Learning non-maximum suppression

Jan Hosang Rodrigo Benenson Bernt Schiele

Max Planck Institut für Informatik
Saarbrücken, Germany
firstname.lastname@mpi-inf.mpg.de

Abstract

Object detectors have hugely profited from moving towards an end-to-end learning paradigm: proposals, features, and the classifier becoming one neural network improved results two-fold on general object detection. One indispensable component is non-maximum suppression (NMS), a post-processing algorithm responsible for merging all detections that belong to the same object. The de facto standard NMS algorithm is still fully hand-crafted, suspiciously simple, and — being based on greedy clustering with a fixed distance threshold — forces a trade-off between recall and precision. We propose a new network architecture designed to perform NMS, using only boxes and their score. We report experiments for person detection on PETS and for general object categories on the COCO dataset. Our approach shows promise providing improved localization and occlusion handling.

Figure 1: We propose a non-maximum suppression convnet that will re-score all raw detections (top). Our network is trained end-to-end to learn to generate exactly one high scoring detection per object (bottom, example result).

1. Introduction

All modern object detectors follow a three step recipe: (1) proposing a search space of windows (exhaustive by sliding window or sparser using proposals), (2) scoring/refining the window with a classifier/regressor, and (3) merging windows that might belong to the same object. This last stage is commonly referred to as "non-maximum suppression" (NMS) [10, 9, 21, 7, 20, 16].

The de facto standard for NMS is a simple hand-crafted test time post-processing, which we call GreedyNMS. The algorithm greedily selects high scoring detections and deletes close-by less confident neighbours since they are likely to cover the same object. This algorithm is simple, fast, and surprisingly competitive compared to proposed alternatives.

The most notable recent performance breakthrough in general object detection was marked by R-CNN [10], which effectively replaced features extraction and classifiers by a neural network, almost doubling performance on Pascal VOC. Another significant improvement was to absorb the object proposal generation into the network [21], while other works avoid proposals altogether [21, 20], leading to both speed and quality improvements. We can see a general trend towards end-to-end learning and it seems reasonable to expect further improvements by doing complete end-to-end training of detectors. NMS is one step in the pipeline that, for the most part, has evaded the end-to-end learning paradigm. All of the above detectors train the classifier in a procedure that ignores the fact that the NMS problem exists and then runs GreedyNMS as a disconnected post-processing.

There is a need to overcome GreedyNMS due to its significant conceptual shortcomings. GreedyNMS makes hard decision by deleting detections and bases this decision on one fixed parameter that controls how wide the suppression is. A wide suppression would remove close-by high scoring detections that are likely to be false positives that hurt precision. On the other hand, if objects are close (e.g. in crowded scenes), close-by detections can be true positives, in which case suppression should be narrow to improve re-

call. When objects are close-by, GreedyNMS is doomed to sacrifice precision or recall independent of its parameter.

It is desirable to learn NMS to overcome these limitations. An NMS approach based on neural network could learn to adapt to the data distribution, overcome the trade-off of GreedyNMS, and importantly could be incorporated *into* a detector. In this paper we propose the first "pure NMS network" which is able to do the task of non-maximum suppression without image content or access to decisions of another algorithm. Our network renders the need for final GreedyNMS post-processing superfluous.

In section 3 we start by discussing with the underlying issue: why is NMS needed at all? We discuss the task of detection and how it relates to the specifics of detectors and NMS. We identify two necessary ingredients that current detectors are lacking and design an NMS network that contains these ingredients (section 4); the result is conceptually different than both GreedyNMS and current detectors. In section 5, we report promising results that show that this network is indeed capable of replacing GreedyNMS. We report both single- (PETS pedestrians) and multi-class results (COCO dataset), both showing improvements over GreedyNMS.

We believe this work opens the door to true end-to-end detectors.

2. Related work

Clustering detections. The de facto standard algorithm, GreedyNMS, has survived several generations of detectors, from Viola&Jones [32], over the deformable parts model (DPM) [7], to the current state-of-the-art R-CNN family [10, 9, 21]. Several other clustering algorithms have been explored for the task of NMS without showing consistent gains: mean-shift clustering [6, 35], agglomerative clustering [2], affinity propagation clustering [17], and heuristic variants [25]. Principled clustering formulations with globally optimal solutions have been proposed in [27, 23], although they have yet to surpass the performance of GreedyNMS.

Linking detections to pixels. Hough voting establishes correspondences between detections and the image evidence supporting them, which can avoid overusing image content for several detections [15, 1, 14, 34]. Overall performance of hough voting detectors remains comparatively low. [37, 5] combine detections with semantic labelling, while [36] rephrase detection as a labelling problem. Explaining detections in terms of image content is a sound formulation but these works rely on image segmentation and labelling, while our system operates purely on detections without additional sources of information.

Co-occurrence. One line of work proposes to detect pairs of objects instead of each individual objects in order to handle strong occlusion [24, 29, 19]. It faces an even more complex NMS problem, since single and double detections need to be handled. [22] bases suppression decisions on estimated crowd density. Our method does neither use image information nor is it hand-crafted to specifically detect pairs of objects.

Auto-context. Some methods improve object detection by jointly rescoring detections locally [30, 4] or globally [31] using image information. These approaches tend to produce fewer spread-out double detections and improve overall detection quality, but still require NMS. We also approach the problem of NMS as a rescoring task, but we completely eliminate any post-processing.

Neural networks on graphs. A set of detections can be seen as a graph where overlapping windows are represented as edges in a graph of detections. [18] operates on graphs, but requires a pre-processing that defines a node ordering, which is ill-defined in our case.

End-to-end learning for detectors. Few works have explored true end-to-end learning that includes NMS. One idea is to include GreedyNMS at training time [33, 12], making the classifier aware of the NMS procedure at test time. This is conceptually more satisfying, but does not make the NMS learnable. Another interesting idea is to directly generate a sparse set of detections, so NMS is unnecessary, which is done in [26] by training an LSTM that generates detections on overlapping patches of the image. At the boundaries of neighbouring patches, objects might be predicted from both patches, so post-processing is still required. [13] design a convnet that combines decisions of GreedyNMS with different overlap thresholds, allowing the network to choose the GreedyNMS operating point locally. None of these works actually completely *remove* GreedyNMS from the final decision process that outputs a sparse set of detections. Our network is capable of performing NMS without being given a set of suppression alternatives to chose from and without having another final suppression step.

3. Detection and non-maximum suppression

In this section we review non-maximum suppression (NMS) and why it is necessary. In particular, we point out why current detectors are conceptually incapable of producing exactly one detection per object and propose two necessary ingredients for a detector to do so.

Present-day detectors do not return all detections that have been scored, but instead use NMS as a post-processing step to remove redundant detections. In order to have true end-to-end learned detectors, we are interested in detectors without any post-processing. To understand why NMS is necessary, it is useful to look at the task of detection and how it is evaluated.

Object detection. The task of object detection is to map an image to a set of boxes: one box per object of interest in the image, each box tightly enclosing an object. This means detectors ought to return exactly one detection per object. Since uncertainty is an inherent part of the detection process, evaluations allow detections to be associated to a confidence. Confident erroneous detections are penalized more than less confident ones. In particular mistakes that are less confident than the least confident correct detection are not penalized at all.

Detectors do not output what we want. The detection problem can be interpreted as a classification problem that estimates probabilities of object classes being present for every possible detection in an image. This viewpoint gives rise to "hypothesize and score" detectors that build a search space of detections (e.g. sliding window, proposals) and estimate class probabilities *independently* for each detection. As a result, two strongly overlapping windows covering the same object will both result in high score since they look at almost identical image content. In general, instead of one detection per object, each object triggers several detections of varying confidence, depending on how well the detection windows cover the object.

GreedyNMS. Since the actual goal is to generate *exactly one* detection per object (or exactly one high confidence detection), a common practice (since at least 1994 [3]) is to assume that highly overlapping detections belong to the same object and collapse them into one detection. The predominant algorithm (GreedyNMS) accepts the highest scoring detection, then rejects all detections that overlap more than some threshold ϑ and repeats the procedure with the remaining detections, i.e. greedily accepting local maxima and discarding their neighbours, hence the name. This algorithm eventually also accepts wrong detections, which is no problem if their confidence is lower than the confidence of correct detections.

GreedyNMS is not good enough. This algorithm works well if (1) the suppression is wide enough to always suppress high scoring detections triggered by same object and (2) the suppression is narrow enough to never suppress high scoring detections of the next closest object. If objects are far apart condition (2) is easy to satisfy and a wide suppression works well. In crowded scenes with high occlusion between objects there is a tension between wide and narrow suppression. In other words with one object per image NMS is trivial, but highly occluded objects require a better NMS algorithm.

3.1. A future without NMS

Striving for true end-to-end systems without hand crafted algorithms we shall ask: **Why do we need a hand crafted post processing step? Why does the detector not directly output one detection per object?**

Independent processing of image windows leads to overlapping detection giving similar scores, this is a requirement of robust functions: similar inputs lead to similar outputs. A detector that outputs only one high scoring detection per object thus has to be also conditioned on other detections: multiple detections on the same object should be processed jointly, so the detector can tell there are repeated detections and only one of them should receive a high score.

Typical inference of detectors consist of a classifier that discriminates between image content that contains an object and image content that does not. The positive and negative training examples for this detector are usually defined by some measure of overlap between objects and bounding boxes. Since similar boxes will produce similar confidences anyway, small perturbation of object locations can be considered positive examples, too. This technique augments the training data and leads to more robust detectors. Using this type of classifier training does not reward one high scoring detection per object, and instead deliberately encourages multiple high scoring detections per object.

From this analysis we can see that two key ingredients are necessary in order for a detector to generate exactly one detection per object:

1. A *loss* that penalises double detections to teach the detector we want *precisely one* detection per object.

2. *Joint processing* of neighbouring detections so the detector has the necessary information to tell whether an object was detected multiple times.

In this paper, we explore a network design that accommodates both ingredients. To validate the claim that these are key ingredients and our the proposed network is capable of performing NMS, we study our network in isolation without end-to-end learning with the detector. That means the network operates solely on scored detections without image features and as such can be considered a "pure NMS network".

4. Doing NMS with a convnet

After establishing the two necessary requirements for a convnet (convolutional network) to perform NMS in section 3, this section presents our network that addresses both (penalizing double detections in §4.1, joint processing of detections in §4.2).

Our design avoids hard decisions and does not discard detections to produce a smaller set of detections. Instead, we reformulate NMS as a rescoring task that seeks to decrease the score of detections that cover objects that already have been detected, as in [13]. After rescoring, simple thresholding is sufficient to reduce the set of detections. For evaluation we pass the full set of rescored detections to the evaluation script without any post processing.

4.1. Loss

A detector is supposed to output exactly one high scoring detection per object. The loss for such a detector must inhibit multiple detections of the same object, irrespective of how close these detections are. Stewart and Andriluka [26] use a Hungarian matching loss to accomplish that: successfully matched detections are positives and unmatched detections are negatives. The matching ensures that each object can only be detected once and any further detection counts as a mistake. Henderson and Ferrari [12] present an average precision (AP) loss that is also based on matching.

Ultimately a detector is judged by the evaluation criterion of a benchmark, which in turn defines a matching strategy to decide which detections are correct or wrong. This is the matching that should be used at training time. Typically benchmarks sort detections in descending order by their confidence and match detections in this order to objects, preferring most overlapping objects. Since already matched objects cannot be matched again surplus detections are counted as false positives that decrease the precision of the detector. We use this matching strategy.

We use the result of the matching as labels for the classifier: successfully matched detections are positive training examples, while unmatched detections are negative training examples for a standard binary loss. Typically all detections that are used for training of a classifier have a label associated as they are fed into the network. In this case the network has access to detections and object annotations and the matching layer generates labels, that depend on the predictions of the network. Note how this class assignment directly encourages the rescoring behaviour that we wish to achieve.

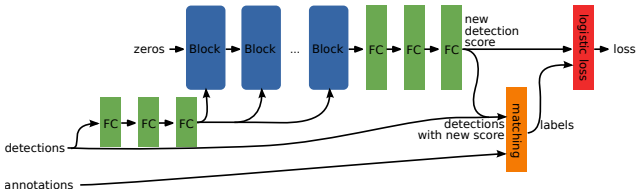
Let d_i denote a detection, $y_i \in \{-1, 1\}$ indicate whether or not d_i was successfully matched to an object, and let f denote the scoring function that jointly scores all detections on an image $f\left([d_i]_{i=1}^n\right) = [s_i]_{i=1}^n$. We train with the weighted logistic loss

$$L(s_i, y_i) = \sum_{i=1}^{N} w_{y_i} \cdot \log\left(1 + \exp\left(-s_i \cdot y_i\right)\right).$$

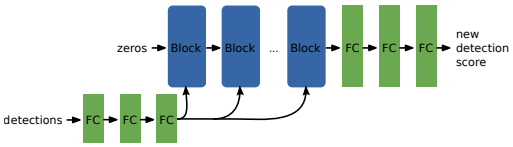
Here loss per detection is coupled to the other detections through the matching that produces y_i.

The weighting w_{y_i} is used to counteract the extreme class imbalance of the detection task. We choose the weights so the expected class conditional weight of an example equals a parameter $\mathbf{E}\left(w_1 I\left(y_i = 1\right)\right) = \gamma$.

When generalising to the **multiclass** setting, detections are associated to both a confidence and a class. Since we only rescore detections, we allow detections to be "switched off" but not to change their class. As a result, we only match detections to objects of the same class, but the classification problem remains binary and the above loss still applies.



(a) Training architecture.



(b) Test architecture.

Figure 2: High level diagram of the Gnet. FC denotes fully connected layers. All features in this diagram have 128 dimensions (input vector and features between the layers/blocks), the output is a scalar.

When representing the detection scores, we use a one-hot encoding: a zero vector that only contains the score at the location in the vector that corresponds to the class. Since mAP computation does not weight classes by their size, we assign the instance weights in a way that their expected class conditional weight is uniformly distributed.

4.2. "Chatty" windows

In order to effectively minimize the aforementioned loss, we need our network to jointly process detections. To this end we design a network with a repeating structure, which we call blocks (sketched in figure 3). One block gives each detection access to the representation of its neighbours and subsequently updates its own representation. Stacking multiple blocks means the network alternates between allowing every detection "talk" to its neighbours and updating its own representation. We call this the **GossipNet** (Gnet), because detections talk to their neighbours to update their representation.

There are two non-standard operations here that are key. The first is a layer, that builds representations for pairs of detections. This leads to the key problem: an irregular number of neighbours for each detection. Since we want to avoid the discretisation scheme used in [13], we will solve this issue with pooling across detections (the second key).

Detection features. The blocks of our network take the detection feature vector of each detection as input and outputs an updated vector (see high-level illustration in figure 2). Outputs from one block are input to the next one. The values inside this $c = 128$ dimensional feature vector are learned implicitly during the training. The output of the last block is used to generate the new detection score for each detection.

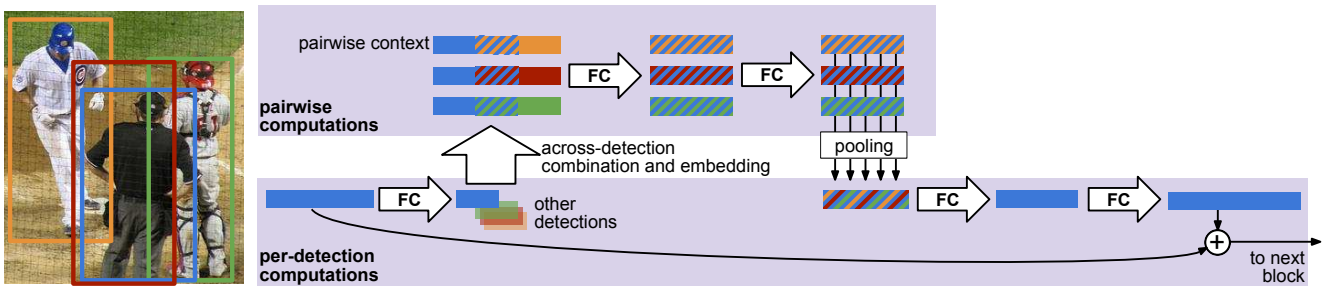


Figure 3: One block of our Gnet visualised for *one* detection. The representation of each detection is reduced and then combined into neighbouring detection pairs and concatenated with detection pair features (hatched boxes, corresponding features and detections have the same colour). Features of detection pairs are mapped independently through fully connected layers. The variable number of pairs is reduced to a fixed-size representation by max-pooling. Pairwise computations are done for each detection independently.

The first block takes an all-zero vector as input. The detections' information is fed into the network in the "pairwise computations" section of figure 3 as described below. In future work this zero input could potentially be replaced with image features.

Pairwise detection context. Each mini-batch consists of all n detections on an image, each represented by a c dimensional feature vector, so the data has size $n \times c$ and accessing to another detection's representations means operating within the batch elements. We use a detection context layer, that, for every detection d_i, generates all pairs of detections (d_i, d_j) for which d_j sufficiently overlaps with d_i (IoU > 0.2). The representation of a pair of detections consists of the concatenation of both detection representations and g dimensional detection pair features (see below), which yields an $l = 2c + g$ dimensional feature. To process each pair of detections independently, we arrange the features of all pairs of detections along the batch dimension: if detection d_i has k_i neighbouring detection that yields a batch of size $K \times l$, where $K = \sum_{i=1}^{n}(k_i + 1)$ since we also include the pair (d_i, d_i). Note that the number of neighbours k_i (the number of pairs) is different for every detection even within one mini-batch. To reduce the variable sized neighbourhood into a fixed size representation, our architecture uses global max-pooling over all detection pairs that belong to the same detection ($K \times l \rightarrow n \times l$), after which we can use normal fully connected layers to update the detection representation (see figure 3).

Detection pair features. The features for each detection pair used in the detection context consists of several properties of a detection pair: (1) the intersection over union (IoU), (2-4) the normalised distance in x and y direction and the normalised l2 distance (normalized by the average of width and height of the detection), (4-5) scale difference of width and height (e.g. $\log(w_i/w_j)$), (6) aspect ratio difference $\log(a_i/a_j)$, (7-8) the detection scores of both detections. In the multi-class setup, each detection provides a scores vector instead of a scalar thus increasing the number of pair features. We feed all these raw features into 3 fully connected layers, to learn the g detection pair features that are used in each block.

Block. A block does one iteration allowing detections to look at their respective neighbours and updating their representation (sketched in figure 3). It consists of a dimensionality reduction, a pairwise detection context layer, 2 fully connected layers applied to each pair independently, pooling across detections, and two fully connected layers, where the last one increases dimensionality again. The input and output of a block are added as in the Resnet architecture [11]. The first block receives zero features as inputs, so all information that is used to make the decision is bootstrapped from the detection pair features. The output of the last block is used by three fully connected layers to predict a new score for each detection independently (figure 2).

Parameters. Unless specified otherwise our networks have 16 blocks. The feature dimension for the detection features is 128 and is reduced to 32 before building the pairwise detection context. The detection pair features also have 32 dimensions. The fully connected layers after the last block output 128 dimensional features. When we change the feature dimension, we keep constant the ratio between the number of features in each layer, so indicating the detection feature dimension is sufficient.

Message passing. The forward pass over serveral stacked blocks can be interpreted as message passing. Every detection sends messages to all of its neighbours in order to negotiate which detection is assigned an object and which detections should decrease their scores. Instead of handcrafting the message passing algorithm and its rules, we deliberately let the network latently learn the messages that are being passed.

4.3. Remarks

The Gnet is fundamentally different than GreedyNMS in the sense that all features are updated concurrently, while GreedyNMS operates sequentially. Since Gnet does not have access to GreedyNMS decisions (unlike [13]), it is surprising how close performance of the two algorithms turns out to be in section 5. Since we build a potentially big network by stacking many blocks, the Gnet might require large amounts of training data. In the experiments we deliberately choose a setting with many training examples.

The Gnet is a pure NMS network in the sense that it has no access to image features and operates solely on detections (box coordinates and a confidence). This means the Gnet cannot be interpreted as extra layers to the detector. The fact that it is a neural network and that it is possible to feed in a feature vector (from the image or the detector) into the first block makes it particularly suitable for combining it with a detector, which we leave for future work.

The goal is to *jointly* rescore all detections on an image. By allowing detections to look at their neighbours and update their own representation, we enable conditional dependence between detections. Together with the loss that encourages exactly one detection per object, we have satisfied both conditions from section 3. We will see in section 5 that the performance is relatively robust to parameter changes and works increasingly well for increasing depth.

5. Experiments

In this section we experimentally evaluate the proposed architecture on the PETS and COCO dataset. We report results for persons, and as well for the multi-class case. Person category is by far the largest class on COCO, and provides both crowded images and images with single persons. Other than overall results, we also report separately high and low occlusion cases. We are interested in performance under occlusion, since this is the case in which non-maximum suppression (NMS) is hard. All-in-all we show a consistent improvement over of GreedyNMS, confirming the potential of our approach.

All results are measured in average precision (AP), which is the area under the recall-precision curve. The overlap criterion (for matching detections to objects) is traditionally 0.5 IoU (as for Pascal VOC, noted as $AP_{0.5}$). COCO also uses stricter criteria to encourage better localisation quality, one such metric averages AP evaluated over several overlap criteria in the range $[0.5, 0.95]$ in 0.05 increments, which we denote by $AP^{0.95}_{0.5}$.

5.1. PETS: Pedestrian detection in crowds

Dataset. PETS [8] is a dataset consisting of several crowded sequences. It was used in [13] as a roughly single scale pedestrian detection dataset with diverse levels

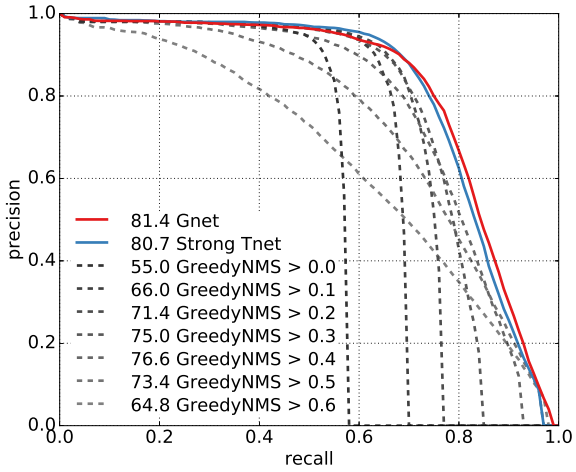


Figure 4: Performance on the PETS test set.

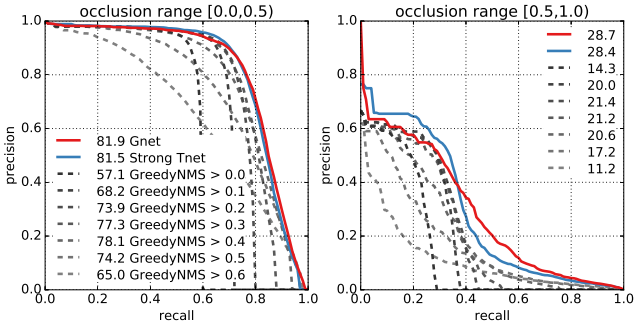


Figure 5: Performance on the PETS test set for different occlusion ranges.

of occlusion. Even though we aim for a larger and more challenging dataset we first analyse our method in the setup proposed in [13]. We use the same training and test set as well as the same detections from [28], a model built specifically to handle occlusions. We reduce the number of detections with an initial GreedyNMS of 0.8 so we can fit the joint rescoring of all detections into one GPU. (Note that these detections alone lead to bad results, worse than "GreedyNMS > 0.6" in 4, and this is very different to having input of GreedyNMS of 0.5 as an input like in [13]).

Training. We train a model with 8 blocks and a 128 dimensional detection representation for 30k iterations, starting with a learning rate of 10^{-3} and decrease it by 0.1 every 10k iterations.

Baselines. We compare to the (typically used) classic GreedyNMS algorithm using several different overlap thresholds, and the Strong Tnet from [13]. Since all methods operate on the same detections, the results are fully comparable.

Analysis. Figure 4 compares our method with the GreedyNMS baseline and the Tnet on the PETS test set. Start-

ing from a wide GreedyNMS suppression with the threshold $\vartheta = 0$ shows almost a step function, since high scoring true positives suppress all touching detections at the cost of also suppressing other true positives (low recall). Gradually increasing ϑ improves the maximum recall but also introduces more high scoring false positives, so precision is decreasing. This shows nicely the unavoidable trade-off due to having a fixed threshold ϑ mentioned in section 3. The reason for the clear trade-off is the diverse occlusion statistics present in PETS.

Tnet performs better than the upper envelope of the GreedyNMS, as it essentially recombines output of GreedyNMS at a range of different thresholds. In comparison our Gnet performs slightly better, despite not having access to GreedyNMS decisions at all. Compared to the best GreedyNMS performance, Gnet is able to improve by 4.8 AP.

Figure 5 shows performance separated into high and low occlusion cases. Again, the Gnet performs slightly better than Tnet. Performance in the occlusion range $[0, 0.5)$ looks very similar to the performance overall. For the highly occluded cases, the performance improvement of Gnet compared to the best GreedyNMS is bigger with 7.3 AP. This shows that the improvement for both Gnet and Tnet is mainly due to improvements on highly occluded cases as argued in section 3.

5.2. COCO: Person detection

Dataset. The COCO datasets consists of 80k training and 40k evaluation images. It contains 80 different categories in unconstrained environments. We first mimic the PETS setup and evaluate for persons only, and report multi-class results in section 5.3.

Since annotations on the COCO test set are not available and we want to explicitly show statistics per occlusion level, we train our network on the full training set and evaluate using two different subsets of the validation set. One subset is used to explore architectural choices for our network (minival, 5k images[1]) and the most promising model is evaluated on the rest of the validation set (minitest, 35k images).

We use the Python implementation of Faster R-CNN [21][2] for generating detections. We train a model only on the training set, so performance is slightly different than the downloadable model, which has been trained on the training and minitest sets. We run the detector with default parameters, but lower the detection score threshold and use detection before the typical non-maximum suppression step. There is no further preprocessing.

Training. We train the Gnet with ADAM for $2 \cdot 10^6$ iterations, starting with a learning rate of 10^{-4} and decreasing it to 10^{-5} after 10^6 iterations. The detection feature dimension is 128, the number of blocks is specified for each experiment.

[1]We use the same as used by Ross Girshick https://github.com/rbgirshick/py-faster-rcnn/tree/master/data.

[2]https://github.com/rbgirshick/py-faster-rcnn

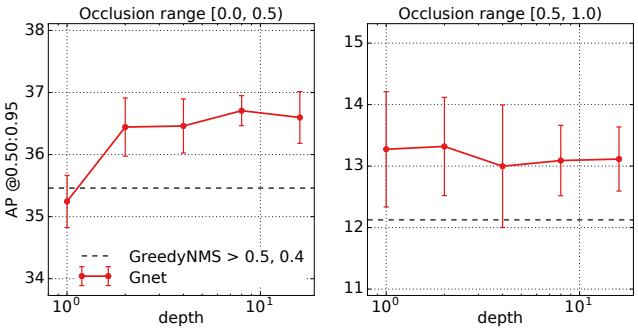


Figure 6: $AP^{0.95}_{0.5}$ versus number of blocks $(2, 4, 8, 16)$ for low and high occlusion respectively on COCO persons minival. Average over six runs, error bars show the standard deviation.

	Method	All $AP_{0.5}$	All $AP^{0.95}_{0.5}$	Occlusion [0, 0.5) $AP_{0.5}$	Occlusion [0, 0.5) $AP^{0.95}_{0.5}$	Occlusion [0.5, 1] $AP_{0.5}$	Occlusion [0.5, 1] $AP^{0.95}_{0.5}$
val	GreedyNMS>0.5	65.6	35.6	65.2	35.2	35.3	12.1
	Gnet, 8 blocks	**67.3**	**36.9**	**66.9**	**36.7**	**36.7**	**13.1**
test	GreedyNMS>0.5	65.0	35.5	61.8	33.8	30.3	11.0
	Gnet, 8 blocks	**66.6**	**36.7**	**66.8**	**36.1**	**33.9**	**12.4**

Table 1: Comparison between Gnet and GreedyNMS on COCO persons minival and minitest. Results for the full set and separated into occlusion levels.

Speed. On average we have 67.3 person detection per image, which the 16 block Gnet can process in 14ms/image on a K40m GPU and unoptimised Tensorflow code.

Baselines. We use GreedyNMS as a baseline. To show it in its best light we tune the optimal GreedyNMS overlap threshold on the test set of each experiment.

Analysis. Figure 6 shows $AP^{0.95}_{0.5}$ versus number of blocks in Gnet. The optimal GreedyNMS thresholds are 0.5 and 0.4 for low and high occlusion respectively. Already with one block our network performs on par with GreedyNMS, with two blocks onwards we see a $\sim$1 AP point gain. As in PETS we see gains both for low and high occlusions. With deeper architectures the variance between models for the high occlusion case seems to be decreasing, albeit we expect to eventually suffer from over-fitting if the architecture has too many free parameters.

We conclude that our architecture is well suited to replace GreedyNMS and is not particularly sensitive to the number of blocks used. Table 1 shows detailed results for Gnet with 8 blocks. The results from the validation set (minival) transfer well to the test case (minitest), provid-

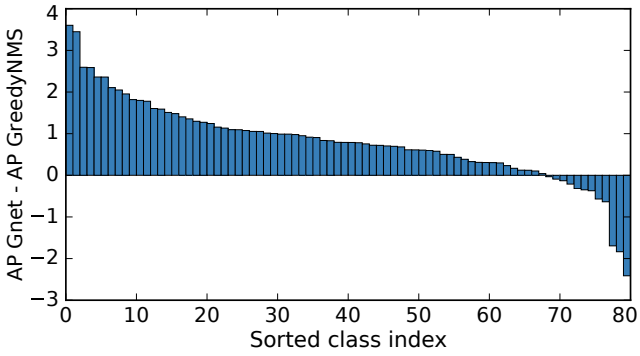


Figure 7: $AP^{0.95}_{0.5}$ improvement of Gnet over the best GreedyNMS threshold for each of the (sorted) 80 COCO classes. Gnet improves on ~ 70 out of 80 categories. On average Gnet provides a ~ 1 $mAP^{0.95}_{0.5}$ point gain over per-class tuned GreedyNMS ($23.5 \rightarrow 24.3\%$ $mAP^{0.95}_{0.5}$).

ing a small but consistent improvement over a well tuned GreedyNMS. Qualitative results are included in the supplementary material.

We consider these encouraging results, confirming that indeed the Gnet is capable of properly performing NMS without access to image features or GreedyNMS decisions.

5.3. COCO multi-class

As discussed in section 4, Gnet is directly applicable to the multi-class setup. We use the exact same parameters and architecture selected for the persons case. The only change is the replacement of the score scalar by a per-class score vector in the input and output (see §4.2). We train one multi-class Gnet model for all 80 COCO categories.

Figure 7 shows the $mAP^{0.95}_{0.5}$ improvement of Gnet over a per-class tuned GreedyNMS. We obtain improved results on the bulk of the object classes, and no catastrophic failure is observed, showing that Gnet is well suited to handle all kind of object categories. Averaged across classes Gnet obtains 24.3% $mAP^{0.95}_{0.5}$, compared to 23.5% for a test-set tuned GreedyNMS. Overall we argue Gnet is a suitable replacement for GreedyNMS.

Supplementary material includes the detailed per-class table.

6. Conclusion

In this work we have opened the door for training detectors that no longer need a non-maximum suppression (NMS) post-processing step. We have argued that NMS is usually needed as post-processing because detectors are commonly trained to have robust responses and process neighbouring detections independently. We have identified two key ingredients missing in detectors that are necessary to build an NMS network: (1) a loss that penalises double detections and (2) joint processing of detections.

We have introduced the Gnet, the first "pure" NMS network that is fully capable of performing the NMS task without having access to image content nor help from another algorithm. Being a neural network, it lends itself to being incorporated into detectors and having access to image features in order to build detectors that can be trained truly end-to-end. These end-to-end detectors will not require any post-processing.

The experimental results indicate that, with enough training data, the proposed Gnet is a suitable replacement for GreedyNMS both for single- or multi-class setups. The network surpasses GreedyNMS in particular for occluded cases and provides improved localization.

In its current form the Gnet requires large amounts of training data and it would benefit from future work on data augmentation or better initialisation by pre-training on synthetic data. Incorporating image features could have a big impact, as they have the potential of informing the network about the number of objects in the image.

We believe the ideas and results discussed in this work point to a future where the distinction between detector and NMS will disappear.

References

[1] O. Barinova, V. Lempitsky, and P. Kholi. On detection of multiple object instances using hough transforms. *PAMI*, 2012. 2

[2] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 2

[3] G. Burel and D. Carel. Detection and localization of faces on digital images. *Pattern Recognition Letters*, 1994. 3

[4] G. Chen, Y. Ding, J. Xiao, and T. X. Han. Detection evolution with multi-order contextual co-occurrence. In *CVPR*, 2013. 2

[5] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *CVPR*, 2015. 2

[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2

[7] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 1, 2

[8] J. Ferryman and A. Ellis. Pets2010: Dataset and challenge. In *AVSS*, 2010. 6

[9] R. Girshick. Fast R-CNN. In *ICCV*, 2015. 1, 2

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *ECCV*, 2016. 5

[12] P. Henderson and V. Ferrari. End-to-end training of object class detectors for mean average precision. In *ACCV*, 2016. 2, 4

[13] J. Hosang, R. Benenson, and B. Schiele. A convnet for non-maximum suppression. In *GCPR*, 2016. 2, 3, 4, 6

[14] P. Kontschieder, S. Rota Bulò, M. Donoser, M. Pelillo, and H. Bischof. Evolutionary hough games for coherent object detection. *CVIU*, 2012. 2

[15] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 2008. 2

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. Reed. Ssd: Single shot multibox detector. In *ECCV*, 2016. 1

[17] D. Mrowca, M. Rohrbach, J. Hoffman, R. Hu, K. Saenko, and T. Darrell. Spatial semantic regularisation for large scale object detection. In *ICCV*, 2015. 2

[18] M. Niepert, M. Ahmed, and K. Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016. 2

[19] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. In *CVPR*, 2013. 2

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1

[21] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 1, 2, 7

[22] M. Rodriguez, I. Laptev, J. Sivic, and J.-Y. Audibert. Density-aware person detection and tracking in crowds. In *ICCV*, 2011. 2

[23] R. Rothe, M. Guillaumin, and L. Van Gool. Non-maximum suppression for object detection by passing messages between windows. In *ACCV*, 2014. 2

[24] M. A. Sadeghi and A. Farhadi. Recognition using visual phrases. In *CVPR*, 2011. 2

[25] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. In *ICLR*, 2014. 2

[26] R. Stewart and M. Andriluka. End-to-end people detection in crowded scenes. In *CVPR*, 2016. 2, 4

[27] S. Tang, B. Andres, M. Andriluka, and B. Schiele. Subgraph decomposition for multi-target tracking. In *CVPR*, 2015. 2

[28] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele. Learning people detectors for tracking in crowded scenes. In *ICCV*, 2013. 6

[29] S. Tang, M. Andriluka, and B. Schiele. Detection and tracking of occluded people. In *BMVC*, 2012. 2

[30] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *PAMI*, 2010. 2

[31] A. Vezhnevets and V. Ferrari. Object localization in imagenet by looking out of the window. In *BMVC*, 2015. 2

[32] P. Viola and M. Jones. Robust real-time face detection. In *IJCV*, 2004. 2

[33] L. Wan, D. Eigen, and R. Fergus. End-to-end integration of a convolutional network, deformable parts model and non-maximum suppression. In *CVPR*, 2015. 2

[34] P. Wohlhart, M. Donoser, P. M. Roth, and H. Bischof. Detecting partially occluded objects with an implicit shape model random field. In *ACCV*, 2012. 2

[35] C. Wojek, G. Dorkó, A. Schulz, and B. Schiele. Sliding-windows for rapid object class localization: A parallel technique. In *DAGM*, 2008. 2

[36] J. Yan, Y. Yu, X. Zhu, Z. Lei, and S. Z. Li. Object detection by labeling superpixels. In *CVPR*, 2015. 2

[37] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 2