Ph.D. DISSERTATION

Lifelong Learning of Everyday Human Behaviors using Deep Neural Networks: Dual Memory Architecture and Incremental Moment Matching

깊은 신경망 기반 일상 행동에 대한 평생 학습: 듀얼 메모리 아키텍쳐와 점진적 모멘트 매칭

BY

Sang-Woo Lee

AUGUST 2018

Department of Electrical Engineering & Computer Science
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Ph.D. DISSERTATION

Lifelong Learning of Everyday Human Behaviors using Deep Neural Networks: Dual Memory Architecture and Incremental Moment Matching

깊은 신경망 기반 일상 행동에 대한 평생 학습: 듀얼 메모리 아키텍쳐와 점진적 모멘트 매칭

BY

Sang-Woo Lee

AUGUST 2018

Department of Electrical Engineering & Computer Science
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Lifelong Learning of Everyday Human Behaviors using Deep Neural Networks: Dual Memory Architecture and Incremental Moment Matching

깊은 신경망 기반 일상 행동에 대한 평생 학습: 듀얼 메모리 아키텍쳐와 점진적 모멘트 매칭

지도교수 장 병 탁

이 논문을 공학박사 학위논문으로 제출함

2018년 04월

서울대학교 대학원

전기.컴퓨터공학부

이 상 우

이 상 우 의 공학박사 학위논문을 인준함

2018년 06월

위 원 장	최 진 영
부위원장	장 병 탁
위 원	윤 성 로
위 원	김 건 희
위 원	하 정 우

Abstract

Learning from human behaviors in the real world is imperative for building human-aware intelligent systems. We attempt to train a personalized context recognizer continuously in a wearable device by rapidly adapting deep neural networks from sensor data streams of user behaviors. However, training deep neural networks from the data stream is challenging because learning new data through neural networks often results in loss of previously acquired information, referred to as catastrophic forgetting. This catastrophic forgetting problem has been studied for nearly three decades but has not been solved yet because the mechanism of deep learning has been not understood enough.

We introduce two methods to deal with the catastrophic forgetting problem in deep neural networks. The first method is motivated by the concept of complementary learning systems (CLS) theory - contending that effective learning of the data stream in a lifetime requires complementary systems that comprise the neocortex and hippocampus in the human brain. We propose a dual memory architecture (DMA), which trains two learning structures: one gradually acquires structured knowledge representations, and the other rapidly learns the specifics of individual experiences. The ability of online learning is achieved by new techniques, such as weight transfer for the new deep module and hypernetworks for fast adaptation.

The second method is incremental moment matching (IMM) algorithm. IMM incrementally matches the moment of the posterior distribution of neural networks, which is trained for the previous and the current task, respectively. To make the search space of posterior parameter smooth, the IMM procedure is

complemented by various transfer learning techniques including weight transfer, L2-norm of the old and the new parameter, and a variant of dropout with the old parameter.

To provide an insight into the success of two proposed lifelong learning methods, we introduce an insight by introducing two online learning methods of sum-product network, which is a kind of deep probabilistic graphical model. We discuss online learning approaches which are valid in probabilistic models and explain how these approaches can be extended to the lifelong learning algorithms of deep neural networks.

We evaluate proposed DMA and IMM on two types of datasets: the various artificial benchmarks devised for evaluating the performance of lifelong learning and the lifelog dataset collected through the Google Glass for 46 days. The experimental results show that our methods outperform comparative models in various experimental settings and that our trials for overcoming catastrophic forgetting are valuable and promising.

Keywords: Lifelong learning, Lifelog dataset, Sum-product networks, Deep neural networks, Dual memory architecture, Complementary learning systems, Incremental moment matching, Sequential Bayesian

Student Number: 2012-20835

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Wearable Devices and Lifelog Dataset

It is important to learn from human behavior in the real world for building human-aware intelligent systems in personalized digital assistants. Recently, various types of wearable devices, including smart watches and Google Glass, have gained considerable attention. It is noticeable that these devices can see and hear what the device user sees and hears; this property differentiates them from the classical agents in personal computers or the smartphones. To simulate this environment, we collected a lifelog dataset through the Google Glass over 46 days from three participants. This dataset has two properties. First, high-level contexts are hidden in the raw-level data stream, for example, an egocentric video recorded during a meeting includes various types of high-level contexts, although the data is only a stream of pixels and audio signals. Second, the data streams are often non-stationary, for example, the life patterns of a student during vacations and semesters are different. It is in our interest

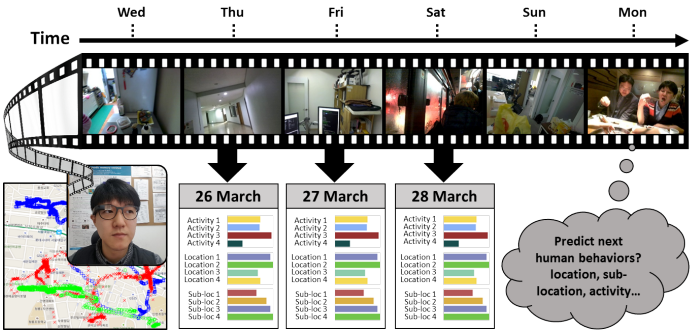

Figure 1.1 The Lifelong learning framework discovered in this work

to continually adapt the context-aware activity recognizer rapidly from human behaviors gathered through wearable devices. To treat these properties, two algorithmic techniques are required. First, the deep learning method is necessary to manage raw-level data efficiently. Second, an online learning algorithm is required to keep track of fast-changing life patterns of user behavior.

1.2 Lifelong Learning and Catastrophic Forgetting

Deep learning algorithms, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs), deliver state-of-the-art performances for various fields including computer vision (He et al., 2015; Noh et al., 2015), speech recognition (Graves et al., 2013; Sainath et al., 2015), and natural language processing (Sundermeyer et al., 2012; Sutskever et al., 2014), implying that representation learning with DNNs is essential for various context-aware problems. Further, this type of deep learning technique is applicable for context-aware tasks in egocentric videos (Doshi et al., 2015; Bettadapura et al., 2015; Simonyan and Zisserman, 2014; Yu et al., 2015).

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). To handle such non-stationary data streams, it is important to learn deep representations in an online manner.

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 (McCloskey and Cohen, 1989; French, 1999). Assume that you trained a neural network with the first training data as user data of the first week. Subsequently, the second training data as user data of the second week becomes available. You can train the neural network with the second training data; however, the information of the first training data can be lost, especially when the data stream is non-stationary. In short, if new data becomes available, the neural network often forgets the old data. The catastrophic forgetting problem has resurfaced with the renaissance of deep learning research (Goodfellow et al., 2013; Srivastava et al., 2013; Li and Hoiem, 2016; Kirkpatrick et al., 2017).

1.3 Approach and Contribution

We introduce two approaches to deal with this catastrophic forgetting problem. The first method is dual memory architecture (DMA) which processes slow-changing global patterns as well as keeps track of fast-changing local behaviors over a lifetime (Lee et al., 2016, 2017c). The second method is incremental

moment matching (IMM) which merges networks trained for different datasets or tasks (Lee et al., 2017b).

In our first approach, the DMA trains two memory structures: one is an ensemble of DNNs, and the other consists of a shallow network that uses hidden representations of the DNNs as input. These two memory structures are designed to use different strategies. The ensemble of DNNs learns new information to adapt its representation to new data, whereas the shallow network aims to manage non-stationary distribution and unseen classes more rapidly.

Moreover, some techniques for online deep learning are proposed in this study. First, the transfer learning method via weight transfer is applied to maximize the representation power of each neural module in online deep learning (Yosinski et al., 2014). Second, the multiplicative Gaussian hypernetwork (mGHN) and its online learning method are developed. 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. The mGHN possesses two good properties for online learning. First, it can learn from every new one instance rapidly, even if the instance is from a new class. Second, it can handle incremental input features, which property is essential when a new DNN is constructed, and new input features appear in the fast memory.

In our second approach, on the other hand, IMM uses the framework of Bayesian neural networks, which implies that uncertainty is introduced on the parameters in neural networks, and that the posterior distribution is calculated (MacKay, 1992; Blundell et al., 2015). The dimension of the random variable in the posterior distribution is the number of the parameters in the neural networks. IMM approximates the mixture of Gaussian posterior with each component representing parameters for a single task to one Gaussian distribution for

a combined task. To merge the posteriors, we introduce two novel methods of moment matching. One is *mean-IMM*, which simply averages the parameters of two networks for old and new tasks as the minimization of KL-divergence between the approximated posterior distribution and the mixture of two Gaussian posteriors. The other is *mode-IMM*, which merges the parameters of two networks using a Laplacian approximation (MacKay, 1992) to approximate a mode of the mixture of two Gaussian posteriors, which represent the parameters of the two networks.

In general, it is too naïve to assume that the final posterior distribution for the whole task is Gaussian. To make our IMM work, the search space of the loss function between the posterior means needs to be smooth and convex-like. In other words, there should not be high cost barriers between the means of the two networks for an old and a new task. To make our assumption of Gaussian distribution for neural network reasonable, we applied three main transfer learning techniques on the IMM procedure: weight transfer, L2-norm of the old and the new parameters, and our newly proposed variant of dropout using the old parameters.

To explain the success of the proposed lifelong learning methods of deep neural networks in the dissertation, we provide insight by introducing two online learning methods of sum-product networks (SPNs), a kind of deep probabilistic graphical models (PGMs). The relation between deep PGMs and deep neural networks have long been studied. For example, deep neural networks can be understood as a discriminative version of deep belief networks or deep Boltzmann machine. Though the original SPNs is a generative model, there also is a discriminative version of SPNs which can also be used for classification tasks. We propose two methods for the online learning of SPNs. One method, 'online incremental structure learning of sum-product networks' (OISSPN), continuously

extends the structure of the SPN to estimate the probability density function (Lee et al., 2013). The other method, 'non-parametric Bayesian sum-product networks' (NPBSPN), continuously estimates the probability of the SPN by sequential Bayesian (Lee et al., 2014). These two methods for SPN can correspond to DMA, which use dual structure learning algorithms, and IMM, which use sequential Bayesian and Bayesian neural networks to estimate the probability distribution of the whole dataset and task.

1.4 Organization of the Dissertation

The remained part of this dissertation is organized as follows.

Chapter 2 describes the related works of lifelong learning. First, we introduce various fields related to the lifelong learning research. Then, we review two approaches for solving lifelong learning, the ensemble approach and the single network learning approach.

Chapter 3 introduces two online learning methods of sum-product networks (SPNs), a kind of deep probabilistic graphical models (PGM). One method is OISSPN, which continuously extends the structure of the SPN to estimate the probability density function. The other method is NPBSPN, which continuously estimates the probability of the SPN by sequential Bayesian. The discussion section explains the follow-up study after our studies on online learning of SPNs and argues connections between these SPN studies and the lifelong learning of deep neural networks.

Chapter 4 introduces dual memory architecture (DMA) for tackling the lifelong learning problem with structure learning of the dual deep-memory system. First, we explain complementary learning systems (CLS) theory which motivates our proposed model. Then, we introduce the general concept of DMA and

its multiplicative-Gaussian Hypernetworks (mGHNs). After that, we present the experimental results for the lifelong learning of DNNs. Finally, we discuss the potential contribution of DMA.

Chapter 5 introduces incremental moment matching (IMM) for tackling the lifelong learning problem with sequential Bayesian. First, we propose mean-IMM and mode-IMM, the method of merging many networks in the perspective of Bayesian neural network and Gaussian posterior assumption. Then, we explain various transfer techniques that can be applied to enhance the performance of mean-IMM and mode-IMM. Specifically, we propose drop-transfer, a novel transfer method devised in the paper. After that, we show that proposed IMM makes state-of-the-art performance in a variety of datasets of lifelong learning. Lastly, we discuss geometrical properties in IMM and a Bayesian perspective of modern deep neural networks.

Finally, we summarize the dissertation and its contributions in Chapter 6.

Chapter 2

Related Works

2.1 Lifelong Learning

Lifelong learning refers to the learning of multiple consecutive tasks with never-ending exploration and continuous discovery of knowledge from data streams. This task also has been referred to as lifelong learning, but in this case, the directing point of preventing catastrophic forgetting along with minimizing to access previous data is emphasized. Lifelong learning 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). Some studies have proposed methods that maintaining a sparsely shared basis of the shallow network for all task models (Ruvolo and Eaton, 2013). However, these approaches cannot learn deep shared representation over all tasks, which degrades the performances of these models. Online deep learning has useful properties from the perspective of lifelong learning because deep neural networks show high performance in transfer and multi-task learning (Heigold

et al., 2013; Yosinski et al., 2014).

There are a variety of categories consisting of or related to lifelong learning. Among these categories of lifelong learning, our primary interest is preventing catastrophic forgetting. In other words, we focus on how the model does not forget the information of the previous tasks while it is only accessible to the new task, not previous tasks, for learning.

2.2 Application-driven Lifelong Learning

Various studies have been conducted on online learning from data stream during an entire lifetime. Carlson et al. (Carlson et al., 2010; Mitchell et al., 2015) proposed the Never-Ending Language Learner (NELL), which extracts a variety of information from the web and constructs a structured knowledge base. Chen et al. (Chen et al., 2013) extended the NELL to develop the Never-Ending Image Learner (NEIL), which extracts both language and visual information from the web. Ha et al. (Ha et al., 2015) presented a model of deep concept hierarchy that enables progressive abstraction of concept knowledge at multiple levels in an online manner. They tested their algorithm on several hundred episodes of cartoon videos and showed that it can build vision-language concept hierarchies from non-stationary data streams. In our study, we attempt to extract abstracted concepts from continuous sensing data of wearable devices or video streams.

2.3 Classical Approach for Preventing Catastrophic Forgetting

In this section, we categorize the classical approach to training deep networks in an online manner from data streams into three categories. The first approach

is *online fine-tuning*, which is online learning of an entire neural network based on typical stochastic gradient descent (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. Because of the difficulty in optimizing the extremely large non-convex search space of a neural network, limited theoretical studies exist on online learning of DNNs (Bottou, 1998). Recently, in (Nam and Han, 2016), online fine-tuning of a CNN with simple online SGD was used 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:

$$y = softmax(f(h^{\{1\}}(x))) \tag{2.1}$$

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.

Other studies use implicit distributed storage of information in SGD learning. These methods use the idea of dropout, maxout, or neural module for distributively storing the information for each task by making use of the large capacity of the neural network (Srivastava et al., 2013). Unfortunately, most studies following this approach had limited success and failed to preserve performance on the old task when an extreme change to the environment occurred (Goodfellow et al., 2013).

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:

$$y = \delta(w^T \phi(h^{\{1\}}(x))). \tag{2.2}$$

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:

$$y = \frac{1}{d} \sum_{i}^{d} softmax(f_d(h^{\{d\}}(x)). \tag{2.3}$$

Several studies have adopted this incremental bagging method, whereby a weak learner is made to use the online dataset, and these multiple weak learners are combined to obtain a better predictive performance (Polikar et al., 2001). Unfortunately, in our experiment, a simple voting method with weak learners trained from a relatively small online dataset was unsuccessful; we presumed that a relatively smaller online dataset is insufficient for learning highly expressive representations of DNNs.

Recent researcher for preventing catastrophic forgetting also have studied on extending the structures of neural networks (Lee et al., 2017a). In the study of progressive network (Rusu et al., 2016), when a new task arrives, the algorithm makes a new network but also shares the representation of old networks between the tasks. However, this method still has a complexity issue, especially in inference, because the number of networks increases with the number of new tasks that need to be learned.

Fernando et al. (Fernando et al., 2017) proposed PathNet, which extends the idea of the ensemble approach for parameter reuse (Rusu et al., 2016) within a single network. In PathNet, a neural network has ten or twenty modules in each layer, and three or four modules are picked for one task in each layer by an evolutionary approach. This method alleviates the complexity issue of the ensemble approach to lifelong learning in a plausible way.

2.4 Learning Parameter Distribution for for Preventing Catastrophic Forgetting

2.4.1 Sequential Bayesian

Sequential Bayesian. It has long been studied updating the parameters not only for shallow networks but also complex hierarchical models by sequential

Bayesian estimation (Ghahramani, 2000). Sequential Bayes was also used to learn topic models from stream data in Broderick et al. (Broderick et al., 2013).

Similar to our method, Bayesian moment matching is used for sum-product networks, a kind of deep hierarchical probabilistic model (Rashwan et al., 2016). Though sum-product networks are usually not scalable to large datasets, their online learning method is useful, and it achieves similar performance to the batch learner. Our method using moment matching focuses on lifelong learning and deals with significantly different statistics between tasks, unlike the previous method.

Bayesian Neural Networks. Bayesian neural networks (BNN) assume an uncertainty for the whole parameter in neural networks so that the posterior distribution can be obtained (Blundell et al., 2015). Previous studies have argued that BNN regularizes better than NN, and provides a confidence interval for the output estimation of each input instance. Current research on BNN, to the best of our knowledge, uses Gaussian distributions as the posteriors of the parameters. In the Gaussian assumption, because tracking the entire information of a covariance matrix is too expensive, researchers usually use only the diagonal term for the covariance matrix, in which case the posterior distribution is fully factorized for each parameter. However, the methods using full covariance was also suggested recently (Louizos and Welling, 2016). For estimating a covariance matrix most studies use stochastic gradient variational Bayes (SGVB), where a sampled point from the posterior distribution by Monte Carlo is used in the training phases (Kingma and Welling, 2013). Alternatively, Kirkpatrick et al. (Kirkpatrick et al., 2017) approximated the covariance matrix as an inverse of a Fisher matrix. This approximation makes the computational cost of the inference of a covariance matrix smaller when the update of covariance information is not needed in the training phase. Our method follows the

approach using the Fisher matrix.

2.4.2 Approach to Simulating Parameter Distribution

Recently, the concept of applying a regularization function to a network trained by the old task to learning a new task has received much attention. This approach can be interpreted as an approximation of sequential Bayesian. Representative examples of this regularization approach include elastic weight consolidation (EWC) (Kirkpatrick et al., 2017). EWC succeeded in some experiments where their own assumption of the regularization function fits the problem.

In EWC, the posterior distribution trained by the previous task is used to update the new prior distribution. This new prior is used for learning the new posterior distribution of the new task in a Bayesian manner. EWC assumes that the covariance matrix of the posterior is diagonal and there are no correlations between the nodes. Though this assumption is tough, EWC performs well in some domains. EWC maximizes the following terms by gradient descent to get the solution $\mu_{1:K}$.

$$\begin{aligned}
\log p_{1:K} &\approx \log p(y_K|X_K,\theta) + \lambda \cdot \log p_{1:(K-1)} + C \\
&\approx \log p(y_K|X_K,\theta) + \lambda \cdot \sum_{k=1}^{K-1} \log q_{1:k} + C \\
&= \log p(y_K|X_K,\theta) - \frac{\lambda}{2} \cdot \sum_{k=1}^{K-1} (\theta - \mu_{1:k})^T \tilde{\Sigma}_k^{-1} (\theta - \mu_{1:k}) + C'
\end{aligned} \tag{2.4}$$

p_k is empirical posterior distribution of kth task, and $q_k \sim N(\mu_k, \Sigma_k)$ is an approximation of p_k. In EWC, $\tilde{\Sigma}_k^{-1}$ is also approximated by the diagonal term of Fisher matrix $\tilde{F}_k$ with respect to $\mu_{1:k}$ and X_k.

When moving to a third task, EWC uses the penalty term of both first and

second network (i.e., μ_1 and $\mu_{1:2}$). Although this heuristic works reasonably in the experiments in their paper, it does not match to the philosophy of Bayesian.

2.5 Learning Data Distribution for Preventing Catastrophic Forgetting

The approach with simulating data distribution also have received much attention. Learning without forgetting (LwF) is one example of this approach, which uses the pseudo-training data from the old task (Li and Hoiem, 2016). Before learning the new task, LwF puts the training data of the new task into the old network, and uses the output as pseudo-labels of the pseudo-training data. By optimizing both the pseudo-training data of the old task and the real data of the new task, LwF attempts to prevent catastrophic forgetting. This framework is promising where the properties of the pseudo training set is similar to the ideal training set. Although LwF does not explicitly assume Bayesian, the approach can be represented nonetheless as follows:

$$\log p_{1:K} \approx \log p(y_K|X_K, \theta) + \lambda \cdot \sum_{k=1}^{K=1} \log p(\hat{y}_k|X_K, \theta) \tag{2.5}$$

Where $\hat{y_k}$ is the output from μ_k with input X_K. This framework is promising where the properties of a pseudo training set of kth task (X_K, $\hat{y}_k$) is similar to the ideal training set (X_k, y_k).

Many researchers recently believe that generative models can help prevent catastrophic forgetting. Shin et al. use generative adversarial networks to learn data distribution of the whole previous tasks to prevent catastrophic forgetting (Shin et al., 2017). It is also noticeable that their pair of generative data distri-

bution learner and discriminative classifier was also motivated by the concept of generative replay in complementary learning systems theory (McClelland et al., 1995).

Chapter 3

Preliminary Study: Online Learning of Sum-Product Networks

3.1 Introduction

As learning the structure of graphical models is one of the most important issues in machine learning fields, many researchers have contributed to its study. Noteworthy examples of these research studies are Bayesian networks (Chickering et al., 1997), Markov networks (Ravikumar et al., 2010), deep networks (Bengio et al., 2009), and sum–product networks (SPNs) (Gens and Pedro, 2013). However, studies on online learning, especially for complex models, are limited because it is not easy to change the form of probability tables without information about forgotten training data. Nevertheless, online learning is an essential problem in machine learning, and there are some learning environments in which it should be applied, such as large-scale data learning or lifelong learning.

In a successful study, a denoising autoencoder was learned using online incremental structure learning (Zhou et al., 2012). It has been verified that this model learns the changing probability distribution of data. In order to solve the online learning problem using enough representation power, we focus on the SPN, deep architectures that can learn and infer at low computational costs (Poon and Domingos, 2011). In early research on SPNs (Gens and Pedro, 2013), Gens and Domingos used hierarchical biclustering to learn the structure of SPNs well. The framework that they used is, however, not suitable for online learning because entire data should be used in the structure learning steps.

We introduce two approaches to make an incremental model of SPNs. In the first approach, we proposed the 'online incremental structure learning of sum-product networks' (OISSPN), online structure learning algorithm that used an online clustering algorithm. We note that structure learning of SPNs is highly dependent on clustering instances. We devised a simple mini-batch clustering algorithm which can modify the number of clusters dynamically on incoming data, and apply it to incremental structure learning. In other words, our online learning of SPNs is formulated as an online clustering problem, in which a local assigning instance corresponds to modifying the tree-structure of the SPN incrementally. In the method, the number of hidden units and even layers are evolved dynamically on incoming data. The experiments show that it outperforms the online version of the previous method (Gens and Pedro, 2013), and achieves the performance of batch structure learning.

In the second approach, we define two models for 'non-parametric Bayesian sum-product networks' (NPBSPN), which train SPNs in an online manner using non-parametric Bayesian. The first is a tree structure of Dirichlet Processes; the second is a dag of hierarchical Dirichlet Processes. These generative models for data implicitly define a prior distribution on SPN of tree and of dag structure.

They allow MCMC fitting of data to SPN models, and the learning of SPN structure from data. It can be also understood that NPBSPN also utilizes an online clustering algorithm, but in a non-parametric Bayesian manner.

3.2 Sum-Product Networks

3.2.1 Representation of Sum-Product Networks

SPNs are one of probabilistic graphical models (PGMs) which have specialized struc-ture for fast inference. They are constrained to have tree structures (rooted directed acyclic graphs), and their leaves represent univariate distribution such as multinomial distribution for discrete variables, Gaussian, Poisson and other continuous distribu-tions. Internal nodes in the tree represent products or sums of their children with the corresponding weights as the Figure 3.1. Recursive definition of SPNs introduced in (Gens and Pedro, 2013) is as follows.

Definition 1 An SPN is defined as follows:

1. A tractable univariate distribution is an SPN.
2. A product of SPNs with disjoint scopes (i.e. variable sets) is an SPN.
3. A weighted sum of SPNs with the same scope is an SPN, provided that all weights are positive and sum to 1.
4. Nothing else is an SPN.

First, let us define the probability distribution corresponding to an SPN. Let there be D variables $X_1, \ldots, X_D$; the SPN specifies a joint probability distribution over these variables. We assume throughout that $D \geq 2$. Each node in a SPN defines a joint probability distribution over a subset of the variables; we refer to this subset of the variables as the *scope* of the sum or product node.

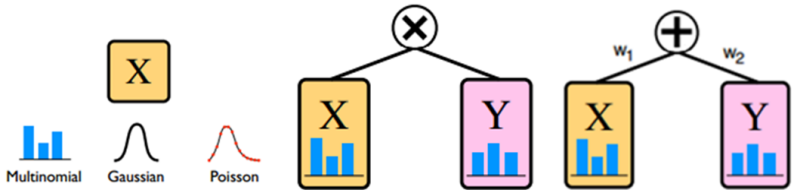

Figure 3.1 Types of nodes in SPN: Univariate node (left); Product node (middle); Sum node (right)

We use s to denote a scope as a subset of indices; in other words a scope with k indices is $s = \{s_1, \ldots, s_k\} \subseteq \{1, \ldots, D\}$. The indexed set of the corresponding variables is written $X_s = (X_{s_1}, \ldots, X_{s_k})$. The number of elements in the scope of a node is termed the *level* of the node. In this paper we consider only binary data. Each node of level 1 (a univariate node) defines a Bernoulli distribution for the (binary) variable that is its scope. Univariate nodes have no children. We require that each product node and each sum node must have level at least 2.[1] We write P_s and S_s to denote a product or sum node respectively with scope s, where $s \subseteq \{1, \ldots, D\}$, and write children$(P_s)$ and children$(S - s)$ for the (indexed) sets of children of product and sum nodes.

A **product-node** defines the p.d. over its scope that is the product of the p.d.s of its children, which are sum-nodes or univariate nodes, with disjoint scopes. Each product node P defines a partition of its scope; we require this partition to have at least two elements. For a product node P of level k, let

$$\text{partitition}(\mathrm{P}) = (s_1, \ldots, s_k)$$
$$\text{children}(\mathrm{P}_s) = (S_{s_1}, \ldots, S_{s_k})$$

[1]For non-binary variables – for example, for real variables – it could be meaningful to have sum-nodes of level one, which would represent mixtures of univariate distributions, but this is not useful for binary data considered here.

where the s_i are non-empty, disjoint, and $s_1 \cup \cdots \cup s_k = \{1, \ldots, D\}$,and the children $(\mathrm{S}_{s_1}, \ldots, \mathrm{S}_{s_k})$ are sum nodes or univariate nodes. If $|s_i|$, which is level(S_{s_i}), is equal to 1, then S_{s_i} is a univariate distribution node, otherwise S_{s_i} is a sum-node. Each product node P only has |partitition(P)| child nodes.

The probability distribution denoted by a product note P is denoted $P(\cdot)$; and the p.d. of a sum-node or univariate node S as $S(\cdot)$. Then product node P_s with children $(S_1, \ldots, S_k)$ defines a joint p.d. over X_s, which is

$$P_s(X_s) = \prod_{i=1}^{k} S(X_{s_i})$$

Note that a product node always has D or fewer children, because the scopes of its children are required to be disjoint.

A **sum-node** is either a univariate node, or else it defines a mixture distribution over its children, which are product-nodes, all with the same scope. In our model, each sum-node S_s may have a countable infinity of children $(P_s^1, P_s^2, P_s^3, \ldots)$ For each child there is a corresponding non-negative weight $w_1, w_2, w_3, \ldots$, such that $\sum_{i=1}^{\infty} w_i = 1$. The distribution defined S_s is:

$$S_s(X_s) = \sum_{i=1}^{\infty} w_i P_s^i(X_s)$$

The top node of the SPN is the sum-node $S_{\{1,\ldots,D\}}$, which we also denote S_{top}.

Many previous works related to SPNs, including sigma-pi neural networks (Zhang and Muhlenbein, 1994), arithmetic circuits (Darwiche, 2003), AND-OR graphs (Dechter and Mateescu, 2007), and other compact representations exist. However, SPNs are a more general probabilistic model that enjoys enough representation power. Not only is the inference and learning process of SPNs simple and fast, but they also perform well in terms of representing the probability distribution of data. SPNs show a remarkable performance in image

classification tasks (Gens and Domingos, 2012) and video learning (Amer and Todorovic, 2012).

3.2.2 Structure Learning of Sum-Product Networks

In the seminal work of SPNs (Poon and Domingos, 2011), the structure of SPNs was built in the application-oriented manner. After that, structure learning for SPNs has been introduced in (Gens and Pedro, 2013; Dennis and Ventura, 2012). Dennis & Ventura (Dennis and Ventura, 2012) collected variables using regional relationship to find better structure to solve the image completion task. Gens & Domingos (Gens and Pedro, 2013) firstly proposed batch-type structure learning method by splitting the variables into mutually independent subset toward compact representation of joint distribution minimizing the representation power loss.

As a brief introduction, structure learning in (Gens and Pedro, 2013) is a recursive procedure given dataset T and set of variables V. At first, it checks whether the variables can be split into mutually independent subsets. If possible, the split recursions are done for each sub-set, and return the products of the resulting SPNs (building internal product nodes). Otherwise, the instances T are clustered into similar subsets, and it returns the weighted sum of the resulting SPNs (building internal sum nodes). Weights for child nodes are determined to be proportional to the number of the assigned instances. At the end of the recursive process, all leaves consist of a univariable node.

At the clustering process, they use a naïve Bayes mixture model to pick most likely component in the mixture, where all variables are independent conditioned on the cluster.

$$P(V) = \sum_i P(C_i) \prod_j P(V_j|C_i) \tag{3.1}$$

where C_i is the i-th cluster and V_j is the j-th variable. They use a hard expectation-maximization (hard EM) algorithm, where each instance is wholly assigned to only its most probable cluster in expectation.

It is reported (Gens and Pedro, 2013; Rooshenas and Lowd, 2014) that SPNs outperformed other PGMs in variable retrieval problems along with a specialized structure for faster inference.

3.3 Online Incremental Structure Learning of Sum-Product Networks

3.3.1 Methods

Online structure learning of PGMs is challenging task because there is no guarantee that new incoming instances follow the learnt model. If new instances are not explained with the model, they should be modified. Fortunately, in the case of SPNs, online structure learning is deeply related to clustering, and it can be converted to online clustering problem. It means that SPNs able to learn incrementally regardless of any value of instances coming in.

To explain online incremental SPN structure learning method, we first suggest simple mini-batch incremental clustering problem in Algorithm 1. In Algorithm 1, new instances are assigned to one of the existing clusters. New clusters are added with new instances if they are needed. To find an appropriate number of clusters k, we increase the number of clusters one by one until likelihood does not increase any more than threshold.

We can extend the above clustering process to learn the structure of SPNs. Algorithm 2 and Figure 3.2 illustrate online incremental learning algorithm for

Algorithm 1 IncrementalClustering(T,V,M)

Input: set of mini-batch instances T, set of variables V, and a cluster model M

Output: sets of instances assigned to the existing cluster $\{T_i\}$, sets of instances assigned to the new cluster $\{T_j\}$

likelihood = -inf

while true **do**

  while model M is converge **do**

    assign T with variable V and model M

    update model M with T

  end while

  calculate likelihood

  if increase of likelihood is less than threshold **then**

    break

  end if

end while

SPNs. The algorithm hierarchically adds new child nodes onto the sum nodes in whole layers. The clustering process is used in algorithm 2 as one part, and it basically uses the distributions of child nodes. However, there is no model for applying to new cluster, so we use naïve Bayes model, as the structure learning on the previous study does. After clustering, existing child nodes are augmented recursively, whereas new child nodes are constructed by previous structure learning methods. This hierarchical hard clustering strategy is also used in parameter learning of SPNs and verified by previous study (Poon and Domingos, 2011).

If the model explains new data well, the structure will not be changed.

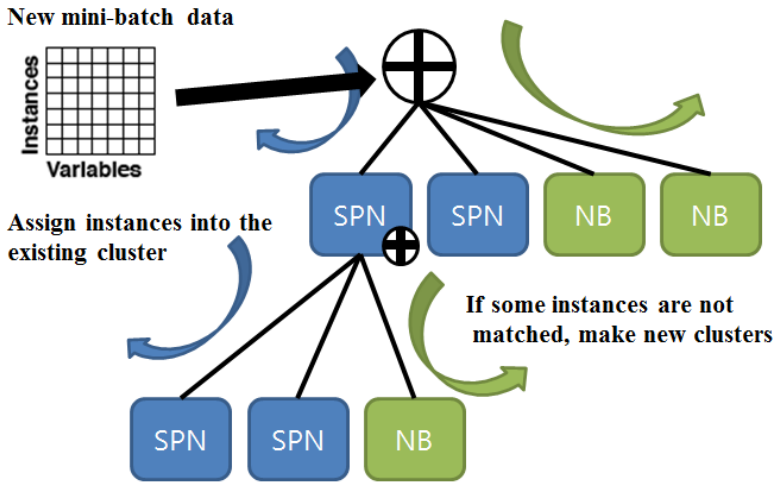

Figure 3.2 The proposed online incremental structure learning algorithm of SPNs.

Otherwise, the SPN increases their nodes to express new data. Note that, on the other hands, the variable subsets split by the product node will not change after the product node is generated once. If they are not mutual independent on new data, the product node cannot fully explain them. It may cause the model heavier with meaningless product nodes. It is a major limitation of our method.

3.3.2 Experiments

We evaluated online incremental structure learning methods on a tiny dataset to illustrate our argument. The main datasets are variants of the "hand-written optic digits" (Lichman, 2013). These digit data include an image pixel and a digit class. However, pixels are binarized for our experiments. We make two settings: 'homo' and 'hetero.' In the hetero setting, we only reorder the dataset according to the class label. In other words, the models first meet all the digit0

Algorithm 2 AugmentSPN(T,V,M)

Input: set of mini-batch instances T, set of variables V, and an SPN M

Output: an augmented cluster model M'

if root of M is univariate node **then**

  M = ParameterLearnNode(T,V,M)

else if root of M is product node **then**

  for each child node M_k of root **do**

    M_k = AugmentSPN(T,V_k,M_k)

  end for

else if root of M is sum node **then**

  ($\{T_i\}$,$\{T_j\}$) = IncrementalClustering(T,V,M)

  for each instance sets of i-th existing cluster **do**

    M_i = ParameterLearnNode(T_i, V, M_i)

    M_i = AugmentSPN(T_i, V, M_i)

  end for

  for each instance sets of j-th new cluster **do**

    M_j = StructureLearnSPN(T_j, V)

  end for

end if

images, then the digit1 images, and so on. The homo setting uses only the dataset's own order, and it yields a stable probability distribution of data. The hetero setting, however, yields a dynamically changing probability distribution of data in the mini-batch task.

We evaluated the suggested model in a mini-batch environment. The number of mini-batches was 16. Three models were compared. The first is "classical online learning", which uses only the first mini-batch for structure learning. The

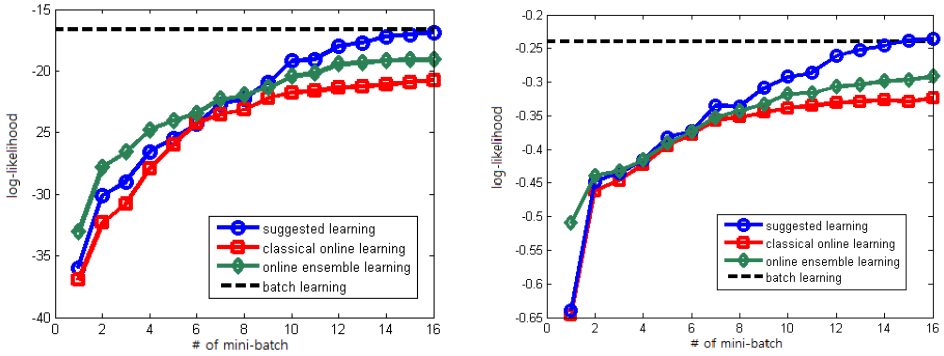

Figure 3.3 Log-likelihoods (left). Average conditional log-likelihoods for arbitrary query and evidence (right).

method for structure learning follows previous research studies. The second is "online ensemble learning" which ensembles 16 SPNs constructed by "classical online learning" method. "Classical online learning" methods, however, in general may include smaller number of nodes than the suggested model. It is possible that more complex model may do better, which is why this second model used in the comparison. However, online ensemble models also have a slightly larger number of nodes than the suggest-ed model. The third model is "batch learning," which uses the whole dataset for struc-ture learning and is exactly same as the classical methods used in previous studies.

We tested two methods for evaluating performances, one of which measures likelihood. The other goal is to infer the probability of a subset of the variables (the query) given the values of another (the evidence). We used 50% of the variables as the query and 30% of the variables as the evidence in the experiments.

Figure 3.3 shows the performance results of the various learning methods as a hetero handwritten dataset arrives. The suggested model not only out-

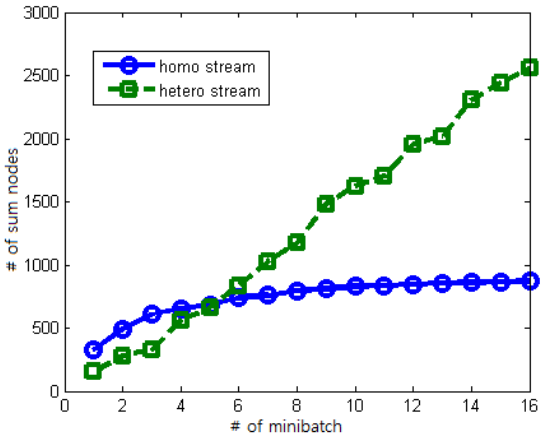

Figure 3.4 Growth of complexity. If incoming data are similar to the distribution of the model, the complexity of the model converges. Otherwise, the complexity of the model increases.

performs na¨ıve online models, but also achieves the performances of batch structure learning. The results imply that the suggested learning method represents well the probability distribution of the data. We also catch that "nline ensemble learning" method do better than "lassical online learning"methods, which means that previous studies may not have fully tuned their own model.

We also investigated the form of the structure of changing SPNs. According to dif-ferent characteristics of the order in which the data arrive, the structure changes dif-ferently. First, the complexities of SPNs are different. Figure 3.4 shows that, in the hetero setting, the models need more nodes to represent the probability distribution as new data arrive. Second, the structure of SPNs or the numbers of child nodes of a root differ. Figure 3.5 shows that the different sequence-order of data evolves different forms of structure. Figure 3.5 (a, c) shows the results of the first mini-batch step, and Figure 3.5 (b, d) shows the result of the final mini-batch step. Figure 3.5 (a-b) shows the structure change

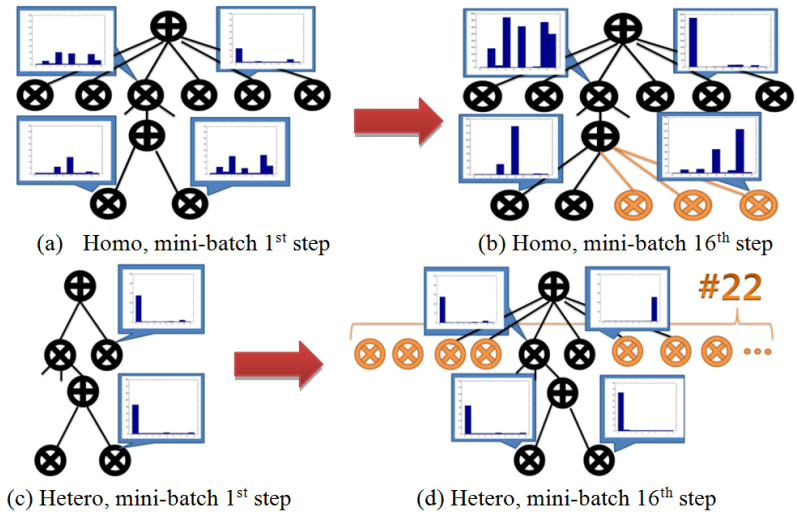

Figure 3.5 Different structures evolve according to different orders of datasets.

of the homo setting when distribution of data is balanced. When data with similar distribution arrive, SPNs increase their depth. Figure 3.5 (c-d) shows the structure change of the hetero setting when the order of data is according to the class. When data with dynamically changing distribution arrive, SPNs increase their width.

3.4 Non-Parametric Bayesian Sum-Product Networks

3.4.1 Model 1: A Prior Distribution for SPN Trees

We describe two non-parametric generative models for multivariate data, which indirectly specify non-parametric prior distributions over SPNs. In the first model, the SPN is always a tree, in that no product node is ever a child of two sum-nodes, and no sum-node is ever a child of two product nodes. In the second model, the SPN is a dag, in that every product node is potentially the child of many sum-nodes, but no sum-node is ever the child of two product nodes. The

first model is simpler; the second model is more general. The prior distributions are defined recursively, node by node.

In both the tree and the dag models, the prior distribution of each sum-node S_s is a Dirichlet process (Teh et al., 2005), with concentration parameter α_s and base distribution denoted by $\mathrm{G_P}(s)$. The base distribution $\mathrm{G_P}(s)$ is a probability distribution over the set of possible product nodes with scope s. The distribution $\mathrm{G_P}(s)$ is a basic input to the model which expresses prior beliefs about the form of product nodes. In principle, any distribution over product nodes could be used, but a simple and in some sense elegant choice is specify $\mathrm{G_P}(s)$ as a probability distribution over the partitions of s. Only the partition of s is chosen at this level; the prior distribution for each child sum-node of the product node is defined recursively, in the same manner, at its own level. if the partition generated contains singletons, $\mathrm{G_P}(s)$ must also specify the a p.d. over the possible univariate distributions; for binary data, a natural choice is a beta distribution parametrised as a vague prior, such as $\mathrm{Beta}(\frac{1}{2}, \frac{1}{2})$.

Each sum-node S_s has a countably infinite number of child product-nodes (note that a sum-node can have multiple child product nodes with the same partition). The probability distribution over these children is a Dirichlet Process over base distribution $\mathrm{G_P}(\mathrm{scope}(\mathrm{S}), \alpha_s)$.

In this model, sum-nodes with identical scopes are distinct and independent. In short, the prior distribution over SPNs is specified as a tree of Dirichlet Processes over product-nodes; each product-node has a finite branching factor, and each sum-node has an infinite branching factor. The tree has maximal depth D, and there is a unique top sum-node $\mathrm{S}_{Top} = S_{\{1,\ldots,D\}}$.

The prior is parametrised by:

- For each s, a concentration parameter $\alpha_s > 0$. These may, for example, plausibly be a function of $|s|$.

- For each s, a probability distribution $\mathrm{G_P}(s)$ over partitions of s. These distributions may express significant prior knowledge concerning the desired shape of the tree, and also which partitions of the particular variables indexed by s are plausible. For example, different variables may have different meanings, and there may be prior beliefs that some partitions of s are more probable than others.

- For each $i \in \{1, \ldots, D\}$, a prior distribution on the univariate distribution $S_{\{i\}}(\cdot)$. For binary data, this is plausibly a vague beta prior.

Computation is greatly simplified if the predictive distribution for a new product node P_s, not yet associated with any data, is of a simple known form. For example, a newly generated product node P_s with no data might predict the uniform distribution over X_s.

Generating data with the tree model

Sampling and inference with the tree model are rather straightforward: we place a Blackwell-MacQueen urn process (Blackwell and MacQueen, 1973) at each sum node, and sample from each such process as required, starting from the top sum-node S_{Top}.

Each sample is generated recursively, as a tree of data requests: the root request, for all D variables is sent to S_{Top}. The recursion is as follows. For a node of type:

univariate-node : a sample from the univariate distribution defined by the node is returned.

sum-node : A child product-node P is sampled from the Blackwell-MacQueen urn process at the sum-node, and the sample request is then sent to that product node. The sample received from the product node is returned.

product-node : Let the node partition be $(s_1, \ldots, s_k)$. The sample request is partitioned into k requests with scope s_1 to s_k, and these requests are sent to the corresponding child nodes. When the samples from the k child nodes are returned, they are recombined to form a sample for the scope requested, and returned to the parent node.

This recursive sampling process generates an exchangeable sequence of samples. To carry out this sampling procedure, each sum-node (and each distribution-node) must maintain state information sufficient for its urn process: this information consists of the indexed sequence of samples taken from that node's generative process.

The univariate distribution nodes may simply be fixed probability distributions on one variable, or the may have an arbitrarily complex structure. To ensure exchangeability of the samples from the entire structure, it is sufficient that each univariate distribution node should provide an exchangeable sequence of samples, separately from all other nodes.

This generative model, with urn-processes, can be used in several ways:

- to generate an exchangeable sequence of samples
- to fit a distribution to given data, using Gibbs sampling, Metropolis-Hastings, or many other MCMC methods. Gibbs and Metropolis-Hastings can be performed recursively throughout the tree, and in parallel on different branches.
- for any given state of the sampling system, an SPN is defined implicitly by the predictive sampling probabilities for the next data item at each node. This SPN can then be used for inference as described in (Poon and Domingos, 2011)

Algorithm 3 Update_Instance_Full_MH

```
Input: instances {x_1, ..., x_N}
Initialize M, l
for lo = 1 to nLoop do
  for n = 1 to N do
    M' ← Update_Instance_One_MH(M, root, x_n, n)
    temp_l ← likelihood(M', x_n)
    if rand() < min(1, temp_l/l(n)) then
      l(n) ← temp_l
      M ← M'
    end if
  end for
end for
```

Learning algorithm for the tree structure

In this section, we explain two sampling methods for learning SPN topology. Algorithm 3 explains a Metropolis-Hastings (MH) sampling method, whereas Algorithm 4 explains a Gibbs sampling method. Algorithm 5 explains the MH learning rule used for updating the tree structure with one instance at a time. α is parameter of DP. Gibbs learning rule for updating one instance is a little difficult compared to MH method, because likelihood of choosing one child should be calculated in each step. It is more difficult to use the Gibbs learning rule for updating one instance at a time compared to the learning rule of the MH method. This is because the Gibbs sampling requires the calculation of the likelihood of choosing one child in each step. However, as calculating the likelihood in SPN is tractable, this step can be performed in a less complicated manner.

Algorithm 4 Update_Instance_Full_Gibbs

Input: SPN M, instances $\{x_1, ..., x_N\}$

Initialize M

for $lo = 1$ **to** $nLoop$ **do**

  for $n = 1$ **to** N **do**

    $M \leftarrow Update\_Instance\_One\_Gibbs(M, root, x_n, n)$

  end for

end for

The MH and Gibbs sampling scheme was evaluated using a "hand-written optic digits" UCI dataset which includes 8x8 pixels and digit classes. Only the image pixels with binarization used in the experiments. Partitions were randomly split every time a new product node was made. In Figure 3.6, Algorithm 3 is used for the 'Tree with MH' condition, whereas Algorithm 4 is used for the 'Tree with Gibbs'. In this graph, the x-axis represents the number of instances used by the model. This simulation only used the instances once. We found the Gibbs sampling scheme performs better than the MH sampling scheme.

3.4.2 Model 2: A Prior Distribution for a Class of dag-SPNs

Model 1 is a tree, and this may be undesirable in some applications: this is because sampling requests follow the tree, and nodes deep in the tree must therefore typically handle only a small fraction of all sample requests. Models of this type have been used (Gens and Pedro, 2013), but (Delalleau and Bengio, 2011) intriguingly suggests that deep networks can express a more interesting class of distributions.

It is straightforward to alter the model to allow the sampling-request paths to form a dag. Each individual sampling request will take the form of a tree

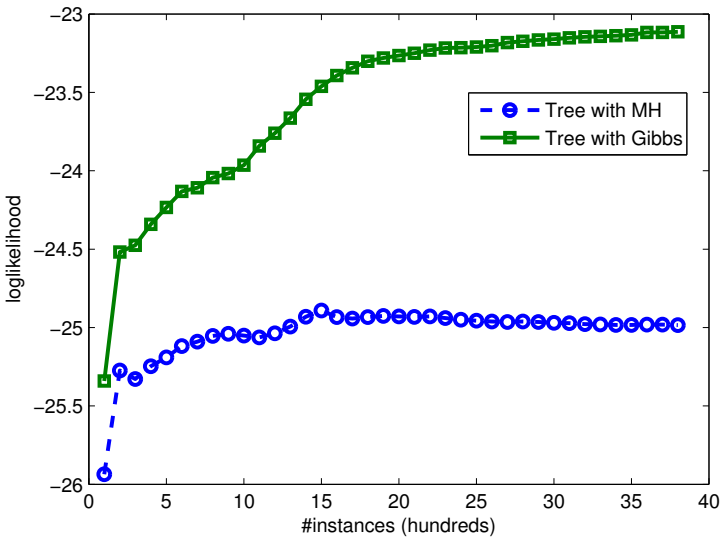

Figure 3.6 Loglikelihoods with different sampling schemes

within the dag, but the totality of sampling requests will lie on the dag. In such a model, even nodes deep in the dag may handle a large fraction of sampling requests, so that 'deep learning' becomes possible.

We alter model 1 as follows. For each of the $2^D - D - 1$ possible scopes s of level at least 2, we set up a 'sum-node-group' $\mathbf{S}_s$, which consists of a hierarchical Dirichlet Process (Teh et al., 2005). A hierarchical DP consists of a set of 'layer 1' DPs which share a common 'layer 0' base distribution, which is itself a DP. For $\mathbf{S}_s$, the base distribution for the layer 0 DP is $\mathrm{G_P}(s)$, which is the same as the base-distribution of a sum-node with scope s in the tree-SPN model above.

For layer 1 of $\mathbf{S}_s$, we set up a separate layer 1 DP for each possible parent node from which sampling requests may come. More specifically, for any product-node with a child sum-node with scope s, then that child sum-node

is placed as a DP in layer 1 of $\mathbf{S}_s$. In other words, all sum-nodes with scope s share the same base-distribution of product-nodes, which is itself a DP with base distribution $\mathrm{G_P}(s)$. The effect of this is that all sum-nodes with scope s in the dag will tend to share, and route requests to, common child product-nodes. Hence sampling requests from different sum-nodes can be routed to the same child product-node. Samples from a hierarchical DP are exchangeable, hence samples from the entire model are exchangeable, as before.

Remarkably, the only new parameters required are the concentration parameters of the layer 1 and layer 2 Dirichlet processes for each scope: once again, these may plausibly be functions only of the size of the scope.

Algorithm 3 and 4 can also be used for learning both the tree and dag structure. Algorithm 6 explains the MH learning rule at updating dag structure with one instance. α and γ are parameters of the hierarchical DP. Algorithm 4 uses the inference scheme explained in Ch 5.3 of (Teh et al., 2005).

MCMC sampling using the dag-SPN (our model 2) appears to be a particularly elegant possibility, which is as far as we know unexplored. The construction appears both modular and generic, and could be applied to the generation of many types of structured object, provided the generative decisions take the form of a tree, and node-scopes respect the same partial ordering for all objects.

Learning in product nodes is, however, problematic because there are many possible partitions and it is hard to find a good partition by Gibbs sampling, proposing random partitions. This is slow because the merit of a good partition only becomes apparent when many data-requests have been transferred to the new product node: when a product node is first generated, even if its partition is optimal, it has no data assigned to it, and so at first it predicts a uniform distribution over its scope variables. This means that every new product node is at an initial disadvantage compared to existing competitor product nodes which

have data currently assigned to them. An effective sampling method, therefore, should make proposals of altering the product partitions directly: however, such proposals are expensive, since if high-level partition elements are altered in this way, then lower level partition elements need to be changed as well. We are investigating such algorithms.

In the above experiment, random partition priors were explored, but other partition priors could also be used. It would be possible to have a hybrid method, in which the 'prior' for partition of variables would be made sensitive to the number of allocated instances; when few instances are allocated to some product node, the partition of this node would be fully separated into univariate nodes. To solve this problem, we are examining other SPN structure leraning methods to find good initialisations. One possibility is the LearnSPN algorithm in (Gens and Pedro, 2013). When a tree structure is made with the DP, the whole algorithm becomes a non-parametric Bayesian biclustering. If a dag-SPN is made with the HDP, the algorithm becomes more interesting and is an idea we will explore in the future. Additionally, (Lowd and Rooshenas, 2013; Peharz et al., 2013) study the bottom-up learning of SPN and the arithmetic circuit. These ideas can be used for making good candidates of product nodes. It is also interesting to note that (Rooshenas and Lowd, 2014) uses the hybrid approach of top-down and bottom-up.

In summary, we have defined prior distributions on SPNs that specify only mixture-distribution concentration parameters, priors on partitions of scopes, and vague priors on univariate distributions. However, finding effective sampling methods is a challenging problem.

3.5 Discussion

3.5.1 History of Online Learning of Sum-Product Networks

The follow-up studies of the structure learning are as follows. Other researchers also studied infinite online incremental structure learning. In (Hsu et al., 2017), the structure is evolved by Gaussian leave.

In another direction, there are works on sequential Bayesian (Rashwan et al., 2016; Zhao et al., 2016). Rashwan et al. apply moment matching with the Dirichlet distribution assumption on their SPN (Rashwan et al., 2016). In their moment matching of each parameter, there is a small loss on the likelihood. It would be not interesting in the perspective of probabilistic models, as counts on categorical distributions are added in an approximated way. By the way, it is somewhat surprised in the perspective of deep models, as it overcomes the catastrophic forgetting easily.

3.5.2 Toward Lifelong Learning of Deep Neural Networks

We introduced online learning methods of SPNs, a kind of deep PGMs. The properties of PGMs and their online learning is relatively well known, and the relation between deep PGMs and deep neural networks have long been studied. There are two approaches to online learning of deep PGMs. One approach continuously extends the structure of the model and estimate the probability density function. The other approach continuously estimates the probability of the model by sequential Bayesian.

Two methods are an example of SPN online learning method based on structure learning and sequential Bayesian, respectively. We extend the discovery from these studies of online learning of SPNs to the studies of lifelong learning in deep neural networks in the following two chapters. These two methods for

SPN correspond to DMA, which use dual structure learning algorithms, and IMM, which use sequential Bayesian and Bayesian neural networks to estimate the probability distribution of the whole dataset and task.

3.6 Summary

In the section, we introduced two online learning methods of SPN, a kind of deep PGMs. In the first method, an online incremental strucutre learning method is used. In the method, the number of hidden units and even layers are evolved dynamically on incoming data to follow the changing distribution of data. In the second method, a non-parametric Bayesian method is used. In our study, two prior distributions on SPNs are defined, which specify mixture-distribution concentration parameters, priors on partitions of scopes, and vague priors on univariate distributions.

The relation between deep PGMs and deep neural networks have long been studied. For example, deep neural networks can be understood as a discriminative version of deep belief networks or deep Boltzmann machine. Though the original SPNs is a generative model, there also is a discriminative version of SPNs which can also used for classification tasks. These two methods for SPN can be corresponds to DMA, which use dual structure learning algorithms, and IMM, which use sequential Bayesian and Bayesian neural networks to estimate the probability distribution of the whole dataset and task.

Algorithm 5 Update_Instance_One_MH_Tree

Input: SPN M, sumnode index i, instances $\{x_1, ..., x_N\}$, instance index n

Output: SPN M'

$M' \leftarrow M$

$\{pidx_1, ..., pidx_K\} \leftarrow$ indexes of child nodes of sum node $M'.S_i$

if $M.S_i.allocate(n) \neq empty$ **then**

  $M'.P_{[M.S_i.allocate(n)]}.w -= 1$

end if

for $k = 1$ **to** K **do**

  $p(k) = \frac{(M'.P_{[pidx_k]}.w)}{[(\sum_k^K M'.P_{[pidx_k]}.w)+\alpha]}$

end for

$p(K+1) = \frac{\alpha}{[(\sum_k^K M'.P_{[pidx_k]}.w)+\alpha]}$

select $k \sim p(k)$

if $k \leq K$ **then**

  $M'.S_i.allocate(n) = pidx_k$

  $P_{[M'.S_i.allocate(n)]}.w += 1$

  $\{sidx_1, ..., sidx_L\} \leftarrow$ indexes of child nodes of product node $M'.P_{pidx_k}$

  for $l = 1$ **to** L **do**

    $M' \leftarrow Update\_Instance\_One\_MH\_Tree\ (M, sidx_l, x_n, n)$

  end for

else

  $M' \leftarrow make\_prodnode\_in\_sumnode(M', i, x_n, n)$

end if

Algorithm 6 Update_Instance_One_MH_DAG

Input: SPN M, sumnode index i, instances $\{x_1, ..., x_N\}$, instance index n

Output: SPN M'

$M' \leftarrow M$

$\{pidx_1, ..., pidx_K\} \leftarrow$ indexes of product nodes which have same *scope* to sum node $M'.S_i$

if $M.S_i.allocate(n) \neq empty$ **then**

  $M'.P_{[M.S_i.allocate(n)]}.w_i -= 1$

end if

for $k = 1$ **to** K **do**

  $p(k) = \frac{(M'.P_{[pidx_k]}.w_i + \alpha \times M'.S_i.\beta_k)}{(\sum_k^K M'.P_{[pidx_k]}.w_i) + \alpha]}$

end for

$p(K+1) = \frac{(\alpha \times M'.S_i.\beta_{K+1})}{[(\sum_k^K M'.P_{[pidx_k]}.w_i) + \alpha]}$

select $k \sim p(k)$

if $k \leq K$ **then**

  $M'.S_i.allocate(n) = idx_k$

  $P_{[M'.S_i.allocate(n)]}.w_i += 1$

  $\{sidx_1, ...sidx_L\} \leftarrow$ indexes of child nodes of product node $M'.P_{pidx_k}$

  for $l = 1$ **to** L **do**

    $M' \leftarrow Update\_Instance\_One\_MH\_DAG\ (M, sidx_l, x_n, n)$

  end for

else

  $M' \leftarrow make\_prodnode\_in\_sumnode(M', i, x_n, n)$

  $M'.S_i.\beta \leftarrow sample\_beta(M'.S_i.allocate, \alpha, \gamma)$

end if

Chapter 4

Structure Learning for Lifelong Learning: Dual Memory Architecture

4.1 Introduction

In this chapter, we explain a dual memory architecture (DMA) 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 contributions of the study described in this chapter are as follows: 1) Proposal of the DMA and the mGHN; 2) Empirical analysis of various online learning methods on real-life datasets; 3) Proof of the online parameter learning method of mGHNs; 4) Cognitive neuroscience perspective of DMA; 5) Making the lifelog dataset publically available; and 6) Implications of DMA on other machine learning fields.

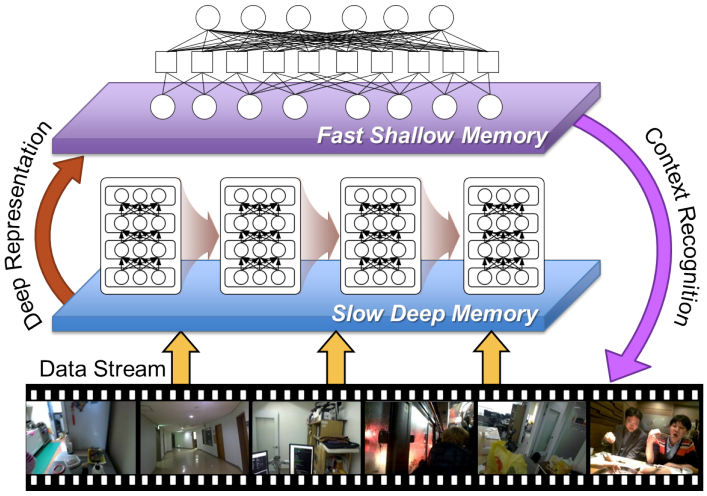

Figure 4.1 Lifelong Learning framework and the dual memory architecture (DMA)

The remainder of this chapter is organized as follows. Section 4.2 explains complementary learning systems (CLS) theory, which motivated our DMA research. Section 4.3 introduces the general concept of DMA. Section 4.4 discusses the multiplicative-Gaussian Hypernetworks (mGHN) and its online learning method. Section 4.5 presents experimental results for the online learning of DNNs and analyzes the performances of the DMA. Section 4.5.1 explains the results for a non-stationary variant of the CIFAR-10 dataset, and Section 4.5.2 contains the description and the results for the Google Glass lifelog dataset. Section 4.6 discusses our philosophy of parameter-decomposability which is extended to the IMM algorithm, and the implication of the DMA algorithm on Bayesian optimization.

4.2 Complementary Learning Systems Theory

To devise the DMA algorithm for solving the catastrophic forgetting problem, we apply the concept of complementary learning systems (CLS) theory a framework that suggests a dual learning system structure in the brain (McClelland et al., 1995; O'Reilly et al., 2014). According to the CLS theory, there are two critical areas in the brain that affect online learning: the neocortex and hippocampus, which complement each other's functionality. From the learning perspective, the neocortex is analogous to a deep neural module that can gradually learn to extract a structure from the real-world sensor streams (Guyonneau et al., 2004). Further, corroborative evidence from cognitive neuroscience shows that the behaviors of performance-optimized deep CNNs closely match to the neural responses in higher visual cortex of the brain in a monkey (Yamins et al., 2014). However, the key limitations introduced by the perspective of online learning is that the neocortex does not rapidly learn a new concept in a single attempt nor does it process data accordingly at the instance-level to weigh appropriately specific events. In contrast, the hippocampus alleviates this problem by allowing rapid and individuated storage to memorize a new instance (Treves and Rolls, 1992; Knierim and Neunuebel, 2016). There has been evidence that the hippocampus can allow general statistics of the environment to be circumvented by weighting procedures such that statistically unusual but significant events may be afforded a privileged status (Carr et al., 2011; Bendor and Wilson, 2012). However, a hippocampal system alone would be insufficient for continuous learning because of the limitations on memory capacity and generalization ability.

Recently, some machine learning algorithms related to CLS theory have been proposed. A recently published study reviewed the link between the core

principles of the CLS theory and recent themes in machine learning (Kumaran et al., 2016); we follow this perspective.

The first example is model-free episodic control, which is a reinforcement learning method inspired by hippocampal episodic control applied to difficult sequential decision-making tasks (Blundell et al., 2016). In this method, memorization strategy is utilized for learning a value function directly from experience without iterative estimation of the value function. This approach not only makes the learning significantly faster than comparable deep reinforcement learning algorithms, but also performs better on some of the more challenging domains.

The second example is an external memory utilized for the neural network. The neural network researchers found that the capability of a neural network alone is limited in solving complex AI problems including question and answering tasks. Thus, they have proposed coupling neural networks with external memory resources that correspond to hippocampal episodic memory. There are two methods exploiting external memory for neural networks. One method is the neural Turing machine (NTM) (Graves et al., 2014), where the framework consists of an external memory and an RNN controller that control the reading from and writing to an external memory. An external memory writes the intermediate computations of the neural network, which can read out when the results are required. Studies have shown that the NTM can learn the general concepts of a simple algorithm such as copying and sorting. The other method is the memory network (Weston et al., 2014), where the external memory stores knowledge as a vectorized form. The memory network is mainly used for question and answering tasks and makes notable results (Sukhbaatar et al., 2015). In typical question and answering applications, appropriate vectorized knowledge (e.g., 'Tom left the basketball' and 'Tom traveled to the garden') is retrieved by the similarity of a vectorized question (e.g., 'Where is the basketball') to

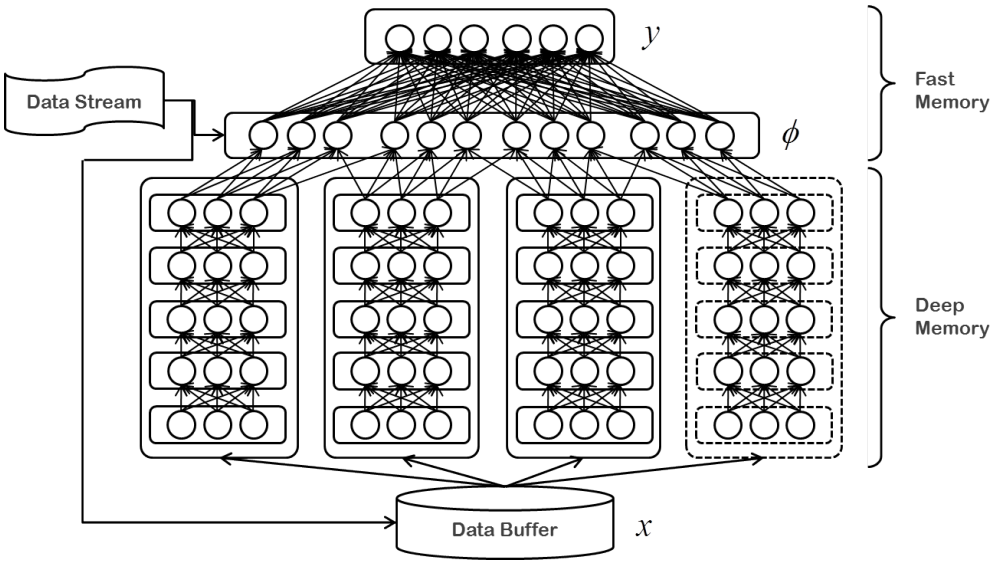

Figure 4.2 A schematic diagram of the dual memory architecture (DMA).

find the right answer ('the garden'). However, some researchers criticize that these methods due to their lower performance compared with that of a simpler model (Kim et al., 2016a).

4.3 Dual Memory Architectures

The dual memory architecture (DMA) is a framework designed to learn continuously from data streams. The DMA framework is illustrated in Figure 4.2, and the online learning process of the DMA is explained in Algorithm 7. With continuously arrived instances of data streams, fast memory updates its shallow network immediately. If a 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. The DMA consists of deep memory and fast memory. The structure of deep memory consists of several deep networks. Each of this network is constructed when

a specific amount of data from an unseen probability distribution is accumulated, thus creates a deep representation of data in a specific time. Examples of deep memory models include deep neural network classifier, 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 is 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) that can be learned online. The shallow network is in charge of making inference in the DMA; deep memory only generates deep representation. The equation used for inference can be described as:

$$y = \delta(w^T \phi(h^{\{1\}}(x), h^{\{2\}}(x), \cdots, h^{\{k\}}(x))) \tag{4.1}$$

where x is the input (e.g., a vector of image pixels), y is the target, ϕ and w are kernel and corresponding weight respectively, 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 to 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 SGD, can be used to guarantee a lower bound to the objective function (Zinkevich, 2003). Therefore, an

Algorithm 7 Online Learning of Dual Memory Architecture

```
if new instances come then
    if a new DNN is required then
        Initialize a new DNN with a previously constructed DNN.
        Train the new DNN with instances in the storage.
        Update the input feature and corresponding structure of the shallow network.
    end if
    Update the parameter of the shallow network.
    Put the new instances into the storage.
    Discard the oldest instances in the storage.
end if
```

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 $1^{st} \sim k$-1th 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 an older deep network to form a new deep network. This transfer technique initializes the weights of a newly trained deep network

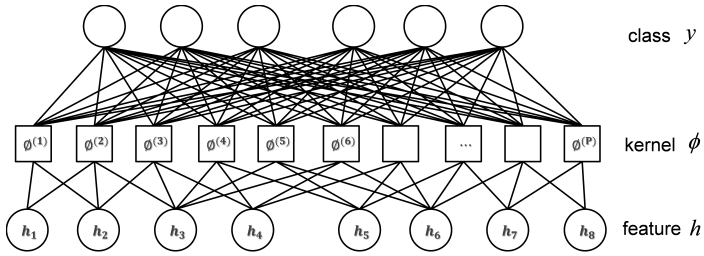

Figure 4.3 A schematic diagram of the multiplicative-Gaussian hypernetworks

W_k by the weights of the most recently trained deep network W_{k-1}. Once the training of the deep network from its own online dataset is complete, 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 the fast memory. This original transfer method assumes that the two networks have the same structure. However, there are some extensions (Chen et al., 2016; Wei et al., 2016) that allow different widths and a number of layers between some networks.

The proposed DMA is a combination of the ideas of three previously proposed methods 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 the older deep networks, as in the online learning of neural networks. Moreover, a shallow network in the fast memory is concurrently trained with deep memory, which is similar to the last-layer fine-tuning approach.

Table 4.1 Notations

Symbol	Explanation
y	class (i.e., location, sub-location, or activity)
ϕ	kernel vector
ϕ_k	k^{th} kernel
h	hidden vector of DNNs
x	input vector (i.e., input pixel)
k	index of kernel set
d	index of small dataset
n	index of instance
$\hat{\mu}$, $\hat{\Sigma}$	empirical sufficient the statistics
$\tilde{\mu}$, $\tilde{\Sigma}$	approximated maximum likelihood solution of statistics
$\phi_{k\cdot(d,n)}$	$(d,n)^{th}$ instance of kth kernel
$\hat{\mu}_{k\cdot d}$, $\hat{\Sigma}_{k\cdot d}$	empirical sufficient statistics of ϕ_k over d^{th} small dataset

4.4 Online Learning of Multiplicative-Gaussian Hypernetworks

4.4.1 Multiplicative-Gaussian Hypernetworks

In this section, we introduce a multiplicative Gaussian hypernetwork (mGHN) as an example of fast memory (Figure 4.3). Table 4.1 summarizes the notation used in this section. mGHNs are shallow networks that use a multiplicative function as an explicit kernel in (4.2):

$$\begin{aligned} \phi &= [\phi^{(1)}, \cdots, \phi^{(p)}, \cdots, \phi^{(P)}]^T, \\ s.t., \phi^{(p)}(h) &= (h_{(p,1)} \times \cdots \times h_{(p,H_p)}), \end{aligned} \tag{4.2}$$

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 hidden activation of DNNs. The set of variables of the pth 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 pth 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, 2008; 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 (4.3):

$$p\left(\begin{bmatrix} y \\ \phi(h) \end{bmatrix}\right) = N\left(\begin{bmatrix} \mu_y \\ \mu_\phi \end{bmatrix}, \begin{bmatrix} \Sigma_{yy} & \Sigma_{y\phi} \\ \Sigma_{\phi y} & \Sigma_{\phi\phi} \end{bmatrix}\right), \tag{4.3}$$

where μ_y, μ_ϕ, Σ_{yy}, $\Sigma_{y\phi}$, $\Sigma_{\phi y}$, 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.

$$E[p(y|h)] = \mu_y + \Sigma_{y\phi} \cdot \Sigma_{\phi\phi}^{-1} \cdot (\phi(h) - \mu_\phi) \tag{4.4}$$

The maximum likelihood solution of mGHNs can be updated immediately from any amount of new instances by online update of the mean and covariance if the number of features does not increase. Note that the distribution over y and ϕ over the first and second datasets (d = 1:2) is as follows:

$$\begin{bmatrix} y \\ \phi_1 \end{bmatrix}|_{d=1:2} \sim N\left(\begin{bmatrix} \hat{\mu}_{y\cdot 1:2} \\ \hat{\mu}_{1\cdot 1:2} \end{bmatrix}, \begin{bmatrix} \Sigma_{yy\cdot 1:2} & \Sigma_{y1\cdot 1:2} \\ \Sigma_{1y\cdot 1:2} & \Sigma_{11\cdot 1:2} \end{bmatrix}\right) \tag{4.5}$$

where $\hat{\mu}_{\cdot 1:2}$, $\hat{\Sigma}_{\cdot 1:2}$ is the function of $\hat{\mu}_{\cdot 1}$, $\hat{\mu}_{\cdot 2}$, $\hat{\Sigma}_{\cdot 1}$, and $\hat{\Sigma}_{\cdot 2}$.

$$\hat{\mu}_{1\cdot 1} = \frac{1}{N_1}\sum_{n=1}^{N_1} \phi_{1\cdot(d=1,n)} \tag{4.6}$$

$$\hat{\mu}_{1\cdot 2} = \frac{1}{N_2}\sum_{n=1}^{N_2} \phi_{1\cdot(d=2,n)} \tag{4.7}$$

$$\hat{\mu}_{1\cdot 1:2} = \frac{1}{N_1 + N_2}\left[\sum_{n=1}^{N_1}\phi_{1\cdot(d=1,n)} + \sum_{n=1}^{N_2}\phi_{1\cdot(d=2,n)}\right] = \alpha\hat{\mu}_{1\cdot 1} + (1-\alpha)\hat{\mu}_{1\cdot 2} \tag{4.8}$$

$$\alpha = \frac{N_1}{N_1 + N_2} \tag{4.9}$$

$$\begin{aligned}\hat{\Sigma}_{11\cdot 1} &= \frac{1}{N_1}\sum_{n=1}^{N_1}(\phi_{1\cdot(d=1,n)} - \hat{\mu}_{1\cdot 1})(\phi_{1\cdot(d=1,n)} - \hat{\mu}_{1\cdot 1})^T \\ &= \frac{1}{N_1}\left[\sum_{n=1}^{N_1}\phi_{1\cdot(d=1,n)}\phi_{1\cdot(d=1,n)}^T\right] + \hat{\mu}_{1\cdot 1}\hat{\mu}_{1\cdot 1}^T\end{aligned} \tag{4.10}$$

$$\hat{\Sigma}_{11\cdot 1:2} = \alpha\hat{\Sigma}_{11\cdot 1} + (1-\alpha)\hat{\Sigma}_{11\cdot 2} + \alpha(1-\alpha)(\hat{\mu}_{1\cdot 1} - \hat{\mu}_{1\cdot 2})(\hat{\mu}_{1\cdot 1} - \hat{\mu}_{1\cdot 2})^T \tag{4.11}$$

In our model, the parameter of two datasets can be decomposed into the parameter of each dataset, which we refer to as *instance-decomposibility*. This property allows our model to weigh specific events appropriately, which we refer to as *instance-scale reweighting*. For example, the model learns significantly r times from recent events as opposed to the penalized previous events by substituting α in (4.9) for following α':

$$\alpha' = \frac{N_1}{N_1 + r \cdot N_2} \tag{4.12}$$

4.4.2 Evolutionary Structure Learning

If the kth deep neural network is formed in the deep memory, the mGHN in the 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 (Zhang et al., 2012), as illustrated in Algorithm 8.

Algorithm 8 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 less-important kernel and adding new kernel. In our experiments, the set of $\phi_{discard}$ was picked by selecting the kernels with the lowest corresponding weights. From Equation (4.4), 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 pth 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.

4.4.3 Online Learning on Incremental 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 the first ($k = 1$) and the second ($k = 2$) 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 ith kernel vector ϕ_i and j^{th} kernel vector ϕ_j corresponding to the distribution of the dth dataset. If these sufficient statistics satisfy the following equations:

$$\phi|_{d=1} \sim N(\hat{\mu}_{1\cdot 1}, \hat{\Sigma}_{11\cdot 1}) \tag{4.13}$$

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} |_{d=2} \sim N\left(\begin{bmatrix} \hat{\mu}_{1\cdot 2} \\ \hat{\mu}_{2\cdot 2} \end{bmatrix}, \begin{bmatrix} \hat{\Sigma}_{11\cdot 2} & \hat{\Sigma}_{12\cdot 2} \\ \hat{\Sigma}_{21\cdot 2} & \hat{\Sigma}_{22\cdot 2} \end{bmatrix} \right) \tag{4.14}$$

$$\phi_1|_{d=1,2} \sim N(\hat{\mu}_{1\cdot 1:2}, \hat{\Sigma}_{11\cdot 1:2}) \tag{4.15}$$

the parametrization of maximum likelihood via the conditional Gaussian distribution can be done as follows.

$$\begin{aligned} p(\phi_2|\phi_1)|_{d=2} &= N(\hat{\mu}_{2|1\cdot 2}, \hat{\Sigma}_{22|1\cdot 2}) \\ &= N(\hat{\mu}_{2\cdot 2} + \hat{\Sigma}_{21\cdot 2}\hat{\Sigma}_{11\cdot 2}^{-1}(\phi_1 - \hat{\mu}_{1\cdot 2}), \hat{\Sigma}_{22\cdot 2} - \hat{\Sigma}_{21\cdot 2}\hat{\Sigma}_{11\cdot 2}^{-1}\hat{\Sigma}_{12\cdot 2}) \end{aligned} \tag{4.16}$$

Note the following properties of Gaussian:

$$E \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \mu \\ A\mu + b \end{bmatrix} \tag{4.17}$$

$$Cov \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} \Lambda^{-1} & \Lambda^{-1}A^T \\ A\Lambda^{-1} & L^{-1} + A\Lambda^{-1}A^T \end{bmatrix} \tag{4.18}$$

where the probability of x and the conditional probability of y given x are as follows:

$$p(x) = N(x|\mu, \Lambda^{-1}) \tag{4.19}$$

$$p(y|x) = N(y|Ax + b, L^{-1}) \tag{4.20}$$

Thus, the maximum likelihood solution represents ϕ as (4.21).

$$\begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} |_{d=1,2} \sim N\left(\begin{bmatrix} \hat{\mu}_{1\cdot 1:2} \\ \tilde{\mu}_2 \end{bmatrix}, \begin{bmatrix} \hat{\Sigma}_{11\cdot 1:2} & \tilde{\Sigma}_{12} \\ \tilde{\Sigma}_{12}^T & \tilde{\Sigma}_{22} \end{bmatrix} \right), \tag{4.21}$$

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

(4.21) can also be updated immediately from a new instance by online update of the mean and covariance. 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), which is not what we desire intuitively.

$$\begin{aligned} \hat{\mu}_{y|x}, \hat{\Sigma}_{y|x} &= \operatorname{argmax} p(y|\phi_{1:k}), \\ p(y|\phi_{1:k}) &\sim N(\hat{\mu}_{y|\phi}, \hat{\Sigma}_{y|\phi}) \end{aligned} \tag{4.22}$$

Moreover, (4.21) can be extended to sequential updates, when there is more than one increment of the kernel set (i.e., $\phi_3, \cdots, \phi_k$), because the equation of maximum likelihood of whole kernel observation ϕ_{obs} can be decomposed as follows:

$$p(\phi_{obs}) = \prod_{i=1}^{k} \prod_{d=1}^{k} \prod_{n=1}^{N_d} p(\phi_{i\cdot(d,n)} | \phi_{1:(i-1)\cdot(d,n)}) \tag{4.23}$$

4.5 Experiments

4.5.1 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 the algorithm was 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 $3 \sim 5$ image classes. Figure 4.4 shows the distribution of the data stream. In particular, the first online dataset contains 40% instances of classes 1 and 2 respectively, and 20% instances of class 3. Meanwhile, the second online dataset contains 40% instances of class 1, and 20% instances of classes 2, 3, and 4 respectively. We used the Network in Network (NIN) model (Lin et al., 2014), a kind of deep CNN, implemented using the MatConvNet toolbox (Vedaldi and Lenc, 2015). In the NIN, the conventional convolution linear filters are replaced by multi-layer DNNs. This idea is utilized by various CNNs including inception networks (Szegedy et al., 2015), the winner of ImageNet Challenge 2014. In our learning phase, 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

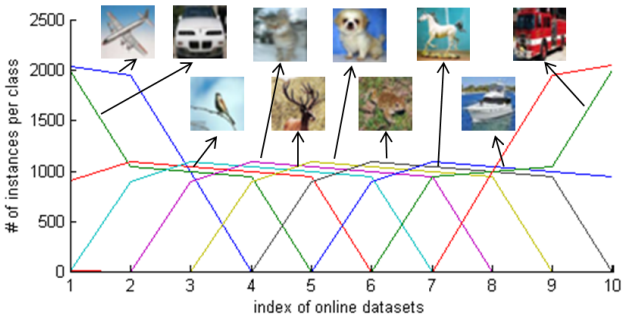

Figure 4.4 Distribution of non-stationary data stream of CIFAR-10 in the experiment

Table 4.2 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			

rate of momentum is 0.9.

We evaluate the performance of DMA with comparative models. The properties of DMA and comparative methods are listed in Table 4.2. Specifically, we propose two comparable learning methods to clarify the concept of DMA. The first one is *incremental bagging with transfer*. Unlike the naïve incremental bagging, this method transfers the weights of the older deep networks to the new deep network, as in DMA. The other is *DMA with last-layer retraining* where a shallow network is retrained in a batch manner. Although this algo-

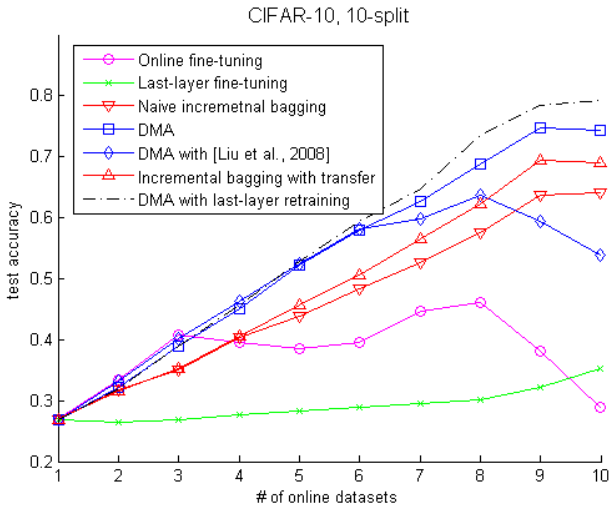

Figure 4.5 Test accuracies of learning algorithms on non-stationary CIFAR-10 data stream

rithm 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.

Figure 4.5 illustrates 10-split experimental results on non-stationary data. The models with only one CNN, which are online fine-tuning and last-layer fine-tuning in this experiment, show inferior results compared to other algorithms. These models did 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. The model has deep representation for discriminating only three image classes. Hence, the performance does not increase significantly. In the case of online fine-tuning, the model loses the previously seen information, which reduces the performance of test accuracy as time progresses. Meanwhile, incremental bagging increases its performance continuously with the non-stationary data stream. Incremental bagging that uses many networks outperforms online fine-tuning that uses only one deep network. On the

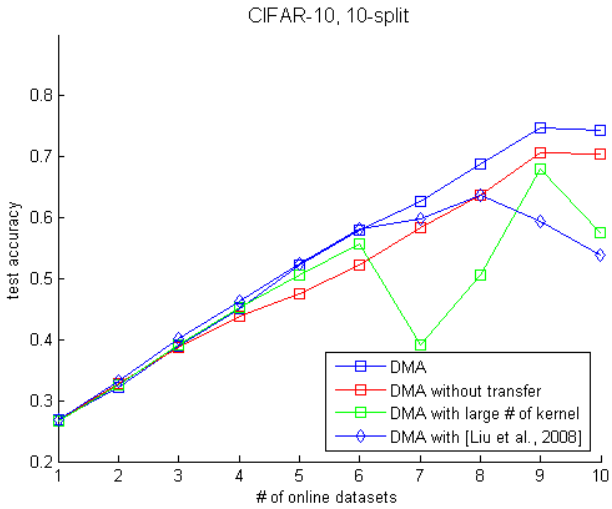

Figure 4.6 Test accuracies of DMA on CIFAR-10 data stream under various settings

other hand, the proposed DMA outperforms incremental bagging consistently. This result shows learning a shallow network and deep networks concurrently is advantageous compared to naïvely averaging softmax output probability of each CNN.

Alternative configurations of the DMA were also explored in our study to discover the characteristics of DMA (Figure 4.6). First, we analyze the effect of weight transfer in the deep memory. The DMA with weight transfer performs better than the DMA without weight transfer (+3.80%), as shown in Figure 4.6. The performance gap is much larger between incremental bagging with weight transfer and the model without weight transfer (+4.89%), as shown in Figure 4.5. Part of the entire data is not sufficient for learning discriminative representations in the weak learner for the whole class. In the experiment, weight transfer alleviates this problem for both the DMA and incremental bagging.

Second, we substitute our generative assumption over class y and kernel ϕ

Table 4.3 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

to the discriminative counterpart suggested by Liu et al. (Liu et al., 2008). The performance of two approaches does not differ at the early phase of the learning. However, the performance of the discriminative approach deteriorates as extreme non-stationary data is encountered continuously. This result supports our argument that a generative approach is one of the key points of successful online learning of mGHNs.

Lastly, we discuss the effect of the number of kernel features in the mGHN. The generative approach of mGHN needs more number of parameter compared to the discriminative approach. For example, the number of parameters of the covariance matrix in our generative approach is directly proportional to square of the number of kernel features, whereas the number of parameters of the weight matrix in the discriminative counterpart is directly proportional to the number of kernel features. The performance of the DMA severely decreases if the kernel size increase from 1,000 to 4,900 variables at the end of training as shown in *DMA with large # of kernel* in Figure 4.6.

4.5.2 Lifelog Dataset

We collected Google Glass lifelog dataset recorded over 46 days from three participants. The 660,000 seconds of the egocentric video stream data reflects the behaviors of participant including the indoor activities, such as 'working in the office' or 'eating in the restaurant', and the outdoor activities, such as

Table 4.4 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)

'walking on the road' or 'waiting for the arrival of the subway car'. The subjects were asked to notate what they were doing and where they were in real-time by using a life-logging application installed on their mobile phones. The notated data was then used as labels for the classification task in our experiments. In this study, the classification task of location, sub-location, and daily activity are considered. For evaluation, the dataset of each subject is separated into the training set and test set in order of time. A frame image of each second is used and classified as one instance. Table 4.3 summarizes the dataset statistics and Table 4.4 presents the distribution of the five major classes in each class type. We allowed the AlexNet features and the labels of class types of the lifelog dataset to be publically available *(version 1)* [1].

Two kinds of neural networks were used to extract the representations in the experiment. One is AlexNet, a prototype network trained by ImageNet (Krizhevsky et al., 2012). The other is referred to as LifeNet, a network trained with our lifelog dataset. The structure of LifeNet is similar to AlexNet, but the number of nodes in LifeNet is half of those in AlexNet. Both AlexNet and LifeNet were implemented using the MatConvNet toolbox. 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

[1] bi.snu.ac.kr/datasets/lifeome/

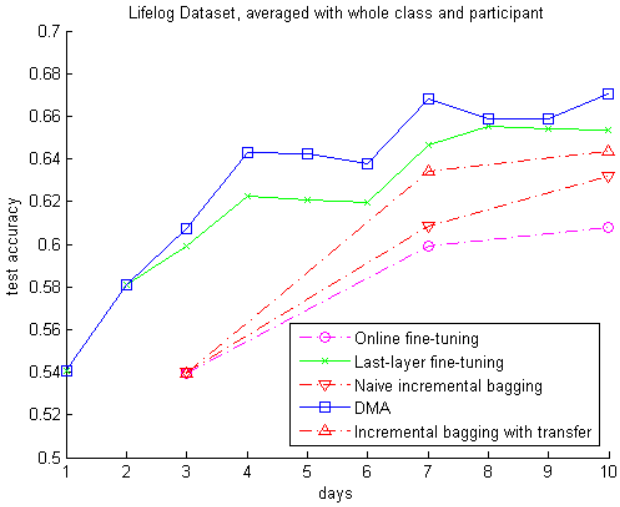

Figure 4.7 Averaged test accuracies of various learning algorithms on the lifelog dataset. The location, sub-location, and activity are classified separately for each of the three subjects.

each scene to be 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 the subjects B and C respectively. However, for subject A, the 13 days of training data was changed into 10 online datasets 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 the 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.

Figure 4.7 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

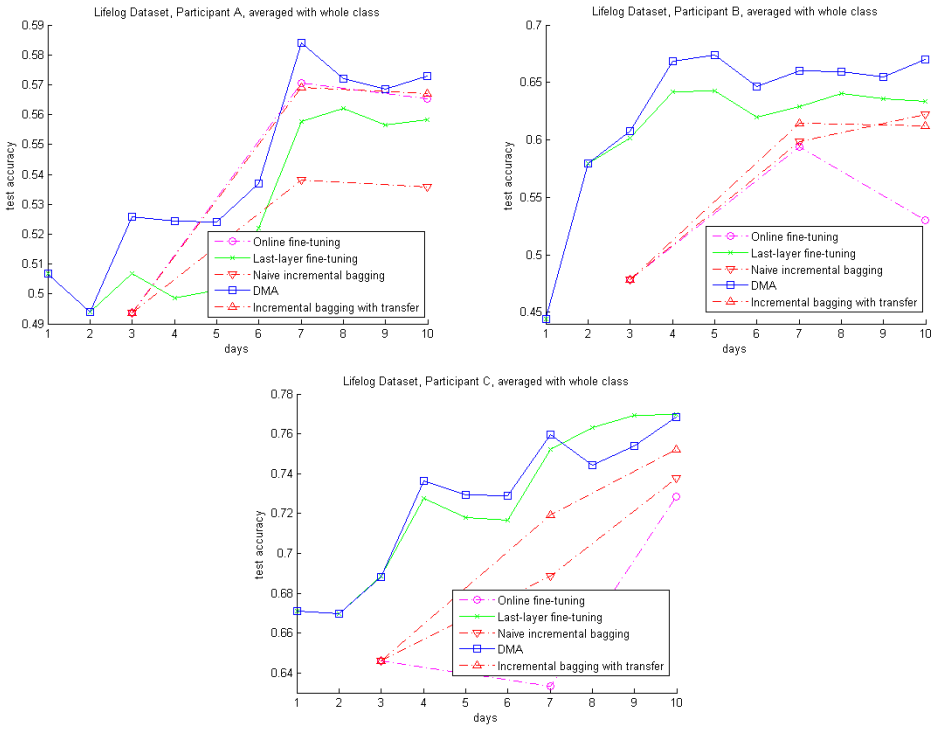

Figure 4.8 Averaged test accuracies of various learning algorithms on the lifelog dataset. The result of each class type are evaluated separately for each participant.

accuracies from a range of learning algorithms are plotted. Unlike the previous result on CIFAR-10, 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. 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 for online deep

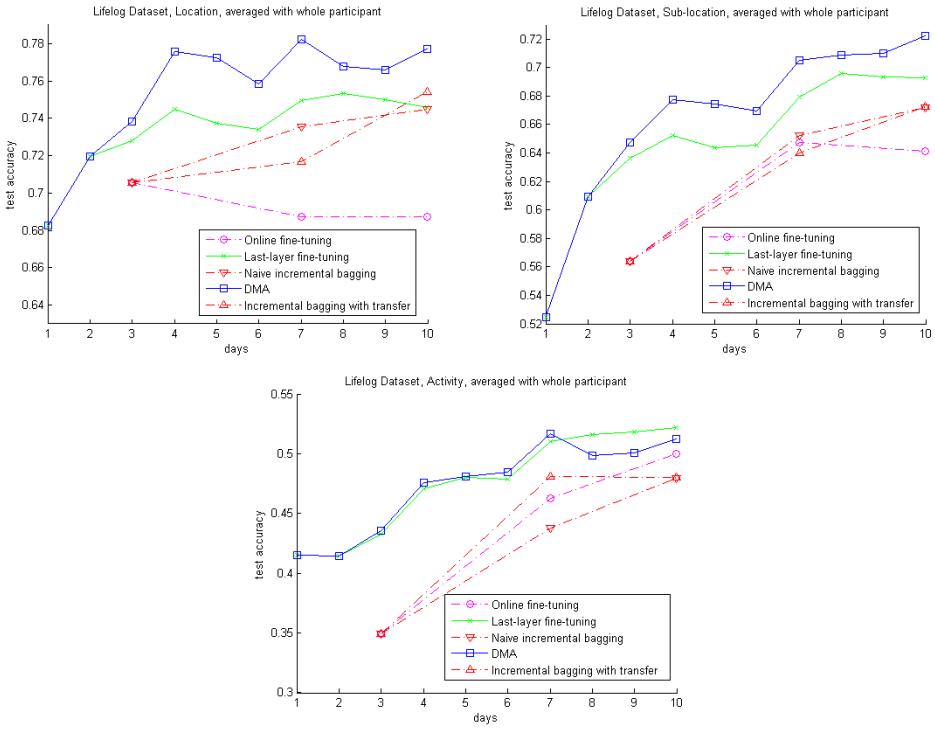

Figure 4.9 Averaged test accuracies of various learning algorithms on the lifelog dataset. The result of each participant are evaluated separately for each of the class type.

learning.

Figure 4.8 and 4.9 show the accuracies by each subject and class type respectively. In some experiments, at times the performance of algorithms decreases with the incoming stream of data. While learning a non-stationary data stream is a natural phenomenon, it would occur in 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 algorithms increase steadily with the incoming stream of data.

4.6 Discussion

4.6.1 Parameter-Decomposability in Deep Learning

We have previously mentioned that the DMA is inspired by the CLS theory. The study not only first raises the catastrophic forgetting problem, but also suggests the dual structure to solve the problem; deep memory corresponds to the neocortex, whereas fast memory corresponds to the hippocampus. Further, the following questions on missing links in the cognitive modeling of CLS are in our interest: First, how can the neocortex as the connectionist parametric model and the hippocampus as the instance-based non-parametric model work together? Second, what is the mechanism of data regeneration? Is it a database-style memorization or, does a more efficient algorithm exists for preserving the information of data pattern?

Two properties of the DMA can be used to answer these questions. First, the parameter μ and Σ of an mGHN is decomposable at each instance although DNN and mGHN infer as one neural network. This instance-decomposability allows one-shot learning (Fei-Fei et al., 2006), learning information about categories, from one, or only a few training instances, and instance-scale reweighting, learning with over-weight and under-weight instances by their importance.

Second, mGHNs can recover old information using an analytic method over the maximum likelihood solution from a pattern-completion perspective. This property accords with the interpretation of the CLS theory that there are areas for pattern classification (dentate gyrus) and pattern generation (CA3) in the hippocampus.

4.6.2 Online Bayesian Optimization

Bayesian optimization is a global black-box optimization technique that can be effectively applied on functions whose evaluation cost is expensive and its dis-

tribution is unknown. For determining next search point, Bayesian optimization uses a surrogate estimator of real search space, which gives an estimated score and its predictive uncertainty for each point. The most typical algorithm for Bayesian optimization is the Gaussian processes (GPs), that models non-linear search space but the speed of inference is high. However, the inference costs of GPs become expensive where the size of training data is large, since GPs scale cubically with the number of observations. Snoek et al. (Snoek et al., 2015) propose DNN-based modeling which alleviates this problem, because the DNN is adaptable for large-scale data. In their model, a Bayesian linear regression model is added to the last layer of the DNN. However, the existing method of DNNs does not fit the stream data to develop deep representations in an online manner. Our method can be applied to this kind of setting. The DMA updates not only the mGHN, which can work as the regression model in the last layer of DNN, but also the deep representation. For using mGHNs as Bayesian models, the prior can be put into the sufficient statistic of Gaussian parameter μ, Σ over class y and kernel ϕ. An example structure of the Bayesian version of DMA is shown in Kim et al.'s work (Kim et al., 2016b), where effective deep representations of a CNN pretrained by ImageNet are selected with the Bayesian least-square support vector machine (LS-SVM) model to classify into a new task.

4.7 Summary

We presented DMA for online deep learning of user behavior in everyday life using a wearable device. Our experimental results showed that the proposed method overcomes catastrophic forgetting in the learning of real non-stationary data. This property was utilized for implementing advanced personalized context recognition system.

This success is an example that solving the engineering problem by obtaining inspiration from the theories in cognitive neuroscience, which in our case was the CLS theory. On the other hand, our DMA implementation was used to discuss how neural network model and instance-based method work together from the CLS theory perspective. We also discussed two variants of the DMA that can potentially contribute to other machine learning fields. One model is the novel Bayesian optimization method having deep representation and online learnability. The other model is a novel online multi-task learner with evolving deep representation.

Chapter 5

Sequential Bayesian for Lifelong Learning: Incremental Moment Matching

5.1 Introduction

In this chapter, we introduce the incremental moment matching (IMM), which uses the Bayesian neural network framework. IMM assumes that the posterior distribution of parameters of neural networks is approximated with the Gaussian distribution and incrementally matches the moment of the posteriors, which are trained for the first and second task.

There are two moment matching methods in our study, mean-IMM and mode-IMM. Mean-IMM simply averages the parameter of two neural networks, whereas mode-IMM tries to find a maximum of the mixture of Gaussian posteriors. For IMM to be reasonable, the search space of the loss function between the posterior means μ_1 and μ_2 should be reasonably smooth and convex-like. To find a μ_2 which satisfies this condition of a smooth and convex-like path from

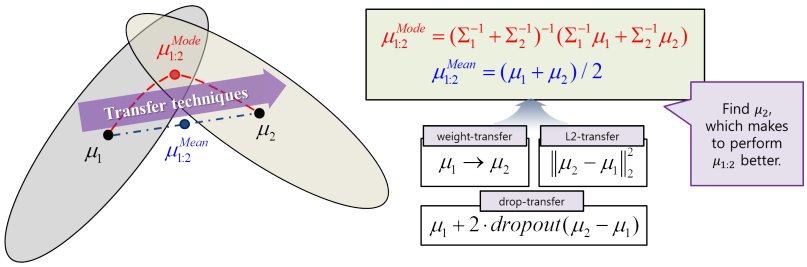

Figure 5.1 Geometric illustration of incremental moment matching (IMM).

μ_1, we propose applying various transfer techniques for the IMM procedure. The transfer techniques include 1) weight-transfer; 2) L2-transfer, L2-norm of old and new parameters; and 3) drop-transfer, a newly proposed variant of dropout using old parameters.

5.2 Incremental Moment Matching

In incremental moment matching (IMM), the moments of posterior distributions are matched in an incremental way. In our work, we use a Gaussian distribution to approximate the posterior distribution of parameters. Given K sequential tasks, we want to find the optimal parameter $\mu_{1:K}^*$ and $\Sigma_{1:K}^*$ of the Gaussian approximation function $q_{1:K}$ from the posterior parameter for each kth task, (μ_k, Σ_k).

$$p_{1:K} \equiv p(\theta|X_1, \cdots, X_K, y_1, \cdots, y_K) \approx q_{1:K} \equiv q(\theta|\mu_{1:K}, \Sigma_{1:K}) \tag{5.1}$$

$$p_k \equiv p(\theta|X_k, y_k) \approx q_k \equiv q(\theta|\mu_k, \Sigma_k) \tag{5.2}$$

$q_{1:K}$ denotes an approximation of the true posterior distribution $p_{1:K}$ for the whole task, and q_k denotes an approximation of the true posterior distribution p_k over the training dataset (X_k, y_k) for the kth task. θ denotes the vectorized parameter of the neural network. The dimension of μ_k and $\mu_{1:k}$ is D, and the

dimension of Σ_k and $\Sigma_{1:k}$ is $D \times D$, respectively, where D is the dimension of θ. For example, in the case of a multi-layer perceptrons (MLP) with [784-800-800-10] as the number of nodes, $D = 1917610$.

Next, we explain two proposed moment matching algorithms for the lifelong learning of modern deep neural networks. Two algorithms have a different form of Gaussian with a different objective function for the same dataset.

5.2.1 Mean-based Incremental Moment Matching (mean-IMM)

Mean-IMM averages the parameters of two networks in each layer, using mixing ratios α_k with $\sum_k^K \alpha_k = 1$. The objective function of mean-IMM is to minimize the KL-divergence between $q_{1:K}$ and the mixture of each q_k. According to Zhang and Kwok (Zhang and Kwok, 2010), the solutions $\mu^*_{1:K}$ and $\Sigma^*_{1:K}$ are as follows:

$$\mu^*_{1:K}, \Sigma^*_{1:K} = \underset{\mu_{1:K}, \Sigma_{1:K}}{\operatorname{argmin}} \; KL(q_{1:K} || \textstyle\sum_k^K \alpha_k q_k) \tag{5.3}$$

$$\mu^*_{1:K} = \textstyle\sum_k^K \alpha_k \mu_k \tag{5.4}$$

$$\Sigma^*_{1:K} = \textstyle\sum_k^K \alpha_k (\Sigma_k + (\mu_k - \mu^*_{1:K})(\mu_k - \mu^*_{1:K})^T) \tag{5.5}$$

Notice that covariance information is not needed for mean-IMM, since calculating $\mu^*_{1:K}$ does not require any Σ_k. A series of μ_k is sufficient to perform the task. The idea of mean-IMM is commonly used in shallow networks (Pathak et al., 2010; Baldi and Sadowski, 2013). However, the contribution of this paper is to discover where and how mean-IMM can be applied in modern deep neural networks and to show it can performs better with other transfer techniques.

5.2.2 Mode-based Incremental Moment Matching (mode-IMM)

Mode-IMM is a variant of mean-IMM which uses the covariance information of the posterior of Gaussian distribution. Though mean-IMM minimizes KL-divergence for a mixture of Gaussian (MoG), usually, a weighted average of two mean vectors of Gaussian distributions is not a mode of MoG. In discriminative learning, the maximum of the distribution is of primary interest. According to Ray and Lindsay (Ray and Lindsay, 2005), all the modes of MoG with K clusters lie on $(K-1)$-dimensional hypersurface $\{\theta|\theta = (\sum_k^K a_k \Sigma_k^{-1})^{-1}(\sum_k^K a_k \Sigma_k^{-1}\mu_k), 0 < a_k < 1 \text{ and } \sum_k a_k = 1\}$. See Appendix A for more detail.

Motivated by the above description, a mode-IMM approximate MoG with Laplacian approximation, in which the logarithm of the function is expressed by a Taylor expansion (MacKay, 1992). Using Laplacian approximation, the MoG is approximated as follows:

$$\log q_{1:K} \approx \sum_k^K \alpha_k \log q_k + C = -\frac{1}{2}\theta^T(\sum_k^K \alpha_k \Sigma_k^{-1})\theta + (\sum_k^K \alpha_k \Sigma_k^{-1}\mu_k)\theta + C' \tag{5.6}$$

$$\mu_{1:K}^* = \Sigma_{1:K}^* \cdot (\sum_k^K \alpha_k \Sigma_k^{-1}\mu_k) \tag{5.7}$$

$$\Sigma_{1:K}^* = (\sum_k^K \alpha_k \Sigma_k^{-1})^{-1} \tag{5.8}$$

Here, we assume diagonal covariance matrices, which means that there is no correlation among parameters. This diagonal assumption is useful, since it decreases the number of parameters for each covariance matrix from $O(D^2)$ to $O(D)$ for the dimension of the parameters D.

For covariance, we use the inverse of a Fisher information matrix, following (Kirkpatrick et al., 2017; Pascanu and Bengio, 2013). The main idea of this

approximation is that the square of gradients for parameters is a good indicator of their precision, which is the inverse of the variance. The Fisher information matrix for the kth task, F_k is defined as:

$$F_k = E\left[\frac{\partial}{\partial \mu_k} \ln p(\tilde{y}|x, \mu_k) \cdot \frac{\partial}{\partial \mu_k} \ln p(\tilde{y}|x, \mu_k)^T\right], \tag{5.9}$$

where the probability of the expectation follows $x \sim \pi_k$ and $\tilde{y} \sim p(y|x, \mu_k)$, where π_k denotes an empirical distribution of X_k.

The theoretical verification is as follows. According to Ray and Lindsay (Ray and Lindsay, 2005), all the critical points θ of a mixture of Gaussian (MoG) with two components are in one curve as the following equation with $0 < \alpha < 1$.

$$\theta = ((1-\alpha)\Sigma_1^{-1} + \alpha\Sigma_2^{-1})^{-1}((1-\alpha)\Sigma_1^{-1}\mu_1 + \alpha\Sigma_2^{-1}\mu_2) \tag{5.10}$$

The proof is as follows. Imagine two Gaussian distribution q_1 and q_2, such as in Equation 5.2.

$$q_1 \equiv q_1(\theta; \mu_1, \Sigma_1) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_1|}} \exp\left(-\frac{1}{2}(\theta-\mu_1)^T\Sigma_1^{-1}(\theta-\mu_1)\right) \tag{5.11}$$

$$q_2 \equiv q_2(\theta; \mu_2, \Sigma_2) = \frac{1}{\sqrt{(2\pi)^D|\Sigma_2|}} \exp\left(-\frac{1}{2}(\theta-\mu_2)^T\Sigma_1^{-1}(\theta-\mu_2)\right) \tag{5.12}$$

D is the dimension of the Gaussian distribution. Mixture of two Gaussian q_1 and q_2 with the equal mixing ratio (i.e., 1:1) is $q_1/2 + q_2/2$. The derivation of the MoG is as follows:

$$\frac{\partial(q_1/2 + q_2/2)}{\partial\theta} = -\frac{q_1}{2}(\Sigma_1^{-1}(\theta-\mu_1)) - \frac{q_2}{2}(\Sigma_2^{-1}(\theta-\mu_2)) = 0 \tag{5.13}$$

If we set Equation 5.13 to 0, to find all critical points, the following equation holds:

$$\theta = (q_1\Sigma_1^{-1} + q_2\Sigma_2^{-1})^{-1}(q_1\Sigma_1^{-1}\mu_1 + q_2\Sigma_2^{-1}\mu_2) \tag{5.14}$$

When set α to $\frac{q_2}{q_1+q_2}$, Equation 5.10 holds.

Note that α_k is a function of θ, so θ cannot be calculated in a closed-form from Equation 5.14. However, the optimal θ is in the set $\{\theta|\theta = ((1-\alpha)\Sigma_1^{-1} + \alpha\Sigma_2^{-1})^{-1}((1-\alpha)\Sigma_1^{-1}\mu_1 + \alpha\Sigma_2^{-1}\mu_2), 0 < \alpha < 1\}$, which motivates our mode-IMM method.

In our study IMM uses diagonal covariance matrices, which means that there is no correlation between parameters. This diagonal assumption is useful, since it decreases the number of parameters for each covariance matrix from $O(D^2)$ to $O(D)$. Based on this, the $\mu^*_{1:2}$ is defined as follows:

$$\mu^*_{1:2,v} = \frac{\mu_{1,v}/\sigma^2_{1,v} + \mu_{2,v}/\sigma^2_{2,v}}{1/\sigma^2_{1,v} + 1/\sigma^2_{2,v}} \tag{5.15}$$

v denotes an index of the parameter vector. $\mu_{\cdot,v}$ and $\sigma^2_{\cdot,v}$ are scalar.

For MoG with two components in K dimension, the number of modes can be at most $K+1$ (Ray and Ren, 2012). Therefore, it is hard to find all modes in high-dimensional Gaussian in general.

The property of critical points of a MoG with two components can be extended to the case of K components. The following equation holds:

$$\theta = (\sum_{k=1}^{K} \alpha_k\Sigma_k^{-1})^{-1}(\sum_{k=1}^{K} \alpha_k\Sigma_k^{-1}\mu_k), \tag{5.16}$$

where $0 < \alpha_k < 1$ for all k and $\sum_k \alpha_k = 1$. There is no tight upper bound on the number of modes of MoG in general. There is a guess that, for all $D, K \geq 1$, the upper bound is $D + K - 1$ choose D (Améndola et al., 2017).

5.3 Transfer Techniques for Incremental Moment Matching

In general, the loss function of neural networks is not convex. If the parameters of two neural networks initialized independently are averaged, it might perform poorly. This results from high cost barriers between the two parameters (Goodfellow et al., 2014). However, we will show that various transfer learning techniques can be used to ease this problem, and make the assumption of Gaussian distribution for neural networks reasonable. In this section, we introduce three practical techniques for IMM, including weight-transfer, L2-transfer, and drop-transfer.

5.3.1 Weight-Transfer

Weight-transfer initialize the parameters for the new task μ_k with the parameters of the previous task μ_{k-1} (Yosinski et al., 2014). In our experiments, the use of weight-transfer was critical to the lifelong learning performance. For this reason, the experiments on IMM in this paper use the weight-transfer technique as a default.

The weight-transfer technique is motivated by the geometrical property of neural networks discovered in the previous work (Goodfellow et al., 2014). They found that there is a straight path from the initial point to the solution without any high cost barrier, in various types of neural networks and datasets. This discovery suggests that the weight-transfer from the previous task to the new task makes a smooth loss surface between two solutions for the tasks, so that the optimal solution for both tasks lies on the interpolated point of the two solutions.

To empirically validate the concept of weight-transfer, we use the linear path analysis proposed by Goodfellow et al. (Goodfellow et al., 2014) (Figure 5.2).

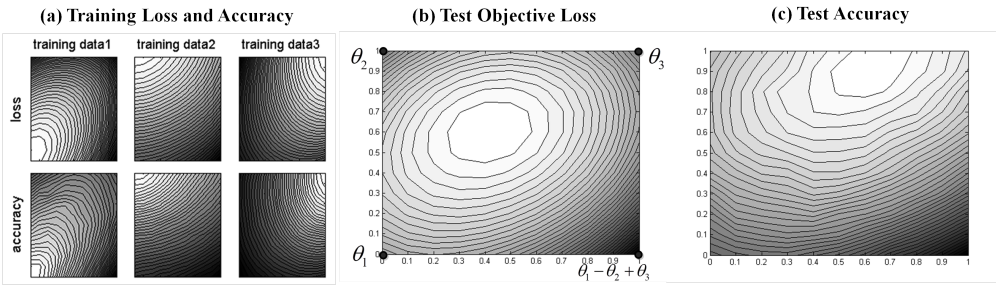

Figure 5.2 Experimental results on visualizing the effect of weight-transfer.

We randomly chose 18,000 instances from the training dataset of CIFAR-10, and divided them into three subsets of 6,000 instances each. These three subsets are used for sequential training by CNN models, parameterized by θ_1, θ_2, and θ_3, respectively. Here, θ_2 is initialized from θ_1, and then θ_3 is initialized from θ_2, in the same way as weight-transfer. In the analysis, each loss and accuracy is evaluated at a series of points $\theta = \theta_1 + \alpha(\theta_2 - \theta_1) + \beta(\theta_3 - \theta_2)$, varying α and β. In Figure 5.2, the loss surface of the model on each online subset is nearly convex.

The geometric property of the parameter space of the neural network is analyzed. Brighter is better. θ_1, θ_2, and θ_3 are the points of trained networks from randomly selected subsets of the CIFAR-10 dataset. This figure shows that there are better solutions between the three locally optimized parameters.

The figure shows that the parameter at $\frac{1}{3}(\theta_1 + \theta_2 + \theta_3)$, which is the same as the solution of mean-IMM, performs better than any other reference points θ_1, θ_2, or θ_3. However, when θ_2 is not initialized by θ_1, the convex-like shape disappears, since there is a high cost barrier between the loss function of θ_1 and θ_2.

5.3.2 L2-transfer

L2-transfer is a variant of L2-regularization. L2-transfer can be interpreted as a special case of EWC where the prior distribution is Gaussian with λI as a covariance matrix. In L2-transfer, a regularization term of the distance between μ_{k-1} and μ_k is added to the following objective function for finding μ_k, where λ is a hyperparameter:

$$\log p(y_k|X_k, \mu_k) - \lambda \cdot ||\mu_k - \mu_{k-1}||_2^2 \tag{5.17}$$

The concept of L2-transfer is commonly used in transfer learning (Evgeniou and Pontil, 2004; Kienzle and Chellapilla, 2006) and lifelong learning (Li and Hoiem, 2016; Kirkpatrick et al., 2017) with large λ. Unlike the previous usage of large λ, we use small λ for the IMM procedure. In other words, μ_k is first trained by Equation 5.17 with small λ, and then merged to $\mu_{1:k}$ in our IMM. Since we want to make the loss surface between μ_{k-1} and μ_k smooth, and not to minimize the distance between μ_{k-1} and μ_k. In convex optimization, the L2-regularizer makes the convex function strictly convex. Similarly, we hope L2-transfer with small λ will help find a μ_k with a convex-like loss space between μ_{k-1} and μ_k.

5.3.3 Drop-transfer

Drop-transfer is a novel method devised in this paper. Drop-transfer is a variant of dropout where μ_{k-1} is the zero point of the dropout procedure. In the training phase, the following $\hat{\mu}_{k,i}$ is used for the weight vector corresponding to the ith node $\mu_{k,i}$:

$$\hat{\mu}_{k,i} = \begin{cases} \mu_{k-1,i}, & \text{if } i\text{th node is turned off} \\ \frac{1}{1-p} \cdot \mu_{k,i} - \frac{p}{1-p} \cdot \mu_{k-1,i}, & \text{otherwise} \end{cases} \tag{5.18}$$

Table 5.1 The averaged accuracies on the disjoint MNIST for two sequential tasks. For IMM with transfer, only α is tuned. The numbers in the parentheses refer to standard deviation. Every IMM uses weight-transfer.

Disjoint MNIST Experiment	Explanation of Hyperparam	Natural Hyperparam	Natural Accuracy
SGD (Goodfellow et al., 2013)	epoch per dataset	10	47.72 (± 0.11)
L2-transfer (Evgeniou and Pontil, 2004)	λ in (5.17)	-	-
Drop-transfer	p in (5.18)	0.5	51.72 (± 0.79)
EWC (Kirkpatrick et al., 2017)	λ in (2.5)	1.0	47.84 (± 0.04)
Mean-IMM	α_2 in (5.4)	0.50	90.45 (± 2.24)
Mode-IMM	α_2 in (5.7)	0.50	91.49 (± 0.98)
L2-transfer + Mean-IMM	λ / α_2	0.001 / 0.50	78.34 (± 1.82)
L2-transfer + Mode-IMM	λ / α_2	0.001 / 0.50	92.52 (± 0.41)
Drop-transfer + Mean-IMM	p / α_2	0.5 / 0.50	80.75 (± 1.28)
Drop-transfer + Mode-IMM	p / α_2	0.5 / 0.50	93.35 (± 0.49)
L2, Drop-transfer + Mean-IMM	λ / p / α_2	0.001 / 0.5 / 0.50	66.10 (± 3.19)
L2, Drop-transfer + Mode-IMM	λ / p / α_2	0.001 / 0.5 / 0.50	**93.97** (± 0.32)

Disjoint MNIST Experiment	Explanation of Hyperparam	Tuned Hyperparam	Tuned Accuracy
SGD (Goodfellow et al., 2013)	epoch per dataset	0.05	71.32 (± 1.54)
L2-transfer (Evgeniou and Pontil, 2004)	λ in (5.17)	0.05	85.81 (± 0.52)
Drop-transfer	p in (5.18)	0.5	51.72 (± 0.79)
EWC (Kirkpatrick et al., 2017)	λ in (2.5)	600M	52.72 (± 1.36)
Mean-IMM	α_2 in (5.4)	0.55	91.92 (± 0.98)
Mode-IMM	α_2 in (5.7)	0.45	92.02 (± 0.73)
L2-transfer + Mean-IMM	λ / α_2	0.001 / 0.60	92.62 (± 0.95)
L2-transfer + Mode-IMM	λ / α_2	0.001 / 0.45	92.73 (± 0.35)
Drop-transfer + Mean-IMM	p / α_2	0.5 / 0.60	92.64 (± 0.60)
Drop-transfer + Mode-IMM	p / α_2	0.5 / 0.50	93.35 (± 0.49)
L2, Drop-transfer + Mean-IMM	λ / p / α_2	0.001 / 0.5 / 0.75	**93.97** (± 0.23)
L2, Drop-transfer + Mode-IMM	λ / p / α_2	0.001 / 0.5 / 0.45	**94.12** (± 0.27)

where p is the dropout ratio. Notice that the expectation of $\hat{\mu}_{k,i}$ is $\mu_{k,i}$.

There are studies (Srivastava et al., 2014; Baldi and Sadowski, 2013) that have interpreted dropout as an exponential ensemble of weak learners. By this perspective, as the marginalization of output distribution over the whole weak learner is intractable, the parameters multiplied by the inverse of the dropout rate are used for the test phase in the procedure. In other words, the parameters of the weak learners are, in effect, simply averaged oversampled learners by dropout. At the process of drop-transfer in our lifelong learning setting, we hypothesize that the dropout process makes the averaged point of two arbitrary

Table 5.2 The averaged accuracies on the shuffled MNIST for three sequential tasks. The results of SGD and EWC with tuned setting is from (Kirkpatrick et al., 2017)

Shuffled MNIST Experiment	Explanation of Hyperparam	Natural Hyperparam	Natural Accuracy
SGD (Goodfellow et al., 2013)	epoch per dataset	60	89.15 (± 2.34)
L2-transfer (Evgeniou and Pontil, 2004)	λ in (5.17)	-	-
Drop-transfer	p in (5.18)	0.5	94.75 (± 0.62)
EWC (Kirkpatrick et al., 2017)	λ in (2.5)	-	-
Mean-IMM	α_3 in (5.4)	0.33	93.23 (± 1.37)
Mode-IMM	α_3 in (5.7)	0.33	98.02 (± 0.05)
L2-transfer + Mean-IMM	λ / α_3	1e-4 / 0.33	90.38 (± 1.74)
L2-transfer + Mode-IMM	λ / α_3	1e-4 / 0.33	**98.16 (± 0.08)**
Drop-transfer + Mean-IMM	p / α_3	0.5 / 0.33	90.79 (± 1.30)
Drop-transfer + Mode-IMM	p / α_3	0.5 / 0.33	97.80 (± 0.07)
L2, Drop-transfer + Mean-IMM	λ / p / α_3	1e-4 / 0.5 / 0.33	89.51 (± 2.85)
L2, Drop-transfer + Mode-IMM	λ / p / α_3	1e-4 / 0.5 / 0.33	97.83 (± 0.10)

Shuffled MNIST Experiment	Explanation of Hyperparam	Tuned Hyperparam	Tuned Accuracy
SGD (Goodfellow et al., 2013)	epoch per dataset	-	∼95.5
L2-transfer (Evgeniou and Pontil, 2004)	λ in (5.17)	1e-3	96.37 (± 0.62)
Drop-transfer	p in (5.18)	0.2	96.86 (± 0.21)
EWC (Kirkpatrick et al., 2017)	λ in (2.5)	-	**∼98.2**
Mean-IMM	α_3 in (5.4)	0.55	95.02 (± 0.42)
Mode-IMM	α_3 in (5.7)	0.60	98.08 (± 0.08)
L2-transfer + Mean-IMM	λ / α_3	1e-4 / 0.65	95.93 (± 0.31)
L2-transfer + Mode-IMM	λ / α_3	1e-4 / 0.60	**98.30 (± 0.08)**
Drop-transfer + Mean-IMM	p / α_3	0.5 / 0.65	96.49 (± 0.44)
Drop-transfer + Mode-IMM	p / α_3	0.5 / 0.55	97.95 (± 0.08)
L2, Drop-transfer + Mean-IMM	λ / p / α_3	1e-4 / 0.5 / 0.90	97.36 (± 0.19)
L2, Drop-transfer + Mode-IMM	λ / p / α_3	1e-4 / 0.5 / 0.50	97.92 (± 0.05)

sampled points using Equation 5.18 a good estimator.

We investigated the search space of the loss function of the MLP trained from the MNIST handwritten digit recognition dataset for with and without dropout regularization, to supplement the evidence of the described hypothesis. Dropout regularization makes the accuracy of a sampled point from dropout distribution and an average point of two samples, from 0.450 (± 0.084) to 0.950 (± 0.009) and 0.757 (± 0.065) to 0.974 (± 0.003), respectively. For the case of both with and without dropout, the space between two arbitrary samples is empirically convex, and fits the second-order equation. Based on this exper-

Algorithm 9 IMM with weight-transfer, L2-transfer

Input: data $\{(X_1, y_1),...,(X_K, y_K)\}$, balancing hyperparameter α
Output: $w_{1:K}$
$w_0 \leftarrow$ InitializeNN()
for $k = 1{:}K$ **do**
  $w_{k*} \leftarrow w_{k-1}$
  Train(w_{k*}, X_k, y_k) with $L(w_{k*}, X_k, y_k) + \lambda \cdot ||w_{k*} - w_{k-1}||_2^2$
  if type is mean-IMM **then**
    $w_{1:k} \leftarrow \sum_t^k \alpha_t w_{t*}$
  else if type is mode-IMM **then**
    $F_{k*} \leftarrow$ CalculateFisherMatrix(w_{k*}, X_k)
    $\Sigma_{1:k} \leftarrow (\sum_t^k \alpha_t F_{t*})^{-1}$
    $w_{1:k} \leftarrow \Sigma_{1:k} \cdot (\sum_t^k \alpha_t F_{t*} w_{t*})$
  end if
end for

iment, we expect not only that the search space of the loss function between modern neural networks can easily be nearly convex (Goodfellow et al., 2014), but also that regularizers, such as dropout, make the search space smooth and the point in the search space has a good accuracy in lifelong learning.

5.3.4 IMM Procedure

Two moment matching methods: mean-IMM and mode-IMM, and three transfer learning techniques: weight-transfer, L2-transfer, and drop-transfer, are combined to make a variety of lifelong learning algorithms in our study. Algorithm 9 describes mean-IMM and mode-IMM with weight-transfer and L2-transfer.

5.4 Experimental Results

We evaluate our approach on four experiments, which settings are intensively used in the previous works (Srivastava et al., 2013; Kirkpatrick et al., 2017; Li and Hoiem, 2016; Lee et al., 2016).

5.4.1 Disjoint MNIST Experiment

The first experiment is the disjoint MNIST experiment (Srivastava et al., 2013). In this experiment, the MNIST dataset is divided into two datasets: the first dataset consists of only digits {0, 1, 2, 3, 4} and the second dataset consists of the remaining digits {5, 6, 7, 8, 9}. Our task is 10-class joint categorization, unlike in the previous work, which considers two independent tasks of 5-class categorization. Because the inference should include the decision whether a new instance comes from the first or the second task, our task is more difficult than the task from the previous work.

We evaluate the models both on the natural setting and the tuned setting. The natural setting refers to the most natural hyperparameter in the equation of each algorithm. The tuned setting refers to using heuristic hand-tuned hyperparameters. Consider that tuned hyperparameter-setting is often used in previous works of lifelong learning as it is difficult to define a validation set in their setting. For example, when the model needs to learn from the new task after learning from the old task, a low learning rate or early stopping without a validation set, or arbitrary hyperparameter for balancing is used (Goodfellow et al., 2013; Kirkpatrick et al., 2017). We discover hyperparameters in the tuned setting not only for finding the oracle performance of each algorithm, but also for showing that there exist some paths consisting of the point that performs reasonably for both tasks. Hyperparam in Table 5.1 and 5.2 denotes hyperparameter mainly searched in the tuned setting. Table 5.1 and Figure 5.3 (Left) shows the experimental results from the disjoint MNIST experiment.

The natural setting refers to the most natural hyperparameter in the equation of each algorithm, whereas the tuned setting refers to using heuristic hand-tuned hyperparameters. Hyperparam denotes the main hyperparameter of each

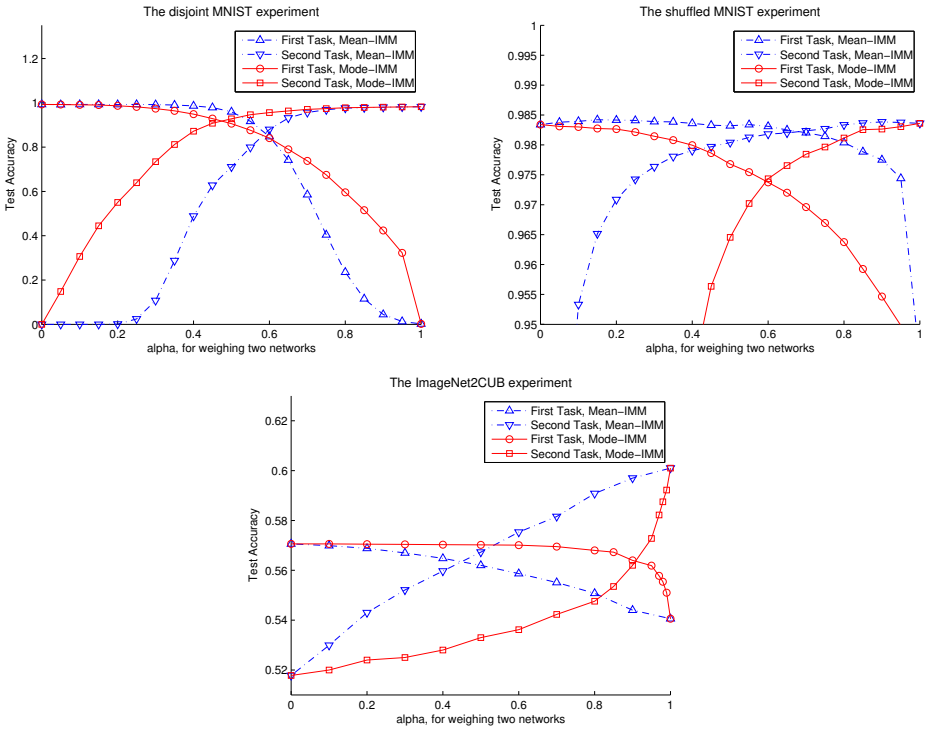

Figure 5.3 Test accuracies of two IMM models with weight-transfer on the disjoint MNIST (Left), the shuffled MNIST (Middle), and the ImageNet2CUB experiment (Right).

algorithm.

α is a hyperparameter that balances the information between the old and the new task.

We first explain the natural setting and the tuned setting in detail. The natural setting refers to the most natural hyperparameter in the equation of each algorithm, whereas the tuned setting refers to using heuristic hand-tuned hyperparameters. For mean-IMM, it is most natural to evenly average K models and $1/K$ is the most natural α_k value for K sequential tasks. For EWC, 1 is the most natural λ value in Equation 2.4, because EWC is derived from the equation of sequential Bayesian. For L2-transfer, there is no natural hyperparameter

value in Equation 5.17, so we need to heuristically choose a λ value, which is not too small but does not damage the performance of the new network for the new task.

In the SGD, epoch per dataset for the second task corresponds to the hyperparameter. The unit is how much of the network is trained from the whole data at once. In the L2-transfer and EWC, λ in Equations 5.17 and 2.4 corresponds to their hyperparameter, respectively. In the mean-IMM and mode-IMM, α_K in Equations 5.4 and 5.7 corresponds to the hyperparameter, respectively. In the drop-transfer, dropout ratio p in Equation 5.18 corresponds to the hyperparameter.

All of the explained hyperparameters are devised to balance the information between the old and new tasks. If $\lambda/(1+\lambda) = 1$ or $\alpha_1 = 1$, the final network of the algorithms is the same as the network for the first task. If $1/(1+\lambda) = 1$ or $\alpha_K = 1$, the final network is the same as the network for the last task.

We used multi-layer perceptrons (MLP) with [784-800-800-10] as the number of nodes, ReLU as the activation function, and vanilla SGD as the optimizer for all of the experiments. We set the epoch per dataset to 10, unless otherwise noted. The entire IMM model uses weight-transfer to smooth the loss surface of the model. Without weight-transfer, our IMM model does not work at all. In our experiments, all models only use one 10-way softmax output layer. In only SGD, dropout is used as proposed in Goodfellow et al. (Goodfellow et al., 2013), but dropout does not help much.

Each accuracy was measured by repeating 10 experiments. In the experiment, IMM outperforms comparative models by a significant margin. In the tuned experiment, the performance of the IMM models exceeds 90%, and the performance increases more when more transfer techniques are applied. Among all the models, weight-transfer + L2-transfer + drop-transfer + mode-IMM

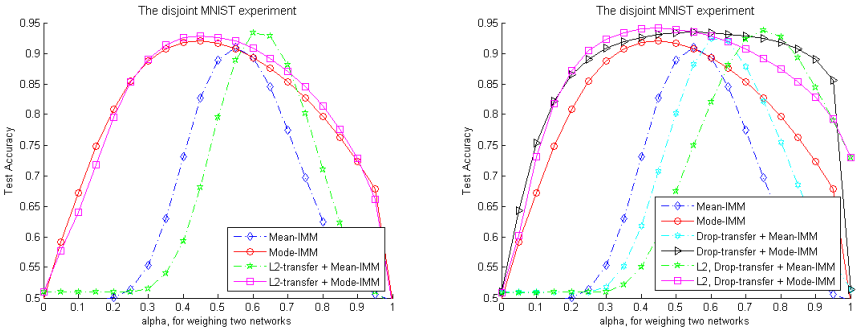

Figure 5.4 Test accuracies of IMM with various transfer techniques on the disjoint MNIST.

performs the best and at greater than 94%. However, the comparative models fail to reach greater than 90%. Existing regularizer including dropout does not improve the comparative models. In our experimental setting, the usual SGD-based optimizers always perform less than 50%, because the biases of the output layer for the old task are always pushed to large negative values, which implies that our task is extremely hard. Figure 5.4 also shows that mode-IMM is robust with α and the optimal α of mean-IMM is larger than 1/2 in the disjoint MNIST experiment.

Both L2-transfer and drop-transfer boost the performance of IMM and make the optimal value of α larger than 1/2. However, drop-transfer tends to make the accuracy curve more smooth than L2-transfer does.

5.4.2 Shuffled MNIST Experiment

The second experiment is the shuffled MNIST experiment (Goodfellow et al., 2013; Kirkpatrick et al., 2017) of three sequential tasks. In this experiment, the first dataset is the same as the original MNIST dataset. However, in the second dataset, the input pixels of all images are shuffled with a fixed, random permutation. In previous work, EWC reaches the performance level of the

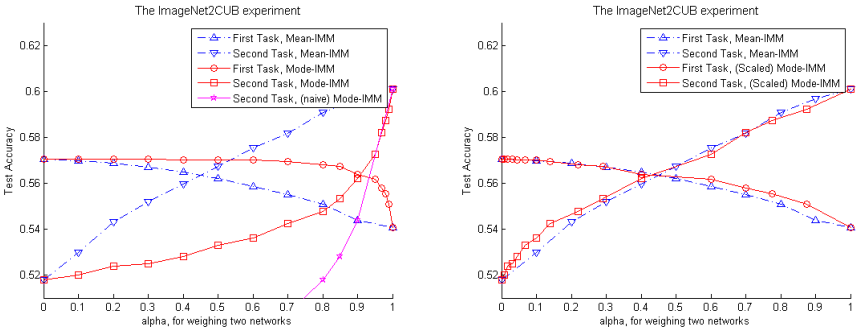

Figure 5.5 (Left) Illustration of the effect of the strategy of re-weighing on the new last-layer. (Right) The results of mode-IMM with changing the balancing hyperparameter α to the re-scaled balancing hayperparameter $\hat{\alpha}$ with the scale of the Fisher matrix of each network.

batch learner, and it is argued that EWC overcomes catastrophic forgetting in some domains. The experimental details are similar to the disjoint MNIST experiment, except all models are allowed to use dropout regularization. In the experiment, the first dataset is the same as the original MNIST dataset. However, in the second and the third dataset, the input pixels of all images are shuffled with a fixed, random permutation. Therefore, the difficulty of the three datasets is the same, though a different solution is required for each dataset.

Table 5.2 and Figure 5.3 (Middle) shows the experimental results from the shuffled MNIST experiment. Notice that accuracy of drop-transfer ($p = 0.2$) alone is 96.86 ($\pm$ 0.21) and L2-transfer (λ = 1e-4) + drop-transfer ($p = 0.4$) alone is 97.61 ($\pm$ 0.15). These results are competitive to EWC without dropout, whose performance is around 97.0.

The second experiment is the shuffled MNIST experiment for three sequential tasks. For the hyperparameter of IMM, we set α_1 and α_2 as the same value, and tune only α_3. Table 5.2 shows the experimental results. The performance of SGD + dropout and EWC + dropout comes from the report in (Kirkpatrick

et al., 2017). Changing only the epoch does not increase the performance by much in SGD. Results show that our IMM paradigm does work and performs similarly to EWC in a case where it is known that EWC performs well. Dropout regularization in the task makes both our models and comparative models perform better.

In our IMM framework, weight-transfer, L2-transfer, and drop-transfer all takes μ_{k-1} as the reference models of the transfer for training μ_k. In other words, weight-transfer initializes μ_k with μ_{k-1}, L2-transfer uses a regularization term to minimize the Euclidean distance between μ_{k-1} and μ_k, and drop-transfer uses a μ_{k-1} as the zero point of the dropout procedure. All three transfer techniques can be considered to change the reference point to, for example, $\mu^{mean}_{1:(k-1)}$ or $\mu^{mode}_{1:(k-1)}$, as previous works do (Kirkpatrick et al., 2017). However, these alternatives all make worse performances in our shuffled MNIST experiment. We argued that our utilization of transfer techniques are devised not to minimize the distance between μ_{k-1} and μ_k, but to help find a μ_k with a smooth and convex-like loss space between μ_{k-1} and μ_k. This interpretation seems to be related to the result.

5.4.3 ImageNet to CUB Dataset

The third experiment is the ImageNet2CUB experiment (Li and Hoiem, 2016), the lifelong learning problem from the ImageNet dataset to the Caltech-UCSD Birds-200-2011 fine-grained classification (CUB) dataset (Wah et al., 2011). The number of classes of ImageNet and CUB dataset is around 1K and 200, and the number of training instances 1M and 5K, respectively. In the ImageNet2CUB experiment, the last-layer is separated for the ImageNet and the CUB task. The structure of AlexNet is used for the trained model of ImageNet (Krizhevsky et al., 2012). In our experiment, we match the moments of the last-layer fine-

tuning model and the LwF model, with mean-IMM and mode-IMM.

Figure 5.3 (Right) shows that mean-IMM moderately balances the performance of two tasks between two networks. However, the balance point of mode-IMM is far from $\alpha = 0.5$. It is because the scale of the Fisher matrix F is different between the ImageNet and the CUB task. As the number of training data of the two tasks is different, it is natural that the mean of the square of the gradient, which is the definition of F, is different. This implies that the assumption of mode-IMM does not always hold for heterogeneous tasks. See Appendix D.3 for more information including learning methods of IMM in the case of a different class output layer and a different scale of datasets.

Our results of IMM with LwF exceed the previous state-of-the-art performance, whose model is also LwF. This is because, in the previous works, the LwF model is initialized by the last-layer fine-tuning model, not directly by the original AlexNet. In this case, the performance loss of the old task is decreased, but the performance gain of the new task is also decreased. The accuracies of our mean-IMM ($\alpha = 0.5$) are 56.20 and 56.73 for the ImageNet task and the CUB task, respectively. The gains compared to the previous state-of-the-art are +1.13 and -1.14. In the case of mean-IMM ($\alpha = 0.8$) and mode-IMM ($\alpha = 0.99$), the accuracies are 55.08 and 59.08 (+0.01, +1.12), and 55.10 and 59.12 (+0.02, +1.35), respectively.

In cases where a different class output layer of each task, IMM requires additional techniques. There is no counterpart in the last-layer of the first network representing the second task, or the second last-layer of the first network. To tackle this problem, we add the training process of the last-layer fine-tuning model to the IMM procedure; we match the moments of the last-layer fine-tuning model and the original new network for the new task. Last-layer fine-tuning is the model the last-layer is only fine-tuned for each new task; thus it

does not make a performance loss for the first task, but does not often learn enough for new tasks.

The technique utilizing the last-layer fine-tuning model makes mean-IMM work in the case of different class output layers, but it is not enough for mode-IMM. It is not possible to calculate a proper Fisher matrix of the second last-layer in the first network for the first dataset. As the Fisher matrix is defined with the gradient from the loss of the first task, elements of the Fisher matrix have a value of zero. However, a zero matrix not only is what we want but also degenerates the performance of mode-IMM. To tackle this problem, we apply mean-IMM for the last-layer with a re-scaling. We change the mixing ratios $\alpha_1 : \alpha_2$ to $\hat{\alpha}_1 : \hat{\alpha}_2 = \alpha_1 : \alpha_2 \cdot \frac{|\hat{w}_1|}{|\hat{w}_1|+|\hat{w}_2|}$ for the re-scaling, where $|\hat{w}_1|$ and $|\hat{w}_2|$ is the average of the whole element of weight matrix in the layer before the last-layer, in the first and the second task.

In our ImageNet2CUB experiment, the moments of the last-layer fine-tuning model and the LwF model are matched. Though LwF does not perform well in our previous experiments, it is known that LwF performs well when the size of a new dataset is small relative to the old dataset, as is in the ImageNet2CUB experiment.

Figure 5.5 (Left) compares the performances of mode-IMM models with different assumptions on the Fisher matrix. In naïve mode-IMM, the Fisher matrix of the second last-layer of the first network is a zero matrix. In other words, the second last-layer of the final naïve mode-IMM is the second last-layer of the second network. Naïve mode-IMM does not yield a good performance as we expect.

Mode-IMM refers to the original mode-IMM devised for the ImageNet2CUB experiments. In naïve mode-IMM, the second last-layer of the second network is used for the second last-layer of the final IMM model.

Table 5.3 Experimental results on the Lifelog dataset. Mean-IMM uses weight-transfer. Classification accuracies among different classes (Top) and different subjects (Bottom).

Algorithm	Location	Sub-location	Activity
Dual memory architecture (Lee et al., 2016)	**78.11**	72.36	52.92
Mean-IMM	77.60	73.78	52.74
Mode-IMM	77.14	**75.76**	**54.07**
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

Algorithm	A	B	C
Dual memory architecture (Lee et al., 2016)	67.02	58.80	77.57
Mean-IMM	67.03	57.73	**79.35**
Mode-IMM	**67.97**	**60.12**	78.89
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

In Figure 5.5, scaled mode-IMM denotes the results of mode-IMM re-plotted by the $\hat{\alpha}$ as we defined above. The result shows that re-scaled mode-IMM performs similarly to mean-IMM in the ImageNet2CUB experiment.

5.4.4 Lifelog Dataset

Lastly, we evaluate the proposed mean-IMM on the Lifelog dataset (Lee et al., 2016). The Lifelog dataset consists of 660,000 instances of egocentric video stream data, collected over 46 days from three participants using Google Glass (Lee et al., 2017c). Three class categories, location, sub-location, and activity, are labeled on each frame of video. In the Lifelog dataset, the class distribution changes continuously and new classes appear as the day passes. Table 5.3 shows that mean-IMM and mode-IMM are competitive to the dual memory architec-

ture, the previous state-of-the-art ensemble model, even though IMM uses only one network.

In the experiment, our IMM paradigm achieves competitive results with the approach using an ensemble network, without additional cost for inference and learning.

The Lifelog dataset is the dataset recorded from Google Glass over 46 days from three participants. The 660,000 seconds of the egocentric video stream data reflects the behaviors of the participants. The dataset consists of 10 days of training data and 4 days of test data in order of time for each participant respectively. In the framework of Lee et al. (Lee et al., 2016), the network can be updated every day, but a new network can be made for the 3rd, 7th, and 10th day, with training data of 3, 4, and 3 days, respectively. Following this framework, our network is made in the 3rd, 7th, and 10th day, and then merged to previously trained networks. Our IMM used AlexNet pretrained by the ImageNet dataset (Krizhevsky et al., 2012) as the initial network. The experimental results on the Lifelog dataset are in Table 5.3, where the performance of models is from Lee et al. (Lee et al., 2016) except IMM.

5.5 Discussion

5.5.1 A Shift of Optimal Hyperparameter via Space Smoothing

The tuned setting shows there often exists some α which makes the performance of the mean-IMM close to the mode-IMM. However, in the natural hyperparameter setting, mean-IMM performs worse when more transfer techniques are applied. This is because an assumption of the mean-IMM is broken in that the training of the network is only affected by the current task or the prior for the task is rarely informed by the previous task. Fortunately, mode-IMM works more robustly than mean-IMM where transfer techniques are applied.

Figure 5.4 in Appendix D.1 illustrates the change of the test accuracy curve corresponding to the applied transfer techniques and a following shift of the optimal α in mean-IMM and mode-IMM.

5.5.2 Bayesian Approach on lifelong learning.

Kirkpatrick et al. (Kirkpatrick et al., 2017) interpreted that the Fisher matrix F as weight importance in explaining their EWC model. In the shuffled MNIST experiment, since a large number of pixels always has a value of zero, the corresponding element of the Fisher matrix is also zero. Therefore, EWC does work by allowing weights to change, which are not used in the previous task. On the other hand, mode-IMM also works by selectively balancing between two weights using variance information. However, these assumptions on weight importance do not always hold, especially in the disjoint MNIST experiment. The most important weight in the disjoint MNIST experiment is the bias term in the output layer. Nevertheless, these elements of the Fisher matrix are not guaranteed to be the highest value nor can they be used to balance the class distribution between the first and second task. We believe that using only the diagonal term for the covariance matrix in Bayesian neural networks is too naïve in general and that this is why EWC failed in the disjoint MNIST experiment. We think this could be alleviated in future work by using a more complex prior, such as a matrix Gaussian distribution while assuming correlations between nodes in the network (Louizos and Welling, 2016).

5.5.3 Balancing the Information of an Old and a New Task.

The IMM procedure produces a neural network without a performance loss for kth task μ_k, which is better than the final solution $\mu_{1:k}$ in terms of the performance of the kth task. Furthermore, IMM can easily weigh the importance

of tasks in IMM models in real time. For example, α_t can be easily changed for the solution of mean-IMM $\mu_{1:k} = \sum_t^k \alpha_t \mu_t$. In actual service situations of IT companies, the importance of the old and the new task frequently changes in real time, and IMM can handle this problem. This property differentiates IMM to other lifelong learning methods using the regularization approach, including LwF and EWC.

5.6 Summary

The contributions of the section are four fold. First, we applied mean-IMM to the lifelong learning of modern deep neural networks. Mean-IMM makes competitive results to comparative models and balances the information between an old and a new network. We also interpreted the success of IMM by the Bayesian framework with Gaussian posterior. Second, we extended mean-IMM to mode-IMM with the interpretation of mode-finding at the mixture of Gaussian posterior. Mode-IMM outperforms mean-IMM and comparative models in a variety of datasets. Third, we introduced drop-transfer, a novel method devised in the paper. Experimental results showed that drop-transfer alone performs well and is similar to the EWC without dropout, in the domain which it was argued that EWC rarely forgets. Fourth, We applied various transfer techniques by the IMM procedure to make our assumption of Gaussian distribution reasonable. We argued that not only the search space of the loss function among neural networks can easily be nearly convex, but also regularizers, such as dropout, make the search space smooth and the point in the search space have a good accuracy. Experimental results showed that applying transfer techniques often boost the performance of IMM. Overall, we made state-of-the-art performance in a variety of datasets of lifelong learning and explored geometrical properties and a Bayesian perspective of modern deep neural networks.

Chapter 6

Concluding Remarks

6.1 Summary of Methods and Contributions

In the dissertation, we proposed two algorithms for overcoming catastrophic forgetting. First, we presented dual memory architecture (DMA) for online deep learning of user behavior in everyday life using a wearable device. In the method, We utilize the concept of complementary learning systems theory - contending that effective learning of the data stream in a lifetime requires complementary systems that comprise the neocortex and hippocampus in the human brain. A dual memory architecture (DMA) trains two learning structures: one gradually acquires structured knowledge representations, and the other rapidly learns the specifics of individual experiences. Our experimental results showed that the proposed method overcomes catastrophic forgetting in the learning of real non-stationary data. This property was utilized for implementing advanced personalized context recognition system.

Second, we proposed incremental moment matching (IMM), which uses

Bayesian neural networks to merge many neural networks. In the study, We discussed not only how moment matching works but also how three transfer techniques in the paper make our Gaussian assumption reasonable. Four experimental results showed that IMM achieves state-of-the-art performance in a variety of datasets. It was also shown that our IMM procedure performs better, by applying the methods proposed in the previous works, including dropout regularization and L2-transfer. We believe our IMM could also be applied together with other methods, including EWC and PathNet, to boost the performance further. Using our lifelong learning setting, we discovered geometrical properties and a Bayesian perspective of modern deep neural networks.

6.2 Suggestions for Future Research

There are numerous directions for the future works on the problem we solved. We mention some possible directions for the future research.

Learning with Few Labels

Lifelong learning with few labels is a more plausible situation in practice (Nichol and Schulman, 2018). For example, in the case of lifelog dataset, it is not plausible to get the labeled data every day because it is too expensive to label every day. In the ideal lifelong learning, the algorithm should learn from the data stream with fewer labels or no label. It is the case where unsupervised learning or semi-supervised learning is required along with lifelong learning. Learning algorithms for two kinds of learning can be used together to solve this problem.

Active Learning

Active learning can also be used for lifelong learning if the agent can interact with the human or environment. For example, in the home-service robot situation, the robot can ask the user whether he knows the concept correctly. we believe the task-oriented dialogue algorithm can be used to achieve this kind of active learning, where the goal is to know the correct label of instances in environment (Han et al., 2017; Lee et al., 2018). Though it is known to be difficult to be achieved by deep neural networks, we hope that some variants ease the problem in the future work.

Bibliography

Améndola, C., Engström, A., and Haase, C. (2017). Maximum number of modes of gaussian mixtures. *arXiv preprint arXiv:1702.05066*.

Amer, M. R. and Todorovic, S. (2012). Sum-product networks for modeling activities with stochastic structure. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1314–1321. IEEE.

Baldi, P. and Sadowski, P. J. (2013). Understanding dropout. In *Advances in Neural Information Processing Systems*, pages 2814–2822.

Bendor, D. and Wilson, M. A. (2012). Biasing the content of hippocampal replay during sleep. *Nature neuroscience*, 15(10):1439–1444.

Bengio, Y. et al. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

Bettadapura, V., Essa, I., and Pantofaru, C. (2015). Egocentric field-of-view localization using first-person point-of-view devices. In *IEEE Winter Conference on Applications of Computer Vision*, pages 626–633.

Blackwell, D. and MacQueen, J. B. (1973). Ferguson distributions via pólya urn schemes. *The annals of statistics*, pages 353–355.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1613–1622.

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., Rae, J., Wierstra, D., and Hassabis, D. (2016). Model-free episodic control. *arXiv preprint arXiv:1606.04460*.

Bottou, L. (1998). *Online learning and stochastic approximations*, chapter 2, pages 9–42. Cambridge University Press.

Broderick, T., Boyd, N., Wibisono, A., Wilson, A. C., and Jordan, M. I. (2013). Streaming variational bayes. In *Advances in Neural Information Processing Systems*, pages 1727–1735.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, E. R., and Mitchell, T. M. (2010). Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, pages 1306–1313.

Carr, M. F., Jadhav, S. P., and Frank, L. M. (2011). Hippocampal replay in the awake state: a potential substrate for memory consolidation and retrieval. *Nature neuroscience*, 14(2):147–153.

Chen, T., Goodfellow, I., and Shlens, J. (2016). Net2net: Accelerating learning via knowledge transfer. In *International Conference on Learning Representations*.

Chen, X., Shrivastava, A., and Gupta, A. (2013). Neil: Extracting visual knowledge from web data. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1409–1416.

Chickering, D. M., Heckerman, D., and Meek, C. (1997). A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 80–89. Morgan Kaufmann Publishers Inc.

Darwiche, A. (2003). A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305.

Dechter, R. and Mateescu, R. (2007). And/or search spaces for graphical models. *Artificial intelligence*, 171(2-3):73–106.

Delalleau, O. and Bengio, Y. (2011). Shallow vs. deep sum-product networks. In *Advances in Neural Information Processing Systems*, pages 666–674.

Dennis, A. and Ventura, D. (2012). Learning the architecture of sum-product networks using clustering on variables. In *Advances in Neural Information Processing Systems*, pages 2033–2041.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning*, pages 647–655.

Doshi, J., Kira, Z., and Wagner, A. (2015). From deep learning to episodic memories: Creating categories of visual experiences. In *Proceedings of the Third Annual Conference on Advances in Cognitive Systems ACS*.

Evgeniou, T. and Pontil, M. (2004). Regularized multi–task learning. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 109–117. ACM.

Fei-Fei, L., Fergus, R., and Perona, P. (2006). One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611.

Fernando, C., Banarse, D., Blundell, C., Zwols, Y., Ha, D., Rusu, A. A., Pritzel, A., and Wierstra, D. (2017). Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*.

French, R. M. (1999). Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Gens, R. and Domingos, P. (2012). Discriminative learning of sum-product networks. In *Advances in Neural Information Processing Systems*, pages 3239–3247.

Gens, R. and Pedro, D. (2013). Learning the structure of sum-product networks. In *International Conference on Machine Learning*, pages 873–880.

Ghahramani, Z. (2000). Online variational bayesian learning. In *NIPS workshop on Online Learning*.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. (2014). Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*.

Graves, A., Mohamed, A.-r., and Hinton, G. (2013). Speech recognition with deep recurrent neural networks. In *IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649.

Graves, A., Wayne, G., and Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Guyonneau, R., VanRullen, R., and Thorpe, S. J. (2004). Temporal codes and sparse representations: a key to understanding rapid processing in the visual system. *Journal of Physiology-Paris*, 98(4):487–497.

Ha, J.-W., Kim, K.-M., and Zhang, B.-T. (2015). 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.

Han, C., Lee, S.-W., Heo, Y., Kang, W., Jun, J., and Zhang, B.-T. (2017). Criteria for human-compatible ai in two-player vision-language tasks. In *2017 IJCAI Workshop on Linguistic and Cognitive Approaches to Dialogue Agents*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

Heigold, G., Vanhoucke, V., Senior, A., Nguyen, P., Ranzato, M., Devin, M., and Dean, J. (2013). Multilingual acoustic models using distributed deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8619–8623.

Hong, S., You, T., Kwak, S., and Han, B. (2015). Online tracking by learning discriminative saliency map with convolutional neural network. In *Proceedings of the 32th International Conference on Machine Learning*, pages 597–606.

Hsu, W., Kalra, A., and Poupart, P. (2017). Online structure learning for sum-product networks with gaussian leaves. *arXiv preprint arXiv:1701.05265*.

Huynh, T., Fritz, M., and Schiele, B. (2008). Discovery of activity patterns using topic models. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 10–19. ACM.

Kienzle, W. and Chellapilla, K. (2006). Personalized handwriting recognition via biased regularization. In *Proceedings of the 23rd international conference on Machine learning*, pages 457–464. ACM.

Kim, J.-H., Lee, S.-W., Kwak, D.-H., Heo, M.-O., Kim, J., Ha, J.-W., and Zhang, B.-T. (2016a). Multimodal residual learning for visual qa. *arXiv preprint arXiv:1606.01455*.

Kim, Y.-D., Jang, T., Han, B., and Choi, S. (2016b). Learning to select pre-trained deep representations with bayesian evidence framework. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5318–5326.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*.

Knierim, J. J. and Neunuebel, J. P. (2016). Tracking the flow of hippocampal computation: Pattern separation, pattern completion, and attractor dynamics. *Neurobiology of learning and memory*, 129:38–49.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Kumaran, D., Hassabis, D., and McClelland, J. L. (2016). What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20(7):512–534.

Lee, J., Yun, J., Hwang, S., and Yang, E. (2017a). Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547*.

Lee, S.-W., Heo, M.-O., and Zhang, B.-T. (2013). Online incremental structure learning of sum–product networks. In *International Conference on Neural Information Processing*, pages 220–227. Springer.

Lee, S.-W., Heo, Y.-J., and Zhang, B.-T. (2018). Answerer in questioner's mind for goal-oriented visual dialogue. *arXiv preprint arXiv:1802.03881*.

Lee, S.-W., Kim, J.-H., Jun, J., Ha, J.-W., and Zhang, B.-T. (2017b). Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*.

Lee, S.-W., Lee, C.-Y., Kwak, D.-H., Ha, J.-W., Kim, J., and Zhang, B.-T. (2017c). Dual-memory neural networks for modeling cognitive activities of humans via wearable sensors. *Neural Networks*.

Lee, S.-W., Lee, C.-Y., Kwak, D. H., Kim, J., Kim, J., and Zhang, B.-T. (2016). Dual-memory deep learning architectures for lifelong learning of everyday human behaviors. In *Twenty-Fifth International Joint Conference on Artificial Intelligencee*, pages 1669–1675.

Lee, S.-W., Watkins, C., and Zhang, B.-T. (2014). Non-parametric bayesian sum-product networks. In *Workshop on Learning Tractable Probabilistic Models*.

Li, Z. and Hoiem, D. (2016). Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer.

Lichman, M. (2013). UCI machine learning repository.

Lin, M., Chen, Q., and Yan, S. (2014). Network in network. In *International Conference on Learning Representations*.

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

Liu, X., Zhang, G., Zhan, Y., and Zhu, E. (2008). 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.

Louizos, C. and Welling, M. (2016). Structured and efficient variational deep learning with matrix gaussian posteriors. *arXiv preprint arXiv:1603.04733*.

Lowd, D. and Rooshenas, A. (2013). Learning markov networks with arithmetic circuits. In *Artificial Intelligence and Statistics*, pages 406–414.

MacKay, D. J. (1992). A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472.

McClelland, J. L., McNaughton, B. L., and O'Reilly, R. C. (1995). Why there are complementary learning systems in the hippocampus and neocortex: insights from the successes and failures of connectionist models of learning and memory. *Psychological review*, 102(3):419.

McCloskey, M. and Cohen, N. J. (1989). Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., et al. (2015). Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 2302–2310.

Nam, H. and Han, B. (2016). Learning multi-domain convolutional neural networks for visual tracking. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4293–4302.

Nichol, A. and Schulman, J. (2018). Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*.

Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528.

Oza, N. C. (2005). Online bagging and boosting. In *Systems, Man and Cybernetics, IEEE International Conference On*, pages 2340–2345.

O'Reilly, R. C., Bhattacharyya, R., Howard, M. D., and Ketz, N. (2014). Complementary learning systems. *Cognitive Science*, 38(6):1229–1248.

Pascanu, R. and Bengio, Y. (2013). Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*.

Pathak, M., Rane, S., and Raj, B. (2010). Multiparty differential privacy via aggregation of locally trained classifiers. In *Advances in Neural Information Processing Systems*, pages 1876–1884.

Peharz, R., Geiger, B. C., and Pernkopf, F. (2013). Greedy part-wise learning of sum-product networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 612–627. Springer.

Polikar, R., Upda, L., Upda, S. S., and Honavar, V. (2001). 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.

Poon, H. and Domingos, P. (2011). Sum-product networks: A new deep architecture. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 689–690. IEEE.

Rashwan, A., Zhao, H., and Poupart, P. (2016). Online and distributed bayesian moment matching for parameter learning in sum-product networks. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 1469–1477.

Ravikumar, P., Wainwright, M. J., Lafferty, J. D., et al. (2010). High-dimensional ising model selection using $\ell 1$-regularized logistic regression. *The Annals of Statistics*, 38(3):1287–1319.

Ray, S. and Lindsay, B. G. (2005). The topography of multivariate normal mixtures. *Annals of Statistics*, pages 2042–2065.

Ray, S. and Ren, D. (2012). On the upper bound of the number of modes of a multivariate normal mixture. *Journal of Multivariate Analysis*, 108:41–52.

Rooshenas, A. and Lowd, D. (2014). Learning sum-product networks with direct and indirect variable interactions. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 710–718.

Rusu, A. A., Rabinowitz, N. C., Desjardins, G., Soyer, H., Kirkpatrick, J., Kavukcuoglu, K., Pascanu, R., and Hadsell, R. (2016). Progressive neural networks. *arXiv preprint arXiv:1606.04671*.

Ruvolo, P. L. and Eaton, E. (2013). Ella: An efficient lifelong learning algorithm. In *Proceedings of the 30th International Conference on Machine Learning*, pages 507–515.

Sainath, T. N., Vinyals, O., Senior, A., and Sak, H. (2015). Convolutional, long short-term memory, fully connected deep neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4580–4584.

Shin, H., Lee, J. K., Kim, J., and Kim, J. (2017). Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999.

Simonyan, K. and Zisserman, A. (2014). Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576.

Snoek, J., Rippel, O., Swersky, K., Kiros, R., Satish, N., Sundaram, N., Patwary, M., Prabhat, M., and Adams, R. (2015). Scalable bayesian optimization using deep neural networks. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2171–2180.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958.

Srivastava, R. K., Masci, J., Kazerounian, S., Gomez, F., and Schmidhuber, J.

(2013). Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448.

Sundermeyer, M., Schlüter, R., and Ney, H. (2012). Lstm neural networks for language modeling. In *Interspeech*, pages 194–197.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2005). Sharing clusters among related groups: Hierarchical dirichlet processes. In *Advances in neural information processing systems*, pages 1385–1392.

Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The tc algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 489–497.

Treves, A. and Rolls, E. T. (1992). Computational constraints suggest the need for two distinct input systems to the hippocampal ca3 network. *Hippocampus*, 2(2):189–199.

Vedaldi, A. and Lenc, K. (2015). Matconvnet – convolutional neural networks for matlab. In *Proceedings of the ACM International Conference on Multimedia*, pages 689–692.

Wah, C., Branson, S., Welinder, P., Perona, P., and Belongie, S. (2011). The caltech-ucsd birds-200-2011 dataset. *Tech. Rep. CNS-TR-2011-001*.

Wei, T., Wang, C., Rui, Y., and Chen, C. W. (2016). Network morphism. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 564–572.

Weston, J., Chopra, S., and Bordes, A. (2014). Memory networks. In *International Conference on Learning Representations*.

Yamins, D. L., Hong, H., Cadieu, C. F., Solomon, E. A., Seibert, D., and DiCarlo, J. J. (2014). Performance-optimized hierarchical models predict neural responses in higher visual cortex. *Proceedings of the National Academy of Sciences*, 111(23):8619–8624.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

Yu, H., Wang, J., Huang, Z., Yang, Y., and Xu, W. (2015). Video paragraph captioning using hierarchical recurrent neural networks. *arXiv preprint arXiv:1510.07712*.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Proceedings of European Conference on Computer Vision*, pages 818–833.

Zhang, B.-T. (2008). Hypernetworks: A molecular evolutionary architecture for cognitive learning and memory. *IEEE computational intelligence magazine*, 3(3).

Zhang, B.-T. (2013). Information-theoretic objective functions for lifelong learning. In *AAAI Spring Symposium: Lifelong Machine Learning*, pages 62–69.

Zhang, B.-T., Ha, J.-W., and Kang, M. (2012). Sparse population code models of word learning in concept drift. In *Proceedings of the 34th Annual Conference of Cogitive Science Society*, pages 1221–1226.

Zhang, B.-T. and Muhlenbein, H. (1994). Synthesis of sigma-pi neural networks by the breeder genetic programming. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pages 318–323. IEEE.

Zhang, K. and Kwok, J. T. (2010). Simplifying mixture models through function approximation. *Neural Networks, IEEE Transactions on*, 21(4):644–658.

Zhao, H., Adel, T., Gordon, G., and Amos, B. (2016). Collapsed variational inference for sum-product networks. In *International Conference on Machine Learning*, pages 1310–1318.

Zhou, G., Sohn, K., and Lee, H. (2012). Online incremental feature learning with denoising autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 1453–1461.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, pages 928–936.

초록

실제 환경에서 사람들의 행동을 통하여 지식을 쌓는 일은 사람을 인지하는 지능 시스템을 구현하는 데 있어서 필수불가결하다. 우리는 깊은 신경망 인식기를 실시간으로 업데이트하는 기술을 개발함으로서 웨어러블 디바이스 안의 개인화된 상황 인식기를 변화하는 환경에 적응할 수 있도록 하고자 하였다. 그러나, 깊은 신경망을 끊임없이 들어오는 데이터로부터 학습시키는 일은 어려운 일인데, 이는 새로운 데이터를 학습하게 되면 깊은 신경망이 종종 기존에 얻은 지식을 잃어버리기 때문이다. 이러한 현상을 파국적 망각이라고 부른다.

본 논문에서는 이러한 파국적 망각 현상을 해결하기 위하여 두 개의 방법, dual memory architecture (DMA) 와 incremental moment matching (IMM)을 제안한다. 첫번째 방법의 구현에 있어, 우리는 인지신경과학의 complementary learning system 이론의 영향을 받았다. 이 이론에 따르면, 사람은 평생 동안 지식을 쌓아나가는 과정에서 두 가지의 큰 시스템, 신피질 시스템과 해마 시스템이 상보적으로 동작한다. 우리가 제안하는 DMA 알고리즘에서는 두 개의 학습 구조, 깊은 신경망과 하이퍼네트워크가 각각 구조화된 표상을 얻는 동시에 매 사건의 개별 경험들을 저장하기 위하여 사용된다. 알고리즘을 위하여 구체적인 기계학습 기법들 역시 제안되었는데, 이는 깊은 신경망 학습을 위한 파라미터 전이와 빠른 개별 경험 학습을 위한 하이퍼네트워크의 온라인 학습 방법을 포함한다.

두번째 방법인 IMM에서는 새로운 과제 혹은 데이터가 들어올 때 데이터에 대한 파라미터의 사후 확률을 추정한 후 이전 데이터에 대한 파라미터의 사후 확률과 매치시켜 전체 데이터를 위한 사후 확률 파라미터로 병합하는 방식으로 동작한다. 파라미터 사후 확률의 탐색 공간을 평탄하게 하기 위하여, IMM 알고리즘에서는 다양한 전이 학습 기법을 사용한다. 이들은 파라미터 전이, L2-norm 방법의 변형, 그리고 dropout 방법의 변형을 포함한다.

우리는 이 제안된 두 방법의 성공을 직관적으로 설명하기 위하여, 깊은 확률 그래프 모델의 일종인 합곱네트워크의 두 가지 온라인 학습 방법을 제안한다. 우리는 두 온라인 학습이 확률 그래프 모델을 학습하기 위한 수학적으로 적합한 온라인 학습 방법임을 보인 후, 이들이 어떻게 깊은 신경망 학습에 확장될 수 이는 지를 보인다.

우리는 제안한 두 개의 알고리즘 DMA와 IMM을 두 스타일의 데이터 셋 군에 데모하였다. 하나는 다양한 벤치마크 셋들과 그 변형들이며, 다른 하나는 구글 글래스를 통하여 수집된 46일간의 lifelog 데이터셋이다. 우리는 다양한 실험 결과를 통하여, 우리가 제안한 방법들이 다른 비교 모델들보다 더 잘 동작함을 보이는 동시에, 우리의 파국적 망각을 해결하려는 연구 방향이 가치있고 전망 있음을 설득한다.

주요어: Lifelong learning, Lifelog dataset, Sum-product networks, Deep neural networks, Dual memory architecture, Complementary learning systems, Incremental moment matching, Sequential Bayesian

학번: 2012-20835