BAR – A reinforcement Learning Agent for Bounding-Box Automated Refinement (BAR)

Problem Statement: Gathering and labelling of large datasets is a hard and time consuming task with a chance that there will be some error when annotating the data mainly due to human intervention. Traditional deep learning approaches uses large amount of the aforementioned approach for object detection. But there might be a chance that the industry standard accuracy may not be met due to dataset being inaccurate. So this paper attempts to solve this problem by using a reinforcement learning agent.

Aim: To suggest a new reinforcement learning based approach to correct the wrong bounded-box approach such as to limit human intervention to correcting or verifying the subset of bounding-boxes instead of redrawing new ones.

Literature Review:

Use of DL to assist human workers throughout the industrial field is quite an old news. Most of the times the application of deep learning in the field of industries is more often than not is computer vision starting from the inventory management using bots, delivering goods, segregating the junk, food items and much more. The only issue with such data driven learning approaches is that these require a large number of labeled data in order to generalize well and achieve high accuracy.

Gathering and labelling the data is one of the biggest and the most time consuming process when using data driven approaches. Human annotations and labelling always comes with a latent issue of potential error which may act and amplify the wrong annotation in deep neural networks with many layers.

So in this paper the authors have taken a different approach to solve the issue at hand involving the labelling and gathering of large amount of data.

BAR (Bounding-Box Automated Refinement) is a reinforcement learning approach in which agent learns to correct the inaccurate bounding-boxes that are weakly generated using any detection method, or maybe wrongly annotated by human. Either way this is one of the different out of the box approaches I have come across.

The paper implements two types of bounding box correction algorithms. One of them being an Offline Implementation (BAR-DRL) and another an Online Implementation (BAR-CB).

The agent in this just corrects and verifies the present bounding boxes instead of drawing new ones.

After labelling the whole dataset the **Region of Interest** (RoI) might change because of industrial circumstances. In these cased the label needs to be adjusted which is an unnecessary overhead and a wastage of time and effort.

This also papers gives out examples in which bounding-box approach for labelling have been done and avoiding crowdsourcing techniques for the same by developing methods that are less time

consuming than manual labeling. Existing methods just generate bounding-boxes but no attempt is done to correct inaccurate b-boxes.

This work aims at enhancing the accuracy of existing b-boxes, i.e. annotations, to obtain "cleaner" data1 to be used in industrial use-cases, while reducing the human intervention needed.

For this purpose, they 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.

Two types of approaches is used in this paper:

- BAR: an offline approach using Deep Reinforcement Learning (DRL) in which the agent is trained by batches
- Online Approach using Contextual Bandits (CB) in which the agent is re-trained after every new image.

Limitations and advantages to both the agents are compared with different initializations, i.e. method to generate the initial b-boxes.

Approach:

Every image contains exactly one annotated target object whose b-box is represented by its upper-left corner (*xmin*, *ymin*) and its lower-right corner (*xmax*, *ymax*). This b-box is considered inaccurate if its IoU with the groundtruth 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. 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.

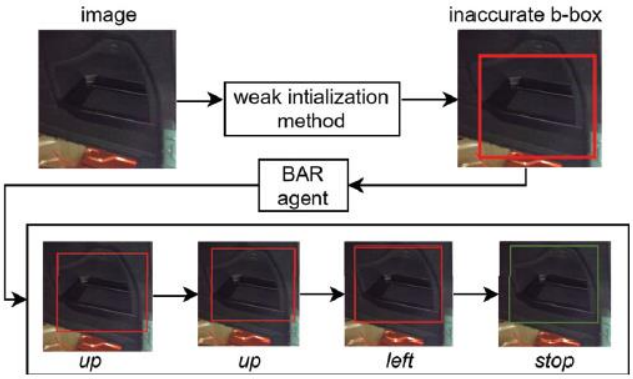

Figure 1: BAR agent workflow during the testing phase. Given an image and an inaccurate b-box enclosing the target object, BAR chooses the path TE with $T = \{up,up,left\}$

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

The dynamics of an episode are as follows:

1) At time step 0, the b-box is given by a weak initialization that generates inaccurate b-boxes.
2) At time step 1, the agent performs an action and determines the quality of the new b-box based on it IoU with the ground truth. If IoU is below a certain threshold β , the agent modifies the b-box in the next time step.
3) If the modification is not required the agent terminates the episode.
4) The episode length is fixed to 10 actions to prevent cases when there is failure to convergence.

Since the goal of agent is sequentially follow steps to arrive at the required solution, the b-box correction problem is formulated as Sequential Decision Making Problem (SDMP), moreover episodic RL problem.

- **Environment** : Image
- State : Feature Vector for offline, and HOG for online.
- Actions : 8 actions, left, right, up, down, fatter, thinner, wider, taller.
- Reward : Change in IoU

Flow:

The agent interacts with the environment (image)

↓

Learns policy to optimize the path it should take to change an initial inaccurate b-box

↓

Path contains translation actions (T), which the agent can take $0 \leq |T| \leq 9$

↓

Episode End (E)

Thus the task of the agent for the path TE is to find :

1) 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

Training Methods:

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 $S\{s1, s2, \ldots\}$, the set of actions $A\{a1, a2, \ldots\}$, the transitional probability function $P(s, a, s\_)$ 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 $R(s, a, s')$.The agent is trained according to the Deep Q-Network algorithm.

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.

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

Fig. Q Value Estimate Equation

The agent is therefore trained to maximize its cumulative reward during each episode.

- Three Main components of BAR-DRL:
  1) State: The state is composed of a feature vector $\in$ R1238 extracted using ResNet50 pre-trained on ImageNet, 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: The 8 actions mentioned in the translational action in the table given and the stop action.
3) Reward: The reward for the translations action *a* at the step *t* is :

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

The higher negative value is necessary when the IoU decreases to prevent the agent from worsening the initial b-box, which defeats the purpose of correcting agent.

The reward for *stop* action is:

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

$$\Gamma = \frac{10-t}{10}, r_1 = 6, c = 4, r_2 = 3.$$

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.

2) BAR-CB:

To train BAR-CB, the LinUCB algorithm was adapted to an episodic scenario. Every image is associated with an episode (either move according to a set of predefined translational actions or stop) 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 *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.

As in previous approach, this approach to has 3 main components:

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 $\in R^{2916}$, 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 discussed above and *stop* action.
3) Reward: Since the rewards are binary in LinUCB algorithm, the reward for action *a* at step *t* is:

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

and for *stop* action:

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

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, BARDRL 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.

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 as an active learning setting in which the agent asks for human feedback while training when it is uncertain.

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.