BAR — A Reinforcement Learning Agent for Bounding-Box Automated Refinement

Morgane Ayle,[1] **Jimmy Tekli,**[2,3] **Julia El-Zini,**[1] **Boulos El-Asmar,**[2] **Mariette Awad**[1]
[1]Maroun Semaan Faculty of Engineering and Architecture, American University of Beirut, Lebanon
[2]Université de Franche Comté, Belfort, France
[3]BMW Group, Munich, Germany
{mna61, jwe04}@mail.aub.edu, mariette.awad@aub.edu.lb,
{jimmy.tekli, boulos.el-asmar}@bmw.de

Abstract

Research has shown that deep neural networks are able to help and assist human workers throughout the industrial sector via different computer vision applications. However, such data-driven learning approaches require a very large number of labeled training images in order to generalize well and achieve high accuracies that meet industry standards. Gathering and labeling large amounts of images is both expensive and time consuming, specifically for industrial use-cases. In this work, we introduce BAR (Bounding-box Automated Refinement), a reinforcement learning agent that learns to correct inaccurate bounding-boxes that are weakly generated by certain detection methods, or wrongly annotated by a human, using either an offline training method with Deep Reinforcement Learning (BAR-DRL), or an online one using Contextual Bandits (BAR-CB). Our agent limits the human intervention to correcting or verifying a subset of bounding-boxes instead of re-drawing new ones. Results on a car industry-related dataset and on the PASCAL VOC dataset show a consistent increase of up to 0.28 in the Intersection-over-Union of bounding-boxes with their desired ground-truths, while saving 30%-82% of human intervention time in either correcting or re-drawing inaccurate proposals.

1 Introduction

Over the years, Deep Learning (DL) has demonstrated great superiority over traditional learning-based approaches in a wide range of computer vision applications (Yang et al. 2019). Object detection and recognition DL techniques allow machines to visualize their environments with high accuracy (Zhang et al. 2018). Hence, companies in many industrial sectors started integrating DL approaches in order to assist human workers and speed up chain processes at their manufacturing sites (Wang et al. 2018) by installing cameras and gathering images specific for their use-cases. Unfortunately, accurately labeling these datasets requires excessive human effort and a considerable amount of time, and the confidentiality of the data renders the use of crowd-sourcing platforms such as Amazon's Mechanical Turk (Buhrmester, Kwang, and Gosling 2011) impossible. Furthermore, the image annotation process is highly prone to human error. For instance, target objects might be wrongly labeled when a human annotator is biased to label one side of the target object. Moreover, after the whole dataset has been labeled, the Region of Interest (RoI) of the target object might change because of industrial circumstances. In these cases, all labels need to be re-adjusted, resulting in unexpected delays and extra costs.

A lot of research (Papadopoulos et al. 2014; 2016; 2017; Yao et al. 2012; Konyushkova et al. 2018) has been recently oriented at reducing the labeling time and avoiding crowd-sourcing techniques by developing methods that are less time consuming than manual labeling. To the best of our knowledge, existing methods generate bounding-boxes (b-boxes) around the target object and no attempts have been made in the literature to correct inaccurate b-boxes.

This work aims at enhancing the accuracy of existing b-boxes, i.e. annotations, to obtain "cleaner" data[1] to be used in industrial use-cases, while reducing the human intervention needed. For this purpose, we implemented BAR (Bounding-box Automated Refinement), a Reinforcement Learning (RL) agent that learns from human examples to correct inaccurate annotations. Hence, instead of manually labeling new images to increase the dataset size, human effort is limited to correcting inaccurate annotations. After learning to find an optimal strategy to correct these annotations, BAR applies its knowledge on new images. We adopt two approaches in order to train BAR: an offline approach using Deep Reinforcement Learning (DRL) in which the agent is trained by batches, and an online approach using Contextual Bandits (CB) in which the agent is re-trained after every new image. We also compare the advantages and limitations of both approaches with different initializations, i.e. methods that generate initial b-boxes. Finally, we show that for all initializations, at least one of the two approaches successfully improves the annotations, with an increase in Intersection-over-Union (IoU) with the ground-truths of up to 0.28, and a decrease in human intervention by 30%-82%.

Our contributions in this work are:

- BAR, an RL agent that improves the quality of existing b-boxes through offline (DRL) and online (CB) methods.

[1]"Clean" data means it is uniformly annotated across the dataset and the b-boxes tightly enclose the RoI of the target object

- BAR-CB, an online-based approach that successfully improves some initializations with the advantage of real-time suggestions of new b-boxes.
- BAR-DRL, an offline-based approach that reaches "cleaner" and more accurate annotations when tested with five different initializations, improving over state-of-the-art work while reducing human input.

The remaining of this paper is organized as such: the literature is presented and compared to this work in section 2. Then, the problem is formally defined in section 3 and both training methods are introduced in section 4. Finally experiments are run and the advantages of BAR are shown in section 5, while section 6 concludes our work.

2 Background and Preliminaries

In this section, we highlight the main applications related to our proposed offline and online approaches, and compare our scheme to the most related literature.

2.1 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) (Mnih et al. 2015) has gained interest in object detection, mainly to reduce the search space during training. In (Bueno et al. 2017), the authors present an agent that learns to progressively zoom in on a target object. At every step, the agent divides its observation into 5 sub-regions: upper-left, upper-right, lower-left, lower-right and center. It then chooses the best sub-region to zoom in on, and repeats this process until reaching an accurate b-box. (König et al. 2019) refine further the b-box by using 4 additional actions: left, right, up or down.

In (Caicedo and Lazebnik 2015), the agent learns to zoom in on the object by sequentially deforming a b-box: starting with a b-box covering the whole image, the agent modifies it by choosing one of its possible transformation actions (right, left, up, down, bigger, smaller, fatter or taller), or ends the detection by selecting the terminal action.

Similar work includes Tree-RL (Jie et al. 2016), which localizes objects using sequential scaling and local translation actions, and (Wu et al. 2018) which applies Tree-RL in a parameterized action space, where actions are associated with continuous rather than discrete parameters.

2.2 Contextual Bandits

A multi-armed bandit (MAB) (Vermorel and Mohri 2005) problem is comparable to a gambler facing different slot machines, each of which will give the player a distinct gain or loss. At every time step, the gambler has to choose one machine and receives a feedback for his/her choice, in an attempt to maximize cumulative rewards. Contextual Bandits (CB) are a derivation of MAB (Li et al. 2010; Chu et al. 2011) in which the agent has additionally access to a representation of the state. A popular application of CB is personalized news articles recommendation (Li et al. 2010) where the agent determines which articles should appear on a user's web-page based on his/her preferences collected from previous articles he/she clicked on. The LinUCB algorithm developed in (Li et al. 2010) allows for re-training on every incoming input and therefore real-time improvements of personalized suggestions. Other applications of MAB and CB range from healthcare and finance to specific problems in machine learning, such as active learning in (Bouneffouf and Rish 2019) which aims at reducing labeling time in a supervised classification setting by selecting the most useful French utterances to label. To the best of our knowledge, CB have not yet been utilized in object detection or recognition.

2.3 Related Work

Some of the existing work aims at replacing the manual drawing of b-boxes with easier and less time consuming tasks, such as eye tracking (Papadopoulos et al. 2014), click-supervision (Papadopoulos et al. 2017), and human verification (Papadopoulos et al. 2016). An interactive object annotation method is presented in (Yao et al. 2012) through which an object detector is incrementally updated while additional annotations are provided by humans. Finally, in (Konyushkova et al. 2018), an agent is trained to iteratively ask the human to verify or correct a b-box in order to reach an accurate annotation in a minimal amount of time.

In summary, existing work focuses on reducing the time spent by human annotators on manual labeling. While, in the literature, this goal is achieved by finding alternative ways to generate b-boxes proposals, our work focuses on learning to correct inaccurately generated b-boxes to later refine annotations regardless of their initialization.

3 Problem Formulation

Every image contains exactly one annotated target object whose b-box is represented by its upper-left corner (x_{min}, y_{min}) and its lower-right corner (x_{max}, y_{max}). This b-box is considered inaccurate if its IoU with the ground-truth is below a threshold, denoted by β. Given an image and an inaccurate b-box enclosing the target object, the goal of the agent is to correct the b-box as shown in figure 1. The agent achieves this goal by executing a series of actions that modify the position and aspect-ratio of the b-box. This series of actions corresponds to an episode that ends with the final correction of the agent.

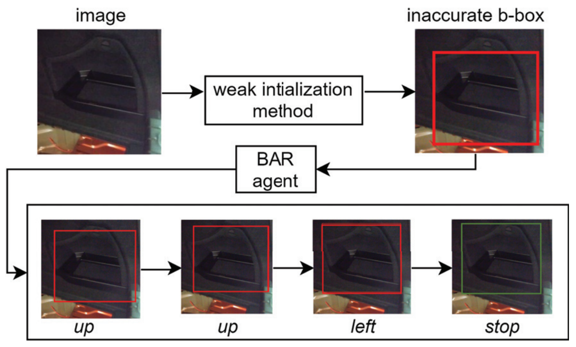

Figure 1: BAR agent workflow during the testing phase. Given an image and an inaccurate b-box enclosing the target object, BAR chooses the path TE with $T = \{up,up,left\}$

The dynamics of an episode are as follows: At time step 0, the b-box is given by a weak initialization, i.e. a method that generates inaccurate b-boxes. At time step 1, the agent performs an action and determines the quality of the new b-box based on its IoU with the ground-truth. If the IoU is below β, the agent needs to modify the b-box in the next time step; otherwise, the agent stops and the episode terminates. The length of an episode is limited to 10 actions to prevent cases where the agent fails to converge to an accurate b-box.

During an episode, the agent is allowed to either move according to a set of predefined translation actions listed in table 1, or to end the episode by choosing the *stop* action. Table 1 reports the equations according to which each of the translation actions changes the corners of the b-box, where c_1 is the percentile of the box's current dimensions (width or height) that is added to or removed from its coordinates. To prevent the b-box from over-scaling, c_2 is set to $c_1/2$.

action	corresponding equations	action	corresponding equations
up	$x_{min} - c_1 \times height$ $x_{max} - c_1 \times height$	*wider*	$y_{min} - c_2 \times width$ $y_{max} + c_2 \times width$
down	$x_{min} + c_1 \times height$ $x_{max} + c_1 \times height$	*taller*	$x_{min} - c_2 \times height$ $x_{max} + c_2 \times height$
left	$y_{min} - c_1 \times width$ $y_{max} - c_1 \times width$	*fatter*	$x_{min} + c_2 \times height$ $x_{max} - c_2 \times height$
right	$y_{min} + c_1 \times width$ $y_{max} + c_1 \times width$	*thinner*	$y_{min} + c_2 \times width$ $y_{max} - c_2 \times width$

Table 1: The set of translation actions

Given the goal of the agent and the dynamics of an episode, the b-box correction problem can be formulated as a Sequential Decision Making Problem (SDMP); specifically, an episodic RL problem. The agent interacts with the environment (as images) to learn a policy that determines the optimal path to change an initial inaccurate b-box into an accurate one. The path is composed of a list of translation actions T, where $0 \leq |T| \leq 9$, and a terminal action E that ends the correction. The task of the agent for the path TE is therefore to find 1) the optimal actions T that modify the b-box and 2) the step at which to end the episode when an accurate b-box is reached.

4 Training Methods

To solve this SDMP, two approaches are proposed: an offline-based with BAR-DRL, where images are accumulated by batches before training starts, and an online-based with BAR-CB, where training is done after every image.

4.1 BAR-DRL

In this method, the problem is cast as a Markov Decision Process (MDP). An MDP is characterized by the following factors: the set of states $\mathcal{S}\{s_1, s_2, \dots\}$, the set of actions $\mathcal{A}\{a_1, a_2, \dots\}$, the transitional probability function $\mathcal{P}(s, a, s')$ where s' is the state reached from s through action a, the discount factor γ where $0 \leq \gamma \leq 1$ and the reward function $\mathcal{R}(s, a, s')$.

The agent is trained according to the Deep Q-Network algorithm presented in (Mnih et al. 2015). The agent's environment is an image, and it takes actions that modify the state, namely the region enclosed in the b-box. The Deep Q-Network relies on Q-Learning, a method that uses a neural network to approximate a value Q at every step and for every action when the transitional probability function is unknown. Specifically, at time step t, the agent updates its Q-value estimate in the following manner:

$$Q_{t+1}(s,a) = (\alpha - 1)Q_t(s,a) + \alpha(r + \gamma \max_{a'} Q_t(s', a')) \tag{1}$$

The agent is therefore trained to maximize its cumulative reward during each episode. The three main components of BAR-DRL are:

1. State: The state is composed of a feature vector $\in \mathbb{R}^{1238}$ extracted using ResNet50 (Szegedy et al. 2017) pre-trained on ImageNet (Deng et al. 2009), and a history vector. The b-box enclosed region is resized to 224×224, then fed to the feature extractor that outputs a vector of size 2048. The history vector encodes the 10 actions of the episode, each of which is represented as a one hot encoder.
2. Actions: These are the eight translation actions shown in table 1 and the *stop* action.
3. Reward: The reward for a translation action a at step t is:

$$r(a_t) = \begin{cases} 1, & \text{if } IoU_t > IoU_{t-1} \\ -3, & \text{otherwise} \end{cases} \tag{2}$$

A higher negative value is necessary when the IoU decreases to prevent the agent from worsening the initial b-box, which defeats the purpose of a correcting agent. The reward for the *stop* action is:

$$r(a_t) = \begin{cases} +r_1 + c * \Gamma, & \text{if } IoU_t \geq \beta \\ -r_2, & \text{otherwise} \end{cases} \tag{3}$$

where: $\Gamma = \frac{10-t}{10}$, $r_1 = 6$, $c = 4$, $r_2 = 3$.
Starting off with heuristics from the literature, the effect of coefficients r_1, c and r_2 were determined through many experiments. The reward r_1 should be a high positive value in order to prevent long episodes that could worsen the b-box, while r_2 follows the same dynamics as for a translation action. An additional reward c discounted by the relative number of steps Γ encourages the agent to find the optimal sequence of actions in the lowest number of steps. This characterization of the reward essentially favors short episodes, while still encouraging the agent to take long ones when needed.

4.2 BAR-CB

To train BAR-CB, the LinUCB (Li et al. 2010) algorithm was adapted to an episodic scenario. Every image is associated with an episode as described in section 3, during which BAR-CB fills its knowledge matrix with experiences (s, a, p). When the episode terminates, it re-fits on the newly accumulated experiences.

At every time step t, BAR-CB observes the state s of the environment and the set of actions $\mathcal{A}$. It constructs a feature vector for every action that summarizes information of the state and that action. Based on previous payoffs, it chooses an action a and receives an associated payoff p. Finally, it improves its strategy by adding the new experience (s, a, p) to its knowledge matrix. The three main components of BAR-CB are:

1. State: BAR-CB learns in an online fashion which necessitates the use of a simple yet meaningful representation of the current observation. The Histogram of Oriented Gradients (HOG) capture well the edges and the outline of the object of interest which makes it suitable to represent the state. To obtain the feature vector $\in \mathbb{R}^{2916}$, the b-box enclosed region is first resized to 224×224. Then, a bilateral filter is applied to eliminate the noise and make the hard edges more salient in the image. Finally, the features are extracted using 9 orientation bins, 12×12 pixels per cell and 1×1 cell per block with an L2 block normalization.
2. Actions: These are the eight translation actions shown in table 1 and the *stop* action.
3. Reward: Since rewards are binary in the LinUCB algorithm, the reward for a translation action a at step t is:

$$r(a_t) = \begin{cases} 1, & \text{if } IoU_t > IoU_{t-1} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

and for the *stop* action:

$$r(a_t) = \begin{cases} 1, & \text{if } IoU_t \geq \beta \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

5 Experiments

5.1 Experimental Setup

The experiments were run on a GeForce GTX 1080 with 11,000 MiB memory and Cuda version 10.0. The dataset used consists of four *car industry*-related classes as shown in figure 2. There are approximately 800 available images for each class, 200 of which are used for testing. The classes *First Aid Kit (FA-Kit) Right, Left* and *Middle* correspond to first aid kits located in the back of a car, and the class *Stickers* corresponds to a sticker in the front of the car that indicates the presence of airbags. Each image of the dataset contains one object and its ground-truth provided by a human expert. For a fair comparison with the literature, BAR-DRL is additionally evaluated on the *aeroplane* class of the publicly available PASCAL VOC dataset (Everingham et al. 2010).

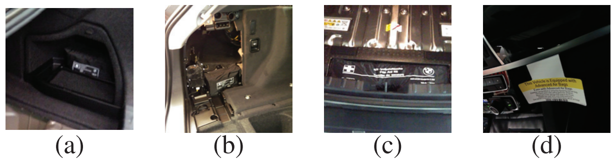

Figure 2: (a) *FA-Kit Right*, (b) *Left*, (c) *Middle* and (d) *Stickers* classes

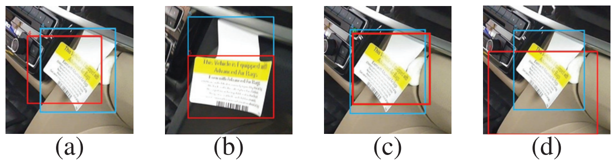

Figure 3: Initial b-boxes (red) vs ground-truth (blue) on instances of *Stickers* class with (a) distribution-based, (b) human, (c) RetinaNet and (d) template matching initializations

5.2 Experimental Design

First, initial b-box proposals are generated using one of the initialization methods detailed in this section. The corresponding ground-truth b-boxes are obtained by letting human annotators correct these generated b-boxes. The agent is then trained with parameters β and c_1 chosen according to the quality of the initial proposals. Finally, BAR is used to correct future inaccurate b-boxes, leaving only a subset of annotations to be corrected by humans. Five experiments are designed, each with a different initialization method and tested with BAR-DRL and/or BAR-CB training methods:

Exp. 1: Distribution-Based Initialization This experiment determines the effect of hyper-parameters on the performance of BAR. The initial proposals are generated according to the following IoU distribution: 5% in the range 0.4-0.55, 35% in the range 0.55-0.6, 50% in the range 0.6-0.7, 5% in the range 0.7-0.75 and 5% in the range 0.75-0.95. Then, BAR-DRL is trained for $\beta = 0.70, 0.75$ and 0.80. The size of the training set varies from 50 to 100 and 150 with $c_1 = 0.1$. BAR-CB is trained with a single configuration of 50 training images, $\beta = 0.75$ and $c_1 = 0.1$.

Exp. 2: Hierarchical Detector (Bueno et al. 2017) Initialization The goal of this experiment is to test the performance of the agent on the public dataset PASCAL VOC and compare to (Bueno et al. 2017) and (König et al. 2019). To obtain initial proposals, a detector is trained as in (Bueno et al. 2017) on the *aeroplane* class of PASCAL 2012 trainval, and then used to obtain b-box proposals on PASCAL 2007 trainval. BAR-DRL is trained on the latter, with 309 training examples, $\beta = 0.6$ and $c_1 = 0.2$. The pipeline is tested on PASCAL 2007 test data and compared to (Bueno et al. 2017) who trained their detector on PASCAL 2007 and 2012 trainvals, and the extended approach in (König et al. 2019). BAR-CB is not used for this experiment, since instances of the *aeroplane* class vary greatly and both the algorithm and the HOG features cannot generalize well when the data changes in shape, texture, context, viewpoint etc.

Exp. 3: Human Annotator Initialization The goal of this experiment is to test the agent in an actual industrial scenario where annotators wrongly annotated an object across an entire dataset. Four cases are considered on our industrial classes: a shift to the right for *FA-Kit Right*, a missing region of the RoI for *FA-Kit Left*, an up-scaling for *FA-Kit Middle* and a missing region of the RoI for the *Stickers* class as shown in figure 3. BAR-DRL and BAR-CB are trained using 50 training images, $\beta = 0.85$ and $c_1 = 0.1$.

Exp. 4: RetinaNet (Lin et al. 2017) Initialization This experiment tests the performance of BAR with good initial proposals. RetinaNet, pretrained on ImageNet (Deng et al. 2009), provides better initial proposals than previous methods as shown in figure 3 when trained on our 4 classes using 20 training images. BAR-DRL is trained on 30 images, with $\beta = 0.80$ and $c_1 = 0.05$. BAR-CB is not used for this initialization since HOG features fail at capturing minor changes.

Exp. 5: Template Matching (Bradski and Kaehler 2008) Initialization This experiment shows the performance of BAR for the widely known template matching (TM) detection method, which detects objects by comparing an input image to a template. Initial proposals are obtained using OpenCV (Bradski and Kaehler 2008) TM function with cross-coefficient normalized similarity index on the 4 classes. These proposals are highly inaccurate compared to the ones provided by other initializations as shown in figure 3. BAR-DRL and BAR-CB are trained on 50 images with $\beta = 0.70$ and $c_1 = 0.1$.

5.3 Parameters of Training Methods

The neural network consists of two fully-connected layers of 500 neurons each with Relu activation and random normal initialization, and an output layer of 9 neurons with linear activation. Mean square error loss with Adam optimizer and learning rate of 0.001 are used, and the discount factor γ for the Q-function is set to 0.90. BAR-DRL is trained for 80 epochs using an experience replay with maximum memory length of 2000, and epsilon-greedy exploration with $\epsilon = 1.0$ initially and linearly annealing to 0.1 for 15 epochs.

BAR-CB is trained using the implementation of (Cortes 2018). To kick-start its training, it is initially fitted to expert human examples: first, a human annotator corrects half the data, then the series of actions used in the correction is determined by a script and the agent is trained on these examples. Second, the agent corrects the remaining half of the b-boxes while receiving human feedback as a b-box verification if it is accurate or a manual correction otherwise. [2]

5.4 Experimental Results

Metrics and Evaluation The metrics reported in the results are the Average Precision (AP), the number of True Positives (TP) and False Positives (FP). The notation X@xx represents the metric X (AP, TP or FP) at an IoU of 0.xx, with xx=$50, 70, 75$ or 80, depending on the desired quality of new detections. The mean Average Precision (mAP) is the AP averaged over all classes. The value of TP@xx (FP@xx resp.) corresponds to the number of predictions whose IoU is higher (lower resp.) than 0.xx.

The percentage of human intervention saved is determined by the ratio of the number of additional TP after the correction stage to the number of FP before it.

Exp. 1 The initial IoU for this case is 0.60 on average as reported in table 2. With BAR-DRL, an increase of 0.14 in IoU on average and an improvement to up to 80% in the AP@70 are achieved as seen in table 3. As expected, setting a higher β increases the final IoU reached, and often compensates for a small number of images: training on 50 images and a β of 0.80 performs similarly to training on 100 images and a β of 0.70. However, this is not always the case: the *FA-Kit Middle* class, which has a lower initial AP than other classes, performs better with lower β values, especially with smaller amount of images. Therefore, when the initial proposals are far from the ground-truths, a high β can negatively affect the training.

With BAR-CB, the AP@70 and number of TP@70 increased for all classes as seen in table 4. However, the AP@50 decreased, indicating that some of the b-boxes considerably worsened due to the inability of BAR-CB to generalize well when the initial proposals vary randomly.

Class	FA-Kit right	FA-Kit left	FA-Kit middle	stickers
Average IoU	0.60	0.59	0.60	0.59
AP@50	84.64	87.58	75.18	84.64
TP@50	184	186	158	184
FP@50	16	14	42	16
AP@70	1.39	1.73	1.4	1.14
TP@70	21	22	20	17
FP@70	179	178	180	183

Table 2: Metrics of initial proposals in Exp. 1

Exp. 2 As seen in table 4, the initial detector trained on PASCAL 2012 performs worse than the one in (Bueno et al. 2017) trained with PASCAL 2007 and 2012. However, it outperforms it after the correction trained on PASCAL 2007 especially for recall > 0.1. This highlights the added value of using a subset of the training images for an additional correction stage rather than an exclusive detection stage. Table 5 shows that the initial detector resulted in a 7.9% AP@50, 32% of TP@50 and 68% of FP@50. BAR-DRL outperforms (König et al. 2019) by increasing the AP@50 to 23.5% and the TP@50 to 57% versus a maximum achieved of 39% TP@50 for (König et al. 2019) as reported in table 5.

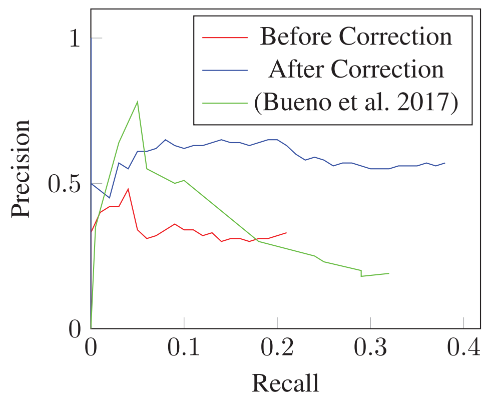

Figure 4: Precision-Recall curve in Exp. 2

Exp. 3 This initialization causes the mAP@70 of initial proposals to decrease to 13% compared to the desired ground-truths for 3 of the 4 classes as seen in table 6, and the

[2]Code available at https://gitlab.com/MorganeAyle/bar

Class	FA-Kit right									FA-Kit left								
Number of images	50			100			150			50			100			150		
β	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80
Average (Avg) IoU	0.72	0.73	0.75	0.74	0.77	0.80	0.76	0.80	0.81	0.73	0.76	0.78	0.74	0.76	0.80	0.75	0.77	0.80
AP@50	92.2	91.8	91.5	95.6	97.9	97.8	**99.1**	97.3	98.6	95.2	95.6	95.4	96.9	**99.4**	98.0	99.2	98.6	99.2
TP@50	191	191	191	195	197	198	199	197	198	194	195	194	197	199	198	199	199	199
FP@50	9	9	9	5	3	2	1	3	2	6	5	6	3	1	2	1	1	1
AP@70	39.1	45.2	52.5	48.4	68.0	71.0	63.2	77.9	**82.1**	48.3	59.0	72.2	63.0	60.0	79.0	57.0	70.5	**84.9**
TP@70	124	133	145	136	164	168	160	176	181	132	149	166	156	152	175	148	163	178
FP@70	76	67	55	64	36	32	42	24	19	68	51	34	44	48	25	52	37	22
Class	FA-Kit middle									stickers								
Number of images	50			100			150			50			100			150		
β	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80	0.70	0.75	0.80
Avg IoU	0.67	0.68	0.68	0.69	0.71	0.70	0.68	0.70	0.69	0.71	0.75	0.76	0.73	0.74	0.79	0.73	0.77	0.79
AP@50	78.5	85.6	79.5	**91.1**	85.1	87.5	84.7	89.7	89.3	95.5	98.1	95.8	97.5	97.6	99.0	98.9	98.2	**100**
TP@50	175	184	177	190	183	185	181	187	187	195	198	195	197	197	199	199	198	200
FP@50	25	16	23	10	17	15	19	13	13	5	2	5	3	3	3	1	2	0
AP@70	21.1	23.4	27.8	25.2	**33.9**	30.1	29.4	31.7	26.8	34.9	52.7	58.0	46.6	48.9	**76.8**	45.7	66.6	75.6
TP@70	90	96	104	97	115	108	103	107	101	114	143	148	130	134	171	126	158	170
FP@70	110	104	96	103	85	92	97	93	99	86	57	52	70	66	29	73.7	41.7	30

Table 3: Results of BAR-DRL in Exp. 1. Best case across a) fixed number of images underlined, b) all class cases in bold

Class	FA-Kit right	FA-Kit left	FA-Kit middle	stickers
Avg IoU	0.66	0.64	0.59	0.57
AP@50	81.0	79.3	65.3	60.4
TP@50	180	178	160	151
FP@50	20	22	40	49
AP@70	14.4	12.0	6.0	6.89
TP@70	75	69	47	48
FP@70	125	131	153	152

Table 4: Results of BAR-CB in Exp. 1

	AP@50	% of TP@50	% of FP@50
Initial	7.9	32	68
BAR-DRL	23.5	57	43
(König et al. 2019)	–	39	56

Table 5: Initial and BAR-DRL metrics in Exp. 2 compared to (König et al. 2019)

AP@80 for the *FA-Kit Left* class to decrease to 73.3%. Using BAR-DRL, the mAP@70, TP@70, mAP@80 and TP@80 increase by 56%, 47%, 43% and 49% respectively. Starting off with 6 TP@70 for the *FA-Kit Middle* class, the agent increases this value to 192, hence correcting 96% of the inaccurate b-boxes. BAR-CB also increases the AP and TP for all classes as shown in table 6. For example, TP@70 and TP@80 improve by 66 and 62 respectively on average.

Exp. 4 RetinaNet sometimes generates more than one b-box for the same object, or no b-box at all. That is why initially $TP + FP \geq 200$, and after correction, $TP + FP < 200$ since BAR acts only on the b-box with greatest confidence. Starting with an initial mAP@75 of 39.2%, BAR-DRL consistently increases the AP@75 and decreases the FP@75 as shown in table 7. For example, with an initial IoU of 0.74 and 10.8% AP@75 for the *Stickers* class, BAR-DRL successfully corrects 43% of the detections. This shows that even with a good initialization, BAR can further enhance the detections.

	Class	FA-Kit right	FA-Kit left	FA-Kit middle	stickers
	Bias Action	shift right	missing part	up-scale	missing part
Initial	AP@70	7.4	100	0.3	31.3
	TP@70	52	200	6	108
	FP@70	148	0	194	92
	AP@80	0.7	73.3	0.0	1.5
	TP@80	14	168	0	24
	FP@80	186	32	200	176
BAR-DRL	AP@70	83.6	96.1	93.1	77.9
	TP@70	182	196	192	168
	FP@70	18	4	8	32
	AP@80	50.4	90.0	64.2	39.5
	TP@80	141	187	160	108
	FP@80	59	13	41	92
BAR-CB	AP@70	72.9	98.4	50.8	41.3
	TP@70	170	198	142	121
	FP@70	30	2	58	79
	AP@80	44.3	84.19	22.6	7.0
	TP@80	132	181	94	45
	FP@80	68	19	106	155

Table 6: Initial, BAR-DRL and BAR-CB metrics in Exp. 3

	Class	FA-Kit right	FA-Kit left	FA-Kit middle	stickers
Initial	Avg IoU	0.78	0.79	0.76	0.74
	AP@75	53.6	53.5	38.8	10.8
	TP@75	138	135	113	61
	FP@75	86	66	98	139
	TP+FP@75	224	201	211	200
BAR-DRL	Avg IoU	0.80	0.81	0.79	0.81
	AP@75	59.9	69	48	64.2
	TP@75	154	161	145	147
	FP@75	45	38	48	34
	TP+FP@75	199	199	193	181

Table 7: Initial and BAR-DRL metrics in Exp. 4

Exp. 5 Table 8 shows that template matching (TM) provides annotations with very low initial IoU (0.36 on average) and that the initial AP for class *Stickers* is lower than for the other classes. This can be explained by the fact that TM provides many initial proposals with 0 IoU, i.e. an observation that doesn't contain any part of the desired object.

BAR-DRL increases the IoU by 0.16 on average and the mAP@50 by 46.5% as seen in table 8, with the classes *FA-Kit Right* and *Left* showing notable increases of 0.20 and

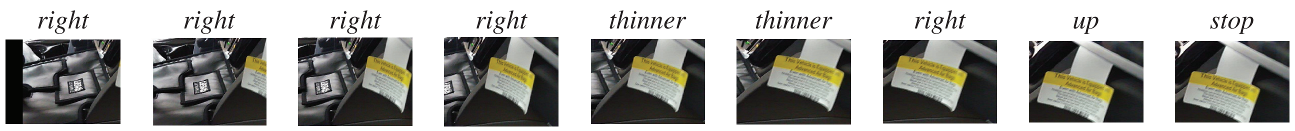

Figure 5: Example of an episode on *Stickers* class in Exp. 5

0.28 respectively. The average number of TP@50 increases by 70. Since many b-boxes for the *Stickers* class had an initial IoU of 0, some of the training parameters need to be adapted. By setting $c_1 = 0.2$ and $\beta = 0.5$, the agent can reach an accurate b-box in fewer steps and is rewarded for lower IoU values. It can now be seen from figure 5 that even with initial proposals completely missing the object, the agent moves towards the target object.

Table 8 shows that BAR-CB increases the AP for the first three classes, but not for the *Stickers* class due to initial b-boxes completely missing the object. The AP@50 for class *FA-Kit Right* doubles and the TP@50 increases by 20%, for a total of 60% TP@50.

	Class	FA-Kit right	FA-Kit left	FA-Kit middle	stickers
Initial	Avg IoU	0.51	0.40	0.35	0.16
	AP@50	25.01	2.7	8.41	5.06
	TP@50	99	33	58	32
	FP@50	101	167	142	168
BAR-DRL	Avg IoU	0.70	0.68	0.48	0.21
	AP@50	88.5	79.9	18.3	8.9
	TP@50	187	177	85	51
	FP@50	13	23	115	149
BAR-CB	Avg IoU	0.60	0.55	0.48	0.17
	AP@50	50.2	41.2	25.1	5.23
	TP@50	141	124	99	30
	FP@50	59	76	101	170

Table 8: Initial, BAR-DRL and BAR-CB metrics in Exp. 5

Human Intervention Depending on the initialization method, our agents successfully correct a subset of the b-box proposals, relieving humans from either correcting them or re-drawing accurate b-boxes. Even though a few b-boxes could get worsened when the agent fails to converge, the number of TP (FP) always increases (decreases), and the overall number of b-boxes to be corrected by a human is always lower than it was before the correction stage. As seen in figure 6, BAR-DRL and BAR-CB save up to 82% and 58% of human time respectively (c.f. section 5.4: Metrics and Evaluation). It is to be noted that the reported values at 0% correspond to cases where the agent could not be applied. The highest achieved percentages are obtained in Exp. 3, which corresponds to the actual industrial scenario. A correction agent is therefore of great advantage for industrial applications in saving human time and achieving higher accuracies. Additionally, the inference time of BAR-DRL and BAR-CB on 1 image are 15ms and 2.5sec on average respectively, with BAR-CB having the convenient online property. Clearly, both agents result in a lower total correction time than letting human annotators perform the task.

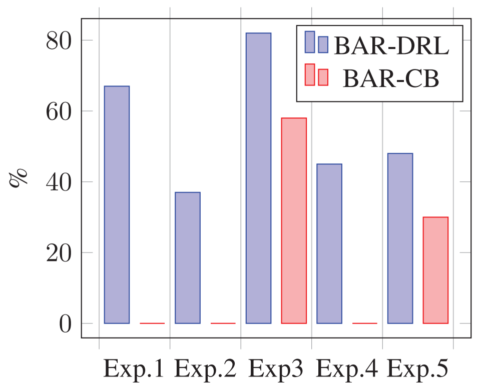

Figure 6: Percentage of human intervention saved for the 5 experiments. BAR-CB could not be used in Exp.1, 2 and 4.

6 Conclusion

In this work, BAR, a novel RL agent, was implemented to accurately correct "weak" b-boxes while reducing human intervention. During the training phase, our agent requires labelers to correct few inaccurate b-boxes instead of drawing new ones as in existing methods, and learns an optimal strategy to determine the correction path on new b-boxes during testing. The correction agents BAR-DRL and BAR-CB were tested on a real industrial dataset, and BAR-DRL was additionally tested on PASCAL VOC. Depending on the initialization method and on the desired quality of b-boxes, BAR-DRL and BAR-CB allow for a reduction in human intervention needed by 37%-82% and 30%-58% respectively while increasing the IoU of b-boxes by up to 0.28. BAR-DRL performs almost twice as good as BAR-CB, which only improved the proposals of 2 out of the 5 initializations considered. However, BAR-CB has the advantages of a faster training time, an online setting, and the ability to suggest acceptable b-boxes after seeing only one training image. Finally, BAR was able to converge to the target object even when the initial detection completely missed it.

7 Future Work

This work focuses on images with a single object, and the proposed methods are easily applicable to multiple objects as long as they do not overlap. A potential extension is to exploit approaches that can handle overlapping objects.

Since human intervention through correction is an important factor in training, the problem could be formulated in an active learning setting in which the agent asks for human feedback while training when it is uncertain. This could also help in reducing the effect of noisy data provided by humans.

Another approach is to introduce Hybrid BAR, which chooses to use either the CB or DRL method depending on the use-case and image structural similarity of the dataset.

Finally, to obtain a better measure of the human intervention being saved, real-life measurements could be performed on a sample population. Using a continuous rather than fixed parameter for translation actions would also allow BAR to easily adapt to different types of initializations independently from parameter tuning.

References

Bouneffouf, D., and Rish, I. 2019. A survey on practical applications of multi-armed and contextual bandits. *arXiv preprint arXiv:1904.10040*.

Bradski, G., and Kaehler, A. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.".

Bueno, M. B.; Giró-i Nieto, X.; Marqués, F.; and Torres, J. 2017. Hierarchical object detection with deep reinforcement learning. *Deep Learning for Image Processing Applications* 31(164):3.

Buhrmester, M.; Kwang, T.; and Gosling, S. D. 2011. Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data? *Perspectives on psychological science* 6(1):3–5.

Caicedo, J. C., and Lazebnik, S. 2015. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, 2488–2496.

Chu, W.; Li, L.; Reyzin, L.; and Schapire, R. 2011. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 208–214.

Cortes, D. 2018. Adapting multi-armed bandits policies to contextual bandits scenarios. *arXiv preprint arXiv:1811.04383*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision* 88(2):303–338.

Jie, Z.; Liang, X.; Feng, J.; Jin, X.; Lu, W.; and Yan, S. 2016. Tree-structured reinforcement learning for sequential object localization. In *Advances in Neural Information Processing Systems*, 127–135.

König, J.; Malberg, S.; Martens, M.; Niehaus, S.; Krohn-Grimberghe, A.; and Ramaswamy, A. 2019. Multi-stage reinforcement learning for object detection. In *Science and Information Conference*, 178–191. Springer.

Konyushkova, K.; Uijlings, J.; Lampert, C. H.; and Ferrari, V. 2018. Learning intelligent dialogs for bounding box annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9175–9184.

Li, L.; Chu, W.; Langford, J.; and Schapire, R. E. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, 661–670. ACM.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Papadopoulos, D. P.; Clarke, A. D.; Keller, F.; and Ferrari, V. 2014. Training object class detectors from eye tracking data. In *European conference on computer vision*, 361–376. Springer.

Papadopoulos, D. P.; Uijlings, J. R.; Keller, F.; and Ferrari, V. 2016. We don't need no bounding-boxes: Training object class detectors using only human verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 854–863.

Papadopoulos, D. P.; Uijlings, J. R.; Keller, F.; and Ferrari, V. 2017. Training object class detectors with click supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6374–6383.

Szegedy, C.; Ioffe, S.; Vanhoucke, V.; and Alemi, A. A. 2017. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Vermorel, J., and Mohri, M. 2005. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, 437–448. Springer.

Wang, J.; Ma, Y.; Zhang, L.; Gao, R. X.; and Wu, D. 2018. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems* 48:144–156.

Wu, Z.; Khan, N. M.; Gao, L.; and Guan, L. 2018. Deep reinforcement learning with parameterized action space for object detection. In *2018 IEEE International Symposium on Multimedia (ISM)*, 101–104. IEEE.

Yang, W.; Zhang, X.; Tian, Y.; Wang, W.; Xue, J.-H.; and Liao, Q. 2019. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*.

Yao, A.; Gall, J.; Leistner, C.; and Van Gool, L. 2012. Interactive object detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 3242–3249. IEEE.

Zhang, Q.; Yang, L. T.; Chen, Z.; and Li, P. 2018. A survey on deep learning for big data. *Information Fusion* 42:146–157.