Image Classification by Reinforcement Learning with Two-State Q-Learning [3]

Original Paper by: Abdul Mueed Hafiz

Abstract

Reinforcement Learning (RL) is a powerful technique for sequential decision-making tasks. Combining RL with pre-trained Convolutional Neural Networks (CNNs) for image classification is an active vision approach that aims to improve accuracy. This report examines the work of Abdul Mueed Hafiz, which integrates a two-state Q-Learning algorithm with pre-trained ResNet50 and other classification network models for improved classification performance. The RL algorithm optimizes actions (rotations and translations) applied to images before classification. Hafiz's method outperforms baseline CNNs across multiple datasets [3]. This report replicates and critiques Hafiz's approach, providing implementation insights, results, and suggestions for future improvements. Despite mixed results in the replicated experiments, RL shows potential in addressing hard-to-classify image problems, especially with careful action-space design and dataset selection.

1. Introduction

Reinforcement Learning (RL) is a branch of machine learning useful for sequential decision-making problems. It aims to solve the problem of learning an agent's best decision-making policy by interacting with the environment and receiving feedback [15]. In computer vision, Convolutional Neural Networks (CNNs) are often used to classify or extract meaningful feature embeddings from images. To improve the accuracy of CNNs and other classification networks, prior work has explored the integration of RL with CNNs by introducing an active learning process [6]. Active vision, which is the acquisition of multiple visual observations where the model selects the next observation, is frequently used to improve image classification [5]. The work of Hafiz introduces an approach which combines pre-trained deep learning networks with a simple two-state Q-Learning algorithm to improve classification accuracy. The control actions (which are sought to be optimized) for this algorithm are rotations and translations, applied before classification, that help to clarify misclassified images [3]. Hafiz's 2022 paper presents a method where a pre-trained CNN, such as ResNet50 or InceptionV3, is used to extract feature maps from an image. Classification, using an also pre-trained classification structure, is performed on these maps. Before being fed into the classification structure, RL is used to improve the classification performance. The RL algorithm generates a more easily identifiable image by applying a learned best transformation. The simple RL algorithm takes place on a two-state space and three-action space and updates a Q-table many times to learn optimal actions to apply to each image in each state. After the transformation (or lack thereof) occurs, the updated feature maps are reclassified. For the RL algorithm, the agent's reward and value function are based on a standard deviation metric which measures how the classification confidence changes after the transformation [3].

The paper implements several variations (using different deep learning models) and considers the respective classification accuracy improvement. The results show that the proposed method outperforms the baseline CNN model and other state-of-the-art methods on several datasets, including Caltech-101, ImageNet, and Cats and Dogs Dataset [3]. The paper's results do suggest that the integration of simple RL with CNNs can improve image classification accuracy.

This report discusses the context, methodology, implementation, results, and strengths and weaknesses of Hafiz's technique. It includes some replicated implementations to verify the results of the original paper and some critiques and suggestions for improvement.

2. Contextual Literature Review

The development of RL was inspired by human trial-and-error learning. RL is specifically useful in sequential decision-making frameworks where dynamic programming is expensive or infeasible. Early and simple applications of RL showed its potential for solving problems that require adaptive strategy optimization or autonomous control. As RL was developed, Temporal Difference (TD) learning emerged as a combination of Dynamic Programming and Monte Carlo Methods. TD learning updates *estimates* of value functions at each state transition [14]. The Q-Learning algorithm, is a TD learning method that defines a Q-value table for each state-action pair. This algorithm allows an agent to learn the best action at a given state without knowledge of a model of the entire environment [18].

Because the Q-learning algorithm requires maintaining a Q-table, historically it was not applicable to large-space, high-dimensional, or continuous problems. In 2015, DeepMind introduced the Deep Q-Network (DQN) algorithm, which combined Q-learning with deep learning. DQN uses a deep Neural Network (NN) to approximate the Q-value function, allowing it to be applied to high-dimensional problems. DQN has been used in a variety of applications, including video games, robotics, and was thereafter extended to image problems [9].

Since its introduction to images, RL has been used in many computer vision models. Early applications included resolution selection for object detection [17] and feature-map learning of digits [13]. In this work and many other classification and RL models, feature maps and images were considered as the states of a Q-Learning algorithm. Many other similar models have been developed which use zoom and on images as actions for an improved classification accuracy [8] [11].

2.1. Novelty and Contributions of [3]

The paper's main contributions are its two-state algorithm and a new rotation action. The two-state algorithm offers low computational complexity relative to previous RL methods which have a state for each training sample. It should be noted that this implies the RL will be applied on each image, during the testing phase of implementation. This is different from past models, where RL was applied globally during a training phase. The algorithm also uses a novel action, rotation. The author notes that rotation for better comprehension of an image is a human ability, and thus should be incorporated in an image classification model, to simulate human visual understanding [3].

3. Background Theory

3.1. RL and Q-Learning

A Markovian environment is one in which the future state of a system depends only on the current state and action. This property is known as the *Markov property*. Formally, the state process $\{x_t\}_{t\geq 1} \in \mathcal{X}$ is a Markov chain if the following holds:

$$P(x_t \mid x_s, s < t) = P(x_t \mid x_{t-1}).$$

When a Markov chain is influenced by an agent's actions, it is referred to as a *controlled Markov chain*. For an process $\{u_t\}_{t\geq 1} \in \mathcal{U}$, a Markov process (or chain) has the following dynamics:

$$x_t = f(x_{t-1}, u_{t-1}, w_{t-1}),$$

where f is a deterministic or stochastic function, and w_t represents independent random noise in the system. A controlled Markov chain still has a Markov property, but the agent's actions influence the transition probabilities, that is:

$$P(x_t \mid x_s, u_s, s < t) = P(x_t \mid x_{t-1}, u_{t-1}).$$

A *policy* $\gamma = \{\gamma_t\}_{t\geq 1}$ is a sequence of decision rules that map the current state to an action: $\gamma_t(x_t) = u_t$. RL seeks to learn an *optimal policy* γ^* that maximizes the long-term expected reward.

This optimization occurs with respect to a pre-defined reward function $r(x_t, u_t)$. The goal is to maximize the cumulative discounted reward starting from an initial state x_0. This optimization is can be expressed as seeking:

$$\pi^* = \arg\max_{\pi\in\Gamma} J(\gamma, x_0),$$

where

$$J(\gamma, x_0) = \mathbb{E}^{\gamma}_{x_0}\left[\sum_{t=1}^{\infty} \beta^t r(x_t, u_t)\right],$$

and $\beta \in (0, 1]$ is the discount factor that prioritizes short-term rewards over distant ones. The Bellman optimality equation provides a recursive way to compute the maximum reward achievable from any state:

$$V^*(x_t) = \max_{u_t\in\mathcal{U}} \left[r(x_t, u_t) + \beta\mathbb{E}[V^*(x_{t+1}) \mid x_t, u_t]\right],$$

where $V^*(x)$ is the optimal value function, and x' is the next state determined by the system dynamics. In the context of Q-learning, the action-value function, $Q(x_t, u_t)$, plays a similar role to the optimal value function:

$$Q^*(x_t, u_t) = r(x_t, u_t) + \beta\mathbb{E}[\max_{u_{t+1}} Q^*(x_{t+1}, u_{t+1}) \mid x_t, u_t].$$

In a real, finite-state-action space, the optimal action-value function can be approximated by a Q-table. At each time step, this table is updated as follows, for $(x, u) \in \mathcal{X} \times \mathcal{U}$:

$$Q(x, u) \leftarrow Q(x, u) + \alpha\left[r(x, u) + \beta \max_{u'} Q(x', u') - Q(x, u)\right], \tag{1}$$

where α is the learning rate, and x' is the observed next state. The Q-learning algorithm iteratively updates the Q-table until convergence, at which point the optimal policy can be extracted from the table as a map for each state to an action as:

$$\gamma^*(x) = \arg\max_{u\in\mathcal{U}} Q(x, u).$$

Noting from [19] that stationary policies in a simple Q-learning algorithms have been shown to have optimal performance, the optimal policy will be represented as γ^*, without the time index.

4. Method

4.1. Model Architecture

The model implemented by Hafiz consists of several deep learning components as well as the RL algorithm. The three components of the model are the following [3].

- **Feature Extracting CNN:** The model uses a pre-trained CNN, such as ResNet50 [4] or InceptionV3 [16], to extract feature maps from the input images. This CNN is responsible for capturing the essential features of the images that are useful for classification.
- **Classifying Structure:** After extracting the feature maps, a secondary classifier,like a NN or a Support Vector Machine (SVM) [12], is used to classify the images based on the extracted features. This classifier is trained on the feature maps obtained from the CNN of the training image set.
- **Reinforcement Learning Algorithm:** The model incorporates a two-state Q-Learning algorithm that uses rotations and translations as control actions to improve classification. The algorithm learns the best transformation to apply to an image improve classification accuracy. For each test image, the Q-table is updated using the Bellman equation during exploration, and the action with the highest Q-value is chosen and applied to the test image.

The model architecture is visible in Figure 1

4.2. Q-Learning Algorithm

The Q-Learning algorithm implemented by Hafiz has the following steps. These steps occur using pre-trained deep learning models, and only on each instance of the test image set.

1. **Initialization:** The Q-table is initialized to zeros. This table will store the expected rewards for each state-action pair.
2. **Feature Extraction:** For a test image, a feature map, f_{sample} is obtained from the pre-trained CNN.
3. **Initial Classification:** f_{sample} is classified, and a metric M is calculated as the standard deviation of the classification predictions.
4. **Action Selection:** An action u_t is randomly selected from the action space, which includes rotations and translations. The randomness of this selection is what makes this the exploration phase of the algorithm.
5. **Transformation Application:** The selected action is applied to the test image, and a new feature map, f'_{sample}, is obtained from the CNN. A new metric M' is calculated.
6. **State Update:** The state is updated based on the comparison of M and M':
$$x_{t+1} = \begin{cases} \text{better}, & \text{if } M' > M \\ \text{worse}, & \text{otherwise.} \end{cases}$$
7. **Reward Calculation:** The reward $r(x_t, u_t)$ is calculated as:
$$r(x_t, u_t) = \begin{cases} 1, & \text{if } M' > M \\ 0, & \text{if } M' = M \\ -1, & \text{if } M' < M \end{cases}$$
8. **Q-Table Update:** The Q-table is updated using the Bellman equation (1).
9. **Action Selection:** After the Q-table is updated $m \times |\mathcal{U}|$ times, the action with the highest Q-value is chosen and applied to the test image. Note that the metric M' is always compared to the original M, not the metric from the previous step in exploration.
10. **Final Classification:** The extracted feature map of the transformed test image is classified using the secondary classifier.

This algorithm requires the choice of several parameters, β, α, and m. These parameters are chosen to balance the learning efficiency and stability of the Q-Learning algorithm. The learning rate α is set to 0.4, the discount factor β is set to 0.3, and the number of iterations m is set to 20. It is assumed that these values were tuned based on empirical testing of the Q-Learning algorithm [3].

5. Results

5.1. Original Paper Implementations

Hafiz conducted several experiments to evaluate the performance of the proposed method. For each dataset, the classifying structure was trained on training set feature maps. The Q-Learning algorithm is only applied to the images in the test set that are deemed *hard-to-classify*. For this implementation, *hard-to-classify* was chosen to be the class of images that were wrongfully classified by the **pre-trained CNN**. Additionally, the chosen set of actions for the Q-Learning was: angular rotation, by $90°$, angular rotation by $180°$, and downward and rightward diagonal translations 15 pixels in each direction [3].

The experiments were conducted on three datasets: Caltech-101 [7], ImageNet (four classes: Bikes, Ships, Tractors, Wagons) [2], and Cats and Dogs Dataset (two classes: cats and dogs) [10], with distributions outlined in Table 1.

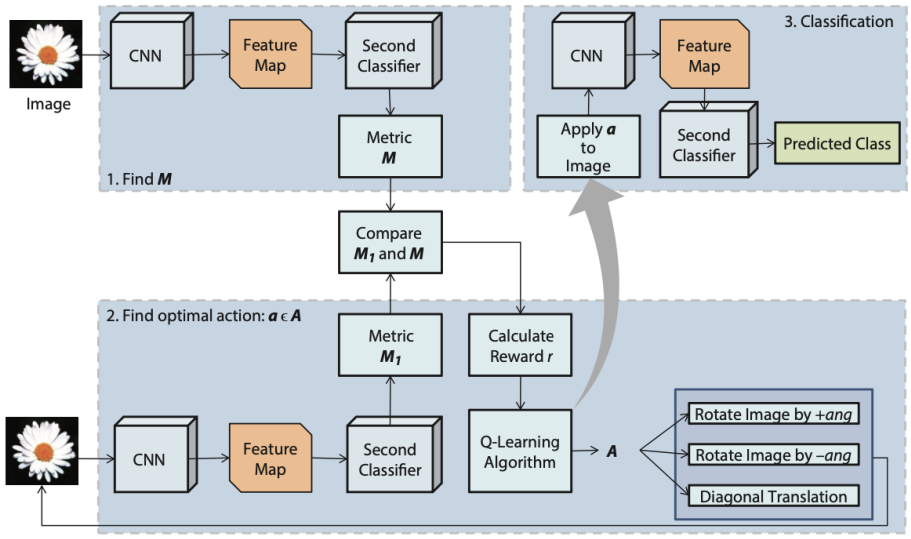

Figure 1. Model Architecture [3]

Table 1. Distribution of data for experiments [3]

Dataset	Classes used	Training images	Validation images	Testing images
ImageNet	4	1,531	788	745
Cats and Dogs	2	2,000	500	500
Caltech-101	50	750	-	1,250

Results of the experiments are summarized in Table 2, using an ImageNet data with an image size of $(150 \times 150 \times 3)$ and Table 3 using Cats and Dogs data with an image size of $(224 \times 224 \times 3)$ for ResNet50 and $(150 \times 150 \times 3)$ for InceptionV3. The proposed method outperformed the baseline CNN model [3].

Table 2. Classification accuracy of proposed model on ImageNet data [3]

Approach	Features extracted at layer	Accuracy (%)
ResNet50	#174: @(conv5_block3_out)	82.42
RL and ResNet50	#174: @(conv5_block3_out)	**83.09**
Inception V3	#228: @(mixed7)	85.64
RL and InceptionV3	#228: @(mixed7)	**86.44**

Table 3. Classification accuracy of proposed model on Cats and Dogs data [3]

Approach	Features extracted at layer	Accuracy (%)
ResNet50	#174: @(conv5_block3_out)	97.80
RL and ResNet50	#174: @(conv5_block3_out)	**98.60**
InceptionV3	#228: @(mixed7)	94.40
RL and InceptionV3	#228: @(mixed7)	**95.20**

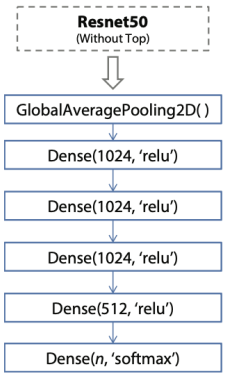

Figure 2. Custom top layers of ResNet50 [3]

5.2. Reproduced Implementations

The replication of these results was attempted on the same four classes from the ImageNet dataset. The dataset was split into training, validation, and testing sets, with 1532, 788, and 748 images, respectively. The images were resized to $(150 \times 150 \times 3)$ pixels. For transfer learning, the ResNet50 model (which will be called the CNN) with custom top layers, shown in Figure 2 was implemented with pre-trained ImageNet image weights. The top of the model was fine-tuned on the training set for 10 epochs.

The secondary classifier was implemented with the same architecture as the custom layers on top of the ResNet50 model in the CNN. The model was trained on the training set feature maps (after they are extracted from the pre-trained CNN) for 10 epochs. These feature maps have shape

Table 4. Classification Test Results Summary

Approach	Total Images	Correctly Classified	Accuracy (%)	Marked-Hard Images	Corrected by RL
NN	736	674	91.58	-	-
NN with RL	736	661	89.81	48	34
CNN	736	687	93.34	-	-
CNN with RL	736	722	98.10	48	34

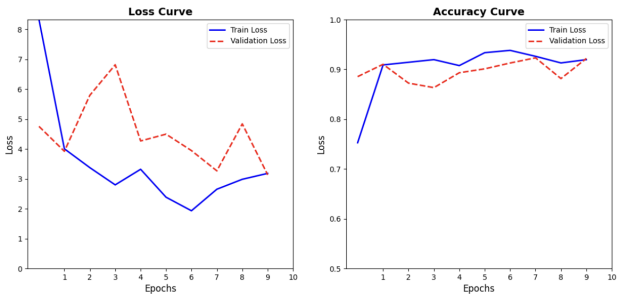

Figure 3. Loss and accuracy of custom ResNet50 CNN during training.

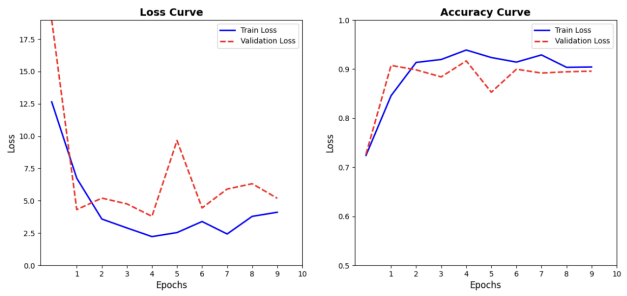

Figure 4. Loss and accuracy of secondary classifying NN during training.

$(7, 7, 2048)$ which is the default shape when they are extracted from layer *conv5_block3_out*.

Training curves of these deep models are visible in Figure 3 and Figure 4. Both the models reach above 85% accuracy within the first few training epochs. For the CNN, this is due to the use of the pre-trained weights, which allow early layers of the model to provide high-quality feature representations. In turn, these strong feature representations are passed to the secondary classifier, which is able to learn a good classification model quickly. The accuracy of both models on the set of test images is presented in Table 4.

For the implementation of the RL model, a few modifications were made to Hafiz's algorithm to attempt to demonstrate meaningful results. Instead of zeros, the Q-table was initialized to random values between -0.01 and 0.01 to encourage exploration, as initial experiments found the Q-table to converge to zero too quickly.

In addition, the Q-table updates were stopped when the update difference crossed a threshold of 0.0001 (achieved convergence). It was found that, across the test set, the maximum number of iterations that it took for the tables to converge was 45, which supports the near-optimality of the learned policy. Otherwise, parameters and design choices from Hafiz's work were maintained.

Two experiments were conducted using the RL algorithm. The first calculated the reward and state update using the standard deviations of the classification predictions from the secondary structure, as explained in [3]. The second experiment used the CNN prediction standard deviations to update the reward and state. The results from both of these are detailed in Table 4.

Code for all reproduced implementations can be found at `https://github.com/evelynhubbard/RLforComputerVision`.

5.3. Discussion of Results

While the overall accuracy results did not completely match those of Hafiz, some promising *proof-of-concept* results were obtained for the introduction of RL to this problem. The RL algorithm consistently converged within 45 iterations and suggested what appear to be helpful transformations. Examples of these transformations are visible in Figure 5. The first image is the original image, the second is the image after the transformation (best action) suggested by the RL algorithm. It is notable that, despite the Q-learning algorithm seemingly being effective, the results are not consistent with Hafiz's work or even displaying improvement when RL is implemented.

Hafiz saw a 0.67% improvement between the classification results of the CNN and the secondary classification using RL. In the reproduced results, a 3.53% decrease in accuracy was observed between these same reproduced structures. However, when using the RL algorithm with predictions from the CNN, the classification accuracy improved by 4.76%. While this result seems promising, it is misleading. Because the *hard-to-classify* images were the ones wrongfully classified by the CNN and were the only images that the RL algorithm was applied to, the classification accuracy could never decrease as a result of introducing RL. It is unclear whether the improvement comes from the RL algorithm learning policies and correcting the CNN's

Figure 5. Examples of transformation suggested by trained RL algorithm.

mistakes. It could be that, after RL takes place, the CNN just has a second chance to classify a random slightly transformed image, and no learning is taking place.

It remains to explain the achieved 3.53% decrease in accuracy compared to Hafiz's results. Some potential reasons for this discrepancy are as follows:

- The RL algorithm was not able to improve the classification accuracy. The secondary classifier is already very good at classifying the images, and the RL algorithm was not strong enough to improve it. The secondary classifier is trained on feature-maps from the CNN which is pre-trained on ImageNet. This transfer learning process is already quite robust to image transformations and it is possible that the RL algorithm was not able to make any impact on classification accuracy [4].

- The RL algorithm is correctly classifying some images, but transforming some already-correct images, decreasing accuracy. The *hard-to-classify* images found by the CNN do not overlap completely with incorrect classifications by the secondary network (on the same test set). In Table 4, it is clear that, at first, the classifier NN misclassifies 62 images. After RL is applied, the classifier misclassifies 75 images but, the RL algorithm claims it corrected 34 of the 48 *hard-to-classify* images. This means that the RL algorithm is not always applied to the images that need it and is sometimes applied to already correct images.

5.4. Suggested Improvements

Some suggestions for improving the results of this model are as follows. First, the original paper lacks the argument that the Q-learning algorithm will achieve policy-convergence for each image. Without this, calling the learned policy *optimal* is not justified. In this report, tests were conducted to show that the Q-tables converged within 45 iterations. This is a good sign that the learned policy is near-optimal, but it is not a guarantee. Showing convergence would also allow the iterations to stop early, saving computational resources.

Aditionally, while the original and reproduced results were not very meaningful, this algorithm still has good potential to improve image classification. A possible extension for this work would be to conduct experiments on a dataset of sets of multiple views of the same objects. Optimal actions, in this environment, could be selecting a most-informative view to pass to the classifying structure. If applied to this situation, this algorithm has potential to cause a more drastic improvement in classification accuracy.

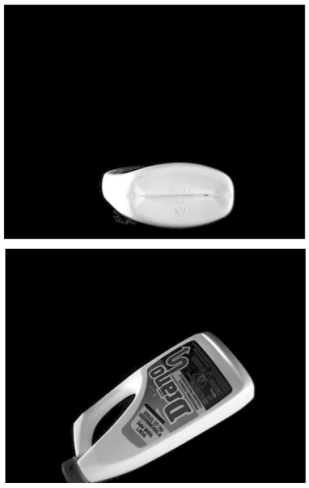

Figure 6. Two views of the same object, with one being much more informative than the other [1].

6. Conclusion

The integration of RL with deep learning for image classification, as proposed by Hafiz, demonstrates some potential for improving classification accuracy. The two-state Q-Learning algorithm is computationally efficient and introduces a novel control action, rotation, to improve image comprehension. While Hafiz's results show slight improvements in accuracy, the replicated experiments demonstrate challenges in reproducing these outcomes.

Further improvements to this approach could involve applying policy convergence tests and exploring multi-view datasets to exploit RL's decision-making capabilities more

effectively. By addressing these limitations, the integration of RL and CNNs could become a more robust solution for tackling hard-to-classify images in real-world applications.

References

[1] Tal Arbel and James J. Clark. Introduction - Fall 2024 - ECSE-626-001 - Statistical Computer Vision. 6

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. 3

[3] Abdul Mueed Hafiz. Image Classification by Reinforcement Learning With Two-State Q-Learning. In *Handbook of Intelligent Computing and Optimization for Sustainable Development*, chapter 9, pages 171–181. John Wiley & Sons, Ltd, 2022. 1, 2, 3, 4, 5

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. 3, 6

[5] Cathy Laporte. Active Vision - Fall 2024 - ECSE-626-001 - Statistical Computer Vision. 1

[6] Ngan Le, Vidhiwar Singh Rathour, Kashu Yamazaki, Khoa Luu, and Marios Savvides. Deep reinforcement learning in computer vision: A comprehensive survey. *Artif Intell Rev*, 55(4):2733–2819, Apr. 2022. 1

[7] Fei-Fei Li, Marco Andreeto, Marc'Aurelio Ranzato, and Pietro Perona. Caltech 101, Apr. 2022. 3

[8] Stefan Mathe, Aleksis Pirinen, and Cristian Sminchisescu. Reinforcement Learning for Visual Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2894–2902, Las Vegas, NV, USA, June 2016. IEEE. 2

[9] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015. 2

[10] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, June 2012. 3

[11] Aleksis Pirinen and Cristian Sminchisescu. Deep Reinforcement Learning of Region Proposal Networks for Object Detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6945–6954, June 2018. 2

[12] Derek A. Pisner and David M. Schnyer. Chapter 6 - Support vector machine. In Andrea Mechelli and Sandra Vieira, editors, *Machine Learning*, pages 101–121. Academic Press, Jan. 2020. 3

[13] Junfei Qiao, Gongming Wang, Wenjing Li, and Min Chen. An adaptive deep Q-learning strategy for handwritten digit recognition. *Neural Networks*, 107:61–71, Nov. 2018. 2

[14] Ashish Kumar Shakya, Gopinatha Pillai, and Sohom Chakrabarty. Reinforcement learning algorithms: A brief survey. *Expert Systems with Applications*, 231:120495, Nov. 2023. 1

[15] Rchard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. Bradford Book, 2 edition, 1992. 1

[16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, Las Vegas, NV, USA, June 2016. IEEE. 3

[17] Burak Uzkent, Christopher Yeh, and Stefano Ermon. Efficient Object Detection in Large Images using Deep Reinforcement Learning, Apr. 2020. 2

[18] C. J. C. H. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992. 1

[19] S. Yüksel and T. Başar. *Stochastic Teams, Games, and Control under Information Constraints*. Springer, Cham, 2024. 2