Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

Online memory learning for active object recognition

1st Dengsheng Chen
AI Institute, Ecovacs Robotics
Nanjing, China
dense.chen@ecovacs.com

2nd Yuanlong Yu*
AI Institute, Ecovacs Robotics
Nanjing, China
jeff.yu@ecovacs.com

3rd Zhiyong Huang
AI Institute, Ecovacs Robotics
Nanjing, China
nn.huang@ecovacs.com

***Abstract*—Traditional CNN-based recognition algorithms are trained for limited labeled data, which may not perform well in a different environment due to the lack of adaptivity of the CNN networks. So the traditional CNN-based recognition algorithms can not play a good role in robot applications because the robots have to work in different environments. However, the robot can continuously perceive new images during its mission. These images contain lots of environment-related features but lack of labels. So the robots must learn the environment-related features adaptively with unlabeled data to further improve the performance of CNN-based recognition algorithms. We call this ability as active object recognition (OBR). In this paper, we designed a dynamic memory structure (DMS) which can adaptively learn the environment-related features online and embedded DMS into a VGG-16 network to implement active object recognition. We also evaluate our dynamic memory network of CIFAR-10 and CIFAR-100 classification dataset. The results show that by learning environment-related features, dynamic memory network achieves a better performance on classification accuracy. More importantly, the network can have the ability to improve itself while many times testing.**

I. INTRODUCTION

Deep convolutional neural networks [1], [2] have led to a series of breakthroughs for image classification [3], [4]. Deep networks naturally integrate low/mid/high-level features and classifiers in an end-to-end multi-layer fashion, the "levels" of features can be enriched by the number of stacked layers(depth). In traditional computer vision tasks, the model was trained in limited labeled data and cannot learn a lot of environment-related features. In robot navigation, we can capture a large number of environment-related images by camera sensor that contains a lot of environment-related features but lack of labels. However, those environment-related images captured in robot navigation cannot be utilized in traditional classification neural network because of the two troubles. The one is that traditional classification network is mostly trained in a supervised learning mode, which means training data must be with label information. However, the captured images do not have any label information at all. The other one is that the parameters of the classification network are fixed once the network has finished the training process. This limit the network to improve itself as a robot does an environment navigation.

How to utilize environment-related features adaptively with unlabeled data to further improve the performance of CNN-based recognition algorithms is a necessary skill for the robot. We call such skill as active object recognition (OBR). In OBR, algorithms must have the ability to learn environment-related features adaptively **without** labels information and update parameters of the network automatically. More importantly, an adaptive learning network's performance can be improved while robots meet the same environment.

Inspired by memory network structure which most used in natural language processing, we combine a traditional classification network with memory structure to tackle ORB. In this paper, we design a dynamic memory structure (DMS) to learn environment-related features adaptively. A dynamic memory structure can record new features and update memory structures which give the original network the ability to update the parameters of the network. The updated memory will provide better environment-related features of the original network while forwarding computing.

The main contributions of our work are:

- we design a dynamic memory structure (DMS) to realize active recognition algorithm by online memory learning. To our best knowledge, it is the first paper to observe how to utilize the environment-related features on unlabeled images to improve the network's performance.
- by adaptive learning, dynamic memory network can achieve a better performance in same specified environment and the performance can be improved after feeding the testing data many times.

The remainder of this paper is organized as follows. Related work is described in Section II, while Section III explains our task definition (Active object recognition) and section IV explains our proposed dynamic memory structure. Section V outlines our active object recognition network and train policy. Our experimental setup and results are presented in Section VI. Finally, Section VII describes our future work and concludes this paper.

II. RELATED WORK

Image classification in computer vision and dynamic memory network in natural language processing are both the most powerful applications of deep learning. The following introduces the most related part of these two fields with our work.

A. Image classification in computer vision

In recent years, we have witnessed the great success of convolutional neural networks (CNN) [2] in a wide range of visual applications, including image classification [5], [6],

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

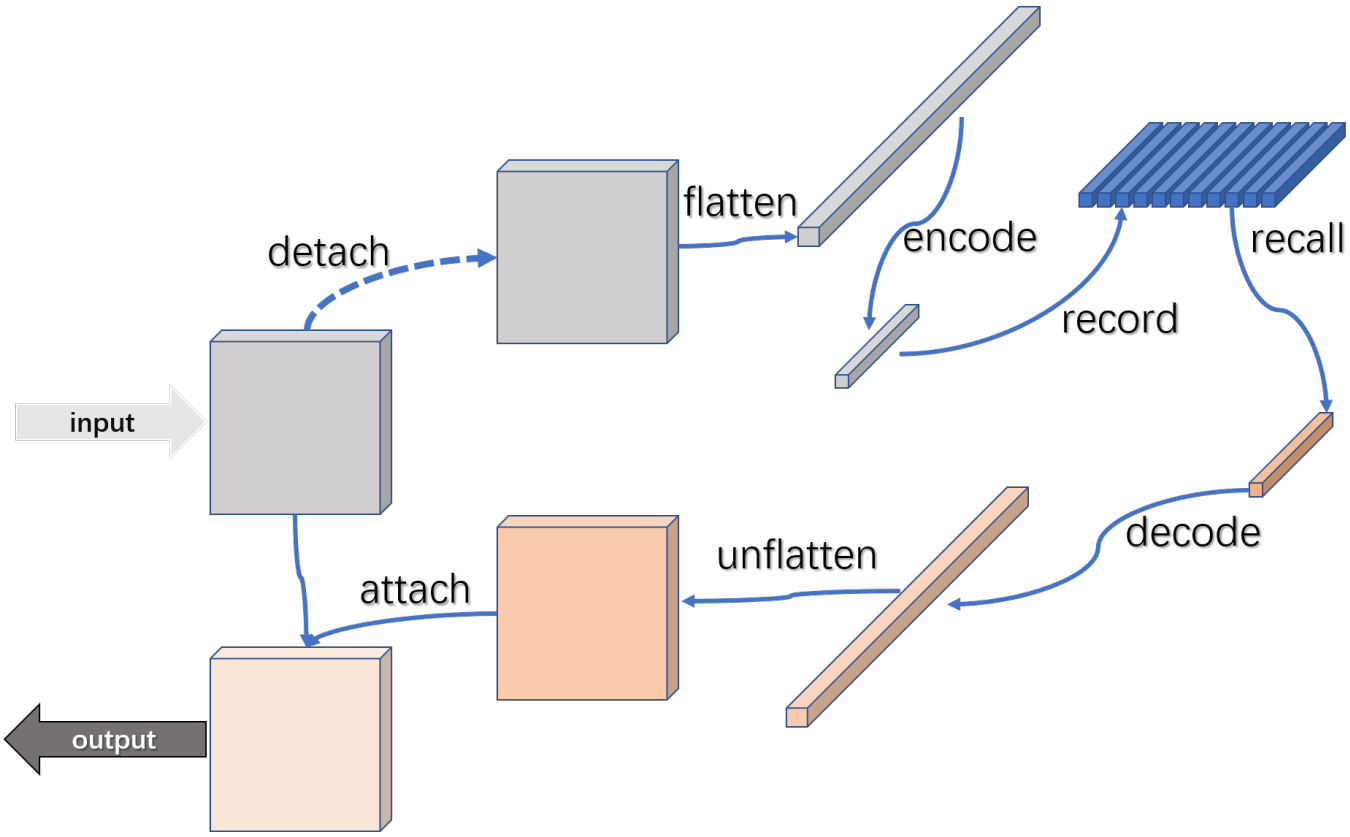


Fig. 1: Architecture of Dynamic Memory Structure (DMS). A DMS unit is consisting of four steps write (or record) operations (detach, flatten, encode, record) and four symmetrical operations (recall, decode, unflatten, attach). Recording operation will write current features into memory structure based on cosine similarity between current features and memory items. Recalling operation will read out the most related features of current input features from memory structure also based on cosine similarity. A more detailed description is listed in Section IV.

object detection [7], [8], [9], etc. This success mainly comes from deeper network architectures as well as the tremendous training data. However, parameters of a network are fixed and the training data is always lack of environment-related features of computer vision task.

During the robot navigation application, it is crucial for a robot to alternate with the new environment by learning environment-related features. And the current existing classification network is incompetent under this demand.

B. Dynamic memory network in natural language processing

Neural networks augmented with [10], [11] have the ability to learn algorithmic solutions to complex tasks. These models appear to promise for applications, such as language modeling [12] and machine translation [13]. One recent architecture, Memory Networks, make use of a content addressable memory that is accessed via a series of read operations [14] and has been successfully applied to a number of questions answering tasks [15]. In these tasks, the memory is pre-loaded using a learned embedding of provided context, such as a paragraph of text, and then the controller, given an embedding of the question, repeatedly queries the memory by content-based reads to determine an answer.

In our work, we designed a dynamic memory structure which is very similar to the memory network but the memory is updated online and store the environment-related features adaptively in robot navigation.

III. Active Object Recognition Task Definition

The main target of active object recognition is to make traditional neural network have the ability to learn environment-related features of observed images without any label information. The network's performance can be improved by feeding the observed images many times. In other words, while robot observing an environment many times, the classification accuracy of the object in the observed environment will be improved.

Here, we design a dynamic memory structure (DMS) to realize active object recognition by recording the environment-related features automatically as robot working. Furthermore, we embedded our DMS into VGG-16 network to valid the adaptive learning ability of dynamic memory network. More detailed instruction about DMS will be described in the next section.

IV. Dynamic Memory Structure

The pipeline of dynamic memory structure (DMS) is shown in Fig 1. For iteration k in layers l, we receive input data $\mathcal{I}$ form lasted layer $l-1$, and generate $\mathcal{O}$ for next layer $l+1$. We will first detach $\mathcal{I}$ from computing graph, use random ferns to get a sparse code expression for current data. We will record this sparse code into our memory, recall a series of related memory items to generate new sparse code for current data $\mathcal{I}$. Finally, a fern decoding operation will be applied to get final features. A more detailed illustration will be described in the following.

A. Input Detach

$$\mathcal{I}' = |\mathcal{D}(\mathcal{I})|$$

where function $\mathcal{D}$ will detach variable from computing graph, which indicates that the gradient of the variable will not be propagating backward while parameters updating to layer $l-1$. $|.|$ will flat the variable to the size of $B \times N$, which B is the batch size while training, N is feature size.

After flatting operation, we can treat all 1-D, 2-D and 3-D data as a 1-D feature code, which means our dynamic memory structure can handle the different dimension data at the same time without any additional operation.

B. Ferns Encode

$$f = \mathcal{F}(\mathcal{I}', \Omega)$$

where Ω is indices set randomly selected from $[0, N]$.

The function $\mathcal{F}$ is defined as:

$$\mathcal{F}(\mathcal{I}', \Omega) = \mathcal{I}'_{i+1} + \mathcal{I}'_{i-1} - 2\mathcal{I}'_i, i \in \Omega$$

In order to get a sparse feature expression of large input data I', we first randomly select an index set Ω of I' at the initialization process of dynamic memory structure. Once we have identified Ω, we will apply it to all input data, and generate a sparse feature code for I'.

C. Memory Record

$$\mathcal{M}_k = \mathcal{S}(f, \omega, \mathcal{M}_{k-1})$$

where ω is trainable recording weights. We will first feed f into a linear with parameters ω and generate a feature vector f_w with size of $B \times width$. $\mathcal{M}$ is a dynamic memory storage with size of $items \times width$. We adapt cosine similarity to measure the distance between each feature vector f and each memory items m, which is:

$$\sigma = \frac{\sum_{i=1}^{width} f_i \times m_i}{\sqrt{\sum_{i=1}^{width} f_i^2} \times \sqrt{\sum_{i=1}^{width} (m_i^2)}}$$

While the recording stage, we regard σ as a weight of writing operation to each item of memory $\mathcal{M}$:

$$m_k = m_{k-1} \times (1 - \sigma) + f \times \sigma$$

And this recording operation can be performed at any time, which makes our dynamic have the ability to realize adaptive learning. It is better to apply cosine similarity and update memory items by each sample to get the updated memory $\mathcal{M}_k$. However, this operation will cost a lot of times going through every sample. Actually, we just calculate the cosine similarity for all batches with $\mathcal{M}_{k-1}$, and using a mean feature vector f_{mean} with a mean cosine similarity σ_{mean} to update $\mathcal{M}_{k-1}$ once getting $\mathcal{M}_k$.

D. Memory Recall

$$s_k^{-1} = \mathcal{S}^{-1}(f, \omega^{-1}, \mathcal{M}_k^{'})$$

where ω^{-1} is trainable recalling weights and $\mathcal{M}_k^{'}$ is the detached $\mathcal{M}_k$:

$$\mathcal{M}'_k = \mathcal{D}(\mathcal{M}')$$

we detach $\mathcal{M}_k$ to avoid creating a circle in the computing graph. First, we select the most similar items from $\mathcal{M}_k$ as current recall feature vector f_r, and pass it through a linear layer with parameters ω^{-1} to get final recalled output from memory. The memory recall and record linear layers consist of an auto-decoder and auto-encoder structure, and $\mathcal{M}$ store the encoded feature. By this auto-encoder and auto-decoder pair, we can efficiently capture the main feature of different data and pick up the most related information from memory by cosine similarity. Different from most existing auto-encoder/decoder structure, our recalling and recording operation have the ability to store new information from current data and add history feature to current data. The history feature from memory can play a very significant role in feature argument for current data, the new information of current data can help the memory to build up a more robust history feature information.

While adaptive learning, we will record a lot of new feature information in memory which may not come across in training data. At this point, the performance of network which has embedded with dynamic memory network will be improved after many times validation without backward propagating, but only forward memory updating. This is our main contribution to this work.

E. Ferns Decode

$$f^{-1} = \mathcal{F}^{-1}(s_k^{-1}, \mathcal{I}', \Omega)$$

After getting the recall feature vector s_k^{-1}, we will decode it respect to current input $\mathcal{I}'$ and set Ω:

$$\mathcal{F}^{-1}(s_k^{-1}, \mathcal{I}', \Omega) = \mathcal{I}_{i+1}^{'} + \mathcal{I}_{i-1}^{'} - 2s_{k,i}^{-1}, i \in \Omega$$

We can regard this process as a reverse operation of ferns encoding operation.

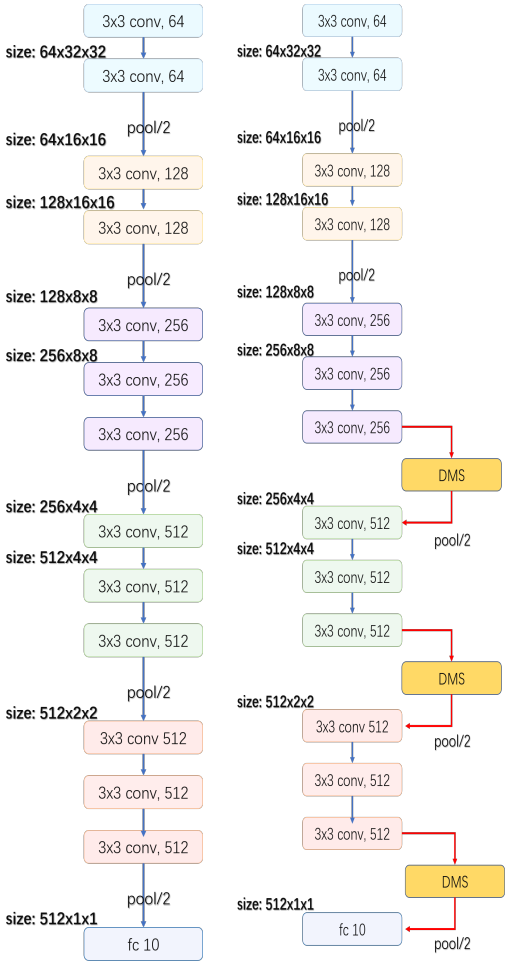
F. Output Attach

$$\mathcal{O} = [\mathcal{N}(\mathcal{I}, f^{-1})]$$

where [.] indicate fold operation to specified shape. And we will merge our decode feature to input data $\mathcal{I}$ by:

$$\mathcal{N}(\mathcal{I}, f^{-1}) = \frac{\mathcal{I}_i + f_i^{-1}}{2}, i \in \Omega$$

Here, we use a weighted sum of decoding ferns data and input data to keep a balance between them.



(a) VGG-16 (b) VGG-16-DMS

Fig. 2: Baseline(VGG-16) and our proposed active object recognition network structure (VGG-16-DMS). We embed three DMS structure before the max-pooling operation.

V. Active Object Recognition Network

As shown in Fig 2, we apply our dynamic memory structure in VGG-16[16] to construct active object recognition network, note as VGG-16-DM.

A. VGG16 Network Architecture

VGG16 network structure is consisting of 16 layers for feature extraction, which mainly consist of convolutional layers and max pooling layers. After each pooling layer, the channels of input data will be double at the first convolutional layer after this operation. We also add a batch normalization layers following each activation layers.

B. VGG16 with Dynamic Memory Structure

We insert the dynamic memory structure into the last three max-pooling layers. It is not recommended to inserting this structure into shadow layers for that the features of shadow layers are variant roughly among different input data. However, the deep layers of the network can obtain more class-related information which is more meaningful to record and recall while adaptive learning in robot working.

C. Supervised Learning and Adaptive Learning Policy

All the parameters of network expect memory $\mathcal{M}$ will be updated with BP algorithms during supervised learning. The memory $\mathcal{M}$ will be updated with record operation during adaptive learning.

During supervised learning, all the training data must assign a correct label to calculate a loss value. However, adaptive learning is based on recording operation, the label of training data is not needed, which allow us feeding a lot of captured images during robot navigation to our network to learn new environment-related features automatically.

However, the adaptive learning process is not stable, which means the record environment-related features may hurt the original network performance. To tackle this problem, we will train our network with supervised and adaptive learning alternatively each epoch.

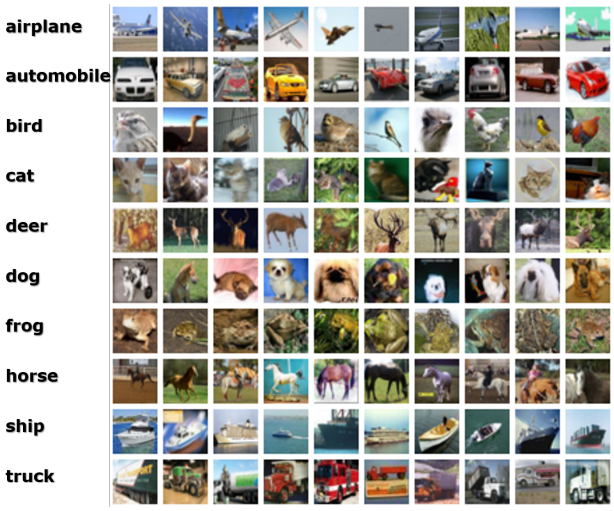


Fig. 3: The different classes of CIFAR-10 datasets.

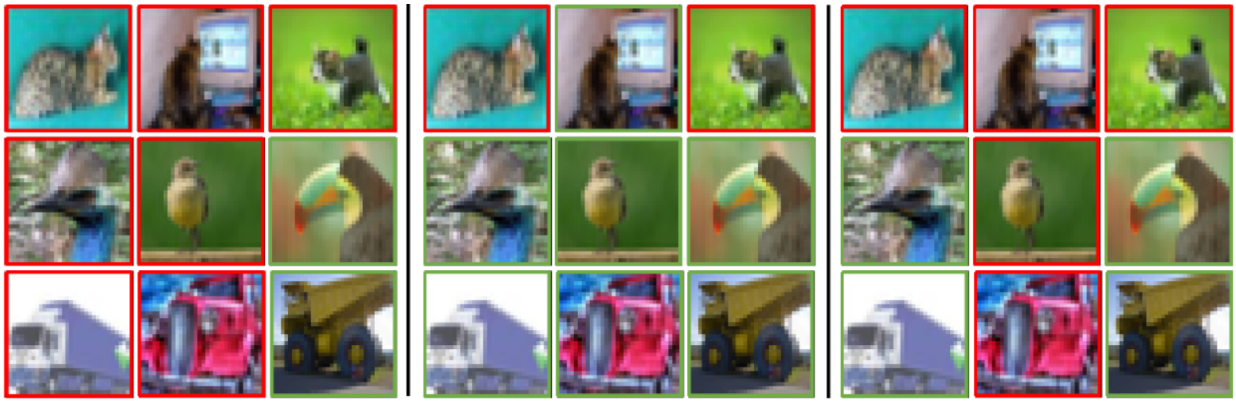


Fig. 4: Result visualization of CIFAR-10. The left most column is the result selected from baselines error result. And the middle and right most result is the corresponding results of baseline in joint-training and adaptive-decouple training. We colorized the wrong label with red bounding box and correct label with greed bounding box. Because our dynamic memory structure can learn new environment-related features automatically, the uncorrect result in baseline can be classified well.

VI. Experiments

We implement our experiments in CIFAR-10 and CIFAR-100 classification dataset [17]. In our experiments, we will feed unlabeled images to network many times to simulate the robot observes in an environment many times.

A. Datasets

1) The CIFAR-10 dataset: The CIFAR-10 dataset consists of 60000 32×32 color images in 10 classes (see Fig.3), with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. In order to validate our adaptive learning ability, we further divided the test batch into two equal parts, $\mathcal{P}_a, \mathcal{P}_b$, and both $\mathcal{P}_a$ and $\mathcal{P}_b$ contains 500 randomly-selected images from each class.

2) The CIFAR-100 dataset: This dataset is just like the CIFAR-10, except it has 100 classes containing 6000 images each. There are 500 training images and 100 testing images per class. And we also divide the testing images into two equal $\mathcal{P}_c, \mathcal{P}_d$, just as CIFAR-10.

B. Experiment Setup

We implement our method based on PyTorch[1]. The memory structure of DMS is with 64 items and 128 length for CIFAR-10, and with 128 items and 256 length for CIFAR-100. Following [18], [19], we employ a poly learning rate policy where the initial learning rate is multiplied by $(1 - \frac{iter}{total-iter})^{0.9}$ after each iteration. The base learning rate is set to 0.01. Momentum and weight decay coefficients are set to 0.9 and 0.0001 respectively. Batch size is setting 128 for both CIFAR-10 and CIFAR-100 datasets. We train the network with 200 epochs in total.

While supervised training, we feed labeled images to network and only adaptive learning we feed both labeled and unlabeled images to update the memory.

[1]https://github.com/pytorch

In order to simulate the robot navigation process, which may come across one object many times and capture many images, we regard $\mathcal{P}_{a,b,c,d}$ as captured images and feed them to our network many times.

C. Adaptive Learning Performance on Robotics Navigation

In order to better to demonstrate the adaptive learning attributes of our dynamic memory network, we have implemented five experiments on CIFAR-10 and CIFAR-100 except.

- Baseline: train VGG-16 on training dataset and test it on testing dataset.
- Joint-Train: train VGG-16-DMS with training data and $\mathcal{P}_a$ dataset jointly, and test on $\mathcal{P}_b$ dataset to demonstrate our DMS can learn environment-related features from unlabeled $\mathcal{P}_a$ training dataset. Note that the $\mathcal{P}_a$ dataset only used in adaptive learning stage without accessing to its label.
- Joint-Train-Minor: This experiment is minor experiment of the previous one.
- Adaptive-Decouple-Train: in this experiment, we first train VGG-16-DMS without accessing to test dataset. In other words, in this experiment, our network cannot adaptively learn any environment-related features while training stage. After training finished, we will feed testing dataset to VGG-16-DMS many times to simulate the robot navigation process. If the accuracy of the testing dataset can be improved by doing this, it well demonstrates the adaptive learning ability of dynamic memory network in robot navigation.

Joint-Train experiments have coupled supervised and adaptive learning together to better utilize the environment-related features while training network. The environment-related features can be well integrating with training dataset's features and will perform best in all experiments. Adaptive-Decouple-Ttrain has decoupled the training dataset's features and environment-related features. We hope the adaptive net-

work should have the ability to learn the environment-related features without the correct of supervised learning.

	CIFAR-10(%)	CIFAR-100(%)
Baseline	92.64	72.93
Joint-Train	93.05	**73.45**
Joint-Train-Minor	**93.10**	73.32

TABLE I: Joint-Train

Feeding Times	0	1	2	3
CIFAR-10(%)	92.70	92.78	92.78	**92.80**
CIFAR-100(%)	73.00	73.11	**73.13**	73.09

TABLE II: Adaptive-Decouple Train

1) experiments result: As shown in table I, after joint-training, the accuracy has been improved by almost 0.5% compared to baseline. In table II, we also list the accuracy of adaptive-only training experiment. We can regard feeding times 0 as a baseline. It is obvious that after we feed the testing dataset to DMN, testing accuracy had been improved lightly, and this shows to us that our dynamic memory structure does have aid in adaptive learning. However, it is hard for us to make sure all the environment-related features can probably aid classification task, the network performance may also be destroyed by adaptive learning. And we found that by feeding testing dataset twice the network will achieve promising performance. In this paper, we only listed top-1 accuracy of CIFAR-100.

In Fig. 4, we also visualize some results of testing images which also demonstrate that adaptive learning can learn environment-related features and get a better result.

VII. CONCLUSION

In this work, we have designed a dynamic memory structure (DMS) to realize adaptive learning by online memory learning and embedded DMS into a VGG-16 network to implement active object recognition (OBR) network. Experiments on CIFAR-10 and CIFAR-100 demonstrated the adaptive learning ability of our network. The adaptive learning of our network is still limited, we can adapt a better dynamic memory structure to avoid recording useless environment-related features. The proposed active object recognition networks can be more suitable for robot applications.

In future work, we will experiment more methods to realize OBR with more fields, such as SLAM system and scene segmentation.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[2] 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, 2014.

[3] M. D Zeiler and R. Fergus, "Visualizing and understanding convolutional neural networks," vol. 8689, 11 2013.

[4] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. Lecun, "Overfeat: Integrated recognition," *Localization and Detection using Convolutional Networks*, pp. 1–16, 01 2013.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, 2012.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Computer Science*, 2014.

[7] R. Girshick, "Fast r-cnn," *Computer Science*, 2015.

[8] S. Ren, R. Girshick, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[9] K. He, G. Gkioxari, P. Dollr, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017.

[10] Ç. Gülçehre, S. Chandar, K. Cho, and Y. Bengio, "Dynamic neural turing machine with soft and hard addressing schemes," *CoRR*, vol. abs/1607.00036, 2016. [Online]. Available: http://arxiv.org/abs/1607.00036

[11] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwiska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, and J. Agapiou, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[12] H. Deng, Z. Lei, and S. Xin, "Feature memory-based deep recurrent neural network for language modeling," *Applied Soft Computing*, vol. 68, 2018.

[13] F. Meng, Z. Tu, Y. Cheng, H. Wu, J. Zhai, Y. Yang, and D. Wang, "Neural machine translation with key-value memory-augmented attention," *CoRR*, vol. abs/1806.11249, 2018. [Online]. Available: http://arxiv.org/abs/1806.11249

[14] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, "End-to-end memory networks," *Computer Science*, 2015.

[15] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. Van Merrinboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *Computer Science*, 2015.

[16] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.

[17] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Computer Science Department, University of Toronto, Tech. Rep*, vol. 1, 01 2009.

[18] Y. Yuan and J. Wang, "Ocnet: Object context network for scene parsing," *CoRR*, vol. abs/1809.00916, 2018. [Online]. Available: http://arxiv.org/abs/1809.00916

[19] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *CoRR*, vol. abs/1706.05587, 2017. [Online]. Available: http://arxiv.org/abs/1706.05587