A-Fast-RCNN: Hard positive generation via adversary for object detection

Xiaolong Wang Abhinav Shrivastava Abhinav Gupta
The Robotics Institute, Carnegie Mellon University

Abstract

How do we learn an object detector that is invariant to occlusions and deformations? Our current solution is to use a data-driven strategy – collect large-scale datasets which have object instances under different conditions. The hope is that the final classifier can use these examples to learn the invariances. But is it really possible to see all possible occlusions in a dataset? We argue that like categories, occlusions and object deformations also follow a long-tail. Some occlusions and deformations are so rare that they will hardly happen; yet we want to learn a model invariant to such occurrences. In this paper, we propose an alternative solution. We propose to learn an adversarial network that generates examples with occlusions and deformations. The goal of the adversary is to generate examples that are difficult for the object detector to classify. In our framework both the original detector and adversary are learned in a joint manner. Our experimental results indicate a 2.3% mAP boost on VOC07 and a 2.6% mAP boost on VOC2012 object detection challenge compared to Fast-RCNN pipeline.

1. Introduction

The goal of object detection is to learn a visual model for concepts such as car and use this model to localize these concepts in an image. This requires the ability to robustly model invariance to illumination, deformation, occlusion and intra-class variations. The standard paradigm to handle these invariances is to collect large-scale datasets which have object instances under different conditions. For example, the COCO dataset [11] has more than 10K examples of cars under different occlusions and deformations. The hope is that these examples capture all possible variations of a visual concept and the classifier can then effectively model invariance to them. We believe this has been one of the prime reasons why ConvNets have been so successful at the task of object detection. They have been able to use all this data to learn the invariances.

However, like the object categories, we believe even occlusions and deformations follow long-tail. That is, some of the occlusions and deformations are so rare that there is a low chance that they will occur even in large-scale dataset. For example, consider the occlusion shown in figure 1, while some of the occlusions occur more than often (occlusion from other cars in parking) other occlusion (from a ladder for example is hardly seen). Similarly, some deformations in animals are common (sitting/standing poses) but other deformations are very rare. How can we sample such occlusions and deformations which lie on the tail? While collecting even larger datasets is one possible solution, it clearly does not seem scalable.

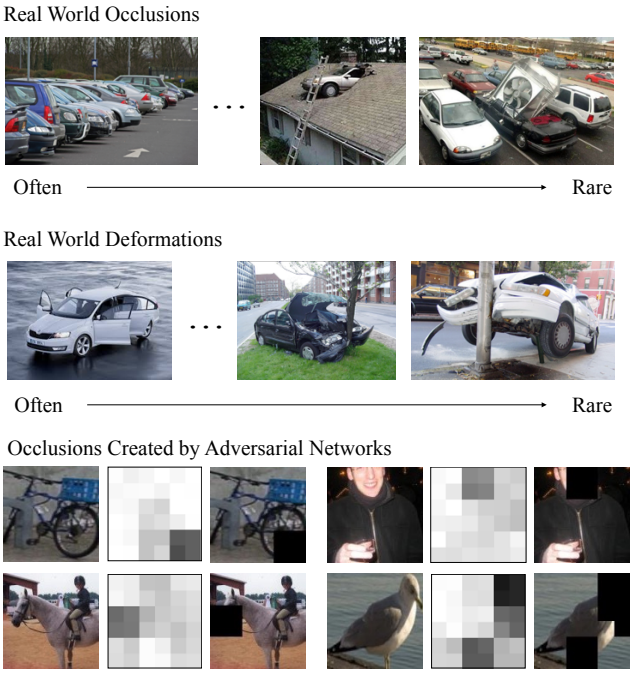


Figure 1: We argue that both occlusions and deformations follow a long-tail distribution. Some occlusions and deformations are rare. In this paper, we propose to use an adversarial network to generate examples with occlusions and deformations that will be hard for object detector to classify. Our adversarial network adapts as the object detector becomes better and better. We show boost in detection performace using this adversarial learning strategy.

Recently, there has been lot of work being done in the field of image/pixel generation. One possible way is to generate realistic looking images which can sample from these deformations. However, since the image generation would require training examples in turn this is not really a feasible solution. Another solution is to generate all possible occlusions and deformations and train object detectors on them. However, since the space of deformations and occlusions is huge, this does not seem like a scalable solution. It has been shown that using all examples is often not the optimal solution [29] and selecting hard examples is better. Is there a way we can generate "hard" positive examples with different occlusions and deformations and without generating the pixels themselves?

How about we train another network: an adversary that creates hard examples by blocking some feature maps spatially or by creating spatial deformations by manipulating feature responses. This adversary will predict what it thinks will be hard for a detector like Fast-RCNN [5] and in turn the Fast-RCNN will adapt itself to learn to classify these adversarial examples. The key idea here is to create adversarial examples in conv5 feature space but not generate the pixels directly since the latter is a substantially harder problem. In our experiments, we show substantial improvement in performance of the adversarial rcnn compared to the standard FRCN pipeline.

2. Related Work

In recent years, significant gains have been made in the field of object detection. The recent success builds upon the success of deep features [10] learned for the task of ImageNet classification [2]. Initially, the R-CNN [6] and OverFeat [22] detectors led this wave with impressive results on PASCAL VOC [4]. However, in recent years, more computationally efficient versions have emerged to train on even larger datasets such as COCO [11]. For example, Fast-RCNN [5] shares the convolutions across different region proposals and provides speed-up. Faster-RCNN [20] incorporates region proposal generation in the framework leading to a completely end-to-end version. As a follow-up on Overfeat, other computationally-efficient sliding-window based approaches have emerged such as YOLO [19] and SSD [12].

Recently, there have been three principal directions to further improve the performance of these object detectors. The first direction focuses on changing the base architecture of these networks. The central idea is that using much deeper networks should not only lead to improvement in classification but also standard object detection pipelines. Some recent work in this direction include ResNet-based object detection [8], Inception-ResNet for object detection [28] etc.

The second area of research has been to use contextual reasoning and other tasks for reasoning and hence improving object detections. For example, [24] use segmentation as a way to contextually prime object detection and provide feedback to initial layers. [1] uses skip-network architecture and uses features from multiple layers of representation in conjunction with contextual reasoning. Other approaches include using a top-down refinement for improving segmentation [16] and hence leading to better detections.

The third direction to improve the performance is to better exploit data itself. It is often argued that the recent success of object detectors are a product of better visual representation and the availability of large-scale data for learning. Therefore, a third class of approaches try to explore how to better use data for improving the performance. One example is to incorporate hard example mining in an effective and efficient setup for training Region-based ConvNets [25]. Other examples of hard example mining include [13, 26, 30].

Our work follows the third direction of research where the focus is on leveraging data in a better manner. However, instead of trying to shift through the data to find hard examples, i.e. we try to generate examples which will be hard for Fast-RCNN to detect/classify. We restrict the space of new positive generation to adding occlusions and deformation to the current existing examples only. Specifically, we learn adversarial networks which try to predict occlusion and deformation that would lead to mis-classification by Fast-RCNN. Our work is therefore related to lot of recent work in adversarial learning [3, 7, 14, 15, 18, 21]. For example, Radford et al. [18] proposed good practices for adversarial learning in image generation. This work is further extended by Salimans et al. [21], which discusses about more techniques for training better image generative model. It is also mentioned in [21] that the adversarial learning can be applied to improve image classification in a semi-supervised setting. However, the experiments in these works are conducted on data which has lesser complexity than object detection datasets, where image generation results are significantly inferior. Our work is also related to recent work on adversarial training in robotics [17]. However, in this case, instead of using an adversary for better supervision; we use the adversary to generate the hard examples.

3. Adversarial Learning for Object Detection

Our goal is to learn an object detector that is robust to different conditions such as occlusion, deformation and illumination. We hyothesize that even in large-scale datasets, it is impossible to sample all possible occlusions and deformations. Instead of relying heavily on dataset or sifting through data to find hard examples, we take an alternative approach. We try to actively generate data which are hard examples for the object detector to learn from. However,

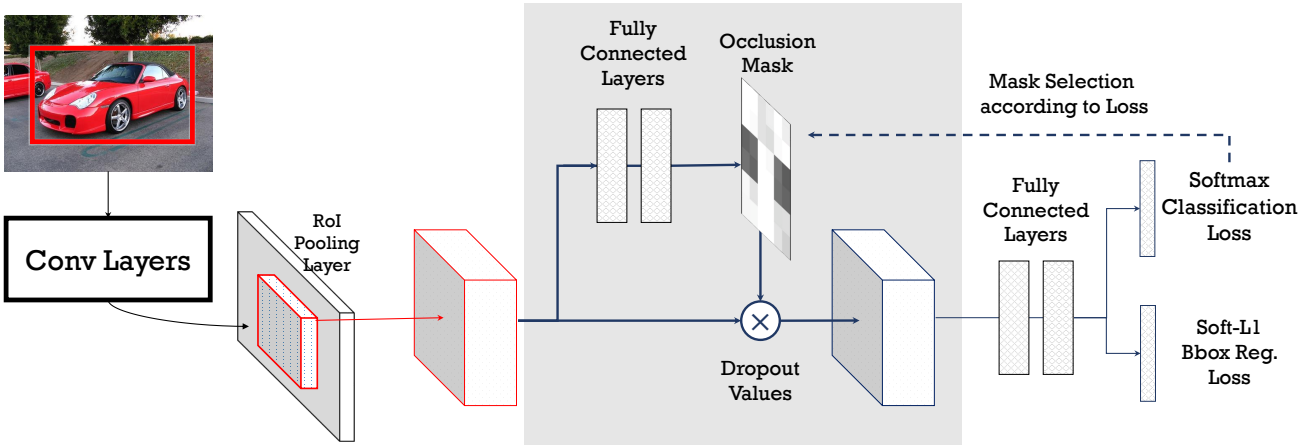


Figure 2: Our network architecture of ASDN and how it combines with Fast RCNN approach. Our ASDN network takes as input image patches with features extracted using RoI pooling layer. ASDN network than predicts an occlusion/dropout mask which is then used to drop the feature values and passed onto classification tower of Fast-RCNN.

instead of generating data in the entire pixel space directly, we focus on a restricted space for generation: occlusion and deformation.

Mathematically, let us assume the original object detector network is represented as $\mathcal{F}(x)$ where x is one of the region proposals. A detector gives two outputs $\mathcal{F}_C$ which represents class output and $\mathcal{F}_L$ represent predicted bbox location. Let us assume that the ground-truth class for x is C with spatial location being l. Our original detector loss can be written down as

$$L_{\mathcal{F}} = L_{\text{softmax}}(\mathcal{F}_C(x), C) + \lambda\,[C \notin \text{bg}]\, L_{\text{bbox}}(\mathcal{F}_L(x), l)$$

The first term is the softmax loss over classes and the second term is the loss based on predicted bounding box location and ground truth box location (only for foreground classes).

Now, let us assume the adversarial network is represented as $\mathcal{A}(x)$ which given a feature x computed on image I, generates new adversarial examples x_A. The loss function for the detector remains the same just that the batches now include few original and few adversarial examples.

However, the adversarial network has to learn to predict images on which the Fast-RCNN would fail. For that we use the following loss function:

$$L_{\mathcal{A}} = 1 - L_{\text{softmax}}(\mathcal{F}_C(\mathcal{A}(x)), C)$$

Therefore, if the feature generated by adversarial network is easy for detector to classify, we get a high-loss. On the other hand, if after adversarial feature generation it is difficult for detector, we get high loss for detector and low loss for adversarial network.

4. A-Fast-RCNN: Approach Details

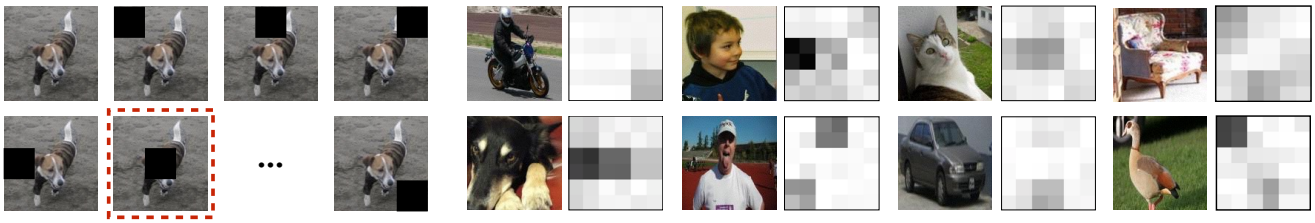
We now describe details of the framework we developed for our experiments. We will first give a brief overview of our base detector which is Fast-RCNN. This is followed by describing the space of adversarial generation. In this paper, we focus on generating different types of occlusions and deformations in the feature maps. Finally, we describe our experimental setup and show the results which indicate significant improvement over baselines.

4.1. Overview of Fast-RCNN

For our experiments, we build upon the Fast-RCNN framework for object detection [5]. Fast-RCNN is composed of two parts: (i) a convolutional network for feature extraction; (ii) an RoI network with a RoI-pooling layer and a few fully connected layers for producing object classes and bounding boxes.

Given an input image, the convolutional network of the Fast-RCNN takes the whole image as an input and extracts convolutional feature maps as the output. Since the operations are mainly convolutions and max-pooling, the spatial dimensions of the output feature map will change according to the input image size. Given the feature map, the RoI-pooling layer is used to project the object proposals into the feature space. The RoI-pooling layer generates fixed size feature vectors for each of the object proposal. These feature vectors are then passed through fully connected layers. The outputs of the fully connected layers are: (i) probabilities for each object class including the background class; and (ii) bounding box coordinates.

For training, the SoftMax loss and regression loss are applied on these two outputs, and the gradients are back propagated though all the layers to perform end-to-end learning



(a) Pre-training via Searching (b) Generated Masks

Figure 3: (a) Model pre-training: Examples of occlusions that are sifted to select the hard occlusions and used as ground-truth to train the ASDN network (b) Examples of occlusion masks generated by ASDN network. The black regions are occluded when passed on to FRCN pipeline.

of the framework.

4.2. Adversarial Networks Design

We consider two types of feature generations by adversarial networks competing against the Fast RCNN detector. The first type of generation is occlusion. Here, we propose Adversarial Spatial Dropout Network (ASDN) which given an object image learns how to occlude the object such that it becomes hard for FRCN to classify. The second type of generation we consider in this paper is deformation. In this case, we propose Adversarial Spatial Transformer Network (ASTN) which learns how to rotate "parts" of the objects and make them hard to recognize by the detector. By competing against these networks and overcoming the obstacles, the Fast-RCNN learns to handle object occlusions and deformations in a robust manner. Note that both the proposed networks ASDN and ASTN are learned simultaneously in conjunction with the Fast-RCNN (FRCN) during training. Joint training allows to make sure that the detector does not overfit to the obstacles created by the fixed policies of generation.

Instead of creating occlusions and deformations on the input images, we find that operating on the feature space is more efficient and effective. Thus, we design our adversarial networks to modify the features to make the object harder to recognize. Note that these two networks are only applied during training to improve the detector. We will first introduce the ASDN and ASTN individually and then combine them together in a unified framework.

4.2.1 Adversarial Spatial Dropout for Occlusion

We propose an Adversarial Spatial Dropout Network (ASDN) to create occlusions on the deep feature space for foreground objects. Recall that in the standard Fast-RCNN pipeline, we can obtain the convolutional features for each foreground object proposal after the RoI-pooling layer. We use these region-based features as the inputs for our adversarial network. Given a feature of an object, the ASDN will try to generate a mask indicating which parts of the feature to dropout (assigning zeros) so that the detector cannot recognize the object.

More specifically, we extract the feature X for an object with size $d \times d \times c$, where d is the spatial dimension and c represents the number of channels (e.g., $c = 256, d = 6$ in AlexNet). Given this feature, our ASDN will predict a mask M with $d \times d$ values which are either 0 or 1 after thresholding. We visualize some of the masks before thresholding as Fig. 3(b). We denote M_{ij} as the value for the ith row and jth column of the mask. Similarly, X_{ijk} represents the value in channel k at location i, j of the feature. If $M_{ij} = 1$, we drop out the values of all the channels in the corresponding spatial location of the feature map F, i.e., $X_{ijk} = 0, \forall k$.

Network Architecture. We use standard Fast-RCNN architecture. We initialize the network using pre-training from ImageNet. Specifically, the parameters before RoI-pooling (from conv1 to pool5 layers in AlexNet and VGGNet) initialized with pre-trained network. The two fully connected layers after RoI-pooling are also initialized with pre-trained ImageNet network. The adversarial network shares first five layers and RoI-pooling layer with FRCN and then uses its own separate fully connected layers. Note that we are not sharing the parameters in our ASDN with Fast-RCNN since we are optimizing two networks to do the exact opposite tasks.

Model Pre-training. In our experiment, we find it important to pre-train the ASDN for the task of creating occlusions before using it to improve Fast-RCNN. Motivated by the Region Proposal Networks in [20], we also apply stage-wise training here. We first train our Fast-RCNN detector without ASDN for 10K iterations. As the detector now has a sense of the objects in the dataset, we train our ASDN model for creating the occlusions by fixing all the layers in the detector.

Initializing ASDN Network. To initialize the ASDN network, given a feature map X with spatial layout $d \times d$, we apply a sliding window with size $\frac{d}{3} \times \frac{d}{3}$ on it. For each sliding window, we drop out the values in all channels whose spatial locations are covered by the window and generate a new feature vector for the region proposal. This feature

vector is then passed through classification and bbox regression layers to compute the loss. Based on the loss of all the $\frac{d}{3} \times \frac{d}{3}$ windows, we select the one with the highest loss. This window is then used to create a single $d \times d$ mask. We generate these spatial masks for n positive region proposals and generate n pairs of training examples $\{(X^1, \tilde{M}^1), ..., (X^n, \tilde{M}^n)\}$ for our adversarial dropout network. The idea is that the ASDN should learn to generate the masks which can give the detector network high losses. We apply the binary cross entropy loss in training the ASDN and it can be formulated as

$$L = -\frac{1}{n}\sum_{p}^{n}\sum_{i,j}^{d}[\tilde{M}_{ij}^{p}A_{ij}(X^p) + (1 - \tilde{M}_{ij}^{p})(1 - A_{ij}(X^p))] \quad (1)$$

where $A_{ij}(X^p)$ represents the outputs of the ASDN in location (i, j) given input feature map X^p. We train the ASDN with this loss for 10K iterations. We show that the network starts to recognize which part of the objects are significant for classification as shown in Fig. 3(b). Also note that our output masks are different from the Attention Mask proposed in [23], where they use the attention mechanism to facilitate classification. In our case, we use the masks to occlude parts to make classification harder.

Thresholding by Sampling. The output generated by ASDN network is not a binary mask but rather a continuous heatmap. Instead of using thresholding, we use importance sampling to select the top $\frac{1}{6}$ blocks to mask out. Note that the sampling procedure incorporates stochasticity and diversity in samples during training.

Joint Learning. Given the pre-trained ASDN and Fast-RCNN model, we jointly optimize two networks in each iteration of training. For training the Fast-RCNN detector, we first use the ASDN to generate the masks on the features after the RoI-pooling during forward propagation. We perform sampling to generate binary masks and use them to drop out the values in the features after the RoI-pooling layer. We then forward the modified features to calculate the loss and train the detector end-to-end. Note that although our features are modified, the labels remain the same. In this way, we create "harder" and diverse examples for training the detector.

While training the ASDN, instead of directly using the loss function, we use an auxiliary procedure. Rather than computing backprops with respect to masks, we compute which masks lead to significant drops in FRCN scores. We use only those hard example masks as groundTruth to train the adversarial network directly using the same loss as described in Eq. 1.

4.2.2 Adversarial Spatial Transformer Network

We now introduce Adversarial Spatial Transformer Network (ASTN). The key idea is to create deformations on the object features and make object recognition by the detector difficult. Our network is built upon the Spatial Transformer Network (STN) proposed in [9]. In their work, the STN is proposed to deform the features to make classification easier. Our network, on the other hand, is doing the exact opposite task. By competing against our ASTN, we can train a better object detector which is robust to object deformations.

STN Overview. The Spatial Transformer Network [9] has three components: localisation network, grid generator and sampler. Given the feature map as input, the localisation network will estimate the variables for deformations (e.g., rotation degree, translation distance and scaling factor). These variables will be used as inputs for the grid generator and sampler to operate on the feature map. The output is a deformed feature map. Note that we only need to learn the parameters in the localisation network. One of the key contribution of STN is making the whole process differentiable, so that the localisation network can be directly optimized for the classification objective via back propagation. More technical details are covered in [9].

Adversarial STN. In our Adversarial Spatial Transformer Network, we focus on feature map rotations. That is, given a feature map after the RoI-pooling layer as input, our ASTN will learn to rotate the feature map to make it harder to recognize. Our localisation network is composed with 3 fully connected layers where the first two layers are initialized with fc6 and fc7 layers from ImageNet pre-trained network as in our Adversarial Spatial Dropout Network.

We train the ASTN and the Fast-RCNN detector jointly. For training the detector, similar to the process in the ASDN, the features after RoI-pooling are first transformed by our ASTN and forwarded to the higher layers to compute the SoftMax loss. For training the ASTN, we try to optimize it so that the detector will classify the foreground objects as the background class. We perform back propagation and only finetune the parameters in the localisation network of the STN.

Implementation Details. In our experiments, we find it very important to limit the rotation degrees produced by the ASTN. Otherwise it is very easy to rotate the object upside down which is the hardest to recognize in most cases. We constrain the rotation degree within 10° clockwise and anti-clockwise. Instead of rotating all the feature map in the same direction, we divide the feature maps on the channel dimension into 4 blocks and estimate 4 different rotation angles for different blocks. Since each of the channel corresponds to activations of one type of feature, rotating channels separately corresponds to rotating parts of the object in different directions which leads to deformations.

In our experiment, we also find that if we use one rotation angle for all feature maps, the ASTN will often predict the largest angle. By using 4 different angles instead of one,

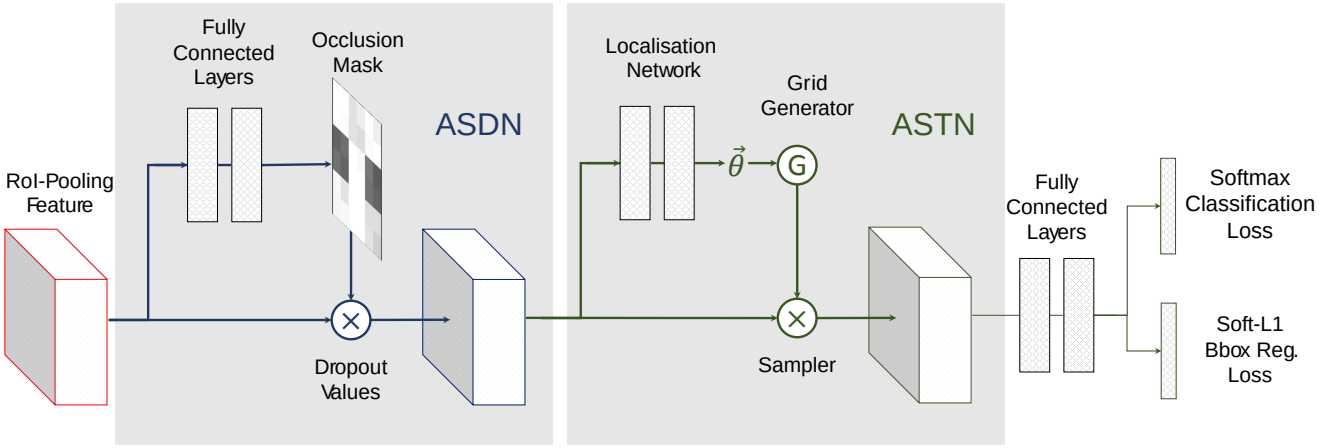


Figure 4: Network architecture for combining ASDN and ASTN network. First occlusion masks are created and then the channels are rotated to generate hard examples for training.

we increase the complexity of the task which prevents the network from predicting trivial deformations.

4.2.3 Adversarial Fusion

The two adversarial networks ASDN and ASTN can also be combined and trained together in the same detection framework. Since these two networks offer different types of information. By competing against these two networks simultaneously, our detector become more robust.

We combine these two networks into the Fast-RCNN framework in a sequential manner. As shown in Fig. 4, the feature maps extracted after the RoI-pooling are first forwarded to our ASDN which drop out some activations. The modified features are further deformed by the ASTN.

5. Experiments

5.1. Experimental settings

We perform our experiments on PASCAL VOC 2007 and PASCAL VOC 2012 dataset [4]. We evaluate our methods with two standard ConvNet architectures: AlexNet [10] and VGG16 [27]. For both dataset, we used the trainval set for training and test set for testing. We follow most of the setup in standard FRCN [5] for training. We apply SGD for 80K to train our models. The learning rate starts with 0.001 and decreases to 0.0001 after 60K iterations. Our minibatch size for training is 256 with 2 images. By using this setting in training, our baseline numbers for both networks are slightly better than the reported number in [5]. During joint learning, to prevent the Fast-RCNN from overfitting to the modified data, we provide one image in the batch without any adversarial occlusions/deformations and apply our approach on another image in the batch.

5.2. PASCAL VOC 2007 Results

We report our results for using ASTN and ASDN during training Fast-RCNN in Table.1. For the AlexNet architecture, our implemented baseline is 57.0% mAP. Based on this setting, joint learning with our ASTN model reaches 58.1% and joint learning with the ASDN model gives higher performance of 58.5%. As both methods are complementary to each other, combining ASDN and ASTN into our full model gives another boost to 58.9% mAP.

For the VGG16 architecture, we conduct the same set of experiments. Firstly, our baseline model reaches 69.1% mAP, much higher than the reported number 66.9% in [5]. Based on this implementation, joint learning with our ASTN model gives an improvement to 69.9% mAP and the ASDN model reaches 71.0% mAP. Our full model with both ASTN and ASDN improves the performance to 71.4%. Our final result gives 2.3% boost upon the baseline.

Note that the performance improvements in VGG are more considerably larger than AlexNet. One possible hypothesis is that the AlexNet has limited capacity and therefore cannot really exploit the deformations/occlusions to learn. On the other hand, VGG model can exploit this extra data to provide significant advances in performance.

5.2.1 Ablative Analysis on the ASDN

We compare our Advesarial Spatial Dropout Network with various dropout/occlusion strategy in training using the AlexNet architecture. The first simple baseline we try is random masking on the feature after RoI-Pooling. For fair comparison, we mask the activations of the same number of neurons as we do in the ASDN network. Each time dropout is also conducted across all channels for a specific spatial location. As Table 2 shows, the performance of ran-

Table 1: **VOC 2007 test** detection average precision (%). FRCN⋆ refers to FRCN [5] with our training schedule.

method	arch	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [5]	AlexNet	55.4	67.2	71.8	51.2	38.7	20.8	65.8	67.7	71.0	28.2	61.2	61.6	62.6	72.0	66.0	54.2	21.8	52.0	53.8	66.4	53.9
FRCN⋆	AlexNet	57.0	67.3	72.1	54.0	38.3	24.4	65.7	70.7	66.9	32.4	60.2	63.2	62.5	72.4	67.6	59.2	24.1	53.0	60.6	64.0	61.5
Ours (ASTN)	AlexNet	58.1	68.7	73.4	53.9	36.9	26.5	69.4	71.8	68.7	33.0	60.6	64.0	60.9	76.5	70.6	60.9	25.2	55.2	56.9	68.3	59.9
Ours (ASDN)	AlexNet	58.5	67.1	72.0	53.4	36.4	25.3	68.5	71.8	70.0	34.7	63.1	64.5	64.3	75.5	70.0	61.5	26.8	55.3	58.2	70.5	60.5
Ours (full)	AlexNet	58.9	67.6	74.8	53.8	38.2	25.2	69.1	72.4	68.8	34.5	63.0	66.2	63.6	75.0	70.8	61.6	26.9	55.7	57.8	71.7	60.6
FRCN [5]	VGG	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
FRCN⋆	VGG	69.1	75.4	80.8	67.3	59.9	37.6	81.9	80.0	84.5	50.0	77.1	68.2	81.0	82.5	74.3	69.9	28.4	71.1	70.2	75.8	66.6
Ours (ASTN)	VGG	69.9	73.7	81.5	66.0	53.1	45.2	82.2	79.3	82.7	53.1	75.8	72.3	81.8	81.6	75.6	72.6	36.6	66.3	69.2	76.6	72.7
Ours (ASDN)	VGG	71.0	74.4	81.3	67.6	57.0	46.6	81.0	79.3	86.0	52.9	75.9	73.7	82.6	83.2	77.7	72.7	37.4	66.3	71.2	78.2	74.3
Ours (full)	VGG	71.4	75.7	83.6	68.4	58.0	44.7	81.9	80.4	86.3	53.7	76.1	72.5	82.6	83.9	77.1	73.1	38.1	70.0	69.7	78.8	73.1

Table 2: **VOC 2007 test** detection average precision (%). Ablative analysis on the Adversarial Spatial Dropout Network.FRCN⋆ refers to FRCN [5] with our training schedule.

method	arch	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN⋆	AlexNet	57.0	67.3	72.1	54.0	38.3	24.4	65.7	70.7	66.9	32.4	60.2	63.2	62.5	72.4	67.6	59.2	24.1	53.0	60.6	64.0	61.5
Ours (random dropout)	AlexNet	57.3	68.6	72.6	52.0	34.7	26.9	64.1	71.3	67.1	33.8	60.3	62.0	62.7	73.5	70.4	59.8	25.7	53.0	58.8	68.6	60.9
Ours (hard dropout)	AlexNet	57.7	66.3	72.1	52.8	32.8	24.3	66.8	71.7	69.4	33.4	61.5	62.0	63.4	76.5	69.6	60.6	24.4	56.5	59.1	68.5	62.0
Ours (fixed ASDN)	AlexNet	57.5	66.3	72.7	50.4	36.6	24.5	66.4	71.1	68.8	34.7	61.2	64.1	61.9	74.4	69.4	60.4	26.8	55.1	57.2	68.6	60.1
Ours (joint learning)	AlexNet	58.5	67.1	72.0	53.4	36.4	25.3	68.5	71.8	70.0	34.7	63.1	64.5	64.3	75.5	70.0	61.5	26.8	55.3	58.2	70.5	60.5

dom dropout is 57.3% mAP which is slightly better than the baseline.

Another dropout strategy we compare to is a similar strategy we apply in pre-training the ASDN (Fig. 3). We exhaustively enumerate different kinds of occlusion and select the best ones for training in each iteration. The performance is 57.7% mAP (Ours (hard dropout)), which is slightly better than random dropout.

As we find the exhaustive strategy can only explore very limited space of occlusion policies, we use the pre-trained ASDN network to replace it. However, when we fix the parameters of the ASDN, we find the performance is 57.5% mAP(Ours (fixed ASDN)) , which is not as good as the exhaustive strategy. The reason is the fixed ASDN has not received any feedbacks from the updating Fast-RCNN while the exhaustive search has. If we jointly learn the ASDN and the Fast-RCNN together, we can get 58.5% mAP, 1.5% improvement compared to the baseline without dropout. This evidence shows that joint learning of ASDN and Fast-RCNN is where it makes a difference.

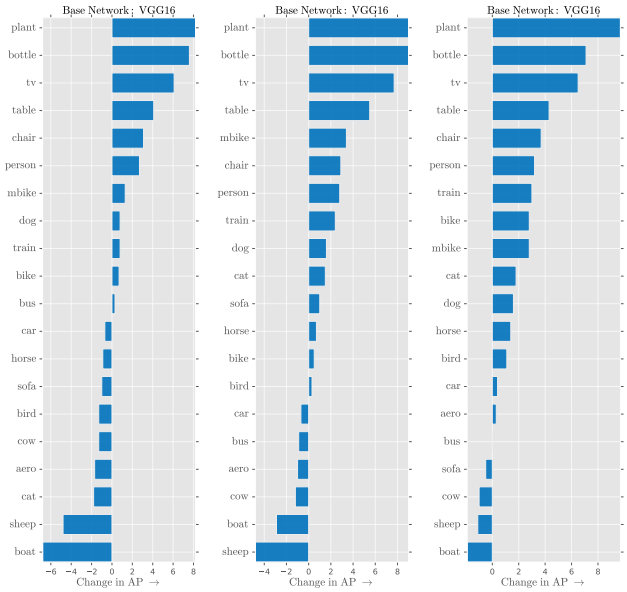


Figure 5: Changes of AP for each compared to baseline FRCN.

5.2.2 Category-based Analysis

Figure 5 shows the graph of how performance of each category changes with the occlusions and deformations. Interestingly the categories that seemed to be helped by both ASTN and ASDN seem to be quire similar. It seems that both `plant` and `bottle` performance improves with adversarial training. However, combining the two transformations together seems to improve performance on some categories which were hurt by using occlusion or deformations alone. Specifically, categories like `car` and `aeroplane` are helped by combining the two adversarial processes.

5.2.3 Qualitative Results

Figure 6 shows some of the false positives of our approach. These examples are hand-picked such that they only ap-

Table 3: **VOC 2012 test** detection average precision (%). FRCN⋆ refers to FRCN [5] with our training schedule.

method	arch	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv
FRCN [5]	VGG	65.7	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7
FRCN⋆	VGG	66.4	81.8	74.4	66.5	47.8	39.3	75.9	69.1	87.4	44.3	73.2	54.0	84.9	79.0	78.0	72.2	33.1	68.0	62.4	76.7	60.8
Ours (full)	VGG	69.0	82.2	75.6	69.2	52.0	47.2	76.3	71.2	88.5	46.8	74.0	58.1	85.6	80.3	80.5	74.7	41.5	70.4	62.2	77.4	67.0

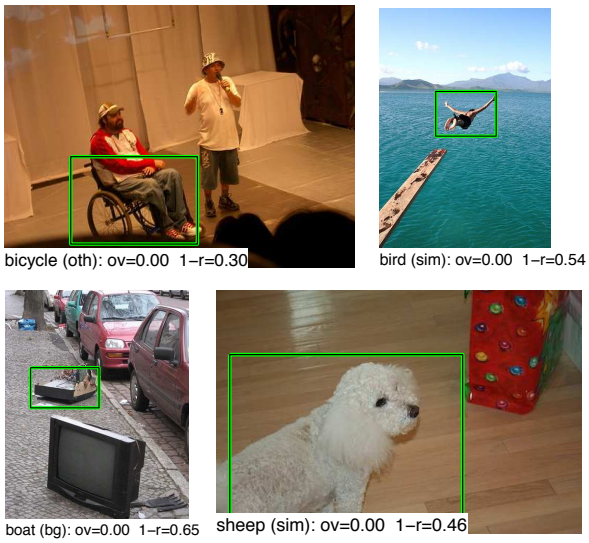


Figure 6: Some of the false positives for our approach. These are top false positives for adversarial training but not the original Fast-RCNN.

peared in the list of false positives for adversarial learning but not the original Fast-RCNN. These results indicate some of the shortcomings of adversarial learning. In some cases, the adversarial learning creates deformations or occlusions which are similar to other object categories and leading to over-generalization of the category. For example, in many instances, our approach hides the wheels of the bicycle. This leads to even a wheel chair being classified as a bike. Similarly in case of boat, the deformations lead to over-generalization and hence the false positive shown in the figure.

5.3. PASCAL VOC 2012 Results

We finally show our results on the PASCAL VOC 2012 dataset in Table. 3, where our baseline performance is 66.4% .Our full approach with joint learning of ASDN and ASTN gives 2.6% boost to 69.0% mAP. This again shows that the performance improvement using VGG on VOC2012 is significant.

Figure 7 shows the changes in AP compared to the baseline. Interestingly, in case of VOC2012, we improve performance of all the categories except sofa. We believe this is probably because of larger diversity in both training and test set.

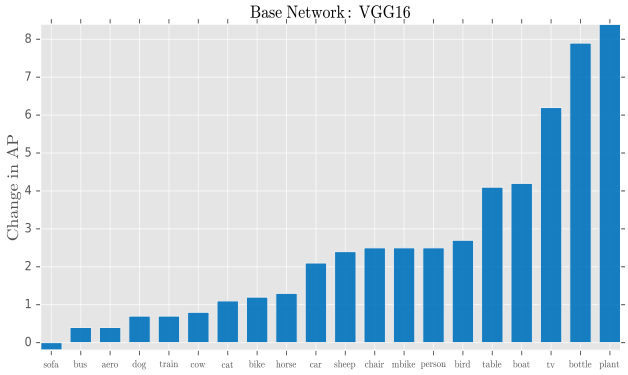


Figure 7: Changes of AP for VOC 2012 compared to baseline FRCN.

6. Conclusion

One of the long-term goals of object detection is to learn object models that are invariant to occlusions and deformations. Current approaches focus on learning these invariances by using large-scale datasets. In this paper, we argue that like categories, occlusions and deformations also follow a long-tail distribution: some of them are so rare that they might be hard to sample even in a large-scale dataset. We propose to learn these invariances using adversarial learning strategy. The key idea is to learn an adversary in conjunction with original object detector. This adversary creates examples on the fly with different occlusions and deformations, such that these occlusions/deformations make it difficult for original object detector to classify. Instead of generating examples in pixel space, our adversarial network modifies the features to mimic occlusion and deformations. We show in our experiments that such an adversarial learning strategy provides significant 2.6% boost in detection performance on VOC 2012 dataset.

References

[1] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks. *arXiv preprint arXiv:1512.04143*, 2015. 2

[2] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2

[3] E. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep gen-

erative image models using a laplacian pyramid of adversarial networks. In *NIPS*, 2015. 2

[4] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 2, 6

[5] R. Girshick. Fast r-cnn. In *ICCV*, 2015. 2, 3, 6, 7, 8

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

[7] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 2

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[9] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu. Spatial transformer networks. In *NIPS*, 2015. 5

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 2, 6

[11] T. Lin, M. Maire, S. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 1, 2

[12] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 2

[13] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *arXiv preprint arXiv:1511.06343*, 2015. 2

[14] M. Mathieu, C. Couprie, and Y. LeCun. Deep multi-scale video prediction beyond mean square error. *CoRR*, abs/1511.05440, 2015. 2

[15] M. Mirza and S. Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014. 2

[16] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *NIPS*, 2015. 2

[17] L. Pinto, J. Davidson, and A. Gupta. Supervision via competition: Robot adversaries for learning tasks. In *CoRR*, 2016. 2

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, abs/1511.06434, 2015. 2

[19] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2015. 2

[20] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 2, 4

[21] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. *CoRR*, 2016. 2

[22] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *CoRR*, abs/1312.6229, 2013. 2

[23] S. Sharma, R. Kiros, and R. Salakhutdinov. Action recognition using visual attention. In *CoRR*, 2016. 5

[24] A. Shrivastava and A. Gupta. Contextual priming and feedback for faster r-cnn. In *ECCV*, 2016. 2

[25] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *CVPR*, 2016. 2

[26] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, and F. Moreno-Noguer. Fracking deep convolutional image descriptors. *arXiv preprint arXiv:1412.6537*, 2014. 2

[27] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 6

[28] stian Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *CoRR*, 2016. 2

[29] M. Takáč, A. Bijral, P. Richtárik, and N. Srebro. Mini-batch primal and dual methods for svms. *arXiv preprint arXiv:1303.2314*, 2013. 2

[30] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015. 2