Dual-Memory Deep Learning Architectures for Lifelong Learning of Everyday Human Behaviors

Sang-Woo Lee[1], **Chung-Yeon Lee**[1], **Dong Hyun Kwak**[2]
Jiwon Kim[3], **Jeonghee Kim**[3], **and Byoung-Tak Zhang**[1,2]
[1]School of Computer Science and Engineering, Seoul National University
[2]Interdisciplinary Program in Neuroscience, Seoul National University
[3]NAVER LABS

Abstract

Learning from human behaviors in the real world is important for building human-aware intelligent systems such as personalized digital assistants and autonomous humanoid robots. Everyday activities of human life can now be measured through wearable sensors. However, innovations are required to learn these sensory data in an online incremental manner over an extended period of time. Here we propose a dual memory architecture that processes slow-changing global patterns as well as keeps track of fast-changing local behaviors over a lifetime. The lifelong learnability is achieved by developing new techniques, such as weight transfer and an online learning algorithm with incremental features. The proposed model outperformed other comparable methods on two real-life data-sets: the image-stream dataset and the real-world lifelogs collected through the Google Glass for 46 days.

1 Introduction

Lifelong learning refers to the learning of multiple consecutive tasks with never-ending exploration and continuous discovery of knowledge from data streams. It is crucial for the creation of intelligent and flexible general-purpose machines such as personalized digital assistants and autonomous humanoid robots [Thrun and O'Sullivan, 1996; Ruvolo and Eaton, 2013; Ha *et al.*, 2015]. We are interested in the learning of abstract concepts from continuously sensing non-stationary data from the real world, such as first-person view video streams from wearable cameras [Huynh *et al.*, 2008; Zhang, 2013] (Figure 1).

To handle such non-stationary data streams, it is important to learn deep representations in an online manner. We focus on the learning of deep models on new data at minimal costs, where the learning system is allowed to memorize a certain amount of data, (e.g., 100,000 instances per online learning step for a data stream that consists of millions of instances). We refer to this task as online deep learning, and the dataset memorized in each step, the online dataset. In this setting, the system needs to learn the new data in addition to the old data in a stream which is often non-stationary.

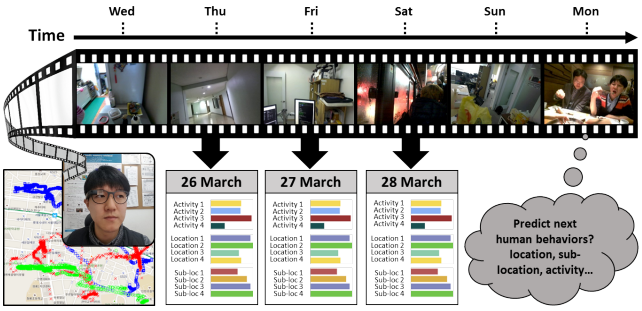

Figure 1: Life-logging paradigm using wearable sensors

However, this task is challenging because learning new data through neural networks often results in a loss of previously acquired information, which is known as catastrophic forgetting [Goodfellow *et al.*, 2013]. To avoid this phenomenon, several studies have adopted an incremental ensemble learning approach, whereby a weak learner is made to use the online dataset, and multiple weak learners are combined to obtain better predictive performance [Polikar *et al.*, 2001]. Unfortunately, in our experiment, simple voting with a weak learner learnt from a relatively small online dataset did not work well; it seems the relatively smaller online dataset is insufficient for learning highly expressive representations of deep neural networks.

To address this issue, we propose a dual memory architecture (DMA). This architecture trains two memory structures: one is a series of deep neural networks, and the other consists of a shallow kernel network that uses a hidden representation of the deep neural networks as input. The two memory structures are designed to use different strategies. The ensemble of deep neural networks learns new information in order to adapt its representation to new data, whereas the shallow kernel network aims to manage non-stationary distribution and unseen classes more rapidly.

Moreover, some techniques for online deep learning are proposed in this paper. First, the transfer learning technique via weight transfer is applied to maximize the representation power of each neural module in online deep learning [Yosinski *et al.*, 2014]. Second, we develop multiplicative Gaussian hypernetworks (mGHNs) and their online learning method.

An mGHN concurrently adapts both structure and parameters to the data stream by an evolutionary method and a closed-form-based sequential update, which minimizes information loss of past data.

2 Dual Memory Architectures

2.1 Dual Memory Architectures

The dual memory architecture (DMA) is a framework designed to continuously learn from data streams. The framework of the DMA is illustrated in Figure 2. The DMA consists of deep memory and fast memory. The structure of deep memory consists of several deep networks. Each of these networks is constructed when a specific amount of data from an unseen probability distribution is accumulated, and thus creates a deep representation of the data in a specific time. Examples of deep memory models are deep neural network classifier, convolutional neural networks (CNNs), deep belief networks (DBNs), and recurrent neural networks (RNNs). The fast memory consists of a shallow network. The input of the shallow network is hidden nodes at upper layers of deep networks. Fast memory aims to be updated immediately from a new instance. Examples of shallow networks include linear regressor, denoising autoencoder [Zhou *et al.*, 2012], and support vector machine (SVM) [Liu *et al.*, 2008], which can be learned in an online manner. The shallow network is in charge of making inference of the DMA; deep memory only yields deep representation. The equation used for inference can be described as (1):

$$y = \delta(w^T \phi(h^{\{1\}}(x), h^{\{2\}}(x), \cdots, h^{\{k\}}(x))) \qquad (1)$$

where x is the input (e.g., a vector of image pixels), y is the target, ϕ and w are a kernel and a corresponding weight, h is values of the hidden layer of a deep network used for the input of the shallow network, δ is an activation function of the shallow network, and k is an index for the last deep network ordered by time.

Fast memory updates parameters of its shallow network immediately from new instances. If a new deep network is formed in the deep memory, the structure of the shallow network is changed to include the new representation. Fast memory is referred to as fast for two properties with respect to learning. First, a shallow network learns faster than a deep network in general. Second, a shallow network is better able to adapt new data through online learning than a deep network. If the objective function of a shallow network is convex, a simple stochastic online learning method, such as online stochastic gradient descent (SGD), can be used to guarantee a lower bound to the objective function [Zinkevich, 2003]. Therefore, an efficient online update is possible. Unfortunately, learning shallow networks in the DMA is more complex. During online learning, deep memory continuously forms new representations of a new deep network; thus, new input features appear in a shallow network. This task is a kind of online learning with an incremental feature set. In this case, it is not possible to obtain statistics of old data at new features. i.e., if a node in the shallow network is a function of $h^{\{k\}}$, statistics of the node cannot be obtained from the

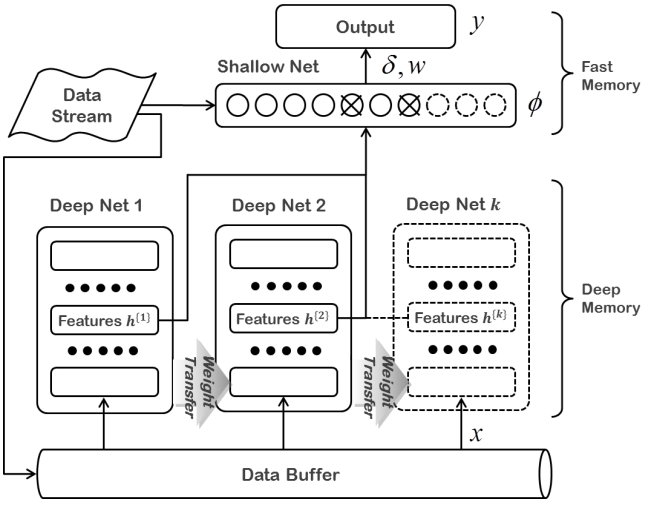

Figure 2: A schematic diagram of the dual memory architecture (DMA). With continuously arrived instances of data streams, fast memory updates its shallow network immediately. If certain amount of data is accumulated, deep memory makes a new deep network with this new online dataset. Simultaneously, the shallow network changes its structure corresponding to deep memory.

$1^{st} \sim k\text{-}1^{th}$ online dataset. In this paper, we explore online learning by shallow networks using an incremental feature set in the DMA.

In learning deep memory, each deep neural network is trained with a corresponding online dataset by its objective function. Unlike the prevalent approach, we use the transfer learning technique proposed by [Yosinski *et al.*, 2014] to utilize the knowledge from a older deep network to form a new deep network. This transfer technique initializes the weights of a newly trained deep network W_k by the weights of the most recently trained deep network W_{k-1}. Although this original transfer method assumes two networks have the same structure, there are some extensions that allow different widths and a number of layers between some networks [Chen *et al.*, 2015]. Once the training of the deep network is complete by its own online dataset, the weights of the network do not change even though new data arrives. This is aimed to minimize changes of input in the shallow network in fast memory.

2.2 Comparative Models

Relatively few studies to date have been conducted on training deep networks online from data streams. We categorize these studies into three approaches. The first approach is *online fine-tuning*, which is simple online learning of an entire neural network based on SGD. In this setting, a deep network is continuously fine-tuned with new data as the data is accumulated. However, it is well-known that learning neural networks requires many epochs of gradient descent over the entire dataset because the objective function space of neural networks is complex. Recently, in [Nam and Han, 2015], online fine-tuning of a CNN with simple online SGD was used

Table 1: Properties of DMA and comparative models

	Many deep networks	Online learning	Dual memory structure
Online fine-tuning		✓	
Last-layer fine-tuning		✓	
Naïve incremental bagging	✓	✓	
DMA (our proposal)	✓	✓	✓
Incremental bagging w/ transfer	✓	✓	
DMA w/ last-layer retraining	✓		✓
Batch			

in the inference phase of visual tracking, which made state-of-the-art performance in the Visual Object Tracking Challenge 2015. However, it does not guarantee the retention of old data. The equation of this algorithm can be described as follows:

$$y \sim softmax(f(h^{\{1\}}(x))) \tag{2}$$

where f is a non-linear function of a deep neural network. This equation is the same in the case of batch learning, where *Batch* denotes the common algorithm that learns all the training data at once, with a single neural network.

The second approach is *last-layer fine-tuning*. According to recent works on transfer learning, the hidden activation of deep networks can be utilized as a satisfactory general representation for learning other related tasks. Training only the last-layer of a deep network often yields state-of-the-art performance on new tasks, especially when the dataset of a new task is small [Zeiler and Fergus, 2014; Donahue *et al.*, 2014]. This phenomenon makes online learning of only the last-layer of deep networks promising, because online learning of shallow networks is much easier than that of deep networks in general. Recently, online SVM with hidden representations of pre-trained deep CNNs using another large image dataset, ImageNet, performed well in visual tracking tasks [Hong *et al.*, 2015]. Mathematically, the last-layer fine-tuning is expressed as follows:

$$y = \delta(w^T \phi(h^{\{1\}}(x))). \tag{3}$$

The third approach is incremental bagging. A considerable amount of research has sought to combine online learning and ensemble learning [Polikar *et al.*, 2001; Oza, 2005]. One of the simplest methods involves forming a neural network with some amount of online dataset and bagging in inference. Bagging is an inference technique that uses the average of the output probability of each network as the final output probability of the entire model. If deep memory is allowed to use more memory in our system, a competitive approach involves using multiple neural networks, especially when the data stream is non-stationary. In previous researches, in contrast to our approach, transfer learning techniques were not used. We refer to this method as *naïve incremental bagging*. The equation of incremental bagging can be described as follows:

$$y \sim \frac{1}{d} \sum_{i}^{d} softmax(f_d(h^{\{d\}}(x)). \tag{4}$$

The proposed DMA is a combination of the three ideas mentioned above. In DMA, a new deep network is formed when a dataset is accumulated, as in the incremental bagging. However, the initial weights of new deep networks are drawn from the weights of older deep networks, as in the online learning of neural networks. Moreover, a shallow network in fast memory is concurrently trained with deep memory, which is similar to the last-layer fine-tuning approach.

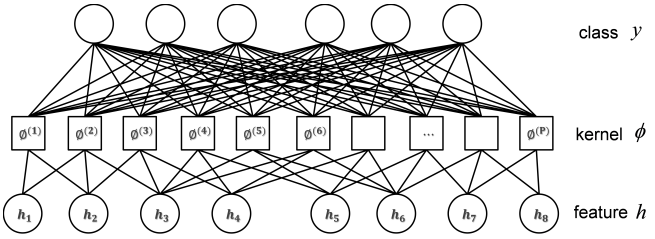

Figure 3: A schematic diagram of the multiplicative-Gaussian hypernetworks

To clarify the concept of DMA, we additionally propose two learning methods. One is *incremental bagging with transfer*. Unlike naïve incremental bagging, this method transfers the weights of older deep networks to the new deep network, as in DMA. The other is *DMA with last-layer retraining* in which a shallow network is retrained in a batch manner. Although this algorithm is not part of online learning, it is practical because batch learning of shallow networks is much faster than that of deep networks in general. The properties of DMA and comparative methods are listed in Table 1.

3 Online Learning of Multiplicative Gaussian Hypernetworks

3.1 Multiplicative-Gaussian Hypernetworks

In this section, we introduce a multiplicative Gaussian hypernetwork (mGHN) as an example of fast memory (Figure 3). mGHNs are shallow kernel networks that use a multiplicative function as an explicit kernel in (5):

$$\begin{aligned} \phi &= [\phi^{(1)}, \cdots, \phi^{(p)}, \cdots, \phi^{(P)}]^T, \\ \phi^{(p)}(h) &= (h_{(p,1)} \times \cdots \times h_{(p,H_p)}), \end{aligned} \tag{5}$$

where P is a hyperparameter of the number of kernel functions, and $\times$ denotes scalar multiplication. h is the input feature of mGHNs, and also represents the activation of deep neural networks. The set of variables of the p^{th} kernel $\{h_{(p,1)}, ..., h_{(p,H_p)}\}$ is randomly chosen from h, where H_p is the order or the number of variables used in the p^{th} kernel. The multiplicative form is used for two reasons, although an arbitrary form can be used. First, it is an easy, randomized method to put sparsity and non-linearity into the model, which is a point inspired by [Zhang *et al.*, 2012]. Second, the kernel could be controlled to be a function of few neural networks.

mGHNs assume the joint probability of target class y and ϕ is Gaussian as in (6):

$$p\left(\begin{array}{c} y \\ \phi(h) \end{array}\right) = N\left(\left[\begin{array}{c} \mu_y \\ \mu_\phi \end{array}\right], \begin{array}{cc} \Sigma_y & \Sigma_{y\phi} \\ \Sigma_{y\phi}^T & \Sigma_{\phi\phi} \end{array}\right), \quad (6)$$

where μ_y, μ_ϕ, Σ_{yy}, $\Sigma_{y\phi}$, and $\Sigma_{\phi\phi}$ are the sufficient statistics of the Gaussian corresponding to y and ϕ. Target class y is represented by one-hot encoding. The discriminative distribution is derived by the generative distribution of y and ϕ, and predicted y is real-valued score vector of the class in the inference.

$$E[p(y|h)] = \mu_y + \Sigma_{y\phi} \cdot \Sigma_{\phi\phi}^{-1} \cdot (\phi(h) - \mu_\phi) \quad (7)$$

Note that the parameters of mGHNs can be updated immediately from a new instance by online update of the mean and covariance if the number of features does not increase [Finch, 2009].

3.2 Structure Learning

If the k^{th} deep neural network is formed in deep memory, the mGHN in fast memory receives a newly learned feature $h^{\{k\}}$, which consists of the hidden values of the new deep neural network. As the existing kernel vector is not a function of $h^{\{k\}}$, a new kernel vector ϕ_k should be formed. The structure of mGHNs is learned via an evolutional approach, as illustrated in Algorithm 1.

Algorithm 1 Structure Learning of mGHNs

repeat
  if New learned feature $h^{\{k\}}$ comes **then**
    Concatenate old and new feature (i.e., $h \leftarrow h \bigcup h^{\{k\}}$.)
    Discard a set of kernel $\phi_{discard}$ in ϕ (i.e., $\hat{\phi} \leftarrow \phi - \phi_{discard}$.)
    Make a set of new kernel $\phi_k(h)$ and concatenate into ϕ (i.e., $\phi \leftarrow \hat{\phi} \bigcup \phi_k$.)
  end if
until forever

The core operations in the algorithm consist of discarding kernel and adding kernel. In our experiments, the set of $\phi_{discard}$ was picked by selecting the kernels with the lowest corresponding weights. From Equation (7), ϕ is multiplied by $\Sigma_{y\phi}\Sigma_{\phi\phi}^{-1}$ to obtain $E[p(y|h)]$, such that weight $w^{(p)}$ corresponding to $\phi^{(p)}$ is the p^{th} column of $\Sigma_{y\phi}\Sigma_{\phi\phi}^{-1}$ (i.e., $w^{(p)} = (\Sigma_{y\phi}\Sigma_{\phi\phi}^{-1})_{(p,:)}$.) The length of $w^{(p)}$ is the number of class categories, as the node of each kernel has a connection to each class node. We sort $\phi^{(p)}$ in descending order of $\max_j |w_j^{(p)}|$, where the values at the bottom of the $\max_j |w_j^{(p)}|$ list correspond to the $\phi_{discard}$ set. The size of $\phi_{discard}$ and ϕ_k are determined by $\alpha|\phi|$ and $\beta|\phi|$ respectively, where $|\phi|$ is the size of the existing kernel set, and α and β are predefined hyperparameters.

3.3 Online Learning on Incrementing Features

As the objective function of mGHNs follows the exponential of the quadratic form, second-order optimization can be applied for efficient online learning. For the online learning of mGHNs with incremental features, we derive a closed-form sequential update rule to maximize likelihood based on studies of regression with missing patterns [Little, 1992].

Suppose kernel vectors ϕ_1 and ϕ_2 are constructed when the first $(d = 1)$ and the second $(d = 2)$ online datasets arrive. The sufficient statistics of ϕ_1 can be obtained for both the first and second datasets, whereas information of only the second dataset can be used for ϕ_2. Suppose $\hat{\mu}_{i\cdot d}$ and $\hat{\Sigma}_{ij\cdot d}$ are empirical estimators of the sufficient statistics of the i^{th} kernel vector ϕ_i and j^{th} kernel vector ϕ_j corresponding to the distribution of the d^{th} dataset. $d = 12$ denotes both the first and the second datasets. If these sufficient statistics satisfy the following equation (8):

$$\begin{gathered} \phi|_{d=1} \sim N(\hat{\mu}_{1\cdot 1}, \hat{\Sigma}_{11\cdot 1}), \\ \left[\begin{array}{c} \phi_1 \\ \phi_2 \end{array}\right]|_{d=2} \sim N\left(\left[\begin{array}{c} \hat{\mu}_{1\cdot 2} \\ \hat{\mu}_{2\cdot 2} \end{array}\right], \begin{array}{cc} \hat{\Sigma}_{11\cdot 2} & \hat{\Sigma}_{12\cdot 2} \\ \hat{\Sigma}_{21\cdot 2} & \hat{\Sigma}_{22\cdot 2} \end{array}\right), \\ \phi_1|_{d=1,2} \sim N(\hat{\mu}_{1\cdot 12}, \hat{\Sigma}_{11\cdot 12}), \end{gathered} \quad (8)$$

the maximum likelihood solution represents ϕ as (9).

$$\left[\begin{array}{c} \phi_1 \\ \phi_2 \end{array}\right]|_{d=1,2} \sim N\left(\left[\begin{array}{c} \hat{\mu}_{1\cdot 12} \\ \tilde{\mu}_2 \end{array}\right], \begin{array}{cc} \hat{\Sigma}_{11\cdot 12} & \tilde{\Sigma}_{12} \\ \tilde{\Sigma}_{12}^T & \tilde{\Sigma}_{22} \end{array}\right), \quad (9)$$

$$\begin{aligned} \tilde{\mu}_2 &= \hat{\mu}_{2\cdot 2} + \hat{\Sigma}_{12\cdot 2}^T \cdot \hat{\Sigma}_{11\cdot 2}^{-1} \cdot (\hat{\mu}_{1\cdot 12} - \hat{\mu}_{1\cdot 2}), \\ \tilde{\Sigma}_{12} &= \hat{\Sigma}_{11\cdot 12} \cdot \hat{\Sigma}_{11\cdot 2}^{-1} \cdot \hat{\Sigma}_{12\cdot 2}, \\ \tilde{\Sigma}_{22} &= \hat{\Sigma}_{22\cdot 2} - \hat{\Sigma}_{12\cdot 2}^T \cdot \hat{\Sigma}_{11\cdot 2}^{-1} \cdot (\hat{\Sigma}_{12\cdot 2} - \tilde{\Sigma}_{12}) \end{aligned}$$

(9) can also be updated immediately from a new instance by online update of the mean and covariance. Moreover, (9) can be extended to sequential updates, when there is more than one increment of the kernel set (i.e., $\phi_3, \cdots, \phi_k$).

Note that the proposed online learning algorithm estimates generative distribution of ϕ, $p(\phi_1, \cdots, \phi_k)$. When training data having ϕ_k is relatively small, information of ϕ_k can be complemented by $p(\phi_k|\phi_{1:k-1})$, which helps create a more efficient prediction of y. The alternative of this generative approach is a discriminative approach. For example, in [Liu *et al.*, 2008], LS-SVM is directly optimized to get the maximum likelihood solution over $p(y|\phi_{1:k})$. However, equivalent solutions from the discriminative method can also be produced by the method of filling in the missing values with 0 (e.g., assume $\phi_2|_{d=1}$ as 0). This is not what we desire intuitively.

4 Experiments

4.1 Non-stationary Image Data Stream

We investigate the strengths and weaknesses of the proposed DMA in an extreme non-stationary environment using a well-known benchmark dataset. The proposed algorithm was tested on the CIFAR-10 image dataset consisting of 50,000 training images and 10,000 test images from 10 different object classes. The performance of these algorithms were evaluated using a 10-split experiment where the model is learned in a sequential manner from 10 online datasets. In this experiment, each online dataset consists of images of only $3 \sim 5$ classes. Figure 4 shows the distribution of the data stream.

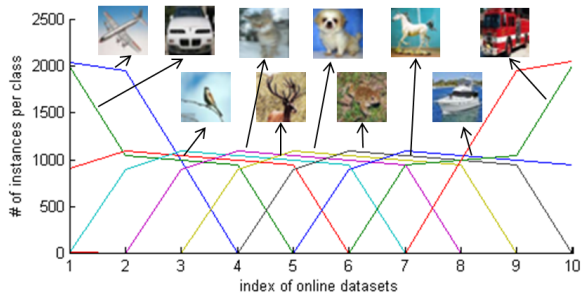

Figure 4: Distribution of non-stationary data stream of CIFAR-10 in the experiment

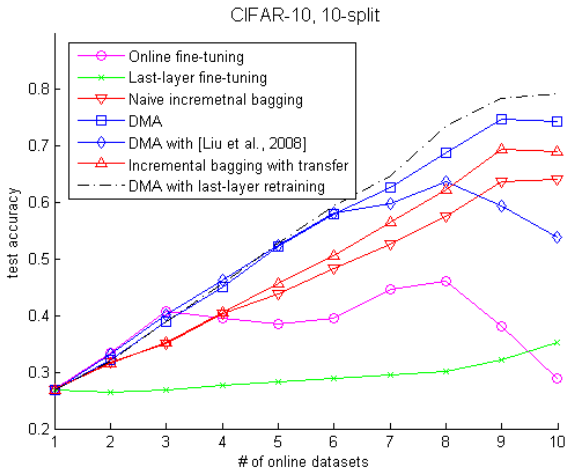

Figure 5: The test accuracy of various learning algorithms on non-stationary data stream of CIFAR-10

We use the Network in Network model [Lin *et al.*, 2014], a kind of deep CNN, implemented using the MatConvNet toolbox [Vedaldi and Lenc, 2015]. In all the online deep learning algorithms, the learning rate is set to 0.25 and then is reduced by a constant factor of 5 at some predefined steps. The rate of weight decay is 5×10^{-4} and the rate of momentum is 0.9.

Figure 5 shows the experimental results of 10-split experiments on non-stationary data. DMA outperforms all other online deep learning algorithms, the result of which supports our proposal. Some algorithms including online fine-tuning and last-layer fine-tuning show somewhat disappointing results.

4.2 Lifelog Dataset

The proposed DMA was demonstrated on a Google Glass lifelog dataset, which was collected over 46 days from three participants using Google Glasses. The 660,000 seconds of the egocentric video stream data reflects the subjects' behaviors including activities in indoor environments such as home, office and restaurant, and outdoor activities such as walking on the road, taking the bus or waiting for arrival of the subway. The subjects were asked to notate activities what they are doing and places where they are on by using a life-logging application installed on their mobile phone in real-time. The notated data was then used as labels for the classification task in our experiments. For evaluation, the dataset of each subject is separated into training set and test set in order of time. An frame image of each second are used and classified as one instance. The statistics of the dataset are summarized in Table 2. The distribution of major five classes in each type of labels are presented in Table 3.

Table 2: Statistics of the lifelog dataset of each subject

	Instances (sec/day)		Number of class		
	Training	Test	Location	Sub-location	Activity
A	105201 (13)	17055 (5)	18	31	39
B	242845 (10)	91316 (4)	18	28	30
C	144162 (10)	61029 (4)	10	24	65

Table 3: Top-5 classes in each label of the lifelog dataset.

Location	Sub-location	Activity
office (196839)	office-room (182884)	working (204131)
university (147045)	classroom (101844)	commuting (102034)
outside (130754)	home-room (86588)	studying (90330)
home (97180)	subway (35204)	eating (60725)
restaurant (22190)	bus (34120)	watching (35387)

Two kinds of neural networks are used to extract the representation in this experiment. One is AlexNet, a prototype network trained by ImageNet [Krizhevsky *et al.*, 2012]. The other is referred to as LifeNet, which is a network trained with the lifelog dataset. The structure of LifeNet is similar to AlexNet, but the number of nodes of LifeNet is half of that of AlexNet. The MatConvNet toolbox is used for both AlexNet and LifeNet. We chose a 1000-dimensional softmax output vector of AlexNet for representation of online deep learning algorithms, as we assume the probability of an object's appearance in each scene is highly related to the daily activity represented by each scene.

The performances on the lifelog dataset were evaluated in a 10-split experiment. Each online dataset corresponds to each day for subjects B and C. However, for subject A, the 13 days of training data was changed into 10 online dataset by merging 3 of the days into its next days. Each online dataset is referred to as a day. LifeNets made from 3 groups of online lifelog datasets, with sets of consecutive 3, 4 and 3 days for each group. In the entire learning of LifeNet, the learning rate is set to 0.0025, the rate of weight decay is 5×10^{-4}, and the rate of momentum is 0.9. In this experiment, LifeNet is used for online fine-tuning and incremental bagging, AlexNet for last-layer fine-tuning and both the LifeNet and AlexNet are used for DMA[1].

Figure 6 shows the experimental results from the lifelog dataset. The experiments consist of three subjects whose tasks are classified into three categories. A total of nine experiments are performed and their averaged test accuracies from a range of learning algorithms are plotted. In some experiments, the performance of the algorithms at times decreases with the incoming new stream of data, which is natural while learning a non-stationary data stream. This would occur in

[1]In DMA, LifeNet corresponds to the 'Deep Net 1' and AlexNet corresponds to the 'Deep Net 2' and 'Deep Net k' in Figure 2.

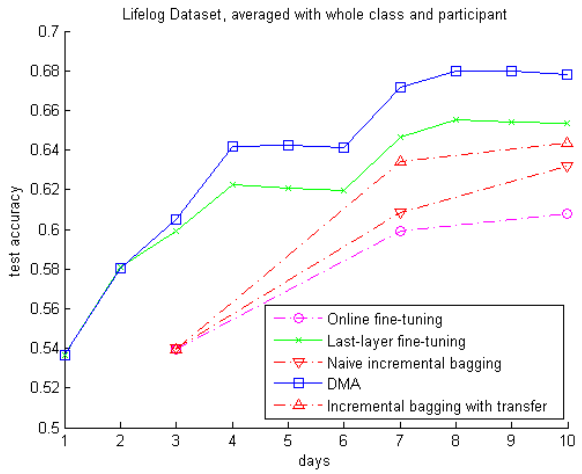

Figure 6: Averaged test accuracy of various learning algorithms on the lifelog dataset. The location, sub-location, and activity are classified separately for each of the three subjects.

situations where the test data is more similar to the training data encountered earlier than later during the learning process. Although, such fluctuations can occur, on average, however, the accuracies of the algorithms increase steadily with the incoming stream of data. In comparison among online deep learning algorithms, last-layer fine-tuning that uses one AlexNet outperforms other online deep learning algorithms that use many LifeNets. However, these learning algorithms perform worse than DMA that uses numerous LifeNets and one AlexNet. Table 4 shows accuracies by each class, and by each subject.

5 Discussion

Performances of online deep learning algorithms are more analyzed and discussed in the chapter for justifying our proposed method. The model with only one CNN does not adapt to extreme non-stationary data streams in the experiment on CIFAR-10. In the last-layer fine-tuning, a CNN trained by the first online dataset was used. So, the model has a deep representation only for discriminating three classes of image objects. Hence, the performance does not increase. In the case of online fine-tuning, the model loses the information about previously seen online datasets. This reduces performance of test accuracy as time progresses.

In the experiment on the lifelog dataset, however, last-layer fine-tuning that uses one AlexNet outperforms other online deep learning algorithms that use many LifeNets. This implies that usage of pre-trained deep networks by a large corpus dataset is effective on the lifelog dataset. From the perspective of personalization, a representation obtained by existing users or other large dataset can be used together with a representation obtained by a new user. However, DMA that uses both AlexNet and LifeNet works better than last-layer fine-tuning, which implies again that using multiple networks is necessary in online deep learning task.

In all the experiments, incremental bagging increases its performance continuously with non-stationary data streams. Incremental bagging that uses many networks outperforms online fine-tuning that uses only one deep network. However, the model does not reach the performance of batch learner, as part of the entire data is not sufficient for learning discriminative representations for the whole class. In the experiment, weight transfer alleviates this problem; the technique decreases error for both DMA and incremental bagging, respectively.

Table 4: Classification accuracies on the lifelog dataset among different classes (top) and different subjects (bottom)

Algorithm	Location	Sub-location	Activity
DMA	**78.11**	**72.36**	**52.92**
Online fine-tuning	68.27	64.13	50.00
Last-layer fine-tuning	74.58	69.30	52.22
Naïve incremental bagging	74.48	67.18	47.92
Incremental bagging w/ transfer	74.95	68.53	49.66
DMA w/ last-layer retraining	78.66	73.23	52.99

Algorithm	A	B	C
DMA	**67.02**	**58.80**	**77.57**
Online fine-tuning	53.01	56.54	72.85
Last-layer fine-tuning	63.31	55.83	76.97
Naïve incremental bagging	62.24	53.57	73.77
Incremental bagging w/ transfer	61.21	56.71	75.23
DMA w/ last-layer retraining	68.07	58.80	78.01

The proposed DMA outperforms incremental bagging consistently. In other words, learning a shallow network and deep networks concurrently is advantageous compared to simply averaging softmax output probability of each CNN. By the way, learning fast memory in the DMA is not trivial. In DMA w/ [Liu *et al.*, 2008] of Figure 5, mGHNs are trained by a discriminative maximum likelihood solution suggested by [Liu et al., 2008]. Their performances are getting worse due to the continuous arrival of extreme non-stationary data. A generative approach, in the online learning of mGHNs, is one of te key points of successful online learning in the paper.

It is worth noting that the performance gap between our algorithms and other algorithms can significantly change for different datasets. If data streams are stationary and in abundance, then incremental bagging can perform better than DMA. The relationship between the performance of online deep learning algorithms and the properties of data streams will be analyzed and described in future work.

6 Conclusion

In this paper, a dual memory architecture is presented for real-time lifelong learning of user behavior in daily life with a wearable device. The proposed architecture represents mutually grounded visio-auditory concepts by building shallow kernel networks on numerous deep neural networks. Online deep learning has useful properties from the perspective of lifelong learning because deep neural networks show high performance in transfer and multitask learning [Heigold *et al.*, 2013; Yosinski *et al.*, 2014], which will be further explored in our future works.

Acknowledgments

This work was supported by the Naver Corp. and partly by the Korea Government (IITP-R0126-16-1072-SW.StarLab, KEIT-10060086-HRI.MESSI, KEIT-10044009-RISF).

References

[Chen *et al.*, 2015] Tianqi Chen, Ian Goodfellow, and Jonathon Shlens. Net2net: Accelerating learning via knowledge transfer. *arXiv preprint arXiv:1511.05641*, 2015.

[Donahue *et al.*, 2014] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proeedings of the 31th International Conference on Machine Learning*, pages 647–655, 2014.

[Finch, 2009] Tony Finch. Incremental calculation of weighted mean and variance. *University of Cambridge*, 2009.

[Goodfellow *et al.*, 2013] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[Ha *et al.*, 2015] Jung-Woo Ha, Kyung-Min Kim, and Byoung-Tak Zhang. Automated construction of visual-linguistic knowledge via concept learning from cartoon videos. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence*, pages 522–528, 2015.

[Heigold *et al.*, 2013] Georg Heigold, Vincent Vanhoucke, Alan Senior, Patrick Nguyen, Marc'Aurelio Ranzato, Matthieu Devin, and Jeffrey Dean. Multilingual acoustic models using distributed deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8619–8623, 2013.

[Hong *et al.*, 2015] Seunghoon Hong, Tackgeun You, Suha Kwak, and Bohyung Han. Online tracking by learning discriminative saliency map with convolutional neural network. In *Proeedings of the 32th International Conference on Machine Learning*, pages 597–606, 2015.

[Huynh *et al.*, 2008] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 10–19, 2008.

[Krizhevsky *et al.*, 2012] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[Lin *et al.*, 2014] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *International Conference on Learning Representations*, 2014.

[Little, 1992] Roderick JA Little. Regression with missing x's: a review. *Journal of the American Statistical Association*, 87(420):1227–1237, 1992.

[Liu *et al.*, 2008] Xinwang Liu, Guomin Zhang, Yubin Zhan, and En Zhu. An incremental feature learning algorithm based on least square support vector machine. In *Proceedings of the 2nd annual international workshop on Frontiers in Algorithmics*, pages 330–338, 2008.

[Nam and Han, 2015] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *arXiv preprint arXiv:1510.07945*, 2015.

[Oza, 2005] Nikunj C Oza. Online bagging and boosting. In *Systems, Man and Cybernetics, IEEE International Conference On*, pages 2340–2345, 2005.

[Polikar *et al.*, 2001] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Transactions on Systems, Man, and Cybernetics Part C: Applications and Reviews*, 31(4):497–508, 2001.

[Ruvolo and Eaton, 2013] Paul L Ruvolo and Eric Eaton. Ella: An efficient lifelong learning algorithm. In *Proeedings of the 30th International Conference on Machine Learning*, pages 507–515, 2013.

[Thrun and O'Sullivan, 1996] Sebastian Thrun and Joseph O'Sullivan. Discovering structure in multiple learning tasks: The tc algorithm. In *Proeedings of the 13th International Conference on Machine Learning*, pages 489–497, 1996.

[Vedaldi and Lenc, 2015] Andrea Vedaldi and Karel Lenc. Matconvnet – convolutional neural networks for matlab. In *Proceedings of the ACM International Conference on Multimedia*, pages 689–692, 2015.

[Yosinski *et al.*, 2014] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

[Zeiler and Fergus, 2014] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Proceedings of European Conference on Computer Vision*, pages 818–833, 2014.

[Zhang *et al.*, 2012] Byoung-Tak Zhang, Jung-Woo Ha, and Myunggu Kang. Sparse population code models of word learning in concept drift. In *Proceedings of the 34th Annual Conference of Cogitive Science Society*, pages 1221–1226, 2012.

[Zhang, 2013] Byoung-Tak Zhang. Information-theoretic objective functions for lifelong learning. In *AAAI Spring Symposium on Lifelong Machine Learning*, 2013.

[Zhou *et al.*, 2012] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1461, 2012.

[Zinkevich, 2003] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proeedings of the 20th International Conference on Machine Learning*, pages 928–936, 2003.