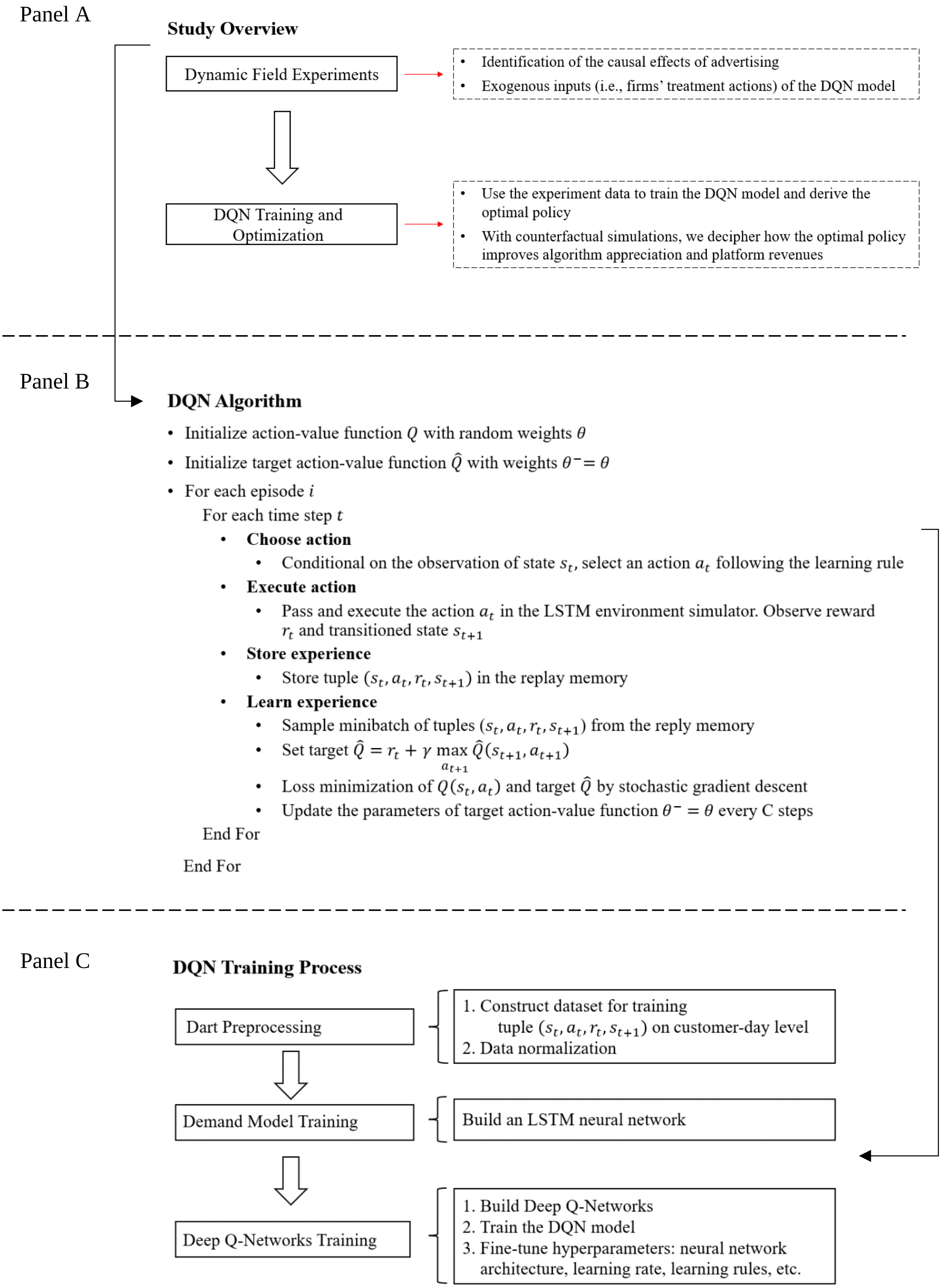
APPENDIX A. STUDY OVERVIEW AND DEEP Q-NETWORKS

Figure A1. Study Overview, DQN Algorithm, and Training Process



Panel A

Study Overview

Dynamic Field Experiments

- Identification of the causal effects of advertising
- Exogenous inputs (i.e., firms' treatment actions) of the DQN model

DQN Training and Optimization

- Use the experiment data to train the DQN model and derive the optimal policy
- With counterfactual simulations, we decipher how the optimal policy improves algorithm appreciation and platform revenues

Panel B

DQN Algorithm

```
• Initialize action-value function $Q$ with random weights $\theta$
• Initialize target action-value function $\hat{Q}$ with weights $\theta^{-} = \theta$
• For each episode $i$
    For each time step $t$
        • Choose action
            • Conditional on the observation of state $s_t$, select an action $a_t$ following the learning rule
        • Execute action
            • Pass and execute the action $a_t$ in the LSTM environment simulator. Observe reward $r_t$ and transitioned state $s_{t+1}$
        • Store experience
            • Store tuple $(s_t, a_t, r_t, s_{t+1})$ in the replay memory
        • Learn experience
            • Sample minibatch of tuples $(s_t, a_t, r_t, s_{t+1})$ from the reply memory
            • Set target $\hat{Q} = r_t + \gamma \max_{a_{t+1}} \hat{Q}(s_{t+1}, a_{t+1})$
            • Loss minimization of $Q(s_t, a_t)$ and target $\hat{Q}$ by stochastic gradient descent
            • Update the parameters of target action-value function $\theta^{-} = \theta$ every C steps
    End For
End For
```

Panel C

DQN Training Process

Dart Preprocessing

1. Construct dataset for training tuple (s_t, a_t, r_t, s_{t+1}) on customer-day level
2. Data normalization

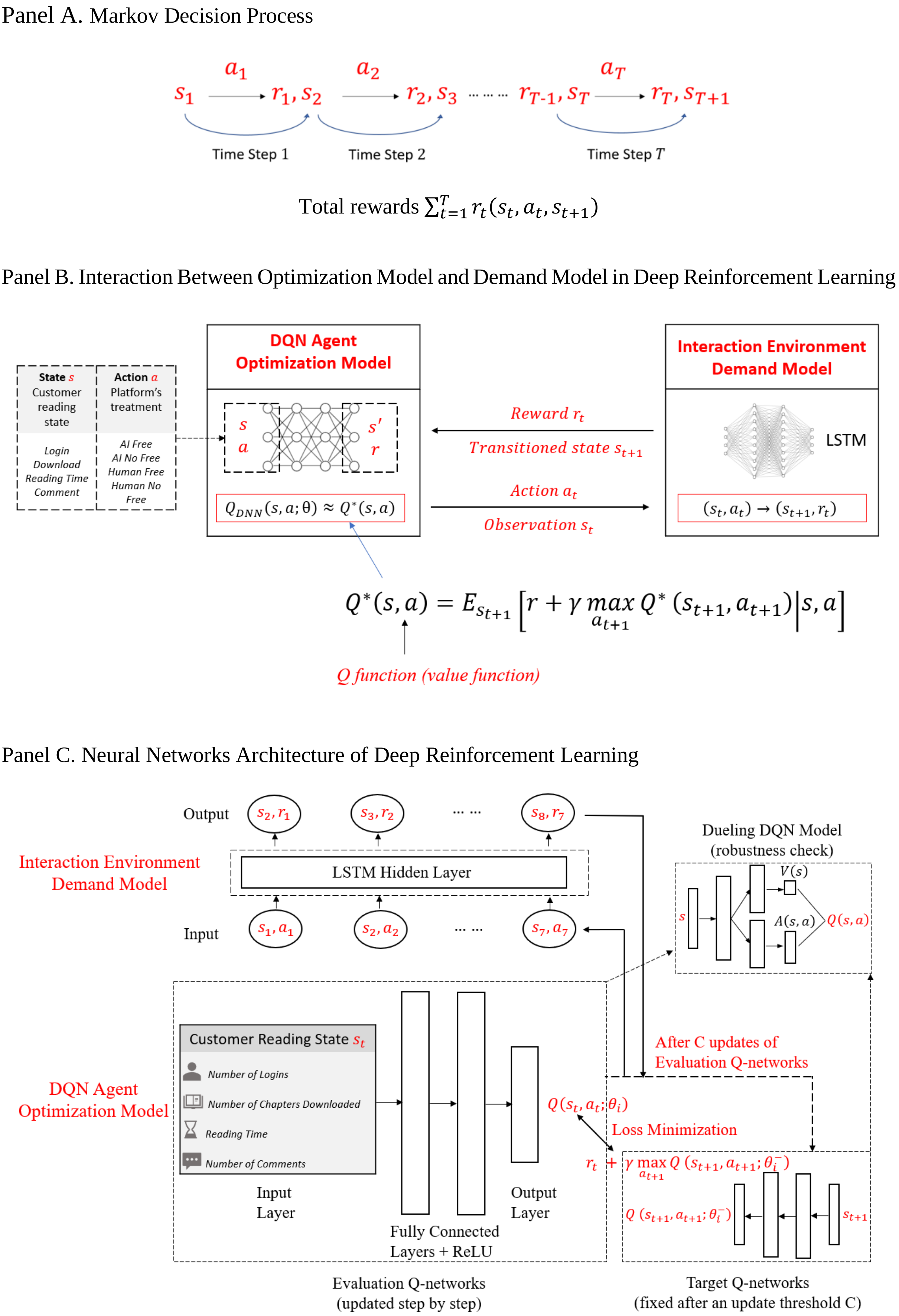
Demand Model Training

Build an LSTM neural network

Deep Q-Networks Training

1. Build Deep Q-Networks
2. Train the DQN model
3. Fine-tune hyperparameters: neural network architecture, learning rate, learning rules, etc.

Figure A2. Deep Reinforcement Learning Model Architecture



Panel A. Markov Decision Process

Panel B. Interaction Between Optimization Model and Demand Model in Deep Reinforcement Learning

Panel C. Neural Networks Architecture of Deep Reinforcement Learning

APPENDIX B. DETAILS OF DEEP Q-NETWORKS

B1. DQN's modifications of standard Q-learning algorithms

Standard Q-learning models may suffer from correlation and dependency (Tsitsiklis and Roy 1997). First, there are correlations among the state variables in s_t and s_{t+1}, and between action values $Q(s, a)$ and the target values $r + \gamma \max_{a'} Q\,(s', a')$. Such correlations make the neural network difficult to learn from more experiences, leading to unstable learning and overfitting. Second, the dependency arises from that the target value is a function of the same parameters θ_i as the action-value function being updated: $Q(s, a\,; \theta_i) = r + \gamma \max_{a'} Q\,(s', a'; \theta_i)$. Therefore, parameters learnt from the current data sample will largely determine the next sample the parameters will be trained on, leading to poor local minimum or parameter divergence (Sutton and Barto 2018).

The deep Q-network (DQN) model employs two modifications to address these issues (Mnih et al. 2015). The first one is *experience replay* (Lin 1992). This technique stores the agent's experience at time-step t, a tuple (s_t, a_t, r_t, s_{t+1}), into the replay memory M, which accumulates over many episodes of interactions. Then multiple Q-learning updates will be performed in a mini-batch at each time step based on random samples $(s, a, r, s') \sim U(M)$, drawn from the replay memory. In this way, a new uncorrelated experience will supply data for the next update, without s_{t+1} necessarily becoming the next s_t as it would in tabular Q-learning. Random samples also reduce the variance of the updates as they break off the correlations (Sutton and Barto 2018). Note that the algorithm is temporal-difference (TD) learning by applying experience replay, which exploits Markov property and learns after every step from incomplete sequences by bootstrapping without knowing the final outcome. In contrast, Monte Carlo (MC) based approach relies on the complete episodes of experiences. Combining MC and dynamic programming, TD is much more efficient, especially when the episode is long.

The second modification *dual network* is utilized to remove update dependency. Specifically, apart from the evaluation Q-network $Q(s, a\,; \theta_i)$, a target Q-network $r + \gamma \max_{a'} Q\,(s', a'; \theta_i^-)$ is generated with the same parameters but delayed weight θ_i^-, a lag of C updates, as the evaluation Q-network.

With these two extensions, we can employ stochastic gradient descent to minimize the loss function in Equation (1), which is computationally tractable and scalable. The gradient is calculated by differentiating the loss function with respect to the weight parameters θ_i.

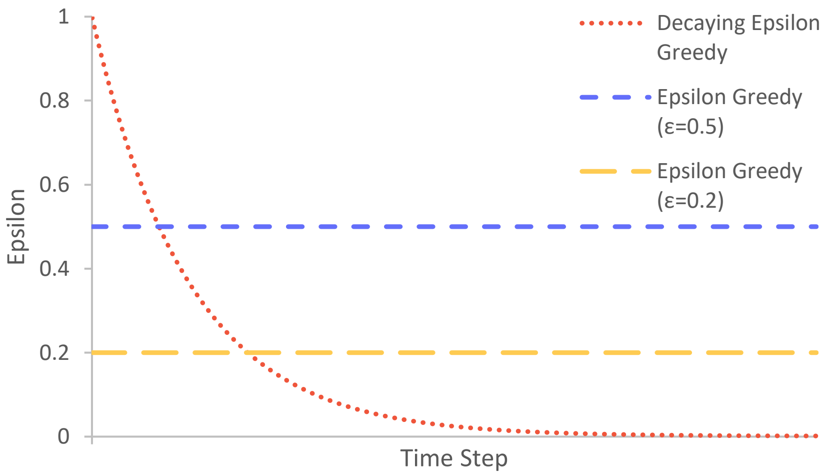
$$L_i(\theta_i) \ = E_{(s,a,r)}\left[\left(E_{s'}\left[r + \gamma \max_{a'} Q\,(s', a'; \theta_i^-)\middle| s, a\right] - Q(s, a; \theta_i)\right)^2\right] \qquad (1)$$

B2. Learning Rules

Following the Bellman Equation, we choose action $a' = arg \max_{a'} Q^* (s', a')$ to optimize $Q^*(s, a)$. However, this greedy rule engages in full *exploitation* and ignores *exploration*, leading to insufficient and inefficient learning.

The tradeoff between exploitation, where managers make the best decision given the current information, and exploration, where managers sacrifice the immediate reward and collect more information for better decisions in the long term, is fundamental in decision making domain (March 1991; Benner and Tushman 2003; Hills et al. 2015) and reinforcement learning literature (Singh et al. 2000; Auer 2002; Osband et al. 2016; Sutton and Barto 2018). Several learning polices are proposed to balance exploration and exploitation, such as epsilon (ε)-greedy, decaying ε-greedy. An ε-greedy rule continues to explore forever with probability ε (e.g. 0.2, 0.5, etc.) selecting a random action, whereas exploits with probability $1 - \varepsilon$ choosing the action $a' = arg \max_{a'} Q^* (s', a')$. In contrast, decaying ε-greedy employs a decreasing exploration parameter ε dependent on training time with a predetermined function, thus starting with exploration dominantly whereas engages in more exploitation as the agent gains more experience, as the example shown in Figure B1.

Figure B1. Exploration Parameter ε as a Function of Learning Rules



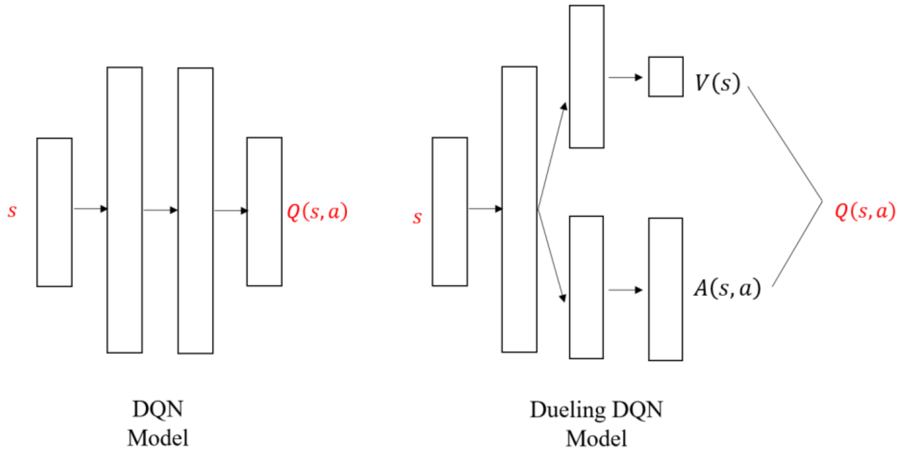
B3. Dueling DQN – A DQN Model with Improved Network Structure

The Q-network has attracted attention in computer science literature. There are revisions of the original Q-network model structure for better training and convergence, such as Dueling Q-network (Wang et al. 2015). We adapt our DQN model with Dueling Q-network, i.e. Dueling DQN. Dueling Q-network can differentiate the valuable states from the trivial states where the actions do not impact the reward in any way by modifying the Q-network structure to two streams: representations of state value function and state-dependent action advantage function, as shown in Equation (2) and Figure B2.

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \qquad (2)$$

where α and β are the parameters of the two streams of network layers.

Figure B2. Dueling Q-Network Structure



Notes: The left is a Q-network with a single stream. The right is Dueling Q-network with separate streams of state value and state-dependent action advantage. Both networks have outputs of each action’s Q-value.

APPENDIX C. ROBUSTNESS CHECKS

Table C summarizes the optimal average customer purchases by DQN model with different values of hyperparameters (the value derived from the model with default hyperparameters we employ is in bold and italic).

There is no rule of thumb in choosing the optimal value of hyperparameters, which depends on particular research context and data. For example, the number of hidden layers in neural networks may vary from one to thousands. A complicated neural network does not necessarily improve, perhaps may decrease, the model performance (Occam's razor principle in machine learning). Results reported in Table C consistently show that the DQN model lifts customer purchases compared to the averaged customer purchases of four treatment groups (5.4940).

Table C. Customer Purchases as a Function of DQN Models Employing Different Hyperparameters

Deep Learning Hyperparameter		*Reinforcement Learning Hyperparameter*		
Model Architecture	Learning Rate	Learning Rule		
		Decaying ε-greedy	ε-greedy (ε=0.5)	ε-greedy (ε =0.2)
Two hidden layers	0.05	***5.9429***	5.9067	6.0376
	0.01	6.0891	5.8483	6.1018
	0.001	6.1759	5.9180	5.9145
Four hidden layers	0.05	5.9489	5.8887	5.8881
	0.01	5.8771	6.5196	5.8970
	0.001	5.8625	5.8416	5.8753
Six hidden layers	0.05	6.0110	5.8421	6.0299
	0.01	6.6759	6.0414	5.8857
	0.001	6.0415	6.1770	5.8766
Dueling DQN	0.05	6.1439	5.9658	5.9564
	0.01	6.2398	5.9643	6.1019
	0.001	6.0692	5.9066	5.9089

TECHNICAL APPENDIX REFERENCES

Auer P (2002) Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov), 397-422.

Benner MJ, Tushman ML (2003) Exploitation, exploration, and process management: The productivity dilemma revisited. *Academy of Management Review*, 28(2), 238-256.

Hills TT, Todd PM, Lazer D, Redish AD, Couzin ID (2015) Cognitive Search Research Group. Exploration versus exploitation in space, mind, and society. *Trends in Cognitive Sciences*, 19(1), 46-54.

Lin, LJ (1992) Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4), 293-321.

March JG (1991) Exploration and exploitation in organizational learning. *Organization Science*, 2(1), 71-87.

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, ... Petersen S (2015) Human-level control through deep reinforcement learning. *Nature*, 518(7540), 529.

Osband I, Blundell C, Pritzel A, Van Roy B (2000) Deep exploration via bootstrapped DQN. *Advances in Neural Information Processing Systems* (pp. 4026-4034).

Singh S, Jaakkola T, Littman ML, Szepesvári C (2000) Convergence results for single-step on-policy reinforcement-learning algorithms. *Machine Learning*, 38(3), 287-308.

Sutton RS, Barto AG (2018) *Reinforcement learning: An introduction*. MIT press.

Tsitsiklis JN, Roy BV (1997) Analysis of temporal-difference learning with function approximation. *Advances in Neural Information Processing Systems* (pp. 1075-1081).

Wang Z, Schaul T, Hessel M, Van Hasselt H, Lanctot M, De Freitas N (2015) Dueling network architectures for deep reinforcement learning. *arXiv preprint* arXiv:1511.06581.