AIF: An Artificial Intelligence Framework for Smart Wireless Network Management

Gang Cao, Zhaoming Lu, Xiangming Wen, Tao Lei, and Zhiqun Hu

***Abstract*— To solve the policy optimizing problem in many scenarios of smart wireless network management using a single universal algorithm, this letter proposes a universal learning framework, which is called AI framework based on deep reinforcement learning (DRL). This framework can also solve the problem that the state is painful to design in traditional RL. This AI framework adopts convolutional neural network and recurrent neural network to model the potential spatial features (i.e., location information) and sequential features from the raw wireless signal automatically. These features can be taken as the state definition of DRL. Meanwhile, this framework is suitable for many scenarios, such as resource management and access control due to DRL. The mean value of throughput, the standard deviation of throughput, and handover counts are used to evaluate its performance on the mobility management problem in the wireless local area network on a practical testbed. The results show that the framework gets significant improvements and learns intuitive features automatically.**

***Index Terms*— Smart wireless network, deep learning, reinforcement learning, resource management.**

I. Introduction

THE Mobile and wireless communications Enablers for Twenty-twenty (2020) Information Society (METIS) project predicts that by 2020, worldwide mobile traffic alone will increase by 33 times that of the 2010 figures [1]. In this situation, the Internet of Things (IoTs) will become dominated by massive wireless devices such as smartphones and sensors. In order to handle this massive data traffic with high mobility, such devices will require more efficient and ubiquitous radio access technologies (RATs) [1], [2]. This implies that the future wireless communication systems will have to evolve the technology of the radio access network (RAN) systems to satisfy at least a 1000-fold traffic volumes, 100 billion connected devices, the large diversity of use cases and specific performance requirements (e.g., reliability, latency and data rates) by 2020 and beyond [1].

When huge numbers of RANs and devices are deployed, the optimal policy of smart wireless network management will be a main challenge. Recently, Ahsan Kazmi *et al.* [3] proposed two different algorithms to solve the underlying mixed-integer resource allocation problem of an underlay small cell network. Lien *et al.* [2] revealed a software-defined design of the cognitive radio resource management. However, all these works require many application specific designs for a particular scenario. Thus, current research area still lacks a general algorithm to solve policy optimizing problem in many scenarios.

A few years ago, Reinforcement Learning (RL) approach was very popular in solving these problems. Yau *et al.* [4] presented a review of RL achievements in wireless network. But it also pointed out that RL is unstable or even divergent when action-value Q function is approximated with a nonlinear function like neural networks.

Nowadays, another data-driven technique called deep learning has made a breakthrough in auto feature extraction area. Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN) are core techniques in deep learning [5]. Silver *et al.* [6] combined the deep learning and monte carlo tree search to beat humans in the game of go which has been seen as a milestone of artificial intelligence area. Mnih *et al.* [7] introduced Deep Q-Network (DQN) and ignited the field of Deep Reinforcement Learning (DRL).

Recently, Sun *et al.* [8] had done very good work on combining deep learning and wireless resource management. The work uses Deep Neural Network (DNN) to approximate the existing algorithm of wireless resource management. It reduces computation complexity while achieving the same performance as the existing algorithm. However, this work does not combine the deep learning, reinforcement learning and wireless resource management which can get a better performance based on the future trends of communication big data.

In this letter, we propose an AI framework to solve the policy optimizing problem and present how to train and infer using this framework. Firstly, we get intuitive features which are learned automatically by CNN and RNN [5]. The features are taken as the new state definitions of RL to generate good policy. Thus there is no need to manually define the states carefully. So it solves the difficulties of designing the states of RL. Secondly, we apply the new DRL technique in wireless network which solves the unstable issues presented in [4]. DRL extracts intuitive features by using deep learning (e.g. CNN, RNN, DNN), and also gets better performance. Using DRL, the framework works in many scenarios of smart wireless network management.

II. AI Framework and Algorithm

In this section, an explicit explanation of the AI framework and related algorithms are presented.

A. An AI Framework

Fig. 1 presents the AI framework, which mainly involves brain model and real environments. The brain model is able to read and observe raw wireless signal information, reward

Manuscript received September 5, 2017; revised October 11, 2017 and November 7, 2017; accepted November 19, 2017. Date of publication November 23, 2017; date of current version February 9, 2018. This work was supported by Beijing Municipal Commission of Education(Grant No. 201501001) and Nation Natural Science Foundation of China (Grant No. 61671073). The associate editor coordinating the review of this letter and approving it for publication was H. Otrok. *(Corresponding author: Gang Cao.)*

The authors are with the Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: caogang440@bupt.edu.cn; lzy0372@bupt.edu.cn; xiangmw@bupt.edu.cn; leitao@bupt.edu.cn; huzhiqun@bupt.edu.cn).

Digital Object Identifier 10.1109/LCOMM.2017.2776917

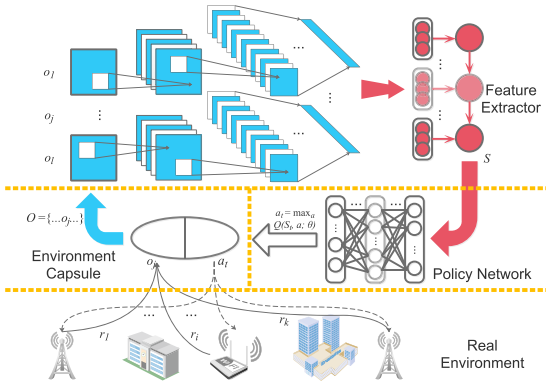

Fig. 1. The AI Framework.

information from the real environments, and it can also take actions to control the real environments.

The brain model can be decomposed into three parts, named environment capsule, wireless signal feature extractor and policy network. The environment capsule is the only part of brain model which communicates with real environment directly, and it mainly has two functionalities. One is stacking the raw information observed from the real environment as a sequential information batch and feeding this batch to the wireless signal feature extractor; the other functionality is constructing the protocol message which is sent to the real environment in terms of the action taken by the policy network.

The wireless signal feature extractor is the core part in the brain model, which can learn to shape useful features automatically by observing the real-time environment states. The features shaped can be categorized into two types, spatial features and sequential features. These two types of features are extracted by CNN and RNN, respectively.

By taking advantages of these two neural networks, the wireless signal feature extractor can forward the sequential information batch through the two networks, and output a hidden state vector of RNN as the feature vector. The feature vector will be sent to policy network. Besides the forward procedure, the extractor takes another backward procedure, back propagating a residual error from the feature vector to all parameters of CNN and RNN and updating the parameters using optimization algorithm such as Stochastic Gradient Descent (SGD) which is presented in [5]. The residual error is a vector passed back by the policy network.

Policy network takes a general full-connected DNN as the core architecture. It takes the feature vector fed by wireless signal feature extractor as input and forwards the vector through the network and gets a probability distribution vector over the action set which is defined in the next part. The final action determined by this part is the action which has the maximum probability in the distribution vector. This final action is pushed to the environment capsule. A similar backward procedure still exists in this part. The residual error is calculated using a deep Q target loss function

B. Algorithm

The details of the algorithm are presented as follows.

1) Observation Definition:

$$O = \underbrace{\{o_1, o_2, \cdots, o_j, \cdots, o_{l-1}, o_l\}}_{l} \tag{1}$$

l is the sequence length which can also be seen as how long it takes to stack the data for training.

$$o_j = \underbrace{\{r_1, r_2, \cdots, r_i, \cdots, r_{k-1}, r_k\}}_{k} \tag{2}$$

o_j is a collection of observed values from access equipments at time step j, and k is the number of access equipments.

r_i is the raw observed value, which can be very simple data in different scenario. Received Signal Strength Indicator (RSSI) can be adopted as r_i in some scenarios of WLAN.

2) State Definition:

$$S = f_{CNN+RNN}(O; \theta_{CNN+RNN}) \tag{3}$$

$f_{CNN+RNN}$ is a nonlinear function represented by CNN and RNN. $\theta_{CNN+RNN}$ is the multidimensional parameter matrix of the non-linear function $f_{CNN+RNN}$.

3) Reward Definition: In order to reflect the user experience which you want to optimize, the algorithm requires designing specific reward R for every scenario of wireless network.

4) Action Set Definition:

$$\mathcal{A} = \underbrace{\{a_1, a_2, \cdots, a_i, \cdots, a_{n-1}, a_n\}}_{n} \tag{4}$$

n is the dimension of action set $\mathcal{A}$, and a_i is the action candidate which may be picked by policy network. The a_i and n should be designed specifically to fit in the target scenario.

5) Q Function Approximation: DQN algorithm uses a DNN to approximate the action value Q function with $Q(S, a_i; \theta)$, which is the probability of taking an action a at observation O using current parameter θ.

$$Q(S, a_i; \theta) = f_{DNN}(S, a_i; \theta) \quad a_i \in \mathcal{A} \tag{5}$$

f_{DNN} is a nonlinear function represented by DNN which is parameterized by θ. A target Q function $\hat{Q}(S, a_i; \hat{\theta})$ is defined almost the same as $Q(S, a_i; \theta)$. It only uses parameter $\hat{\theta}$ instead.

6) Action Definition: Final action a_t at time step t will be picked by policy network from action set $\mathcal{A}$.

$$a_t = \arg\max_{a_i \in \mathcal{A}} Q(S_t, a_i; \theta) \tag{6}$$

7) Loss Function Definition: The deep Q target loss function $L(\theta)$ borrows the same ideas in DQN [7].

The experience replay memory D is defined bellow,

$$D = \underbrace{\{\cdots, (S_t, a_t, R_t, S_{t+1}), \cdots\}}_{m} \tag{7}$$

m is the maximum memory size. The subscript t is the time step. A mini-batch data d is a collection of elements sampled from D. The batch size is l_b.

$$L(\theta) = E_d[\underbrace{(R_t + \gamma \max_{\hat{a}_i} \hat{Q}(O_{t+1}, \hat{a}_i; \hat{\theta})}_{q\ target} - Q(O_{t+1}, a_t; \theta))^2] \tag{8}$$

γ is the discount factor for future rewards. For every training step, we will update θ by calculating $\frac{\partial L(\theta)}{\partial \theta}$ using SGD. After training every C steps, we will update the target network $\hat{Q} = Q, \hat{\theta} = \theta$.

Algorithm 1 Algorithm of finding the optimal policy

Description: Algorithm in Training Stage

Initialize all parameters like k, l, T, n, m, l_b
Initialize replay memory D to capacity m
Initialize $\theta_{CNN+RNN}, \theta, \hat{\theta}$ with random weights
Read the observation to start O_1 from environment capsule
for $t = 1...T$ **do**
  Calculate S_t according to equation (3)
  With probability ϵ select a random action $a_t \in \mathcal{A}$
  otherwise select $a_t = \max_{a_i \in \mathcal{A}} \hat{Q}(S_t, a_i; \hat{\theta})$
  Execute action a_t to environment capsule
  Observe reward R_t and next observation O_{t+1}.
  Store transition (S_t, a_t, R_t, S_{t+1}) in D
  Sample random mini-batch d from D
  Update θ using $\frac{\partial L(\theta)}{\partial \theta}$
  Update $\theta_{CNN+RNN}$ using $\frac{\partial L(\theta)}{\partial \theta_{CNN+RNN}}$
  Perform $\hat{Q} = Q$ every C steps by $\hat{\theta} = \theta$

Description: Algorithm in Inference Stage

Read pre-trained network model θ
Read the observation to start O_1 from environment capsule
for $t = 1...$ **do**
  Calculate S_t according to equation (3)
  Select $a_t = \max_{a_i \in \mathcal{A}} \hat{Q}(S_t, a_i; \theta)$
  Execute action a_t to environment capsule
  Observe reward R_t and next observation O_{t+1}

8) Algorithm: Algorithm 1 can be divided into two stages, training stage and inference stage. It is proved in [9] that this algorithm will converge to the optimal policy after enough iterations. The speed of convergency is related to the reward function which is specifically designed to fit the scenario. For example, if we design the reward to be a very sparse value in a scenario, the speed will be very slow.

III. Experiments and Results

A. Use Case: Mobility Management

1) System Scenario: There are k Access Points (APs) serving for one user in WLAN. Those APs work in the same channel, which can achieve seamless handover with low latency [10]. The user moves around those APs, and only associates with one AP at a time. APs can monitor the RSSI of uplink user traffic data. The optimizing target is to find an optimal access policy for the user while maximizing the user's throughput.

The RSSI always fluctuates due to the time-varying wireless channel. So a simple model can not solve the problem especially when the RSSIs from different AP are close. It will cause ping-pong effect which makes user loss throughput, while a complex model may face the high computation complexity [4], which is not suitable for real-time policy. Thus this problem is non-trivial. Using proposed AI framework, we can get better performance by alleviating the ping-pong effect.

2) Observation Definition: r_i in equation (2) is the RSSI read from the i_{th} access point of the user.

3) Reward Definition: Reward R is the estimated throughput of user calculated by minstrel [11] algorithm.

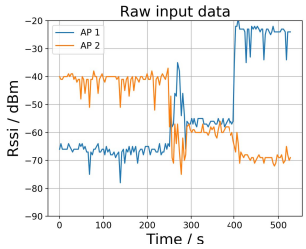

Fig. 2. The observed RSSI values of three stages in Experiment 1.

4) Action Definition: An action a is a specific access point to which the user should be allocated. The action set $\mathcal{A}$ is a collection of all access points. So the dimension n in equation (4) is equal to k in equation (2).

$$\mathcal{A} = \underbrace{\{a_1, a_2, \cdots, a_i, \cdots, a_{k-1}, a_k\}}_{k} \tag{9}$$

a_i is a single access equipment. a is defined in equation (6).

B. Experiments Testbed

This letter presents an experiment using the real-time heterogeneous wireless network testbed demonstrated by [10]. This testbed is an SDN-based heterogeneous wireless network testbed, which has the ability to manage the whole heterogeneous wireless network using CAPWAN protocol proposed in [10]. This testbed can read RSSI values and estimated throughput of user from APs.

The distance between two APs is 10m for the following experiments.

1) Experiment 1: We setup two APs, a mobile terminal, a controller to constitute the testbed which acts as the real environment in our AI framework. In the first experiment, we compare the performance using the following stages. 1) 0s-250s: The mobile terminal stays around AP2 for a while. 2) 250s-400s: It walks to the middle of the two APs, and stands there for a while. 3) 400s-550s: It continues to walk to AP1, and stays around AP1 for a while.

2) Experiment 2: We compare the performance of different framework setups and multiple APs in Table I. We setup 2 APs in one line, 3 APs in a triangle and 4 APs in one line. We collect the throughput data following the stages defined in Experiment 1. Then we use the mean value of throughput, the standard deviation of throughput and handover counts to evaluate the performance.

Three framework setups are defined as follows:

1)*wo AIF:* Setup without AIF and handover policy.

2)*w comp:* Setup with handover policy presented in [10] by comparing the RSSI values from current AP and other APs.

3)*w AIF:* Setup with AIF.

C. Results Analysis

1) Experiment 1: Four results are presented from Fig. 2 to Fig. 5. Fig. 2 shows the RSSI and throughput of a user. In Fig. 3, we compare the throughput in three scenarios, *wo AIF*, *w comp* and *w AIF*. The results show a significant improvement with this framework. The right Y-axis of Fig. 3 shows the action taken by this framework. Fig. 4 shows the feature vectors across each time step, which is automatically

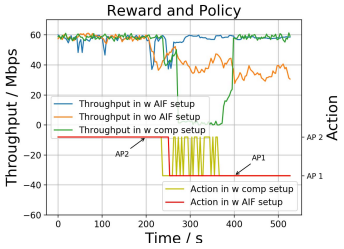

Fig. 3. We compare the performance (throughput and policies) across different framework setups, wo AIF, w comp and w AIF in Experiment 1.

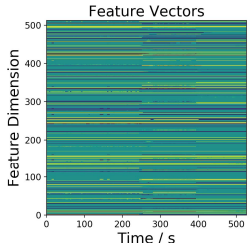

Fig. 4. The learned feature vectors of three stages in Experiment 1.

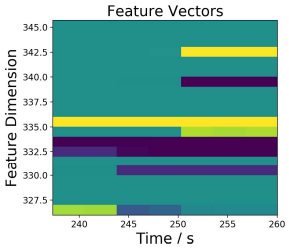

Fig. 5. A segment of feature vectors between 237.5s and 260s during the handover procedure in Experiment 1.

TABLE I

PERFORMANCE ANALYSIS

Experiment	Throughput Avg.	Throughput Std.	Handoff counts
wo AIF(2 APs)	46.00	10.77	0
wo AIF(3 APs)	42.00	11.23	0
wo AIF(4 APs)	15.67	18.29	0
w comp(2 APs)	45.00	23.01	19
w comp(3 APs)	35.76	24.01	45
w comp(4 APs)	42.12	22.01	44
w AIF(2 APs)	**56.89(26.4%)**	**4.05**	**2**
w AIF(3 APs)	**54.09(51.3%)**	**4.96**	**6**
w AIF(4 APs)	**55.23(31.1%)**	**4.12**	**7**

learned. We can explicitly find the pattern changed in about 250 and 400 second, which are the time points moving from AP2 to the middle and moving from the middle to AP1, respectively. This result means that the framework can learn the related feature vector automatically to figure out the user's state, and DRL will use this vector to make better policy. Meanwhile, Fig. 5 zooms in the feature vector in the 250th second, showing the features are exactly different during the handover procedure.

2) Experiment 2: From Table I, we can find significant improvements in throughput with AIF among all setups. Table I also shows how much AIF improves over the *w comp* setup. In *w comp*, it is difficult to set the rule to compare the RSSI values in different environments, which may cause high handover frequency and bad performance. However, the policy taken by AIF alleviates the ping-pong effect and reduces the handover frequency while guaranteeing high throughput.

TABLE II

COST ANALYSIS

Experiment	Memory	Training Time
AIF(2 APs)	1GB	12 hour
AIF(3 APs)	1GB	24 hour
AIF(4 APs)	1GB	26 hour

We analyze the memory and time costs on CPU of training AIF in Table II. The training time is strongly related to the complexity of environment which is the bottleneck. However, this can be solved by using GPU hardware and asynchronous training to accelerate the training speed. Thus, the system can be expanded without a huge penalty in terms of cost.

This framework can extend to multiple users scenario by runnning multiple agents independently, with one agent serving one user. This framework can work because the policy taken by an agent will have potential impacts on the RSSI data of other users, and other agents will try to figure out these changes by learning those parameters. Thus, other agents can make better policy for their own users based on these changes.

IV. CONCLUSION

This letter proposes an AI framework for smart wireless network management. The framework can extract intuitive features automatically and get significant improvements in many scenarios. All the experiments have shown that AIF can achieve 36.3% average improvements in throughput with lower variance. This work is the first to introduce deep reinforcement learning to the field, and provides a simple framework without manually modeling the wireless signal.

REFERENCES

[1] (Jan. 2015). *Mobile and Wireless Communications Enablers for the 2020 Information Society METIS*. [Online]. Available: http://www.metis2020.com

[2] S.-Y. Lien, K.-C. Chen, Y.-C. Liang, and Y. Lin, "Cognitive radio resource management for future cellular networks," *IEEE Wireless Commun.*, vol. 21, no. 1, pp. 70–79, Feb. 2014.

[3] S. M. A. Kazmi, N. H. Tran, W. Saad, L. B. Le, T. M. Ho, and C. S. Hong, "Optimized resource management in heterogeneousy wireless networks," *IEEE Commun. Lett.*, vol. 20, no. 7, pp. 1397–1400, Jul. 2016.

[4] K. L. A. Yau, P. Komisarczuk, and P. D. Teal, "Reinforcement learning for context awareness and intelligence in wireless networks: Review, new features and open issues," *J. Netw. Comput. Appl.*, vol. 35, no. 1, pp. 253–267, 2012.

[5] L. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[6] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[7] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529–533, 2015.

[8] H. Sun, X. Chen, Q. Shi, M. Hong, X. Fu, and N. D. Sidiropoulos. (2017). "Learning to optimize: Training deep neural networks for wireless resource management." [Online]. Available: https://arxiv.org/abs/1705.09412

[9] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1999.

[10] G. Cao, Z. Lu, T. Lei, X. Wen, L. Wang, and Y. Yang, "Demo: SDN-based seamless handover in WLAN and 3GPP cellular with CAPWAN," in *Proc. 13th Int. Symp. Wireless Commun. Syst.*, Poznan, Poland, 2016, pp. 1–3.

[11] D. Xia, J. Hart, and Q. Fu, "Evaluation of the Minstrel rate adaptation algorithm in IEEE 802.11g WLANs," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2013, pp. 2223–2228.