Object Detection with Deep Q-Network Agents

Junlong Liu, Ke Tang, Guiying Li, Chunhui Jiang
USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI)
School of Computer Science and Technology
University of Science and Technology of China, Hefei, China, 230027
Email: junlong@mail.ustc.edu.cn, ketang@ustc.edu.cn

***Abstract*—Object detection can be interpreted as the task of searching for optimal regions in visual space. Most existing methods are one pass sampling process, which are not the usual way we handle search and optimization problems. In this paper, we train closed-loop deep reinforcement learning agents to localize objects starting from random regions without any prior knowledge. In details, we train a deep Q-network as pixels-to-action agent to localize one class objects by curriculum deep Q-learning. The agent automatically discovers visual cues that help localization through reward stimuli. To localize multi-class objects, simply train multiple deep Q-network agents. We train an extra CNN classifier to filter out background regions generated by DQN agents. # The proposed method is evaluated on the object detection task of PASCAL VOC 2007 and 2010, and achieves state-of-the-art performance. #**

I. Introduction

Object detection, as one of the fundamental challenges in computer vision, is a prerequisite step in a broad range of higher-level vision tasks and can support a long list of piratical applications, such as scene content understanding, activity recognition, intelligent video surveillance [1], robot navigation [2], content-based image retrieval [3], etc.

Object detection can be interpreted as the task of searching for optimal regions in visual space [4], and generally feedback scheme is needed to guide the search towards more promising regions and to optimize IoU (Intersection-over-Union) index. However, most existing methods can be viewed as one-pass sampling methods, which composed of region sampling, feature generation and classifier modules [4], like region-based methods [5] [6] [7]. This is not the usual way we handle search and optimization problems.

Reinforcement learning [8] can be utilized to train closed-loop detection agents with feedback scheme. Neuropsychological evidence suggests that human beings learn to extract useful information from visual data in an interactive fashion, without any external supervisor [9]. By evaluating the consequences of our actions on the environment, we gradually learn to focus on visual cues that are beneficial for solving the tasks, which coincides with the basic idea of reinforcement learning [10].

Combination of deep convolutional neural network (CNN) [11] and reinforcement learning has successfully created end-to-end pixels-to-action agents, which achieves significant progress in game playing [12], robotic control [13]. Deep Q-network [12] is one of the representatives, which composes of a few convolutional layers, fully connected hidden layer(s) and action output layer, and achieves human-level performance in Atari [14] games.

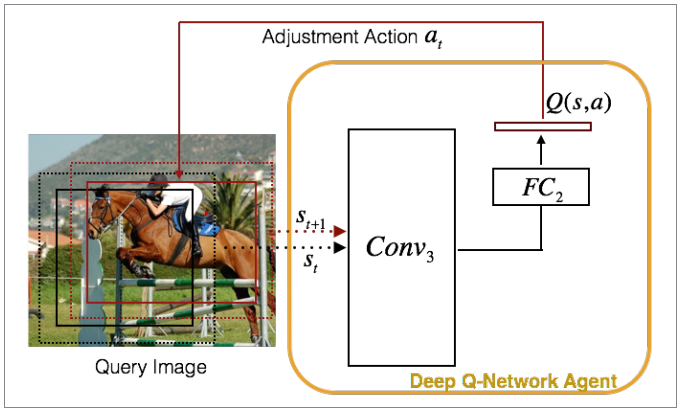

Fig. 1. Object detection with deep Q-network agents. State input of the agent is raw pixels of detection region and corresponding extended region. The agent start from random region and adjust the detection region step by step to localize objects or give up with background regions. A CNN classifier will filter out background regions proposed by the agent.

Combination of CNN and reinforcement learning makes control of object detection region much more feasible and reliable. CNNs have achieved significant breakthrough in visual perception, which makes visual cues in detection region much more sensible to reinforcement learning agents. Visual cues include, but are not limited to, spatial relationship of different objects and of same objects' components. For example, legs are the bottom of human beings, and laptops are usually put on tables. Reinforcement learning makes visual cues discovery automatic through reward stimuli, then the agent can make use of visual cues to search for target objects.

In this paper, we train deep Q-network (DQN) as pixels-to-action agent to localize one-class objects automatically. To detect multi-class objects, simply train multiple deep Q-network agents. The agent starts with a few set of random detection regions of query images, then adjust regions step by step until reaching optimal regions or giving up in background regions. Then an extra CNN classifier is trained to filter out background regions. We adopt curriculum learning to accelerate reward back-propagation and devise guided ε-greedy policy to boost exploration efficiency. # The proposed method is evaluated on the object detection task of PASCAL VOC [15] 2007 and 2010, and achieves state-of-the-art performance. #

II. Related work

Object detection consists of two sub-tasks, object classification and localization. The output of object detection with query images is pairs of bounding box and corresponding category.

Conventional object detection methods generally include three modules: representation of instances and categories, localization of object regions, and supervised classification methods [4]. Object detection procedure can be performed in the following steps: (i) enumerate all possible regions in the input image, (ii) decide whether or not each of them corresponds to any of the predefined categories, and (iii) evaluate all regional responses to obtain the overall detection results [4]. Before Krizhevsky et al. [16] rekindled interest in CNNs, representation of instances and categories are mostly different-level hand-crafted features. Localization strategies include sliding window, voting [17], selective search [18], segmentation.

Recent years, R-CNN (Regions with CNN features [5]) based methods [6] [7], managed to unify the three modules (representation, localization and classification) into one CNN. The key idea of original R-CNN [5] is to obtain all region proposals using techniques like selective search [18], then extract CNN features of these region proposals and classify with SVMs. Fast R-CNN [6] unified representation and classification into one CNN, which is trained with multi-task loss to classify and refine regions simultaneously. To obtain faster R-CNN, Shaoqing Ren et al. [7] designed a Region Proposal Network (RPN) that shares full-image convolutional features with the detection network, which unified feature representation, localization and classification into one CNN.

However, these works rely on one-pass sampling process over whole visual space to generate high-quality region proposals, such as selective search [18], multi-box [19], edge boxes [20], and RPN [7]. As mentioned above, object detection can be viewed as search and optimization problem, and interactive agents with feedback scheme might achieve better results.

III. Background

We consider object localization as tasks that agents interact with the visual image with a sequence of actions, observations and rewards. Starting with a random detection region within the image, at each time-step the agent observes current region x_t and selects an action a_t from the set of adjustment actions, $\mathcal{A} = 1, ..., K$. The action is executed to adjust detection region of the visual image. Then it receives reward r_t representing detection region's IoU improvement with ground truth box. This formalism gives rise to a finite Markov decision process where each x_t or $\Phi(x_t)$ is a distinct state s_t. $\Phi(x_t)$ is mapping function of x_t.

The agent learns to interact with the visual image by selecting actions in a way that maximizes discounted future rewards, namely to locate the ground truth box as soon as possible. The future discounted return at time t is defined as $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r^{t'}$, where T is the time-step at which the detection process terminates. The optimal action-value function $Q^*(s, a)$ is defined as the maximum expected reward return achievable by following any strategy after taking action a at state s, $Q^*(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi]$, where π is a policy mapping states to actions.

The optimal action-value function $Q^*(s, a)$ obeys the Bellman equation, which can be expressed mathematically as:

$$Q^*(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q^*(s', a') | s, a]$$

Many reinforcement learning algorithms estimate the action-value function with the Bellman equation as an iterative update, $Q_{i+1}(s, a) = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q_i(s', a') | s, a]$. Such value iteration algorithms converge to the optimal action-value function, $Q_i \rightarrow Q^*$ as $i \rightarrow \infty$ [8]. However, the action-value function is estimated separately for each state, without any generalization. This basic approach is impractical in practice, and a function approximator is needed to estimate the action-value function, $Q(s, a; \theta) \simeq Q^*(s, a)$. Inspired by [12], we use a convolutional neural network as a non-linear function approximtor for $Q^*(s, a)$. This deep Q-network can be trained by minimizing a sequence of loss functions $L_i(\theta_i)$ that changes at each iteration i,

$$L_i(\theta_i) = \mathbb{E}_{s,a}[(y_i - Q(s, a; \theta_i))^2]$$

where $y_i = \mathbb{E}_{s'}[r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) | s, a]$ is the target for iteration i. The parameters from the previous iteration θ_{i-1} are held fixed when optimizing the loss function $L_i(\theta_i)$. Differentiate the loss function with respect to the weights and update θ_i:

$$\nabla_{\theta_i} L_i(\theta_i) = \mathbb{E}_{s,a,a'}[(r + \gamma \max_{a'} Q(s', a'; \theta_{i-1}) - Q(s, a; \theta_i)) \nabla_{\theta_i} Q(s, a; \theta_i)]$$

Computing the full expectations in the above gradient can be too costly to be practical. It is often computationally expedient to optimize the loss function by stochastic gradient descent (SGD).

IV. our approach

A. Overview

As *Figure 1* shows, a DQN agent manages to localize target class object step by step. At each time step, the agent perceives an extended observable detection region (the paired rectangles with solid and dash lines) as state input s_t, then feeds s_t forward through Q-network, to obtain $Q(s_t, a | a \in \mathcal{A})$. After that the agent chooses the action that maximizes $Q(s_t, a)$ to adjust the detection region. At last, the agent will stop initiatively at regions of interest, or stop passively at background regions due to maximum step limitation. Thus an extra CNN classifier is trained to filter out background regions.

It should be noted that, we train a DQN to localize each class of objects and train the CNN classifier with multi-class datasets. This design not only reduces the training difficulty of DQN compared to multi-class DQN, but also achieves double check on each region. Firstly, the region should be searched out by DQN. Secondly, it must pass the classifier filtering test.

Algorithm 1 curriculum deep Q-learning

Initialize replay memory D to capacity $N, t = 0$
Initialize action-value function Q with random weights
Initialize maximum steps M for exploring each region
for $t < t_{max}$ **do**
  for I_i in training set **do**
    Get starting region x_t with curriculum difficulty λ_t
    for $i = 1, M$ **do**
      Set $s_t = \Phi(x_t)$
      With probability ε to apply guided exploration
      Otherwise select $a_t = \max_a Q(s_t, a; \theta)$
      Execute a_t to transform x_t into x_{t+1} and get r_t
      Set $s_{t+1} = \Phi(x_{t+1})$
      Store (s_t, a_t, r_t, s_{t+1}) in D
      Sample a minibatch of (s_j, a_j, r_j, s_{j+1}) from D
      Set

$$y_j = \begin{cases} r_j, \ if \ a_j = stop \\ r_j + \gamma max_a \hat{Q}(s_{t+1}, a; \theta), \ otherwise \end{cases}$$

      Perform a gradient descent step on $(y_j - Q(s_j, a_j; \theta))^2$ with respect to the parameters θ
      Every C steps reset $\hat{Q} = Q$
      $t = t + 1$
    end for
  end for
end for

Algorithm 2 one class object detection

Input: Quey Image I_q
Output: Set of (region,score) pairs Γ

Initialize $\mathcal{R}$ with regions of different scales(1.0, 0.5, 0.25)
Set maximum steps as M
for $x_0 \in \mathcal{R}$ **do**
  for t=0,M **do**
    Set $s_t = \Phi(x_t)$
    Select $a_t = \max_a Q(s_t, a; \theta)$
    Execute action a_t to adjust x_t into x_{t+1}
    if a_t = stop **then**
      break out from inner loop
    end if
  end for
  Add x_j into Γ, where $j = arg\min_j(\max_a(Q(s_j, a; \theta))$
end for
Filter out background regions in Γ with CNN classifier
Sort regions r_i in Γ by classification score
Kick out r_i in Γ if IoU with better regions > 0.3

B. Formalism of Object Localization as RL

State: The state here refers to raw pixels of extended observable detection regions. Namely each state composes of one detection region, $(x, y, w(idth), h(eight))$, and one extended observable region, $(x, y, w + \frac{w}{2}, h + \frac{h}{2})$. For RGB data, each state composes of 3×2 feature map, and they are all resized to 84×84. Thus, state input s_t is $84 \times 84 \times 6$. These transformation is defined as mapping function Φ.

Reward: Reward of action at time step t by DQN is proportional to the IoU gain of the detection region with ground truth box:

$$r_t = \beta(IoU(x_{t+1}, x) - IoU(x_t, x))$$

where $x = \max_{x \in \Omega(I)} IoU(x_t, x)$, I represents the visual image, $\Omega(I)$ represents the ground truth boxes in I, x_t, x_{t+1} are detection regions in time step $t, t+1$ respectively. β is the magnification factor, and is set as 20 here.

Action: Adjustment actions include "move left, right, up, down, taller, shorter, thinner, wider", with 10% or 50% scale change, and plus "stop" action. 17 action candidates in total.

C. DQN Archtecture and Training Procedure

Architecture: A DQN agent composes of 3 convolutional layers, one hidden fully connected layer, and one fully output layer. The convolutional layers have 32, 64, 64 feature maps, with stride 4, 2, 1, and filter size $8 \times 8, 4 \times 4, 3 \times 3$ respectively. The hidden layer has 512 hidden units, and output layer has 17 action output units. The whole DQN adopts rectify non-linear activation function.

Training: Inspired by deep Q-network learning [12], we utilize a technique known as experience replay [21] where we store the agent's experiences at each time-step, $e_t = (s_t, a_t, r_t, s_{t+1})$ in a data-set $D = e_1, e_2, ..., e_N$, pooled over many time steps into a replay memory. During the inner loop of the algorithm, we apply Q-learning updates, or mini-batch updates, to samples of experience, $e \sim D$, drawn at random from the pool of stored samples. After performing experience replay, the agent selects and executes an action according to guided ε-greedy policy (which will be introduces below) with ε annealed linearly from 1.0 to 0.15. SGD learning rate is set with different scale 0.0003, 0.0002, 0.0001. To achieve better stability, we use a separate network $\hat{Q}$ for generating the targets y_j in the Q-learning update, and clone Q as $\hat{Q}$ every constant steps. The full algorithm is presented in *Algorithm 1*.

Curriculum Learning: To accelerate reward back propagation in Q-learning, we devise curriculum learning [22] for DQN training. More specifically, the initial starting regions during training procedure are evolving from around ground truth boxes to random regions.

Define $\lambda \in [0, 1]$ is the difficulty of curriculum, the initial starting regions $box_\lambda = (x_s, y_s, w_s, h_s)$ equals:

$$\lambda \circ box_g + (1 - \lambda) \circ box_r$$

where box_g, box_r represent ground truth box and random box respectively. To be more specifically, box_λ equals:

$$(\lambda x_g + (1-\lambda)x_r, \lambda y_g + (1-\lambda)y_r, \lambda w_g + (1-\lambda)w_r, \lambda h_g + (1-\lambda)h_r)$$

To apply curriculum learning, set λ as 1 at the beginning, then gradually decreases to 0:

$$\lambda_t = max(0, 1 - \frac{\mu t}{t_{max}})$$

where μ is speed factor to controll curriculum progress, and is set as 1.5 here.

Guided ε-greedy policy: We devised guided ε-greedy policy to find best action occasionally and boost exploration efficiency. More specifically, instead of apply random ε-exploration all the time, we set fixed probability (like 0.1) to choose action that gets direct maximum reward. Otherwise, DQN might stops initiatively at regions with smaller IoU as training procedure goes on. Since neural network is forgetful and there are many action candidates.

D. CNN Classifier Training

Data preparation: we do local search around ground truth regions in detection training datasets, to generate classifier training sets. Regions with IoU>0.5 with any ground truth boxes are marked as target regions with class labels. The rest regions with IoU<0.5 are marked as background regions.

Training: We use the CaffeNet model [23] which is a variation on the well known AlexNet architecture [16]. This model contains 5 convolutional layers and 3 fully connected layers and we replaced the final fully connected layer with a 21-neuron layer since our problem is 21-class (VOC 2007/2010, plus background class) classification. We use the open source Caffe CNN library [24] to train the classifier by SGD optimization at different scale learning rate (0.01, 0.001, 0.0001).

E. Object Detection Procedure

One-class object detection: the procedure includes object localization with DQN agent and background filtering with CNN classifier, as *Algorithm 2* shows. In details, given arbitrary query image, the starting regions $\mathcal{R}$ are compose of $21(1 \times 1 + 2 \times 2 + 4 \times 4)$ different-scale regions. The regions of the same scale cover the whole image. Since Q value is a good indicator of IoU (larger IoU, smaller Q), we take x_t with least Q value as candidate, rather than the last one of each search procedure, as line 9, *Algorithm 2* shows. Usually, the last region of each procedure has smallest Q value, but not always. Then CNN classifier is used to judge the class label of the candidate regions, and output regions that have the same predict label with that DQN aims to localize.

Multi-class object detection: we train multiple DQN agents to localize different-class objects. Given a query image, just repeat Algorithm 2 for each class.

V. Experiments

A. Data Sets and Metrics

The PASCAL Visual Object Challenge data sets [15] are widely used as benchmarks for evaluating algorithms for image understanding tasks, and provide a common evaluation platform for object detection. These data are extremely challenging since the objects vary significantly in size, view angle, illumination, appearance and pose. We use PASCAL VOC 2007, 2010 data sets for experiments in this paper.

Both VOC 2007 and 2010 data sets contain 20 object classes with 9,963 and 21,738 images respectively. The two data sets are divided into "train", "val" and "test" subsets, i.e. 25% for training, 25% for validation and 50% for testing. The annotations for the whole data set of VOC2007 and "train", "val" set of VOC 2010 are provided while the annotations for the "test" set of VOC 2010 are still confidential and can only be evaluated on the web server with limited trials. We employed mean of average precision (mAP) as evaluation metric.

B. Experiment Setup

For DQN agent training, we set maximum iterations t_{max} as 3 million, and it takes nearly 6 hours in one NVIDIA TitanX GPU to train a DQN agent. The algorithms are implemented with Theano [25], with source code available[1].

Before training CNN classifier, we generate around 600 thousand subimages by local search around ground truth boxes in VOC training set, and 200 thousand subimages in validation set. Since the dataset is homogeneous, it only takes 300 thousand iterations to converge to 66% accuracy in validation set of VOC 2007. The batch size is set as 32.

C. Analysis of DQN Agent Behavior

Generally, object localization can be assessed in terms of accuracy and efficiency.

In aspect of accuracy, we presents IoU distribution of initial starting regions and ending regions in test set of VOC 2007, to check DQN agent's ability to search objects and to optimize region. As *Fig.2* shows, ...

As for efficiency, it takes about 0.5 billion floating operations to locate objects by one DQN agent in one image. More than 95% of computation is spent on repeated convolutional operations, which could be reduced to 1 million floating operations (computation load of single pass through convolutional layers) by application of spatial pyramid pooling [26].

To provide a more intuitive comprehension, we presents Q value distribution against IoU index. As *Fig.3* shows, ...

D. Comparison with State-of-the-Art Performance

We also compare the proposed method with the reported state-of-the-art object detection methods on VOC 2007, VOC 2010. The detaild performance comparison results are listed in *Table 1, 2*. The methods compared include [MIT] by Zhu et al. [27] using latent hierachical structural learning, [UCI] by Desai et al. [28] using context of object layout, [INRIA] by Harzallah et al. [29] fusing classification scores, and [UoC] [30] by Felzenswalb et al. using part-based model with context of object-co-occurrence, and [CtxSVM] [31] by Q. Chen et al. contextualizing object detection and classification.

VI. Conclusion

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (Grant Nos. 61329302 and 61175065) and the Program for New Century Excellent Talents in University (Grant No. NCET-12-0512).

[1] www.github.com/omgteam/dqn_detection

TABLE I
COMPARISON ON THE PASCAL VOC 2007 (TRAINVAL/TEST) DATA SET

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
MIT[27]	29.4	55.8	9.4	14.3	28.6	44.0	51.3	21.3	20.0	19.3	25.2	12.5	50.4	38.4	36.6	15.1	19.7	25.1	36.8	39.3	29.6
UCI[28]	28.8	56.2	3.2	14.2	29.4	38.7	48.7	12.4	16.0	17.7	24.0	11.7	45.0	39.4	35.5	15.2	16.1	20.1	34.2	35.4	27.1
INRIA[29]	35.1	45.6	10.9	12.0	23.2	42.1	50.9	19.0	18.0	31.5	17.2	17.6	49.6	43.1	21.0	18.9	27.3	24.7	29.9	39.7	28.9
UoC[30]	31.2	61.5	11.9	17.4	27.0	49.1	59.6	23.1	23.0	26.3	24.9	12.9	60.1	51.0	43.2	13.4	18.8	36.2	49.1	43.0	34.1
CtxSVM[31]	39.8	59.0	18.7	18.9	30.0	54.2	57.2	30.4	23.5	30.9	38.2	20.7	63.8	48.8	41.5	18.7	23.8	42.5	54.8	44.9	38.0
DQN																					

TABLE II
COMPARISON ON THE PASCAL VOC 2010(TRAINVAL/TEST) DATA SET

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
NLPR[15]	53.3	55.3	19.2	21.0	30.0	54.4	46.7	41.2	20.0	31.5	20.7	30.3	48.6	55.3	46.5	10.2	34.4	26.5	50.3	40.3	36.8
MIT[15]	54.2	48.5	15.7	19.2	29.2	55.5	43.5	41.7	16.9	28.5	26.7	30.9	48.3	55.0	41.7	9.7	35.8	30.8	47.2	40.8	36.0
NUS[15]	49.1	52.4	17.8	12.0	30.6	53.5	32.8	37.3	17.7	30.6	27.7	29.5	51.9	56.3	44.2	9.6	14.8	27.9	49.5	38.4	34.2
UVA[15]	56.7	39.8	16.8	12.2	13.8	44.9	36.9	47.7	12.1	26.9	26.5	37.2	42.1	51.9	25.7	12.1	37.8	33.0	41.5	41.7	32.9
CtxSVM[31]	54.6	53.7	16.2	12.5	31.2	54.0	44.2	40.0	16.7	32.2	29.1	30.1	54.3	57.2	43.9	12.5	35.4	28.8	51.1	40.7	36.8
DQN																					

REFERENCES

[1] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[2] V. Krüger, D. Kragic, A. Ude, and C. Geib, "The meaning of action: a review on action recognition and mapping," *Advanced Robotics*, vol. 21, no. 13, pp. 1473–1501, 2007.

[3] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys (CSUR)*, vol. 40, no. 2, p. 5, 2008.

[4] X. Zhang, Y.-H. Yang, Z. Han, H. Wang, and C. Gao, "Object class detection: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 10, 2013.

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 580–587.

[6] R. Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[7] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 91–99.

[8] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, pp. 237–285, 1996.

[9] E. J. Gibson, E. S. Spelke, P. Mussen, J. Flavell, and E. Markman, "The development of perception," *Handbook of Child Psychology, Vol 3*, 1983.

[10] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming (optimization and neural computation series, 3)," *Athena Scientific*, vol. 7, pp. 15–23, 1996.

[11] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[13] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *arXiv preprint arXiv:1504.00702*, 2015.

[14] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *arXiv preprint arXiv:1207.4708*, 2012.

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[17] B. Leibe, A. Leonardis, and B. Schiele, "Combined object categorization and segmentation with an implicit shape model," in *Workshop on statistical learning in computer vision, ECCV*, vol. 2, no. 5, 2004, p. 7.

[18] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders, "Segmentation as selective search for object recognition," in *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1879–1886.

[19] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014, pp. 2155–2162.

[20] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 391–405.

[21] L.-J. Lin, "Reinforcement learning for robots using neural networks," DTIC Document, Tech. Rep., 1993.

[22] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 41–48.

[23] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[24] Y. Jia, "Caffe: An open source convolutional architecture for fast feature embedding," 2013.

[25] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590*, 2012.

[26] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 346–361.

[27] L. Zhu, Y. Chen, A. Yuille, and W. Freeman, "Latent hierarchical structural learning for object detection," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1062–1069.

[28] C. Desai, D. Ramanan, and C. C. Fowlkes, "Discriminative models for multi-class object layout," *International journal of computer vision*, vol. 95, no. 1, pp. 1–12, 2011.

[29] H. Harzallah, F. Jurie, and C. Schmid, "Combining efficient object localization and image classification," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 237–244.

[30] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 32, no. 9, pp. 1627–1645, 2010.

[31] Q. Chen, Z. Song, J. Dong, Z. Huang, Y. Hua, and S. Yan, "Contextualizing object detection and classification," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 37, no. 1, pp. 13–27, 2015.