Eng Int Syst (2022) 4: 293–298

Engineering
Intelligent Systems

Deep Learning for Central Air Conditioning Controller

Dongsheng Xu*

Zhengzhou Preschool Education College, ZhengZhou, HeNan

In current commercial buildings, the most used is central air conditioning, and it is found through the survey that the energy consumption is relatively high after using central air conditioning, so this paper uses a deep learning algorithm to solve the energy consumption problem. The approach used in this paper is to develop a low-cost central air conditioning controller using neural networks, and experimental simulations show that this algorithm we propose has high energy efficiency and also reduces the cost of central air conditioning.Keywords: central air conditioning controller, algorithm, temperature.

1. INTRODUCTION

Heating, ventilation and air conditioning (HVAC) systems are widely deployed and quickly expanding, providing a comfortable indoor environment for commercial buildings. According to the report in [1], the energy costs of buildings, namely HVAC and lighting, account for over 40% of the total energy consumption in the United States, with , HVAC systems consuming the majority of total energy. Therefore, how to improve HVAC controllers to deliver energy savings is a research topic of great interest.

An intelligent and reliable controller plays an essential role in the automation of HVAC control. The existing HVAC functions of commercial buildings include physical condition sensing and intelligent control. Due the advances in sensors and other facilities, the environment sensing module of HVAC systems is sophisticated however, many researchers have made efforts to develop more efficient control strategies for HVAC systems. In [2], the authors design an occupancy-predictive model-based HVAC control algorithm and implement it in a low-cost embedded system, achieving significant energy savings.

A conventional HVAC control relies on rule-based control strategies. This involves complex human-crafted rules and tuning processes, which is unrealistic for real-world implementation. To address this problem, a deep reinforcement learning-based control scheme is proposed in [3]. The agent is able to intelligently learn an effective strategy for operating HVAC systems, which saves more energy costs compared to conventional rule-based methods. But the unknown thermal dynamics of an indoor building is challenging. The authors in [4] present a multi-agent deep reinforcement learning approach that incorporates the attention mechanism, which does not need any prior information. The work in [5] exploits video images to realize thermal comfort inferences and improve the quality of HVAC control.

Although yielding significant improvements, the existing works mostly require a large amount of computing resources to perform the control algorithms, which is not realistic for embedded devices and real implementation. In this paper, we address these problems by utilizing state-of-the-art deep learning algorithms. The contributions are as follows. First, we propose an energy-efficient controller for HVAC systems based on deep reinforcement learning algorithms and improvement schemes. To reduce the implementation cost of the pro- posed design, we utilize neural network pruning techniques to develop a low-complexity HVAC controller. Finally, we provide the experiment results to verify the effectiveness of the proposed algorithms with applications to HVAC control.

The rest of this paper is organized as follows. Section 2 introduces the proposed adaptive controller for HVAC based on deep reinforcement learning. The evaluation and

*Corresponding Author Email: 5695232@qq.com

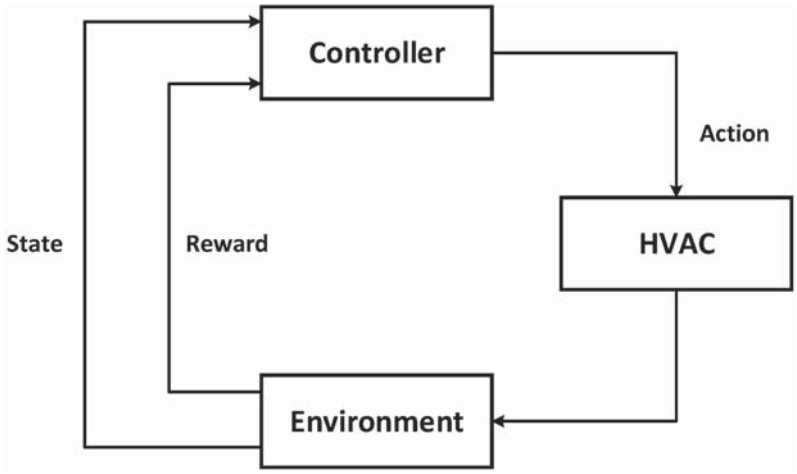

Figure 1 Basic control flow of a HVAC system in a commercial building.

experiments are presented in Section 3. Finally, Section 4 concludes the paper.

2. PROPOSED ADAPTIVE CONTROL SCHEME FOR BUILDING A HVAC SYSTEM

2.1 Overall Control and Design Flow

The HVAC control system inside a building is used to maintain the desired temperature conditions for each building area. The temperature is adjusted based on the current room conditions and the outside environment, such as indoor temperatures and outdoor weather, ensuring a comfortable indoor environment for the occupants. As shown in the analysis in Section 2.2, the HVAC control operation can be treated as a Markov decision process. We formulate the HVAC control process as shown in Figure 1, which illustrates the basic control flow of a HVAC system in a commercial building.

In the case of HVAC control, the controller is regarded as an agent that interacts with a commercial building environment. The controller sends a signal to the HVAC system to produce different levels of air flow. The environment arrives at various states after actions have been performed on the HVAC system. As the actions are taken and the HVAC system generates different effects on the environment, the agent may receive various rewards and state information.

To develop efficient and reliable HVAC controllers, we propose a novel adaptive controller by taking advantage of deep reinforcement learning, optimizations for neural networks, and building simulation techniques. The proposed HVAC control design is a good tradeoff between control accuracy and implementation overhead. Furthermore, due to the adoption of network optimization strategies, the proposed control algorithm can be realized with very low algorithm complexity in an automated manner.

To reach the goal of the precise control of building temperatures, control algorithms should be intelligent and have some degree of adaptiveness to automatically adjust the temperature without a large amount of human intervention. This imposes strict requirements on the core algorithms of control systems. Hence, we exploit state-of-the-art deep reinforcement learning algorithms to realize accurate temperature control and ensure the comfort of the occupants of commercial buildings.

When implementing the HVAC control algorithm, two major factors need to be taken into consideration, namely low algorithm complexity and low response latency, since deep neural networks are computation-dominant [9]. We utilize the existing optimization schemes to prune the neural network models of deep reinforcement learning, thereby reducing the inherent computation complexity as well as the implementation overhead.

Figure 2 illustrates our proposed intelligent HVAC controller for commercial buildings based on deep reinforcement learning algorithms. The diagram involves two phases: the training phase and the inference phase. For the training phase of the proposed algorithm, we use an advanced simulation tool to emulate the thermal dynamics of a specific building due to the limited amount of real data and the training cost. After the training phase has converged, the trained model is optimized via a network pruning algorithm to reduce the computation complexity. Finally, we implement the trained control algorithm and send a control signal to the HVAC system. The experiment results are discussed.

2.2 HVAC Control with Deep Reinforcement Learning

To simplify the controller algorithm design, we consider a HVAC system that produces conditioned air which typically flows at a constant temperature. In this case, the air flow rate can be switched between multiple levels. The HVAC control task for a commercial building can be regarded as the problem of searching for the optimal control sequence for the HVAC system. However, the complex building environment makes it difficult to accurately develop optimal control algorithms that incur a minimum energy cost while maintaining a high comfort level. To solve this problem, similar to [3], we leverage the deep Q-learning algorithms [6] and propose optimization tricks for the application of HVAC control.

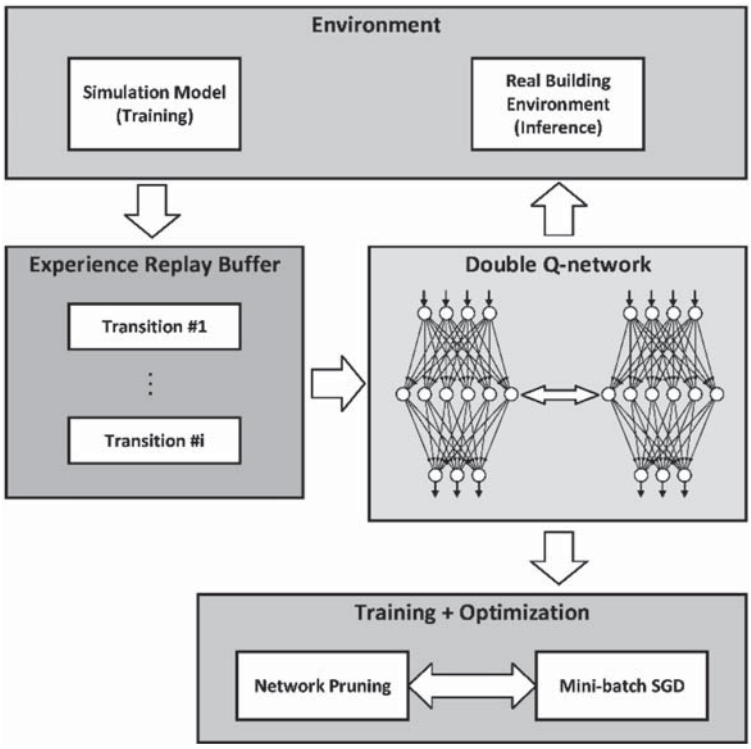

Figure 2 Proposed deep reinforcement learning-based HVAC controller.

Considering the real realizability of simulation, we discretize the control signal of a HVAC system using an infinite resolution of air flow rate. This is because the control algorithm is unable to accept and process sensor data with very high resolution, not only hindering the training convergence but also increasing the difficulty of developing a real-world system. Therefore, we assume that the discrete air flow rate is uniformly distributed between the minimum flow rate $f_{\min}$ and the maximum flow rate $f_{\max}$. As a result, the control signal (or action sequence) is expressed as $A=\{a_1, ..., a_t\}$ from time step 1 to time step t. It is clear that the dimension of the action space will grow exponentially as the time steps and air flow rate levels increase.

The temperature of the next time step after tuning at is determined by the current building states, indoor or outdoor environment conditions, and the control signals of the HVAC system. It is independent from the previous states of the building [3]. Hence, HVAC control algorithms can be regarded as a Markov decision process (MDP). The large search space for the action sequence is significantly reduced under the assumption of MDP.

The reward function guides the agent to take actions and make decisions. Therefore, the reward design plays an important role in deep reinforcement learning. It determines the performance and intelligence level of the trained control algorithms. Based on the state and reward feedback of the environment, the reward should be carefully designed to reflect the goal of the constructed model. As previously mentioned, the goal of the HVAC controller is to maintain the temperature of a building at a comfortable level and minimize the energy consumption over a period of time. We integrate these two targets into a single reward function. The reward function at time t is given as follows:

$$r_t = -\Big(energy(a_{t-1}, s_{t-1}) + \lambda \sum_{i=1}^{k} (|T_t^i - T_{min}| + |T_t^i - T_{max}|)\Big), \quad (1)$$

where the reward function consists of two parts, including the energy consumption term of HVAC and the metric that evaluates the comfort level of the indoor environment. The obtained reward from the environment is determined by the current state of the indoor environment as well as the energy consumption of the previous time step. The comfort level is evaluated based on the difference between the actual temperature and desired comfortable temperature. The reward is negative because the goal is to minimize the energy consumption and ensure minimum temperature fluctuations. The agent achieves the optimal reward function by balancing these two aspects.

Eq. (1) gives the reward for each time step. However, the proposed control algorithms will try to maximize the total reward over a given period of time. The reward for a single time step becomes the cumulative reward. The goal of the agent is to maximize the total reward between the first state and the final state as follows:

$$R_{total} = \sum_{t=1}^{n} \gamma^t r_t, \quad (2)$$

where R_{total} denotes the summation of the reward from the beginning to the end of the action sequence. $\gamma \in [0, 1]$ represents the discount factor which measures the contribution of rewards from short term and long term. For instance, $\gamma = 1$ means that the agent treats the rewards from different time

steps equally. A smaller γ reduces the importance of the reward obtained from previous time steps.

According to [7], the state from the environment and action space is huge thus it is not realistic to store all of them into the Q table of traditional reinforcement learning. To alleviate this problem, a deep neural network is used to estimate the Q-value function in the emerging deep Q-learning algorithms. On the basis of the Bellman equation, the Q-value update is given by:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)],$$

where α denotes the learning rate, $Q(s_t, a_t)$ represents the Q-value function corresponding to the current time, and $Q(s_{t+1}, a)$ represents the Q-value associated with the next time step after taking action a.

The weights of the neural network defined as w are randomly initialized. The weights of the neural networks are iteratively updated using the mini-batch stochastic gradient descent (SGD) method. More specifically, the training of neural networks weights w is done by updating the weights with the gradient of the loss function:

$$w_{k+1} = w_k - \alpha \cdot \mathbb{E}[R_{t+1} + \gamma \max_a Q(s_{t+1}, a; w_k^-) - Q(s_t, a_t; w_k)] \times \nabla_{w_k} Q(s_t, a_t; w_k),$$

where the target network weights at k-th iteration is defined as w_k^-, which are fixed during training and are only updated with the Q-network weights w_k every C training iterations. The used loss function is the mean squared error (MSE) between the output of the neural network and the target Q-value.

According to the analysis in [7], the weight updating in Eq. [4] for the target network and Q-network will lead to overestimating the values of actions. This is because the same network is used to calculate both the predicted value and the target value, resulting in divergence between the calculated results. The phenomenon of overestimation may hinder the control algorithms from choosing the optimal action in many states, thus reducing the overall performance of the algorithms. This problem can be resolved by using a different deep Q-network and target network [7]. In this case, the plain Q-value function is as follows:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[R_{t+1} + \gamma Q'(s_{t+1}, a) - Q(s_t, a_t)],$$

where functions $Q'(\cdot)$ and $a = \max_a Q(s_{t+1}, a)$ are used to estimate the expected Q-value using the selected action a that maximizes the value of the predicted Q-value network.

According to the analysis in [6], the agent training process may experience some unstable circumstances if we only run the deep Q-learning algorithms on the state-action pairs as they occur during simulation. To equip the agent with some capabilities of memory, we introduce the experience replay buffer to store the transition of discovered data and occurred experience. The experience replay is used to reduce the correlation of observed samples, increasing the robustness of the training process. As shown in Figure 2, the experience replay buffer stores the agent's experiences, including the actions taken, states, and rewards over certain time steps.

Exploration and exploitation are the two critical concepts in reinforcement learning. Exploration makes an agent improve the current knowledge about each action while exploitation selects the most rewarded action by exploiting current action-value estimations. To balance exploration and exploitation, ϵ-greedy policy is investigated to randomly choose between exploration and exploitation. It can be expressed by the following equation:

$$a_t = \begin{cases} \max_a Q(s, a), & \text{with probability } 1 - \epsilon, \\ \text{random action}, & \text{with probability } \epsilon, \end{cases}$$

where the agent tends to select the action that maximizes the current Q-value with probability $1 - \epsilon$. In contrast, the agent selects a random action with probability ϵ. In other words, the ϵ-greedy policy will allow the agent to exploit most of the time. The agent only has a small chance to explore new experiences.

2.3 Pruning of Neural Networks

The existing deep neural network models usually require a large amount of computing resources and memory, which becomes a bottleneck when running low-latency models on embedded devices. The training and inference of fully connected networks is computation-intensive and memory-intensive, requiring significant data movement and computation. Hence, embedded devices with limited computing power have difficulty handling real-time inference tasks. Energy efficiency is another major issue when implementing current deep neural network models.

These limitations not only increase the implementation costs but also hinder the deployment of the proposed models in resource-constrained environments. One of the methods to address these challenges is neural network pruning techniques [8,9], which are used to reduce the size of network models by removing redundant connections between fully connected layers. There are various types of network pruning algorithms to optimize neural network structures. The deep compression in [8,10] is one of the promising methods to obtain sparse neural network architectures. The basic idea of deep compression is progressively finding and removing the redundant weights that have relatively small magnitudes.

Figure 3 depicts the neuron connections of original and pruned fully connected layers. The original layer connection is dense and requires an excessive number of weights. Network pruning searches for the most representative and important connections for each layer. Then the trivial connections are removed by setting the weights to zero, thereby effectively cutting the weights and reducing model size as well as computational complexity. After applying the optimizations, the inference of neural networks can be accelerated by several times without loss of performance. The model size will shrink dramatically as a large number of weights are pruned.

However, the metric to prune the networks in deep compression is simple, thus it is difficult to guarantee the model's performance suffers minimal loss. In this work, we adopt another pruning technique ThiNet [11] which prunes the

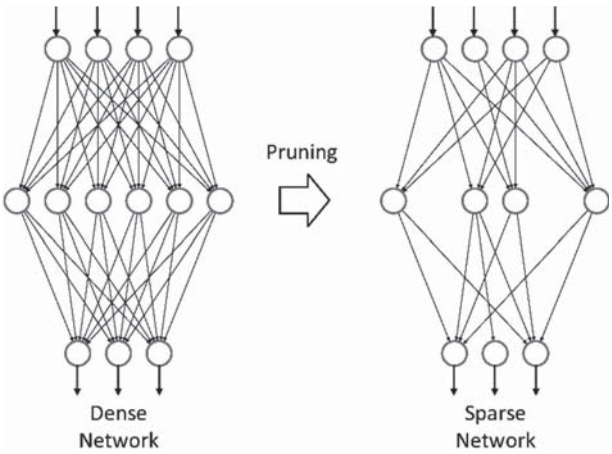

Figure 3 Illustration of original deep neural network (left) and its pruned variant (right).

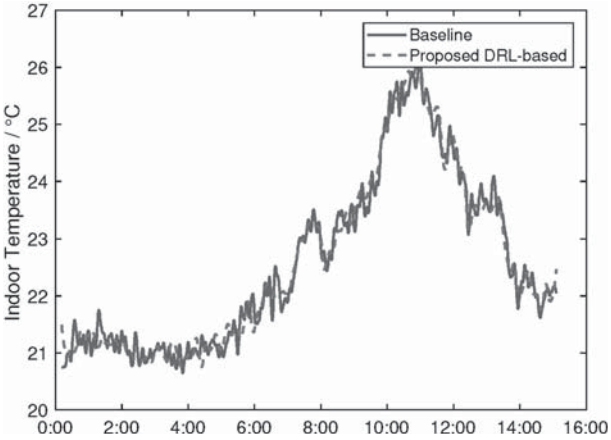

Figure 4 Results of indoor temperature under different HVAC control algorithms from 0: 00 to 16: 00.

model using an analytical metric. To minimize performance degradation and prune as many weights as possible, the authors in [11] use the following reconstruction error:

$$\hat{w} = \arg\min_{w} \sum_{i=1}^{m} (\hat{y}_i - w^T \hat{x}_i')^2,$$

where $\hat{x}_i'$ are the randomly selected samples that are used to evaluate the construction error. Eq. (7) can be understood as a classic linear regression problem. Thus, it has the close-form solution $\hat{w} = (X^T X)^{-1} X^T y$.

3. EXPERIMENTS

In this section, we discuss in detail the experiments to verify the performance of the proposed adaptive control scheme for the HVAC system. A comparison with its counterparts and an analysis are also presented.

3.1 Simulation Setup

We implement the proposed deep reinforcement learning algorithms on the advanced deep learning framework PyTorch. GPU is used to accelerate the training and inference process. Similar to the simulation flow in [3], the thermal dynamics of indoor environments are emulated using the simulation tool EnergyPlus. The comfortable temperature range is set as $T_{\min} = 20$ and $T_{\max} = 26$. The target network and deep Q-network both adopt a five-layer basic structure, where there are 64,128,256,192,96 neurons for each hidden layer.

The input of the trained controller is the concatenation of the outdoor temperature vector, the indoor temperature vector and the current state of the HVAC system. The hyperparameters of training and network structures are given as follows. To achieve a smooth convergence process, the mini-batch size is selected as 16 to make the weight updating stable. As suggested by [6], the ϵ-greedy policy will choose a random action with a probability of 0.1. We assign a small experience replay memory size as 50. The weights of the target network are copied every 6 training iterations. The discount factor of the cumulative reward calculation is set to 0.97. The Adam optimizer [12] is used to yield faster training.

3.2 Experiment Results

Figure 4 illustrates the indoor temperature curve using different HVAC control algorithms. The baseline is the reinforcement learning-based HVAC controller in [3]. A commercial building in Nanjing, China is selected as the indoor environment. The training is carried out on the simulation tool and we perform tests on the real building. It

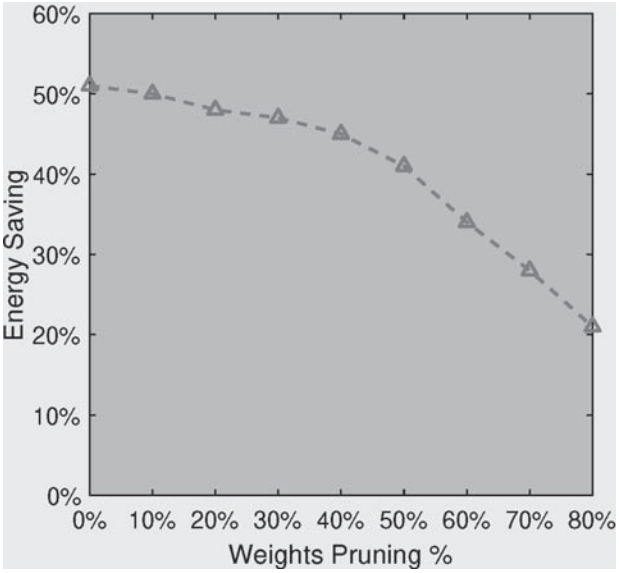

Figure 5 Relationship between the energy saving rate and weight pruning percentage.

can be seen from the figure that the indoor temperature under the control of the proposed algorithms is smoother and there is less temperature fluctuation in comparison with the baseline control algorithms. This demonstrates that our proposed algorithms are more effective in the application of HVAC control.

We use the adopted pruning techniques in [11]. The relationship between the energy saving rate and weight pruning percentage is shown in Figure 5. We do not used the average training reward to evaluate the performance of network pruning since it is difficult for the average reward to reflect the final effects of network pruning. Hence, we use the more realistic metric, energy saving which is defined by the percentage of energy saving using the proposed pruned control algorithms over the baseline algorithms shown in Figure 4. From the figure, we can see that the unpruned network model delivers over 50% energy savings compared to the baseline model. After setting a different pruning threshold, the degraded network performance causes a decline in the energy saving rate. But the energy saving rate declines slowly. The figure provides a guideline for us to decide which network structures to use in different scenarios.

4. CONCLUSION

In this paper, we propose an efficient and effective controller for HVAC systems based on deep reinforcement learning algorithms. We transform the HVAC control problem into a reinforcement learning format. Then, we utilize the advanced optimization methods for deep neural networks to develop a low-cost HVAC controller. The experiment results verify the effectiveness of the proposed algorithms with an application for HVAC control.

REFERENCES

1. L. Pérez-Lombard, J. Ortiz, and C. Pout, "A review on buildings energy consumption information," *Energy and buildings*, vol. 40, no. 3, pp. 394–398, 2008.
2. M. Aftab, C. Chen, C.-K. Chau, and T. Rahwan, "Automatic HVAC control with realtime occupancy recognition and simulation-guided model predictive control in low-cost embedded system," *Energy and Buildings*, vol. 154, pp. 141–156, 2017.
3. T. Wei, Y. Wang, and Q. Zhu, "Deep reinforcement learning for building HVAC control," in *Proceedings of the 54th Annual Design Automation Conference*, 2017, pp. 1–6.
4. L. Yu, Y. Sun, Z. Xu, C. Shen, D. Yue, T. Jiang, and X. Guan, "Multi-agent deep reinforcement learning for HVAC control in commercial buildings," *IEEE Transactions on Smart Grid*, 2020.
5. F. Jazizadeh and W. Jung, "Personalized thermal comfort inference using RGB video images for distributed HVAC control," *Applied Energy*, vol. 220, pp. 829–841, 2018.
6. V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves,
7. 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.
8. H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *AAAI Conference on Artificial Intelligence*, 2016.
9. S. Han, H. Mao, and W.J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," arXiv preprint arXiv:1510.00149, 2015.
10. D. Blalock, J.J.G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" arXiv preprint arXiv:2003.03033, 2020.
11. S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Advances in neural information processing systems*, 2015, pp. 1135–1143.
12. J.-H. Luo, H. Zhang, H.-Y. Zhou, C.-W. Xie, J. Wu, and W. Lin, "ThiNet: pruning CNN filters for a thinner net," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 10, pp. 2525–2538, 2018.
13. D.P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.