluation order of features/classifiers. This idea has seen several refinements including sharing computation across cascades [10] and the possibility to skip evaluations [11]. Recently, a CNN with a cascaded structure for coarse to fine inference of facial feature points has been proposed [12] as well as a boosted cascade trained on CNN features [6]. While previous cascade schemes rely on a fixed ordering of evaluations, our scheme learns to traverse the available features/classifiers in arbitrary orders.

Dynamic selection Most recently, learnt feature and classification deployment schemes based on budget aware policies have shown how to flexibly adapt computation at test time dependent on the input and intermediate results [13, 14]. In particular, we combine dense evaluation of shared, cheap feature with a fine-grained, dynamic execution policy reasoning on window proposals for more expensive features.

3 Method

We consider first approaches which can effectively reorder the sequence of regions to maximize the chance that correct detections will be found, based on inference from relatively lightweight features. We show that basic strategies, such as simply randomly reordering the boxes such that they do not have a degenerate spatial layout, provides a surprising boost, and that very simple features such as region and gradient statistics can provide some signal that is useful to prioritize regions, but that those are relatively weak.

Features based on convolutional mid-level primitives are more powerful, as they are directly related to our final detector. We consider approximate measures computed from a grid of mid-level features; this proves to be a powerful signal, and provides a clear boost when the time required to compute that feature is not considered. But even with a fast image pyramid to compute them globally for all regions, the time required is unfortunately large enough to significantly dilute their impact.

What is thus desired is a method which can compute an early feature which indicates the region priority, and then can use that same feature in the subsequent classification, with minimal computation that cannot be so amortized. We formulate a cascade version of the RCNN architecture which has these desirable properties, and show that either alone or in concert with other methods below, it can effectively learn a "Timely" approach to the detection computation.

We first review the base model that our work is derived from, the R-CNN method, in the following subsection. We then present our approach to region prioritization using simple features based on basic region and gradient properties in subsection 3.2 and subsection 3.3. We then consider mid-level features in subsection 3.4, based on a Pyramid scheme which amortizes the computation across overlapping regions, but does not use them for the subsequent computation. Finally, we present our final model, the Cascade-CNN, in subsection 3.5 which combines elements of all of these approaches into a unified scheme: it can prioritize based on low-level features early in the CNN, or expoit more costly mid-level features, and when it has decided to carry the computation to the final detection

it does so effectively re-using the computation from the earlier tests, and therefore does not waste additional time.

3.1 Base model

We take the Region-CNN (R-CNN) [2] method as our point of departure. As summarized in Figure 1, R-CNN starts with external region-of-interest proposals (ROIs). ROIs are taken in batches: transformed to canonical size, and classified with a CNN, obtaining multi-class scores for each region. After all batches have been scored, ROIs are post-processed with non-maximal suppression and other bounding box refinement to obtain the final detections.

We start with the same ROI proposals, but first score all proposals with a quick-to-compute feature. We then select a batch of highly-scoring ROIs and classify them with the CNN – which can additionally be sped up with cascaded structure. After the batch is scored, we optionally re-score the remaining ROIs, and repeat the process until we're either out of time, or out of regions. Figure 2 summarizes the process.

As shown in Figure 2, the quick-to-compute features are used to select batches of regions in a way that orders the regions most likely to contain ground truth objects earlier than other regions. This process of region selection does not reject regions flat out, but reorders them to be processed later.

We first tried to simply sort the regions by score and take batches in that sorted order. Surprisingly, this naive method results in poor performance, and in fact can be worse than taking regions in an entirely random order. We hypothesize that this is because the highest scoring regions may all be overlapping with each other, and only result in one detection after non-maximal suppression and other post-processing of detections. Guided by this intuition, we consider a more dynamic approach to region selection: we put regions in a random order, set a threshold such that only half of the unexamined regions score above it, take a batch of the above-threshold regions in their order, update the threshold, and repeat.

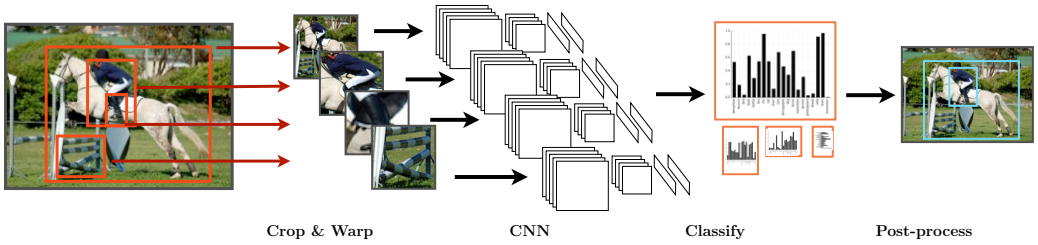


Figure 1: R-CNN architecture: image regions are cropped, resized, and each one fed through a CNN with classification layers. The classifier outputs are post-processed to give the final detections.

3.2 Region statistics

For the quick-to-compute feature, we consider statistics about ROI location and overlap with other regions. We compute a very simple feature for each ROI: its normalized location, its scale($\sqrt{\text{width} \times \text{height}}$), aspect ratio ($\log \sqrt{\text{width}/\text{height}}$), and normalized counts of overlapping regions, at different PASCAL overlap thresholds (0, 0.2, 0.6).

This simple feature works suprisingly well for filtering regions to process first, and is always included in concatenation with the more complex features that are described next.

3.3 CNN pixel gradient computation

One fast-to-compute feature we consider for the preliminary featurization of ROIs is the pixel gradient, back-propagated throgh the classifier CNN applied to the whole image. This gradient corresponds to a kind of saliency map onto the image [7], and can therefore be used to evaluate ROIs.

We have fine-tuned an "AlexNet" [4] CNN using the PASCAL VOC 2007 classification challenge. Unlike the ILSVRC, the PASCAL VOC is a multi-label prediction task, with at times multiple

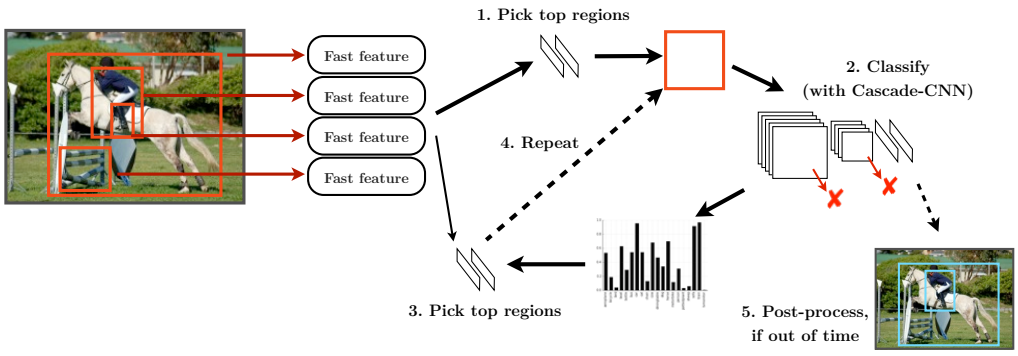


Figure 2: Our method scores each region of interest with a fast feature (we evaluate several), allowing us to pick promising regions first. The regions are classified with the original CNN, or a sped-up Cascade-CNN. The properties of the regions can play a role in selecting the next batch of regions.

correct labels for an image. Accordingly, we implement a cross-entropy loss layer to train a simple multi-label classifier with binary vector.

This pixel-wise gradient can be seen as a first-order approximation to the weight of a pixel in the classification output. To compute it, we first run the CNN forward from image input to the prediction layer. Back-propagating through the model and a last backward pass to the input gives the pixel-wise gradient. A couple of example images are shown in Figure 6.

We compute an integral image on this pixel gradient map, allowing near-instant computation of sums in arbitrary regions. For each region to be evaluated, we compute the total gradient sum, sums for each of the corners, and proportions of in-region vs. out-of-region sums.

3.4 Pyramid amortized R-CNN

Another fast ROI featurization we consider is a variant of R-CNN that processes the whole image, at multiple scales, with the CNN up to the topmost non-fully-connected layer. We call this multiscale CNN output a "feature pyramid". As Figure 3 shows, ROIs are then cropped from the feature pyramid at the best matching scale, warped to a canonical size, and classified. ROIs are highly overlapping, but their featurization is shared through the pyramid, amortizing its construction cost. This doesn't work as well as the original R-CNN system, but is much faster, and can therefore be used as the fast feature for the dynamic region selection.

We explored a variety of choices while designing Pyramid R-CNN. We tried using a single scale or a pyramid with 7 levels each separated by a scale factor of $2^{-1/2}$. For warping, we experimented with canonical sizes of $s \times s$, for $s \in \{5, 6, 7\}$ and resampling with nearest neighbor or bilinear interpolation. We also tried two variants of the feature pyramid: the raw feature pyramid or one where each level is max pooled with a 3×3 pooling window run at a stride of 1 (to avoid subsampling).

Table 1 shows mAP performance on PASCAL VOC 2007 test for these various choices. The best configuration, in terms of mAP, uses a 7 level pyramid, a warp size of 7×7 with bilinear interpolation, and max pooling. The first level of our pyramid is computed from a 1713×1713 pixel image, which yields a 108×108 cell feature map. The gold standard (non-pyramid) R-CNN performance using the same non-fine-tuned CNN is 44.2% mAP. Our best result with Pyramid R-CNN is slightly worse, at 41.9%.

To map a ROI R into the pyramid, we start by computing an "optimal" scale defined by $\alpha^* = 227/\min(h, w)$, where h and w are image height and width of R. We then find the nearest level in the pyramid: $l^* = \text{argmin}_l |\log(\alpha_l) - \log(\alpha^*)|$, where α_l is the scale factor for pyramid level l. This procedure approximates the scale that R-CNN would use, since it operates on 227 pixel inputs.

Table 1: Pyramid R-CNN design choices, vs mAP performance. We additionally show two best and two worst performing classes.

settings	warp 7x7 nearest	warp 7x7 nearest	warp 7x7 bilinear	max pooled 3x3 warp 7x7 bilinear
scales	1	7	7	7
bicycle	36.1	50.1	52.6	57.9
car	39.8	54.3	56.7	60.0
chair	6.7	11.7	11.8	16.0
pottedplant	11.4	17.1	18.6	20.9
mAP	18.7	34.9	37.4	41.9

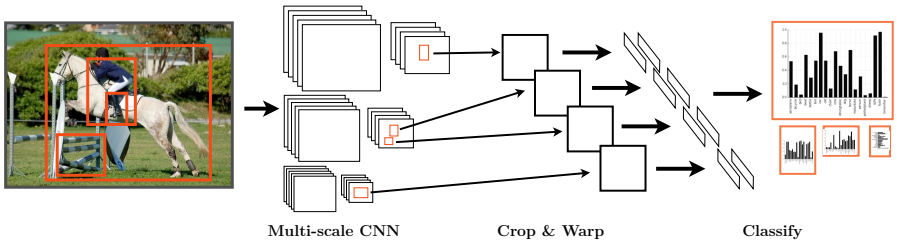


Figure 3: Post R-CNN architecture: the whole image is fed through a CNN up to the highest pooling layer. Regions are cropped from that layer at the best matching scale, resized, and classified.

3.5 Cascaded CNN

Most ROIs that go through the CNN in R-CNN do not contain objects of interest, and are simply background. It would be useful to quickly reject these regions, without expending the full amount of computation on them. Since most of the time is spent in the convolutional layers, we introduce a reject option after `pool1`, `pool2`, and `conv3`, this allows skipping the computation of the following convolutions. These reject layers were trained using the corresponding fine-tuned CNN.

The Cascaded CNN, shown in Figure 4, augments the CNN network with an "Early Reject" option: after some layers, the network decides whether to keep computing the input with the next layer, or to reject the region since it is background. For our purpose, the reject layers are trained to separate Background/Foreground, terminating if they are confident that the input is Background. The last classification layer still outputs the full multi-class scores for the surviving regions.

To set the thresholds we took a sample of 140K regions from the images in the validation set containing positive and negative examples. Then we set the threshold for the rejector after first layer to have 0.8 recall on the positive regions. Similarly we set the threshold in for the rejector after the second layer, using the regions that pass the first rejector, to maintain a 0.8 recall on positive regions. And finally we set the threshold for the rejector after the third layer, using the regions that pass the first two rejectors, to maintain a 0.8 recall on positive regions. After every rejection layer the precision on positive regions increase, going from 0.38 to 0.74, while maintaining a recall of 0.8.

To estimate the saving on time by using the rejectors we timed the time spend to process 1000 regions (10 batches or 100) and the time expended in each of the first 3 layers:

- 1700 ms to process all the layers
- 270 ms (15%) in layer 1 (This includes `conv1`, `relu1`, `pool1`, `norm1`)
- 360 ms (20%) in layer 2 (This includes `conv2`, `relu2`, `pool2`, `norm2`)
- 285 ms (15%) in layer 3 (This includes `conv3`, `relu3`)

Therefore the expected "lifetimes" of regions rejected after layer 1, layer 2, layer 3 and those not rejected are 0.15, 0.35, 0.5, 1.0 of the total time taken per region (1.7 ms).

At test time, we observe how many regions out of the batch size of 100 survive each round of thresholding, and then have a weighted average of a region lifetime in that batch. We multiply the standard batch time of 500ms by that average region lifetime to arrive at the batch time of the cascaded CNN.

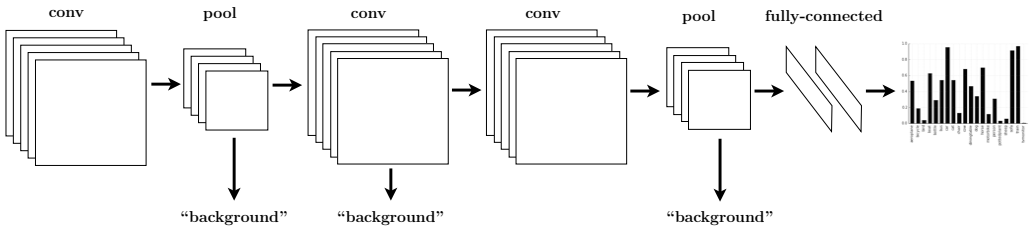


Figure 4: The Cascaded CNN has a reject option after pool1, pool2 and conv3 layers.

4 Evaluation

We evaluate on the standard object detection benchmark: the PASCAL VOC [15]. In all cases, the CNN region classifiers are trained on the PASCAL VOC 2007 trainval set. The parameters of our methods are set by training or cross-validation on the VOC 2007 val set. We evaluate on the VOC 2007 test set.

The R-CNN software was used as available in June 2014[1]. That software relies on Selective Search [3] region proposals. Different images are proposed different numbers of regions. Figure 6 shows the distribution of number of regions on the validation set, with the parameters of the R-CNN. An additional parameter is the size of each batch of regions that goes through the CNN. We set batch size to 100 regions, and observe that it takes on average 500 ms to process them with the CNN. In all experiments, we use Ubuntu 12.04, Intel i5 3.2GHz CPU, and NVIDIA Tesla K40 GPU.

The scoring function for the Region Statistics, Pixel Gradient, and Pyramid R-CNN features is trained by a logistic regression classifier onto the supervised signal of region overlap with ground truth on the validation dataset. The classifier is optimized by stochastic gradient descent, and its regularization parameter is cross-validated.

The result plots and details are shown in Figure 5 and Table 2.

Table 2: Full table of AP vs. Time results on PASCAL VOC 2007. Best performance for each time point is in bold.

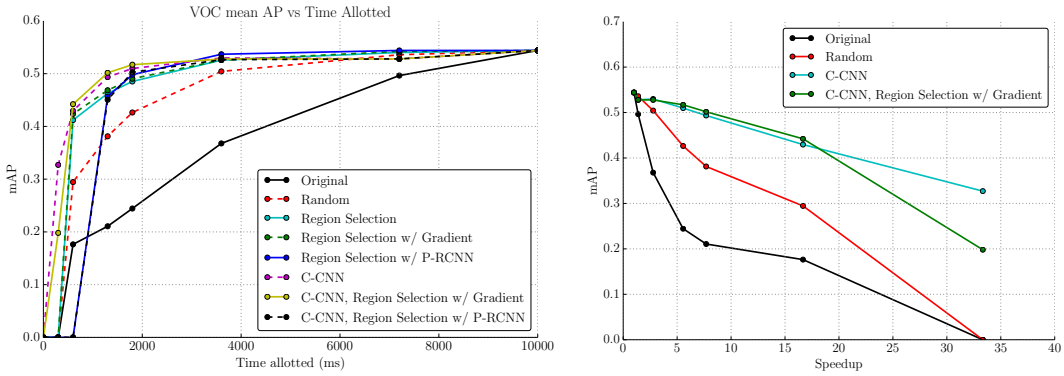
Time allotted (ms)	0	300	600	1300	1800	3600	7200	10000
Original	0	0.000	0.176	0.211	0.244	0.368	0.496	0.544
Random	0	0.000	0.295	0.381	0.426	0.504	0.536	0.544
Region Selection	0	0.000	0.412	0.463	0.485	0.526	0.540	0.544
Region Selection w/ Gradient	0	0.000	0.424	0.469	0.490	0.526	0.542	0.544
Region Selection w/ P-RCNN	0	0.000	0.000	0.457	0.498	**0.537**	**0.544**	0.544
C-CNN	0	**0.327**	0.430	0.493	0.510	0.530	0.528	0.544
C-CNN, Region Selection w/ Gradient	0	0.198	**0.442**	**0.502**	**0.517**	0.528	0.528	0.544
C-CNN, Region Selection w/ P-RCNN	0	0.000	0.000	0.451	0.503	0.527	0.528	0.544

4.1 Experiments

Original

The original order of the Selective Search regions of interest. This order is influenced by the hierarchical segmentation of their method, and so has sequences of highly overlapping regions.

[1] https://github.com/rbgirshick/rcnn



(a) mAP vs. time allotted for detection

(b) mAP vs. speed-up factor

Figure 5: Results on the PASCAL VOC 2007 dataset. (a) On the left-hand mAP vs. Time plot, we can compare APs at a given allotted time point. For example, at 1300 ms, random region selection gets about 0.42 mAP, while our best method (C-CNN with gradient-based region selection) obtains 0.50 mAP. (b) On the right-hand speed-up plot, we can compare speed-ups at a given mAP point. For example, we can see that we should obtain mAP of 0.40 at around 20x speedup with our method.

Random
A completely blind permutation of the original order.

Region Selection
The region statistics feature from subsection 3.2 is always used. Additionally, we consider either the Pixel Gradient and the Pyramid R-CNN features. These two features have a *setup time*: the gradient forward-back propagation takes 20 ms, and the Pyramid R-CNN takes a whopping 800 ms.

Cascaded CNN
The Cascaded CNN model, as described in subsection 3.5. The first experiment (C-CNN) takes batches of regions in a random order. The next two experiments also make use of the Region Selection methodology, and the two fast features of Pixel Gradient and the Pyramid R-CNN.

4.2 Analysis

Since the time to process a full batch with a non-cascaded CNN is 500 ms, there are no results for non-cascaded baselines at 300 ms. At this time, the Cascaded CNN without any region ordering is best. A reason for why C-CNN with Region Selection is not as good at this point is that the region selection presents better region candidates, with fewer rejection opportunities, and thus has less coverage of the image.

At 600 ms, C-CNN method have had more than one batch go through, and the Region Selection is giving it a lead over the simple C-CNN. Both method are better than the baseline non-cascaded methods for this entire duration. This changes once the Pyramid R-CNN has time to compute and to run through a few batches. The later times show best performance from the Region Selection with that feature.

5 Conclusion

We have presented approaches to speed up region-based convolutional object detectors. We consider techniques based on both re-ranking and rejection-filtering based on lightweight features of the observed image region. We re-order regions to process the most promising ones first using one of three "fast features" based on region overlap statistics, CNN pixel gradient statistics, and a convnet feature pyramid. While our work is generally applicable to any windowed detection, we specifically develop a novel cascaded approach for speeding up convolutional neural networks and evaluate it on the recently introduced R-CNN object detection system, which has excellent detection accuracy but

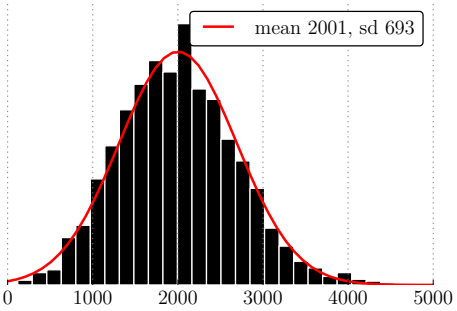


Figure 6: (left) Distribution of number of regions per image. (right) Example of the CNN gradient.

is realtively slow. We show how to effecively amortize the time R-CNN spends classifying regions of interest by sharing computation between overlapping regions, and pruning regions unlikely to be valuable. Our approach introduces a *reject* option between layers in the CNN, allowing the network to terminate computation early, as in a traditional cascade. We reported significant speed-ups with very slight performance degradation. Our scheme allows R-CNN to be used in time-critical applications.

References

[1] Pierre Sermanet and David Eigen. OverFeat: Integrated Recognition , Localization and Detection using Convolutional Networks. In *ICLR*, 2014.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[3] J R R Uijlings, K E A Van De Sande, T Gevers, and A W M Smeulders. Selective Search for Object Recognition. *IJCV*, 2013.

[4] Alex Krizhevsky, Ilyaskever Sut, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. In *NIPS*, 2012.

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[6] Will Y Zou, Xiaoyu Wang, Miao Sun, and Yuanqing Lin. Generic Object Detection with Dense Neural Patterns and Regionlets. Technical report, 2014.

[7] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks : Visualising Image Classification Models and Saliency Maps. In *ICLR*, 2014.

[8] Paul Viola, One Microsoft Way, and Michael J Jones. Robust Real-Time Face Detection. *IJCV*, 57(2):137–154, 2004.

[9] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *CVPR*, pages 2241–2248. IEEE, June 2010.

[10] Piotr Dollar, Ron Appel, and Wolf Kienzle. Crosstalk Cascades for Frame-Rate Pedestrian Detection. In *ECCV*, 2012.

[11] D. Benbouzid, R. Busa-Fekete, and B. Kegl. Fast classifica- tion using sparse decision dags. In *ICML*, 2012.

[12] Y. Sun, X. Wang, and X. Tang. Deep convolutional network cascade for facial point detection. In *CVPR*, 2013.

[13] Sergey Karayev, Tobias Baumgartner, Mario Fritz, and Trevor Darrell. Timely Object Recognition. In *NIPS*, 2012.

[14] Gabriel Dulac-Arnold, Ludovic Denoyer, Nicolas Thome, Matthieu Cord, and Patrick Gallinari. Sequentially generated instance-dependent image representations for classification. In *ICLR*, 2014.

[15] M Everingham, L Van Gool, C K I Williams, J Winn, and A Zisserman. The PASCAL VOC Challenge 2010 Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html, 2010.