國立臺灣大學電機資訊學院電機工程學研究所

碩士論文

Graduate Institute of Electrical Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

減緩卷積類神經網路之災難性失憶問題以有效達成物體辨識

Mitigate Catastrophic Forgetting in Convolutional Neural Networks for Effective Instance Recognition

柯達方

Da-Fang Ke

指導教授：羅仁權博士

Advisor: Ren.C. Luo, Ph.D.

中華民國 106 年 7 月

July 2017

doi:10.6342/NTU201703298

致謝

記得剛進入實驗室時，很多東西都不會，有好一陣子在徬徨，對自己該做甚麼研究感到焦慮，好幾次覺得自己是不是走錯路了，現在回想起來都覺得能熬過那段日子到現在能夠獨立做一些小研究感到不可思議，對我來說，研究所兩年真的過得很快，每一天都覺得有好多東西要學，時間根本不夠用，但真的如指導教授羅仁權老師所講的，在壓力下才可能激發一個人的潛力，在兩年靠著不鬆懈的努力，最後還是找到了研究的方向及興趣，當中特別感謝老師及憲章學長，因為老師的支持及献章學長剛開始的引導，我才得以接觸、認識到稍微深入的研究”深度學習”這門領域。

而在研究所這段旅程中，還要感謝很多人，從修課、比賽到做研究發論文，都受到太多實驗室夥伴的幫助了，感謝晴岡、柏凱、李晟、立揚，一起為上銀比賽準備，走過沒有冷氣、熱到主機熱當的夏天；感謝俊豪、凱鈞及仲凱，實驗室的大小事上總是能夠尋問你們；感謝靖霖、長鈞、晴岡，在長庚比賽為我們贏得了冠軍；感謝孟勳、莉彤、育佑不時的予以建議，討論作業等；感謝超強學弟石崴，一起做關於深度學習的研究。研究所也許就只有這兩年，但一起度過的時光是在腦海中一輩子的事情，希望未來還能與你們共事。還要感謝在這段時期可以偶爾一起打鬧的朋友，不管是從小到大的死黨還是大學曾經熱衷舞社的好友，都在學術研究之外讓我的生活更有趣跟感到不孤單。

特別感謝家人，爸、媽及姐姐，一直以來都是最能讓我心裡感到踏實的家，聽了我好多心裡話，也總是支持我做的決定，讓我可以很踏實、很安心的向前走，非常慶幸能有這樣的家。最後特別感謝陳芝瑩，如同家人一樣陪我度過好多艱困時期，不管是研究所考試、當兵、研究所這兩年，都一直陪在我身邊，也欠了好多大大小小答應過的旅行，今年畢業後就可以還幾個了。

doi:10.6342/NTU201703298

中文摘要

物體辨識為電腦視覺中一項十分重要的研究主題，在機器人中為建立認知系統的重要橋梁，機器人必須能在多變的視覺回饋下萃取有用的特徵，進而轉化為高階的知識語言，才能夠實行一連串複雜的任務，在近年由 Alex Krizhevsky 成功的將深度捲積網路 (Deep Convolutional Neural Networks) 實現並應用在影像分類上後，許多的辨識問題有了突破性的發展，然而，儘管有強健穩定的辨識能力，要實現完整的智慧機器人尚有許多實際層面的考量需要克服。

本篇論文探討以漸進式學習 (incremental learning) 的方式來達到物體辨識，機器人在特定的工作環境中往往需要強健穩定的辨識能力來區分影像中不同狀況下的物體 (例如尺度、亮度、遮蔽變化等)，藉由目前深度學習的方法且在資料充裕的情況下，我們可以得到強健可依賴的辨識系統，然而，其中最大的問題是完整的影像資料在實際中是不存在的，影像資料的收集與標籤化是循序漸進的過程，因此，我們需要發展一個能夠漸進式的學習方法來反映這樣的需求，再者，我們希望漸進式學習能夠模仿人類的學習模式，在學習新的知識時，可以無需無過往資料的再檢視，而能夠在保有原有知識下再學習新的知識，此最大的好處即是我們不需儲存非常大量的影像資料，這對於工作於不同環境中的可適應性的機器人來說是非常具有其效益性的。

根據我們所想的學習情境當中，最大的困難是如何克服”災難性失憶 (catastrophic forgetting)”，災難性失憶是由於類神經網路在學習新的資訊時，新學習的知識將會複寫掉之前學習過的知識，反觀人類的學習，人類只會輕微的遺忘而非如此嚴重的失憶，為模仿這樣的學習模式，我們運用知識萃取當中的一項技術-Pseudorehearsal 作為訓練的機制，並貢獻了兩個能大幅增進表現的想法及理論，第一是引入了 imaging recollection，模仿人腦對某物體映像的概念，另神經網路自行得到最能反映某物體特徵的影像；第二則是提出 pseudo neurons，使在漸進式的訓練過程中，確保在網路最後一層的新類別神經元能夠藉由合理正確的損失函數來訓練，論文所提出的演算法能夠很好的在學習新的物體及保有過往學過的物體間取得很好的平衡點。藉由完整的實

doi:10.6342/NTU201703298

驗我們印證所提出的演算法的可行性並分析及討論，同時也比較其他方法來凸顯我們方法的成效。

關鍵字：漸進式學習、深度學習、災難性失憶、物體辨識

doi:10.6342/NTU201703298

Abstract

Object recognition has remained an important research topic in computer vision for a long time. It plays a critical role in the area of robotics. Robots need to extract useful information from rich visual feedback and convert it to high-level semantic knowledge, and hence can lead to intelligence. However, until the emerging of Convolutional Neural Networks (CNNs), the fundamental ability to recognize objects is still insufficient. Since Alex Krizhevsky successfully applied deep CNNs on large scale image classification, CNNs has been bringing lots of success in the community of computer vision. Yet, lots of practical concerns still need to be overcome to make intelligent systems truly useful.

In this thesis, we focus on a practical issue which requires robots to be able to incrementally learn new objects. We first reason that an intelligent service robot working in a particular environment needs to recognize instances under different imaging conditions (scale, brightness, occlusion, etc.). Through the advanced deep learning method, we are able to train a reliable visual system given sufficient data. The issue, however, is that in the reality of beginning, a complete dataset that covers all instances to be learned and provides sufficient imaging conditions is unavailable. In practice, supervisors collect new data and train recognition systems repeatedly and incrementally. It is necessary to derive an incremental learning approach to meet this requirement. A direct solution would be to reuse of every past data along with new data to ensure performance. While this may be workable, it requires a reservoir of persis-

doi:10.6342/NTU201703298

tent training data for all learning stage, an assumption which may not always hold. To this end, we investigate instance recognition in continuous learning scenarios without the need to access previous data. Under the hood, we are investigating how to mitigate catastrophic forgetting. Catastrophic forgetting is a phenomenon which destroys previously learned knowledge when training Neural Networks on new data. In the thesis, we propose pseudorehearsal with imaging recollection and pseudo neurons to address the forgetting problem. Our approach can achieve a promising tradeoff between learning new knowledge and preserving old knowledge. We demonstrate the feasibility of our approach by experiments and comparison with other approaches. We also provide insights to understand our innovation by experimental analysis.

Keywords: incremental learning, deep learning, catastrophic forgetting, and instance recognition.

doi:10.6342/NTU201703298

Contents

doi:10.6342/NTU201703298

doi:10.6342/NTU201703298

List of Figures

doi:10.6342/NTU201703298

doi:10.6342/NTU201703298

List of Tables

doi:10.6342/NTU201703298

Chapter 1

Introduction

1.1 Instance recognition

service robots to properly operate in real-world environments relies on the knowledge about the surroundings. One of the fundamental elements is reliable visual systems on which high-level techniques such as manipulation, navigation, and object search can be built. For example, in a lab environment, robots are required to distinguish different devices to be qualified assistants. In industry, recognizing objects in convey belt is critical for coherent operation. In a house, specific objects need to be learned by robots to conduct correct tasks assigned by masters. Instance recognition is very basic yet crucial ability. Thus it is a required skill for intelligent systems like robots.

The difficulty of instance recognition is diverse image variances such as lighting, background, rotation, etc (Figure 1.1). It is required for a perceptron system to be robust against such variance. Traditional methods rely on extraction of key points either in 2D or 3D geometry [1] [2]. Then learn a classifier create corresponding boundary conditions. Although these methods achieve some success, they often lack the ability to against variance. Template matching is another commonly used method but requires a large database to store every trained model [3]. For precise recogntion, template matching often uses 3D information like point cloud that might consume too much computation and memory. In recent years, statistical methods has achieved tremendous breakthrough in vision community

doi:10.6342/NTU201703298

thanks to deep learning, or specifically, Convolutional Neural Networks (CNNs) [4] [5]. This novel kind of machine learning method constructed based on large data has reveal a new era of solutions to vision tasks and can be utilized to benefit robotics.

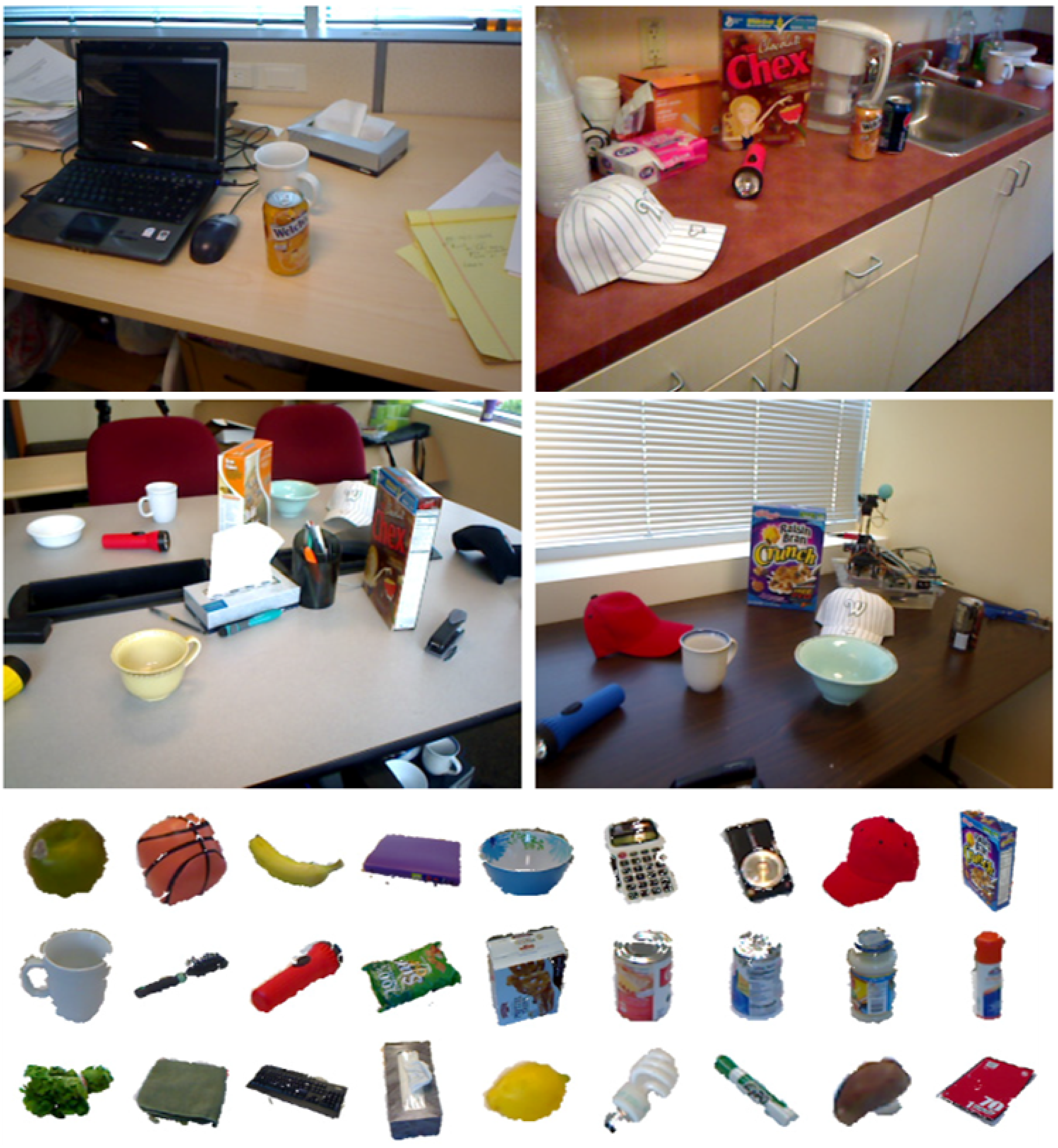


Figure 1.1: Scenario of instance recognition. Instances below may appear in scenes above with different imaging conditions

doi:10.6342/NTU201703298

1.2 Deep learning

Deep learning has gained its popularity recently. One major reason is the appearance of CNNs which are designed for vision community (Figure 1.2). CNNs have been successfully adopted for solving recognition tasks and won many competitions such as ImageNet [6], PASCAL Visual Object Classes [7] and Microsoft Common Objects in Context [8]. A lot of advanced techniques [9] [10] [11] [12] are then further proposed to accelerate the development of relevant research. Resorting to CNNs for solving high level vision task is now the first alternative for many researchers or engineers [13]. Indeed, almost every proposed methods in public competitions are using deep learning skill nowadays.

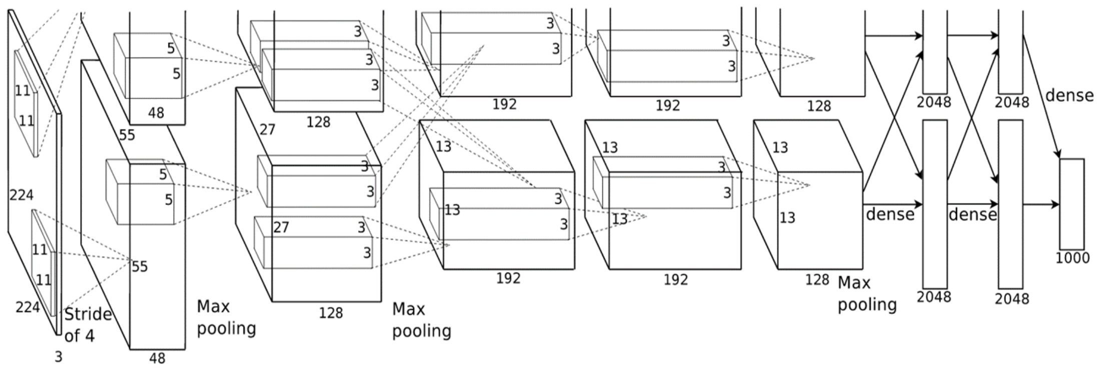


Figure 1.2: AlexNet model structure [5].

Other than CNNs, deep learning is also adopted in many fields. For example, speech recognition [14] and machine translation [15] that posses sequential information are approached using Recurrent Neural Network (RNN) [16] (Figure 1.3), which also belongs to one branch of deep learning. For representation learning, Deep Belief Networks (DBN) [17] and Deep Boltzmann Machine (DBM) [18] (Figure 1.4) are utilized to reconstruct inputs in unsupervised learning. DBN can thus learn a low dimension representations that

doi:10.6342/NTU201703298

encode useful information. Recently, Generative Adversarial Nets (GAN) [19] which act as a generative model are able to create realistic images or even write a Chinese poem. Deep learning has brought lots of possibilities to favour many research domains.

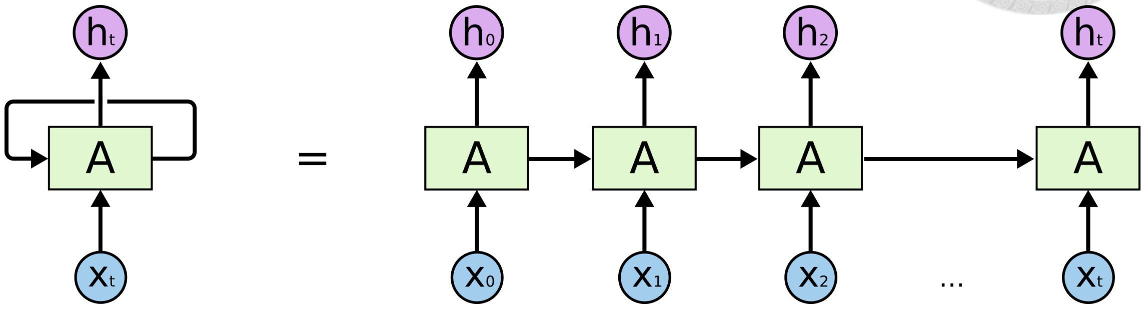


Figure 1.3: Recurrent neural network (RNN).

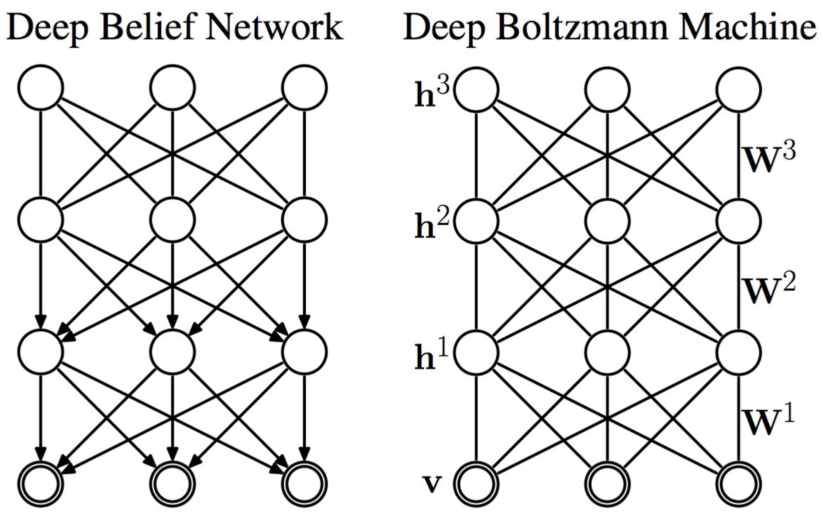


Figure 1.4: Deep Belief Networks (DBN) and Deep Boltzmann Machine (DBM).

The thesis also applies deep learning to address instance recognition. CNN-based method [20] outperforms many state-of-art hand-crafted algorithms. Training CNNs to recognize a bunch of distinct instances without interference dut to different type of image variations is achievable given sufficient annotated data [21] [20]. However, despite the advantage provided by CNNs, there are still numerous practical concerns to address for robotic application. One of the research topic is incremental learning, which we introduce in the next section.

doi:10.6342/NTU201703298

1.3 Incremental learning

One prerequisite for deep learning's success is large amount of available data. For service robots working in custom environments, the assumption of a complete dataset which covers required instances and provides sufficient variety of imaging conditions in the beginning is impractical. Data must be collected by supervisors or users from scratch. Furthermore, the process can be assumed life-long considering dynamic environment with variable objects. To this end, model training should be a adaptive process whenever new training examples are ready for updating systems.

The above scenario brings us to incremental learning, a machine learning paradigm that aims at continuously adapting learner to sequential data (Figure 1.5). This requires model to be class-extensible, i.e. new class can be learned whenever data is available. One solution for both deep learning and incremental learning is to extend final classification layer of CNNs model with new capacities and finetune. Although the strategy works, finetuning needs all previous data being attended in every training stage. It requires a reservoir of persistent training data and thus might consume tremendous storage in long term uasge. A more strict incremental learning [22] should free this constraint by more sophisticated designation. We borrow the rules from Polikar *et al.* [22] that describes precise formulations of incremental learning.

- It should be able to learn additional information from new data.
- It should not require access to the original data, used to train the existing classifier.
- It should preserve previously acquired knowledge.
- It should be able to accommodate new classes that may be introduced with new data.

doi:10.6342/NTU201703298

These rules mimic the behaviour of human learning. Human can learn new items without the need to access previous items and only suffers minor extent of forgetting. Investigating a mechanism that carries out these properties would be crucial development towards real artificial intelligence. Neural networks, however, breaks the third rule because parametric models especially for NNs that store large amount of weights always suffer from ”catastrophic forgetting” and so do CNNs. Catastrophic forgetting causes the model losing previous knowledge drastically if only trained on new items when incremental learning. The thesis hence copes with the forgetting problem and propose possible solutions. We present an effective approach to mitigate catastrophic forgetting in CNNs and apply the approach on instance recognition to enable incremental learning.

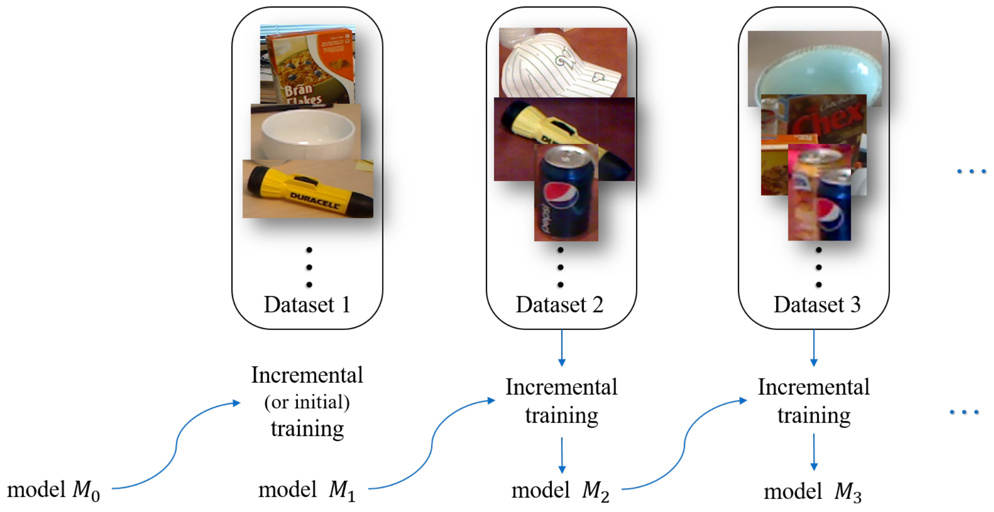


Figure 1.5: Illustration of incremental learning for instance recognition.

1.4 Thesis structure

The organization of the thesis is as follows. In chapter 2, the background and operation of Convolutional Neural Networks will be introduced. Chapter 3 includes detailed phenomenon of catastrophic forgetting and related works which cope with forgetting problem.

doi:10.6342/NTU201703298

The proposed methodology is fully covered in Chapter 4. Chapter 5 contains description of experiment methods such as naive approach and our proposed methods. Other proposed works are also experimented for comparisons. Final conclusion in Chapter 6 summarizes our work and draw some vision for future works.

doi:10.6342/NTU201703298

Chapter 2

Convolutional Neural Networks

The history of Convolutional Neural Networks (CNNs) traces back to Lecun *et al.* [4], who combine stacked parametric filters and the concept of multi-layer perceptron (MLP) to form CNNs (Figure 2.1). This parametric model is then trained by optimizing error functions using gradient descent method to solve character recognition task. Later, Alex *et al.* [5] successfully develop deeper and more complex CNNs and applied the model on large-scale competition [6]. This powerful capability has continued to make a breakthrough on many vision tasks. Understanding the fundamentals of CNNs is important for comprehensibility for subsequent context. In whis chapter, we therefore introduce required knowledge about how CNNs operate and how gradient descent is utilized to train models.

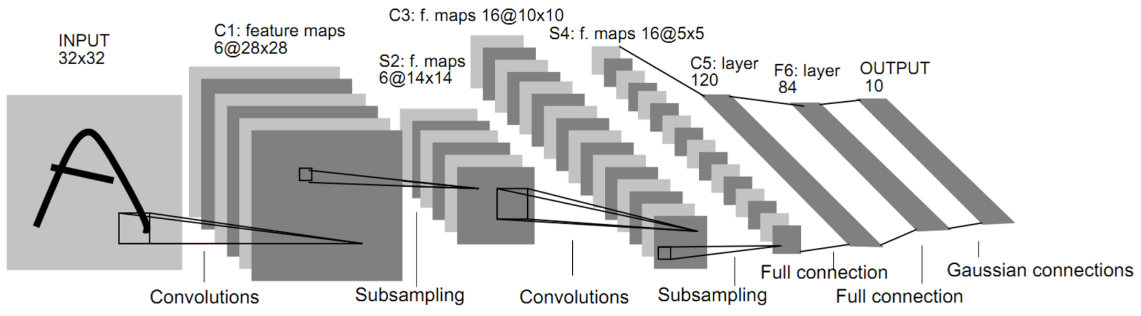


Figure 2.1: LeNet model struture [4].

doi:10.6342/NTU201703298

2.1 Operation

Function of CNNs can be basically described as the result obtained from a series of operations such as *convolution*, *activation*, *pooling*, *flattening* and *dense multiplication*. We use VGG16 [23] as example due to the structure's simplicity (see Figure 2.2).

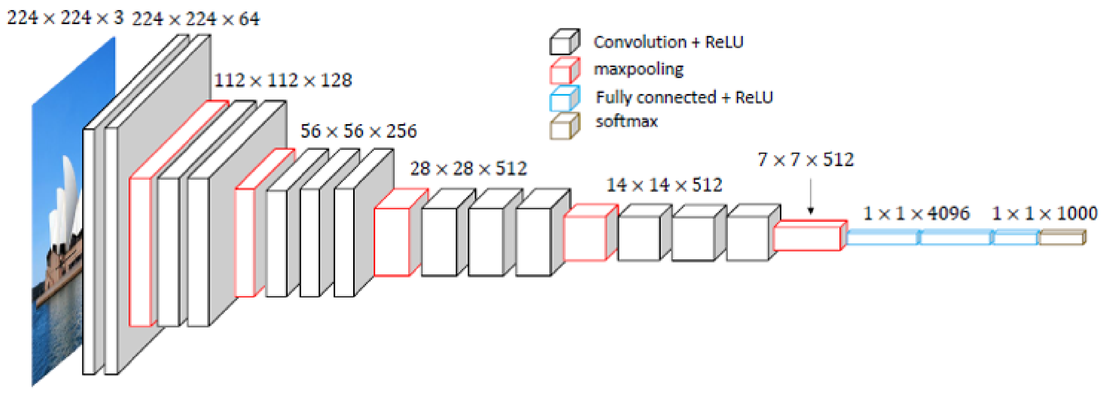


Figure 2.2: VGG16 model struture. Figure extracted from [24].

convolution

A convolutional layer is composed of numerous parametric[1] filters that conduct "convolution" upon input image or intermediate feature maps. Usually we will let the resulting next feature maps have same width and height by padding. For instance, the first convolution in Figure 2.2 takes 3-channel image as input, and then applies 64 filters with stride 1 and padding to output a feature map which has 64-channel and the same width and height as input image. This output map will be inputs to next layer.

activation

Activation functions which are non-linear and are applied after linear operations (convolution, dense multiplication) are used to augment networks' representability. Well-

[1]"Parametric" means there are tunable parameters that will be learned during training by optimization

doi:10.6342/NTU201703298

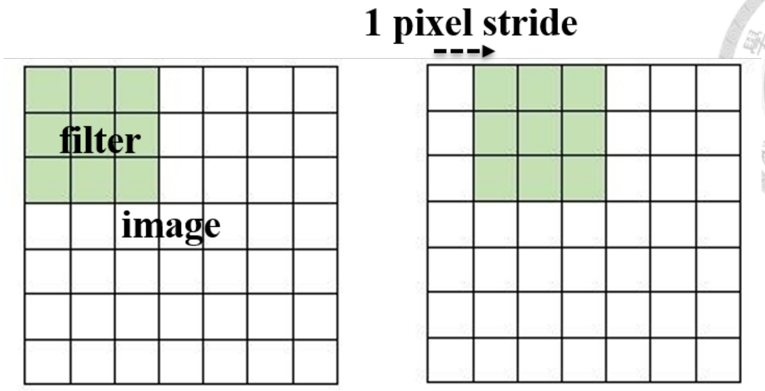


Figure 2.3: Convolution. A filter scans the an image from left-top to right-bottom with stride 1.

known activation functions include *sigmoid*, *tanh*, and *ReLU*.

$$sigmoid(x) = \frac{1}{1 + \exp^{-} x} \tag{2.1}$$

$$tanh(x) = \frac{\exp^{z} - \exp^{-z}}{\exp^{z} + \exp^{-z}} \tag{2.2}$$

$$ReLU = \begin{cases} x, & x > 0 \\ 0, & else \end{cases} \tag{2.3}$$

In CNNs, ReLU is usually selected as activation function for its ability for avoiding ”gradient vanishing[2]”.

pooling

Pooling, or subsampling aim at dimension reduction and manageable usage. It is also adopted to preserve important signal. There are *max pooling* and *average pooling* which are frequently used in CNNs. Figure 2.4 shows an example using max pooling and Fig-

[2]Gradient vanishing occers because derivatives obtained from some kind of activaiton function such as sigmoid and tanh are samller than 1.0. Hence when backpropagating to shallower layers, gradients become very small and fail to improve early layers’ parameters.

doi:10.6342/NTU201703298

ure 2.5 for average pooling.

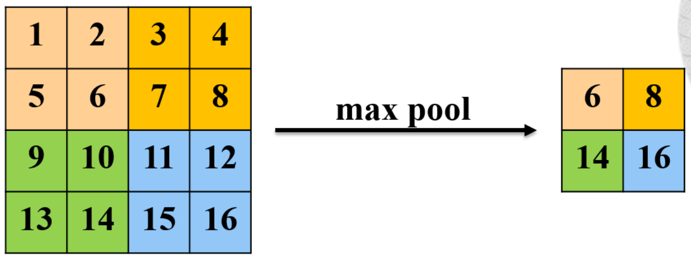


Figure 2.4: Illustration of max pooling.

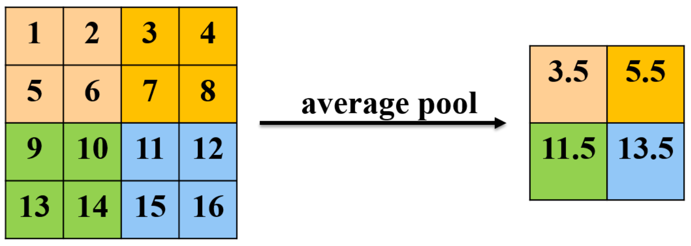


Figure 2.5: Illustration of average pooling.

flattening

Because outputs from convolution layer are image-like 2D feature maps with channels. To feed the output to fully connected layers (MLP), the maps need to be flattened into 1 dimension vector.

dense multiplication

After flattening, dense multiplication contains parametric matrix and applies it on input vector to output a new vector. The operation is a basic element in every neural networks like structures. The corresponding operation in VGG16 is the right part (blue) in Figure 2.2, which will finally output 1000-dimension class scores.

doi:10.6342/NTU201703298

2.2 Optimization

The weights of CNNs are often randomly initialized based on normal distribution. To tune the weights to accommodate tasks ,optimization is applied according to objective function (loss) and updating algorithms. An objective function which is designed to fit a training data set X and parametrized by model weights θ is denoted as $J(X;\theta)$. Minimizing $J(X;\theta)$ renders us an optimized set of weights that can approach desired solutions.

Normally, mini-batch gradient descent is one of the most popular algorithms to perform optimization and by far the most common way to optimize neural networks. With a mini-batch of data $x_{i+1}, x_{i+2}, ..., x_{i+N}$ and current model weights θ^t, the objective function in current iteration is $J(x_{i+1:i+N};\theta^t)$. By derivation and chain rule, we can compute the corresponding gradient with respect to model's parameters $\Delta_\theta J(x_{i+1:i+N};\theta^t)$. Then according to learning rate η, which indicates the size of one step parameters are updating, parameters are updated in the opposite direction to minimize error function.

$$\theta^{t+1} = \theta^t - \eta\Delta_\theta J(x_{i+1:i+N};\theta^t) \tag{2.4}$$

where

$$\Delta_\theta J(x_{i+1:i+N};\theta^t) = \frac{1}{N}\sum_{j=i+1}^{i+N} \Delta_\theta J(x_j;\theta^t) \tag{2.5}$$

There several challenges for this vanilla gradient descent method such as convergence speed, choice to anneal learning rate, and worse convergence. Luckily, deep learning community has develop different kinds of optimizer that can improve training. We list three common optimizers and their updating rules.

- *Momentum* [25]

$$v_t = \gamma v_{t-1} + \eta\Delta_\theta J(\theta) \tag{2.6}$$

doi:10.6342/NTU201703298

$$\theta^{t+1} = \theta^t - v_t \tag{2.7}$$

where γ controls the fraction of momentum. The value is usually set to 0.9.

- *RMSProp* [26]

  For simplicity, we denote $\Delta_\theta J(\theta^t)$ as g_t.

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{2.8}$$

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{E[g^2]_t + \epsilon}} g_t \tag{2.9}$$

  where $E[g^2]_t$ denotes the running average accumulated up to t for each parameter θ. ϵ is a small value to avoid division by zero.

- *Adam* [12]

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.10}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.11}$$

  m_t and v_t are thus estimates of the first moment and the second moment. Furthermore, the author observed that zero initialization of m_t and v_t is unfavourable to training. Therefore they correct the term by division of β with power to t

$$\hat{m_t} = \frac{m_t}{1 - \beta_1^t} \tag{2.12}$$

doi:10.6342/NTU201703298

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.13}$$

Then the update would be

$$\theta^{t+1} = \theta^t - \frac{\eta}{\sqrt{\hat{v}_t + \epsilon}} \hat{m}_t \tag{2.14}$$

In the thesis, we employ *Adam* for its popularity of quick convergence and better performance among other adaptive optimizers. However, altering optimizer is also possible and will not affect our main research.

doi:10.6342/NTU201703298

Chapter 3

Catastrophic Forgetting

3.1 The phenomenon

Catastrophic forgetting is a severe loss of previous memory when trying to learn new information. In a learning environment where new data becomes available, learner is trained on new source to acquire new knowledge. Systems like Multi Layer Perceptron (MLP) or Neural Networks (NNs) will suffer from catastrophic forgetting due to exhausting the capacity to accommodate new information. The weights (the capacity) are tuned to fit new learned problems or items as fast as possible, disregarding previously learned knowledge. Usually one new stage for new items can destroy old knowledge heavily. The phenomenon thus becomes even worse when systems experience several subsequent learning stages, losing the ability to solve previous problems.

Possible solutions to solve or mitigate catastrophic forgetting contains rehearsal methods, reduction of overlapped representation, ensemble of experts, etc. We introduce several works that share similar concepts or differ in essence in the following section, including early attempts to solve forgetting problems in relatively shallow networks and recently deep networks.

doi:10.6342/NTU201703298

3.2 Related Work

3.2.1 Rehearsal

rehearsal method reuses part or all previous data to maintain learned knowledge. Early approaches including *recency rehearsal*, *random rehearsal*, and *sweep rehearsal* [27], each with different sampling strategy. Rebuffi *et al.* [28] latest proposed integrated strategy in deep networks. Their rehearsal method keeps a predefined number of old data to constrain memory usage by constructing an exemplar set. 1 shows the overall algorithm (see Rebuffi *et al.* [28] for more details) Although rehearsal method guarantees certain performance, it still need to access previous data and cope with memory usage, which differs from our scenario.

Algorithm 1 iCaRL

```
Input:
 1: $D_s, ..., D_v$: New training examples in per-class sets
 2: $P = P_0, ..., P_{s-1}$: Current examplar sets
 3: $\theta$: Current model weights
 4: $M$: Memory constraint size
Output: $P, \theta$
 5: $\theta \leftarrow$ UPDATEMODLE$(D_s, ..., D_v, P, \theta)$
 6: $N \leftarrow M/t$: allowed number of example in each exampler set
 7: for $y = 0, ..., s-1$ do
 8:     $P_y \leftarrow$ REDUCEEXAMPLERSET$(P_y, N)$
 9: end for
10: for $y = s, ..., v$ do
11:     $P_y \leftarrow$ CONSTRUCTEXAMPLERSET$(D_y, N)$
12: end for
13: $P = P_0, ..., P_v$
14: return $P, \theta$;
```

3.2.2 Reduce representational overlap

French *et al.* [29] [30] demonstrated that by increasing sparsity of networks, representational overlap between old data and new data is reduced, so parameters that represent

doi:10.6342/NTU201703298

old knowledge are more likely to remain unchanged when training on new data. Similarly, Srivastava *et al.* [31] introduced Local-Winner-Takes-All (LWTA) to let neurons in a block compete and only the max one can pass value to next layer. This results in sparse distributions of network, which can reduce forgetting problem. Their proposed connection structure of network is shown in Figure 3.1, where only one neuron (black) in a block is able to transmit signal to next layer. Kirkpatrick *et al.* [32] also adopt similar rule but also introduce elasticity. By introducing ”elastic weight consolidation” (EWC), The authors record each parameter’s responsibility for how important it is to original task and thus control the changes of those parameters when learning new task. The method finds an optimal updating direction to situate the resulting weights at the proper position which support both new and old tasks (see Figure 3.2).

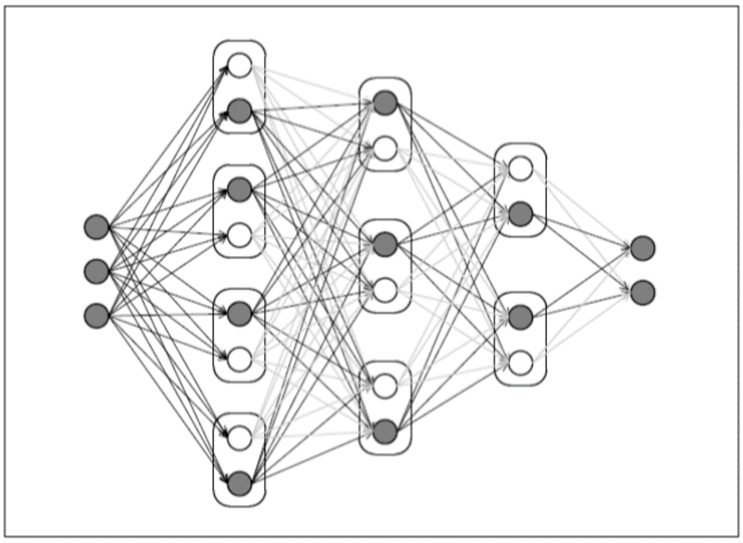


Figure 3.1: A Local Winner Takes All network (figure from [31]).

3.2.3 Ensemble of experts

Polikar *et al.* [22] proposed *Learn* ++. Learn ++ adds new classifiers to accommodate new information. New classifiers are trained to guarantee certain performance on new samples, hence every classifier becomes an ”expert” for specific data distributions.

doi:10.6342/NTU201703298

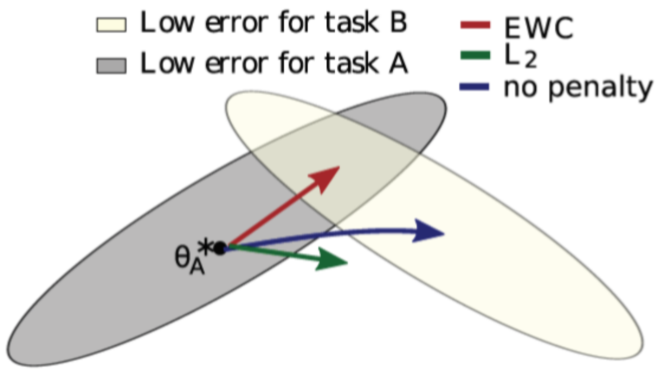


Figure 3.2: Elastic weight consolidation (ELC) (figure from [32]).

Muhlbaier *et al.* [33] further improved Learn ++ by introducing voting and weighting mechanism (Figure 3.3) to address the imbalance problem of classifier's number . Le *et al.* [34] also probed catastrophic forgetting in deep networks by proposing a dual memory architecture consisting of ensemble of three-layer networks, which can continuously learn representations. However, the work proposed by Le *et al.* approach still need to access previous data to accomplish representational learning.

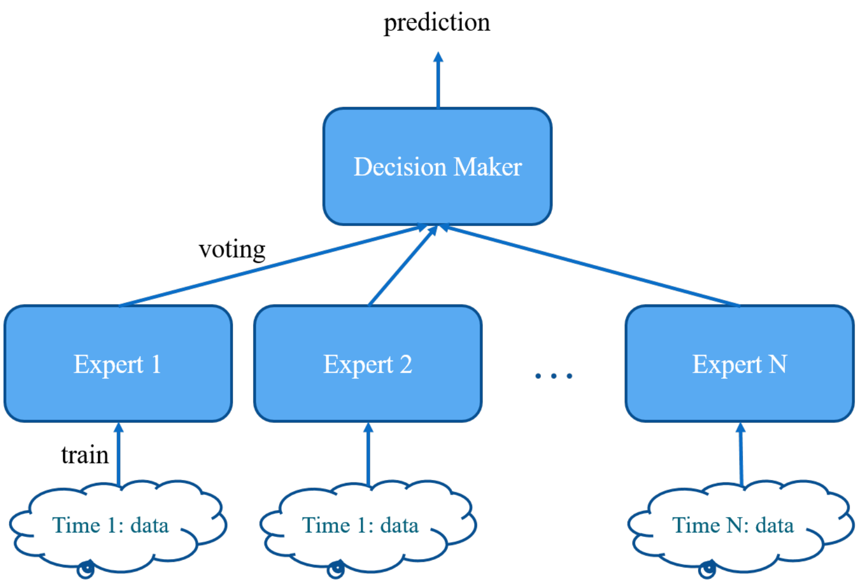


Figure 3.3: Ensemble system consisting of experts

doi:10.6342/NTU201703298

3.2.4 Knowledge distillation

Knowledge distillation is commonly used for model compression [35] , transfer learning [36] [13] [37] and domain adaptation [38] [39] in deep networks. The main idea is to take source model's predictions of unlabelled data as groundtruth labels and use them to train target model. For example, Jung *et al.* [39] incrementally trained model to adapt different input domains by matching logits of the last feature layer. They also reason that the final layer acts as linear classifier and forms decision boundary, which shouldn't be changed. Based on the two criteria, they are able to incrementally train a model to accommodate both color domain and gray level domain. Another example is Li and Hoiem [37]'s work. They propose muti-task learning in incremental manner via distillation loss [35] and show a promising performance on several datasets without suffering severe catastrophic forgetting. In fact, distillation has been applied on mitigating catastrophic forgetting in early stage [40] [28] [39] [37]. Robins [40] proposed *pesudorehearsal* which is able to preserve original model's behaviour by distilling knowledge from pseudo data. This work can be seen as one of the earliest attempts to address forgetting issue and works quite successfully in shallow networks.

Our approach is also related to knowledge distillation and is inspired by the usage of pseudo data [40]. Based on *pesudorehearsal*, we propose two significant improvements to further combat forgetting problem. Playing a critical role as our baseline methodology, we give more details of the mechanism of *pesudorehearsal* in the next chapter.

doi:10.6342/NTU201703298

Chapter 4

Pseudo Rehearsal with Imaging Recollection and Pseudo Neurons

Based on pseudo rehearsal, we propose two significant improvements to enhance the effects against catastrophic forgetting. Namely, the use of recollective images as pseudo data and the joining of pseudo neurons. Our approach allows data of unseen class or existing class but with different imaging conditions to be learned incrementally without catastrophic forgetting. This chapter begins with a problem formulation of our learning scenario. Then a description of how pesudorehearsal works is given. Finally we demonstrate our two innovations. Our core idea and incremental learning pipeline can be seen schematically in Figure 4.1

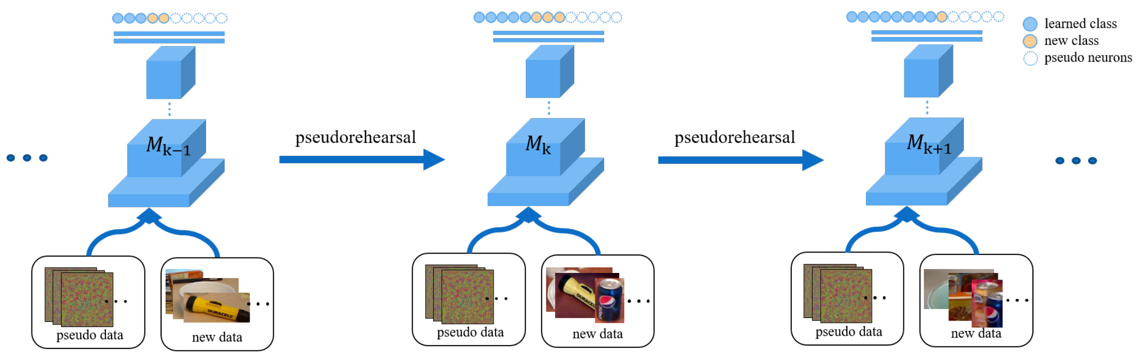


Figure 4.1: Overview of our learning approach. We use the mechanism of pseudorehearsal and combine our two contributed approaches.

doi:10.6342/NTU201703298

4.1 Problem Statement

Data is usually collected sequentially in either supervised or semi-supervised way. Our goal is to enable robots to incrementally learn new information without suffering catastrophic forgetting to fulfil life-long usage. As depicted in Figure 4.1, we denote dataset which becomes available at different time point t as $D^t = \{(x^t_{(1)}, y^t_{(1)}), ..., (x^t_{(N^t)}, y^t_{(N^t)})\}$, where N^t is the number of examples in t^{th} dataset, and $x^t_{(i)}$ is one example with the corresponding one-hot ground truth vector $y^t_{(i)}$. We do not assume newly collected data to belong to only unseen class or existing class so that both kinds of data can be incrementally learned for different purpose, i.e., extension of class list or learning invariance, respectively. Initially we assume a pre-trained model M_0 which has already learned rich hierarchical feature representations is available, e.g. a CNN model trained on ImageNet. Then at each time point $t = k$, we continuously train the current model M on new dataset D^k.

4.2 Pseudorehearsal

We first introduce how pseudorehearsal [40] works as one of the early successful methods to avoid catastrophic forgetting. While common rehearsal methods still rely on previous data to preserve old knowledge, pseudorehearsal is able to rehearse without old items. Innovatively, it manufactures data by randomizing input images (image with random value from 0 to 255 at each pixel) and whatever outputs generated from the current model form the corresponding targets. Before new training process begins, a number of pseudo data is generated and will be trained along with real data. We denote the pseudo dataset formed at current time point $t = k$ as $\widetilde{D}^k = \{(\tilde{x}^k_{(1)}, \tilde{f}^k_{(1)}), ..., (\tilde{x}^k_{(\widetilde{N}^k)}, \tilde{f}^k_{(\widetilde{N}^k)})\}$ where $\tilde{x}^k_{(i)}$ is one pseudo

doi:10.6342/NTU201703298

sample (randomized image in this case), $\tilde{f}^k_{(i)}$ is the corresponding output targets generated from M_{k-1} (see Figure 4.2), and $\widetilde{N}^k$ is the number of psuedo samples. During training, both real data and pseudo data are trained using standard stochastic gradient descent but with different loss functions:

$$\mathcal{L} = \lambda_{dis}\mathcal{L}_{dis} + \lambda_{cls}\mathcal{L}_{cls} \tag{4.1}$$

$$\mathcal{L}_{dis} = \sum_{(\tilde{x},\tilde{f})\in\widetilde{D}} \sum_{i=1}^{C_o} \frac{1}{2}\|f_i(\tilde{x};\theta) - \tilde{f}_i\|_2^2 \tag{4.2}$$

$$\mathcal{L}_{cls} = -\sum_{(x,y)\in D} \sum_{i=1}^{C_o+C_n} y_i \log(g_i(x;\theta)) \tag{4.3}$$

The update with respect to one pseudo sample and one true sample is:

$$\begin{aligned} \frac{\partial \mathcal{L}_{dis}}{\partial \theta} &= \frac{\partial \mathcal{L}_{dis}}{\partial f} \cdot \frac{\partial f}{\partial \theta} \\ &= \sum_{i=1}^{C_o} (f_i(\tilde{x};\theta) - \tilde{y}_i) \cdot \frac{\partial f_i(\tilde{x};\theta)}{\partial \theta} \end{aligned} \tag{4.4}$$

$$\begin{aligned} \frac{\partial \mathcal{L}_{cls}}{\partial \theta} &= \frac{\partial \mathcal{L}_{cls}}{\partial f} \cdot \frac{\partial f}{\partial \theta} \\ &= \sum_{i=1}^{C_o+C_n} g_i(x;\theta) \cdot \frac{\partial f_i(x;\theta)}{\partial \theta} - \frac{\partial f_t(x;\theta)}{\partial \theta} \end{aligned} \tag{4.5}$$

where, λ_{dis} and λ_{cls} are the loss weights of Euclidean loss $\mathcal{L}_{dis}$ and cross-entropy loss $\mathcal{L}_{cls}$, respectively, θ is trainable weights of the model, f_i, g_i, and y_i are the i^{th} values of output vector, output vector before softmax layer and one-hot groundtruth vector, and t is the target index ($y_t = 1$). C_o and C_n represent the old classes and new classes. We use Euclidean loss as distillation loss to penalize the predicted outputs (logits) before softmax layer according to Ba and Caruana [36]. This helps the current model to preserve certain

doi:10.6342/NTU201703298

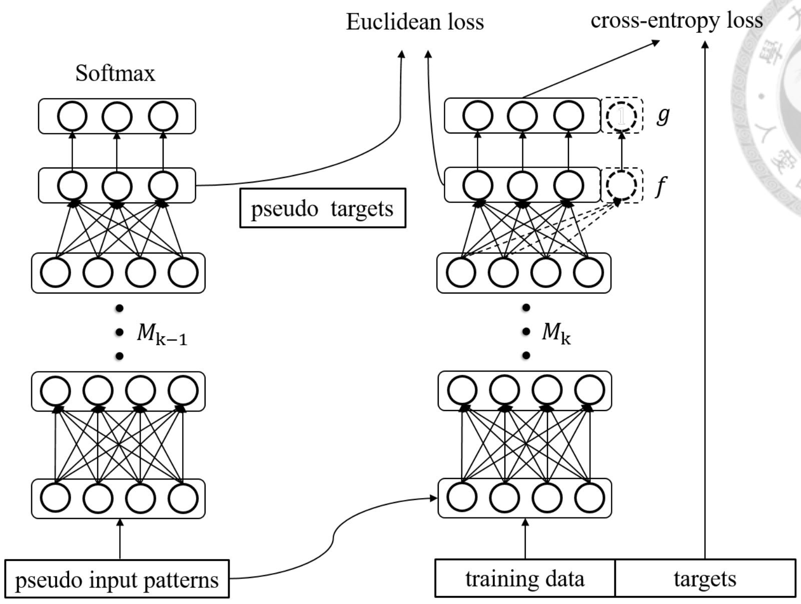


Figure 4.2: Illustration of pseudorehearsal. The dotted neurons on the right side represent new classes (if any).

degree of old model's mapping function by learning the relation between pseudo inputs and pseudo targets. New information can be gradually embedded to the model via standard classification loss. One is free to normalize derivatives $\frac{\partial \mathcal{L}_{dis}}{\partial f}$ and $\frac{\partial \mathcal{L}_{cls}}{\partial f}$ or fine tune λ_{dis} and λ_{cls} to prevent unequally impact introduced by different losses. Our empirical findings show that this may be unnecessary, as neural network seems to be able to reach a stable state by itself, hence we don't adopt this strategy. Schematic illustration of the training process can be seen in Figure 4.2.

4.3 Imaging Recollection

Original work [40] uses randomized input data as pseudo data. However, in the literature [41], the authors pointed out that randomized inputs may blur the abstraction of originally learned data and thus less effective to preserve old knowledge. To obtain more informative data, we borrow the idea from Erhan *et al.*. Erhan *et al.* [42] introduces "activation maximization" to visualize what neural networks "thinks" given a concept of in-

doi:10.6342/NTU201703298

terest. This technique is then combined with regularization terms [43] to synthesize more recognizable images to human (see Figure 4.3 for synthesized images). While both works make the efforts to understand and visualize deep networks, we believe this kind of data is more informative, and can serve as a good data source to conduct pseudorehearsal. We intuitively treat this procedure as kind of ”imaging recollection” as images are inversely generated given a concept (see Figure 4.4). Our intuition, for example, is that when human recalls the object ”coffee mug”, there may be some kind of impression shown as image in our memory. Corresponds to this behaviour, we want to generate an image which can maximally trigger the concept of ”coffee mug” in neural networks. We expect these recollective images to be able to encode more relevant information that can better preserve old knowledge.

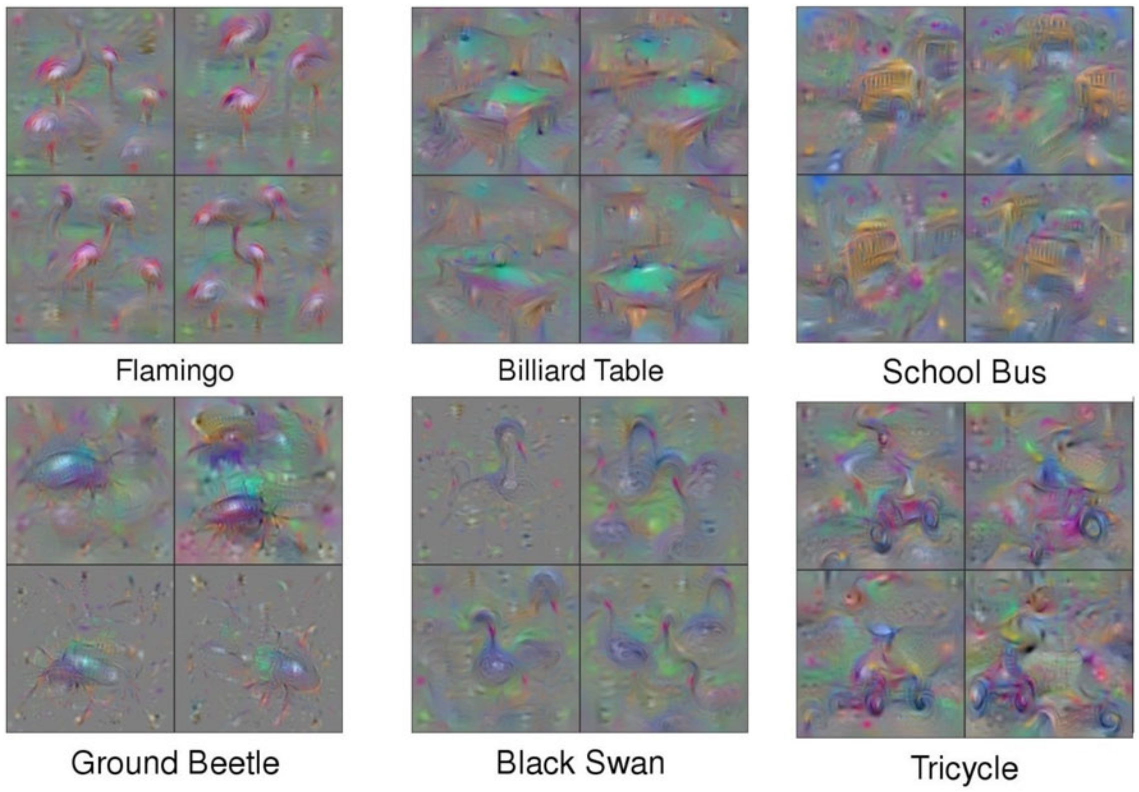


Figure 4.3: Examples of synthsized image by activation maximization and regularization (figure from [43])

doi:10.6342/NTU201703298

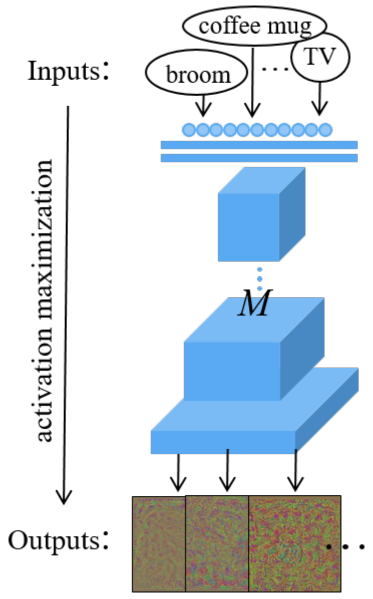


Figure 4.4: Illustration of imaging recollection.

derivation of generated image is described as follows. By first choosing one output unit which represents some specific concept, we form one-hot label vector y where the j^{th} element is the target and equals to 1 while the remaining elements equal to 0. We then minimize the loss in (4.3) by finding the target image $\tilde{x}$ where

$$\tilde{x} = \underset{x}{\operatorname{argmin}} \; \mathcal{L}_{cls}(x, y, \theta) \tag{4.6}$$

which is equivalent to maximize the j^{th} element of networks' output:

$$\tilde{x} = \underset{x}{\operatorname{argmax}} \; g_j(x; \theta) \tag{4.7}$$

This can be solved by rendering an initial random image x and adopting gradient ascent with respect to x. Specifically, for all pixel values x^{τ}_{mn} with the corresponding position m, n and current training step τ. We have

$$x^{\tau+1}_{mn} = x^{\tau}_{mn} + lr \cdot \frac{\partial g_j(x^{\tau}; \theta)}{\partial x^{\tau}_{mn}} \tag{4.8}$$

doi:10.6342/NTU201703298

where lr is learning rate for one training step. We stop the process when confidence value $g_j(x^{\tau+1};\theta)$ surpasses some pre-define threshold, as a high confidence indicates that CNNs have regarded the image $x^{\tau+1}$ as a strong representation of some concept. The resulting pair $(\tilde{x}, f(\tilde{x};\theta))$ (recall that f is the output vector before g) thus form one pseudo sample and the corresponding pseudo target. Before each incremental training begins, the procedure described above is conducted for several times to generate $\tilde{N}^t$ pseudo samples to form the pseudo dataset for current training stage. Note that

$$\tilde{N}^t = \tilde{n}C_o \tag{4.9}$$

where $\tilde{n}$ is a hyperparameter indicating the number of pseudo samples per class.

4.4 Pseudo Neurons

When training on new dataset, we rely on distillation loss to preserve previous behaviour of model and acquire novel knowledge through classification loss. However, such a training process blinds new added neurons from taking account old information introduced by pseudo data, resulting in asymmetrical impacts from objective functions. We here propose *pseudo neurons* to let both information flows across neurons representing old or new classes. We embed the model in the last logits layer with several additional pseudo neurons that initially represent no class. These neurons are only trained to be less activated for the classes initially learned. When new training stage proceeds, where new classes are introduced, these pseudo neurons will be converted to capacities for new classes.

Because of the setting of pseudo neurons, f is now including both new class neurons

doi:10.6342/NTU201703298

and pseudo neurons. Equation (4.2) and (4.3) are now modified to

$$\mathcal{L}_{dis} = \sum_{(\tilde{x},\tilde{f})\in\tilde{D}} \sum_{i=1}^{C_o+C_n+C_{po}} \frac{1}{2}\|f_i(\tilde{x};\theta) - \tilde{f}_i\|_2^2 \tag{4.10}$$

$$\mathcal{L}_{cls} = -\sum_{(x,y)\in D} \sum_{i=1}^{C_o+C_n+C_{po}+C_{pn}} y_i \log(g_i(x;\theta)) \tag{4.11}$$

where C_o, C_n, C_{po}, and C_{pn} are the numbers of old classes, new classes, old pseudo neurons remained, and newly added pseudo neurons respectively. Note that neurons in C_n are obtained by converting some pseudo neurons.

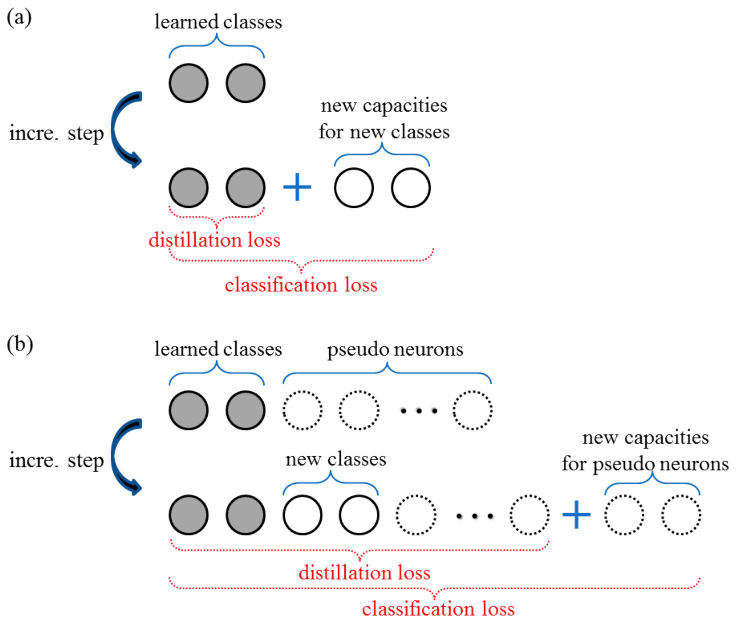


Figure 4.5: The arrangement of output neurons without (a) and with (b) pseudo neurons.

To see how pseudo neurons being arranged compared to the original one without pseudo neurons see Figure 4.5. In Figure 4.5 (a), new neurons of new classes are added directly. Penalizing dissimilarity between previous model and currently trained model by pseudo data, distillation loss can be only applied on already existing neurons. On the other hand, the setting of Figure 4.5 (b) enables distillation loss applied on converted neurons.

doi:10.6342/NTU201703298

To ablate the reason why such a simple arrangement boosts performance, we consider following two cases with or without the addition of pseudo neurons.

We first see original version where no pseudo neuron is added. Consider an extreme case where only one new class is introduced from new data ($C_n = 1$). We form one minibatch consisting only one pseudo sample $(\tilde{x}_{old}, \tilde{f}_{old})$ representing old knowledge and one true sample (x_{new}, y_{new}) of new class. The target index t of new data sample thus lies on the position of $C_o + 1$. Then by (4.2) and (4.3), the summation of two losses given one minibatch is

$$\begin{aligned}\mathcal{L} = &\sum_{i=1}^{C_o} \frac{1}{2}\|f_i(\tilde{x}_{old};\theta) - \tilde{f}_i\|_2^2 \\ &- \sum_{i=1}^{C_o+1} y_i \log(g_i(x_{new};\theta))\end{aligned} \tag{4.12}$$

The optimized state where $\mathcal{L} \approx 0$ can be achieved when

$$f_i(\tilde{x}_{old};\theta) \approx \tilde{f}_i, \qquad for\ i \neq C_o + 1 \tag{4.13}$$

and

$$e^{f_t(x_{new};\theta)} \gg \sum_{i=1}^{C_o} e^{f_i(x_{new};\theta)} \tag{4.14}$$

To see how gradient descent results in this condition, we compute gradients from total loss:

$$\frac{\partial \mathcal{L}}{\partial \theta} = \frac{\partial \mathcal{L}}{\partial f}\frac{\partial f(\tilde{x}_{old};\theta)}{\partial \theta} + \frac{\partial \mathcal{L}}{\partial f}\frac{\partial f(x_{new};\theta)}{\partial \theta} \tag{4.15}$$

doi:10.6342/NTU201703298

and rewrite the equation with respect to derivative $\frac{\partial \mathcal{L}}{\partial f_i}$:

$$\frac{\partial \mathcal{L}}{\partial f_i}\frac{\partial f_i}{\partial \theta} = \begin{cases} (f_i(\tilde{x}_{old};\theta) - \tilde{f}_i) \cdot \dfrac{\partial f_i(\tilde{x}_{old},\theta))}{\partial \theta} & \\ +g_i(x_{new},\theta) \cdot \dfrac{\partial f_i(x_{new},\theta)}{\partial \theta}, & i \neq t \\ (g_i(x_{new},\theta) - 1) \cdot \dfrac{\partial f_i(x_{new},\theta)}{\partial \theta}, & i = t \end{cases} \tag{4.16}$$

To reach optimized state, weights would update toward the direction where all derivative terms $\frac{\partial \mathcal{L}}{\partial f_i}$ approximate to 0[1]. As $f_t(x;\theta)$ is only subject to classification loss, it's free to grow to a large enough value to satisfy (4.14), which yields $g_t(x_{new},\theta)$ very close to 1 and $g_i(x_{new},\theta)$ very close to 0 for $i \neq t$, hence the gradients introduced by classification loss term approximate to 0. Given this condition, $f_i(x;\theta)$ is now only subject to distillation loss as $g_i(x_{new},\theta) \approx 0$ for $i \neq t$. The optimization of distillation loss then results in the condition of (4.13).

We can now reasonably refer that $f(x_{new};\theta)$ would be a logits vector in which the t'th element surpasses other elements, which is desired because the resulting probability of the target class $g_t(x_{new};\theta)$ can be very close to 1. However, such an optimization will cause some confusing situations for images of old class. As encoded representations of images from different class are not possible to be complete orthogonal, they must share some similar information. So normally the resulting representational distributions of x_{old} will also cause a high $f_t(x_{old};\theta)$, which will be a competitive or even exceeding value than the logits $f_{to}(x_{old};\theta)$, where to is the target index of x_{old}. Consequently, the network is confused due to two high logits $f_{to}(x_{old};\theta)$ and $f_t(x_{old};\theta)$[2].

We turn to see the proposed alternative which uses pseudo neurons. For simplicity,

[1]One possible condition to reach stable state is $\frac{\partial f_i}{\partial \theta} \approx 0$ instead of $\frac{\partial \mathcal{L}}{\partial f_i} \approx 0$, which indicates training is stuck at a saddle point. However, such a condition may not exist from the observation by Goodfellow *et al.* [44].

[2]So in this way, neural network does not really forget old knowledge as it still retain a high score for correct class.

doi:10.6342/NTU201703298

we set the number of pseudo neurons to 1, which is converted to the capacity of new class during training, and we don't add new pseudo neuron back to make the equation clear. Thus $C_{po} = C_{pn} = 0$ and $C_n = 1$. Then the gradients of (4.10) and (4.11) become

$$\frac{\partial \mathcal{L}}{\partial f_i}\frac{\partial f_i}{\partial \theta} = \begin{cases} (f_i(\tilde{x}_{old};\theta) - \tilde{f}_i) \cdot \dfrac{\partial f_i(\tilde{x}_{old},\theta))}{\partial \theta} & \\ +g_i(x_{new},\theta) \cdot \dfrac{\partial f_i(x_{new},\theta)}{\partial \theta}, & i \neq t \\ (f_i(\tilde{x}_{old};\theta) - \tilde{f}_i) \cdot \dfrac{\partial f_i(\tilde{x}_{old},\theta))}{\partial \theta} & \\ +(g_i(x_{new},\theta) - 1) \cdot \dfrac{\partial f_i(x_{new},\theta)}{\partial \theta}, & i = t \end{cases} \tag{4.17}$$

The logits $f_t(x;\theta)$ is now also subject to distillation loss. Consequently, $f_t(x;\theta)$ will be high when input image is of new class and, when input is of old class, $f_t(x;\theta)$ will be suppressed because of the penalization from distillation loss during training time. The confusion happens in the original version without pseudo neurons can thus be addressed.

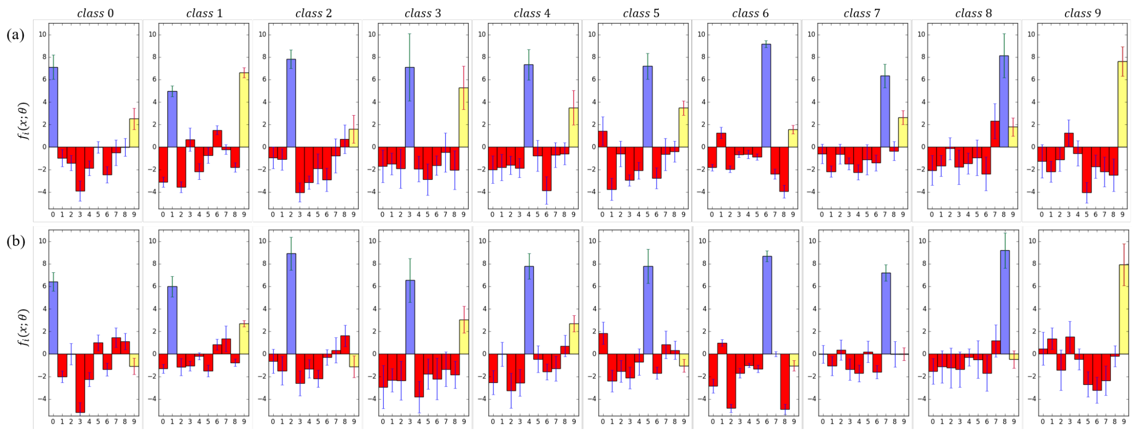


Figure 4.6: Distributions of output logits (a) without pseudo neurons and (b) with pseudo neurons after learning one new class. Each column computes means and standard deviations of one batch of data of same class. Value of correct class index are coloured blue while value of new class index are coloured yellow for clear comparison.

To further demonstrate the phenomenon, we show experimental observation by initially training a CNN model using data of 9 randomly selected instances from RGB-D Object Dataset [45] and then incrementally train the model to learn one new instance class.

doi:10.6342/NTU201703298

Figure 4.6 shows the resulting distributions of $f(x;\theta)$. Each distribution is obtained by averaging the output logits $f(x;\theta)$ among batch of testing images. Images in the same batch belong to the same class,e.g. the 3'rd column is obtained from a batch of data that all belongs to the 3'rd instance class. We can see with pseudo neurons (Figure 4.6 bottom row), $f_t(x_{old};\theta)$ is more suppressed than the one without pseudo neurons (Figure 4.6 top row), meaning that new class neuron is less likely to effect the final decision from CNNs to predict the right answer when seeing images of old class. Also see Table 5.6 in Chapter 5.3 for resulting accuracies. Note that data in RGB-D Object Dataset is recorded on turntable and only has variance of rotation. When using more challenging data or when CNN model has initially learned more classes, the phenomenon will enlarge and lead to a significant gap of performance between the one with pseudo neurons and the one without pseudo neurons (see Chapter 5.1).

doi:10.6342/NTU201703298

Chapter 5

Experiment

We implement all approaches using Caffe [46], an open source library that is specifically built for designing CNN models and support GPU computation. We run our programme on NVIDIA GEFORCE GTX 960M.

5.1 Incremental learning instance

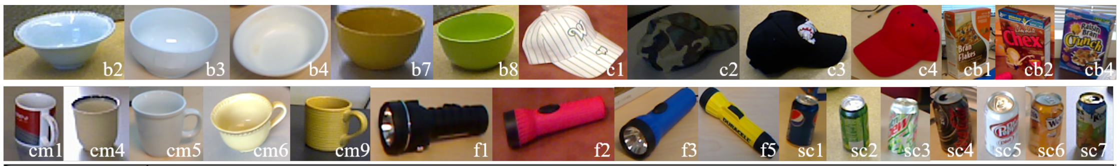
We show the feasibility of our proposed work by several experiments in this chapter. We mainly conduct three experiments. The first experiment focuses on the overall performance about ”whether the approach can incrementally learn new instance without losing previous knowledge”. Then we perform ”whether the approach can learn robustness given more diverse image conditions of an existing class”. Secondly, we give insightful analysis of our two innovations to show each component’s effectiveness. Finally, our recent attempts reveal a new kind of pseudo data, which replaces recollective data and can largely save training time for deriving activation maximized images.

5.1.1 RGB-D Dataset

We demonstrate the feasibility of our approach in practical scenario by using RGB-D Scenes Dataset [45]. The dataset is collected from natural scenes in everyday life. Moreover, we augment the dataset with additional four scene data in RGB-D Scenes Dataset v.

doi:10.6342/NTU201703298

2 [47]. The overall dataset is composed of 11 scene datasets[1] and totally 28 instances[2] (see Figure 5.1). We separate 11 datasets into 8 datasets and 3 datasets. We subsample each of the 8 datasets by taking every fifth video frame to form training data, and the remaining for testing. The 8 testing datasets are thus consisted of instances with different view points compared to training data. We denote the 8 testing datasets as *testset1*. In addition, the other 3 datasets (the last three in Figure 5.1) from totally different scenes with more imaging conditions (illumination, occlusion, background, etc.) which are not seen in training datasets are selected to evaluate the final performance. We denote this second testing set as *testset2*.



Instance / Scene	b2	b3	b4	b7	b8	c1	c2	c3	c4	cb1	cb2	cb4	cm1	cm4	cm5	cm6	cm9	f1	f2	f3	f5	sc1	sc2	sc3	sc4	sc5	sc6	sc7
desk	-	v	v	-	-	-	-	-	v	v	-	-	-	-	v	-	-	v	-	v	v	-	-	-	v	-	v	-
kitchen_small	-	-	v	-	-	v	-	-	-	-	v	-	-	-	v	v	-	-	v	-	v	v	-	-	-	-	v	-
meeting_small	v	v	-	-	-	v	-	v	-	v	v	-	-	-	v	v	-	-	v	-	v	v	-	v	-	v	-	-
table	v	-	-	-	-	v	-	-	v	-	-	v	v	v	-	-	-	-	-	v	-	-	-	-	v	-	-	-
scene_05	-	-	-	v	-	v	-	-	-	-	v	-	-	v	-	-	-	-	-	-	-	-	-	-	-	-	-	-
scene_07	v	-	-	-	-	-	v	-	-	v	-	-	v	-	-	-	-	-	-	-	-	-	-	-	-	-	-	v
scene_09	-	-	-	-	v	-	-	-	-	v	-	-	-	-	-	-	-	-	-	-	-	-	v	-	-	-	-	-
scene_10	v	-	-	-	-	-	-	v	-	-	v	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
table_small	-	-	v	-	-	-	-	-	v	v	-	v	v	-	-	-	-	-	-	-	-	v	-	v	-	-	-	-
scene_13	v	-	-	-	-	-	-	-	-	-	v	-	-	-	-	-	v	-	-	-	-	-	-	-	-	-	v	-
scene_14	-	-	-	-	v	-	v	-	-	-	-	-	-	v	-	-	-	-	-	-	-	-	-	-	-	-	v	-

Figure 5.1: Information table of the 11 scene datasets and appearances of each instance.

5.1.2 Implementation Details

We implement several approaches that also deal with catastrophic forgetting as comparison. For all approaches, we adopt network architecture proposed by Zeiler and Fergus [48] and pre-train models on ImageNet [6]. Specifically, the model we denote as ZF net has 8 layers (see Figure 5.2). When an input image is fed into ZF net, the image is

[1]We merge `desk_[1-3]` to `desk` and `table_small_[1-2]` to `table_small`.

[2]`b`, `c`, `cb`, `cm`, `f` and `sc` are the abbreviations of `bowl`, `cap`, `ceareal_ box`, `coffee_mug`, `flashlight` and `soda_can` respectively

doi:10.6342/NTU201703298

convolved with 96 different 1st layer filters, each of size 7 by 7 and with stride size of 2 in both x and y direction. The resulting feature maps are then: (i) passed through a rectified linear function (ReLU), (ii) max pooling operation within 3x3 regions, using stride size of 2 and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. See Figure 5.2) for detailed difference. After convolutions are two dense layers, which first flatten features from the top convolutional layer to form a 9216-dimension $(6 \cdot 6 \cdot 256)$ vector and then output 4096-dimension feature vector. The final layer is a classification layer which acts on the feature vector and output a predefined C-dimension vector, where C is a prdefined number of classes. To enable incremental learning, we discard the final layer and add extensible classification layer to accommodate new classes. Additionally, we add `ROIPooling` layer [49] right before `fc6` layer so that networks are able to accommodate different size images and classify local patches of an image.

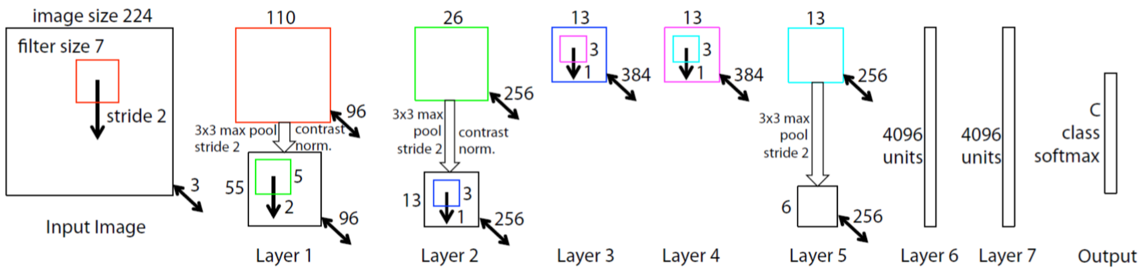


Figure 5.2: ZF model.

During training, we let all parameters after `conv5` be finetuned[3] for all approaches except for *fixed representation*. During training, we set fixed learning rate of 0.0001, weight decay of 0.0005 and mini-batch of 5 images (around 25 samples) per iteration. Each incremental training stage consists of 80 epochs or stops early if average classification loss over one epoch is below 0.005.

[3]Finetuning to early layers reduce the performance for all approaches. This is because finetuning to early layers harms the ability of rich feature representations obtained by pre-training on ImageNet

doi:10.6342/NTU201703298

Baseline

the baseline solution is *fixed representation*, which only learns the weights of new classifiers or existing classifiers depending on whether current training dataset contains old classes or only new classes. This approach avoids forgetting old knowledge by fixing exiting classifiers' weights. Figure 5.3 draws learnable part for the condition where current dataset contains both new class and old class.

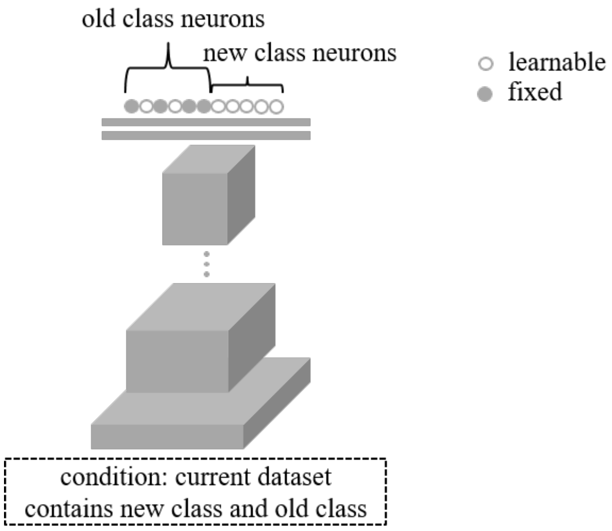


Figure 5.3: Demonstration of trainable weights for method "fix representation" in the condition where current dataset contains both new class and old class.

Less-forgetting Learning in Deep Neural Networks

Jung *et al.*'s work [39] mainly deals with domain adaptation. As the task doesn't require adding new class neuron in final layer, we manually modify their approach to be class-extensible and leave other elements unchanged. Their approach tries to maintain final classifier's decision boundary by fixing weights and keep feature representations from last layer (before classification layer) by using Euclidean loss. In addition, they allow intermediate layers trainable. We show modified version in Figure 5.4 and denote the version as LF*.

doi:10.6342/NTU201703298

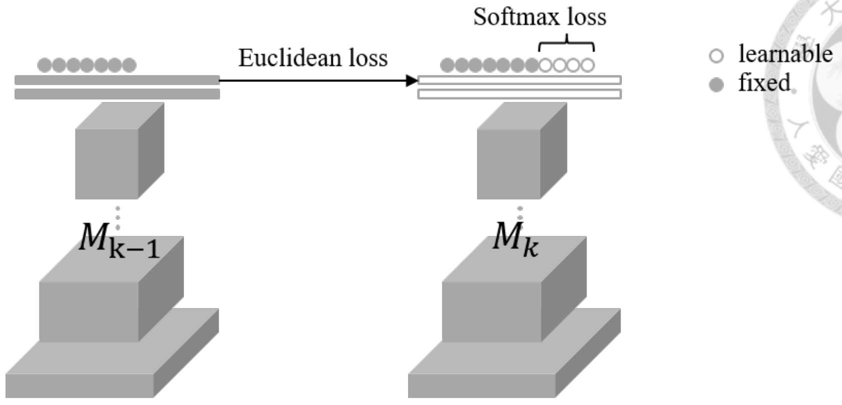


Figure 5.4: Modified version of [39] to allow class-extensible.

Compete to Compute

We implement Local-Winner-Takes-All (LWTA) [31] in `fc6` and `fc7` with blocks of size equal to 16 which yields the best performance among 4, 8 and 16. Training with LWTA is exactly the same as finetuning the current model. The difference is signal transition is now constrained by only permitting max value to be passed between each layer.

Learning without Forgetting

Li and Derek's work [37] can be directly fit into our scenario by simply taking multi-task learning as multi-class learning. We denote the method as LwF. The training rule is very similar to pseudorehearsal that also distil old model's knowledge. The distinction is that instead of using additional data (e.g. pseudo data), LwF leverages current available data as source to preserve knowledge. Besides, LwF uses the distillation loss proposed by Hinton *et al.* [35] rather than L2 distance loss we adopt in our approach. The loss used in LwF [37] is defined as

$$\mathcal{L}_{dis,LwF} = \sum_{(x,y)\in D} \sum_{i=1}^{C_o} g_i(x;\theta^{k-1}) \log g_i(x;\theta^k) \tag{5.1}$$

doi:10.6342/NTU201703298

where $g_i(x; \theta^{k-1})$ is the i^{th} element of output probabilities from old model M_{k-1} with fixed weights θ^{k-1}. θ^k is thr current model's weights, which are trainable. The overall loss is thus summation of common classification loss in (4.3) and LwF distillation loss in (5.1).

Pseudo Rehearsal

For original pseudorehearsal [40], we set $\tilde{n}$ (the number of pseudo sample per class) equal to 80. Before one training stage, randomized images as pseudo data are generated for later usage. During training, loss weights λ_{cls} and λ_{dis} are set to 0.1 and 0.1 respectively. Batch size for pseudo data is set to 16 samples per iteration.

Pseudo Rehearsal with Imaging Recollection and Pseudo Neurons

The training settings of our approach is the same as pseudorehearsal. Additionally, regarding the proposed pseudo neurons, the number of pseudo neurons is set to 10.

5.1.3 Results

Figure 5.5 shows the curve indicating the degree of how much the networks preserve the old knowledge by retrospecting testing data of previous scenes. Among all works that mitigate catastrophic forgetting, ours outperforms all of them in intermediate stages and final stage. The most closest one to us is LwF, which is behind ours by around 5%. Nonetheless, as our goal is to train a robust recognition system in incremental manner, the results by testing on a *testset2* which contains various imaging conditions is more representative. Table 5.1 shows the overall performance on *testset1* and *testset2* after training on the 8 datasets. We can see that after incrementally learning 28 instances from different scenes, our approach learns better invariance compared to others, reaching 74.48% accu-

doi:10.6342/NTU201703298

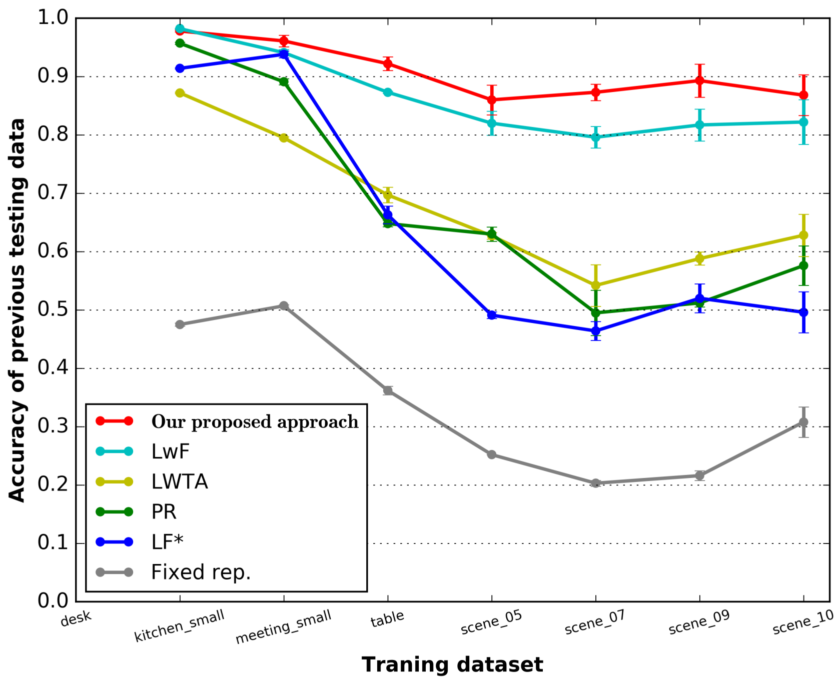


Figure 5.5: Curve demonstrating how much knowledge is preserved by testing model on accumulated testing data.

racy on *testset2* and outperforming the second best LwF with a margin of 13.12%. We further report the accuracies (see Table 5.2) over additional 4 trials with random training order of the 8 datasets to avoid any special case. All the results are consistent and show that our approach outperforms the others.

Table 5.1: % Accuracy on *testset1* and *testset2* after incremental training

Approach	*testset1*	*testset2*
Fix rep.	35.30 ± 1.98	22.53 ± 1.54
LF* [39]	50.38 ± 2.22	39.54 ± 2.50
PR [40]	59.55 ± 3.07	47.93 ± 3.11
LWTA [31]	65.27 ± 3.51	48.79 ± 2.78
LwF [37]	82.54 ± 5.12	61.32 ± 4.72
Ours	$\mathbf{87.23 \pm 3.02}$	$\mathbf{74.48 \pm 2.21}$

doi:10.6342/NTU201703298

Table 5.2: % (*testset1/testset2*) Accuracy over 4 trials

Approach	trial 1	trial 2	trial 3	trial 4	average
Fix rep.	35.3 / 22.5	50.1 / 36.2	62.7 / 44.9	28.1 / 16.0	46.2 / 33.2
LF* [39]	39.6 / 34.8	73.6 / 48.0	55.2 / 32.6	54.8 / 47.8	54.7 / 40.5
PR [40]	64.7 / 47.1	68.6 / 52.9	58.1 / 35.2	64.0 / 54.8	63.0 / 47.6
LWTA [31]	76.8 / 52.6	75.0 / 50.2	67.1 / 42.8	66.8 / 51.6	71.0 / 50.2
LwF [37]	85.0 / 66.7	84.8 / 66.1	86.4 / 63.7	72.1 / 54.1	82.2 / 62.4
Ours	95.2 / 81.9	90.9 / 76.3	88.4 / 75.3	81.5 / 72.7	**88.6** / **76.1**

5.2 Learning Invariance

We have already shown the overall performance of our approach on RGB-D Scenes Dataset that contains data of new classes or different imaging conditions in each incremental training stage. In this part, we separately investigate whether our approach can learn invariance by seeing same objects but with different imaging conditions.

5.2.1 GMU Kitchen Dataset

In this experiment, we rely on GMU Kitchen Dataset [50], which is similar to RGB-D Scene Dataset but with more challenging imaging conditions. The dataset contains 9 cluttered environments each with 9 to 11 objects[4] We select 7 scene datasets that include the 11 objects. Then among the 7 datasets, we randomly chose 3 datasets for incremental training and the remaining 4 for testing. Similar to RGB-D Scene Dataset, training data are collected by subsampling every 5 frames for each training dataset. Implementation settings of approaches are also the same. We conduct 3 trials with different training datasets and testing data. Every trial is run for 5 times to obtain average performance. We report *fixed representation* as baseline, *LwF*, and our approach.

[4]The dataset contains main objects and extra objects. We use main objects here.

doi:10.6342/NTU201703298

5.2.2 Results

Table 5.3 summarizes the results. Surprisingly, all approaches can stably improve invariance by looking more data while previous data is not present, even for *fixed representation*. The gaps between approaches are inapparent in this aspect. The finding also implies the major difference comes from the ability to generalize new classes while not forgetting old classes.

Table 5.3: % Accuracy of invariance learning on GMU datasets

Testing set: `gmu_scene_[05,06,07,09]`

Training order	`gmu_scene_01`	`gmu_scene_03`	`gmu_scene_04`
Fix rep.	54.42 ± 0.35	60.42 ± 0.33	69.40 ± 0.51
LwF [37]	54.14 ± 0.64	61.74 ± 0.14	70.23 ± 0.13
Ours	53.87 ± 0.78	63.48 ± 0.20	72.08 ± 0.21

Testing set: `gmu_scene_[01,03,04,05]`

Training order	`gmu_scene_06`	`gmu_scene_07`	`gmu_scene_09`
Fix rep.	50.94 ± 0.77	62.99 ± 0.33	63.00 ± 0.37
LwF [37]	51.89 ± 0.36	63.97 ± 0.97	63.65 ± 0.58
Ours	51.95 ± 0.73	64.42 ± 0.52	66.87 ± 0.50

Testing set: `gmu_scene_[01,04,06,09]`

Training order	`gmu_scene_03`	`gmu_scene_05`	`gmu_scene_07`
Fix rep.	47.45 ± 1.16	64.18 ± 0.26	67.68 ± 0.48
LwF [37]	48.76 ± 0.53	64.68 ± 0.22	68.99 ± 0.32
Ours	47.55 ± 0.33	64.24 ± 0.16	69.86 ± 0.31

5.2.3 Discussion

CNNs pretrained on ImageNet [6] do not possess the property to be robust against imaging variance according to [51], yet CNNs seems to be able to incrementally learn this property without degeneration. The finding is somehow valuable as it suggests that when vision task is to recognize some certain objects (or human faces) under different imaging conditions, one do not need to keep the whole database but is able to directly train CNNs

doi:10.6342/NTU201703298

model on new data without suffering forgetting problem. In addition, this attribute largely relates to the work proposed by Held *et al.* [20]. We would like to investigate this as one of our future works.

5.3 Analysis of Imaging Recollection and Pseudo Neurons

In this section we provide additional analysis of the proposed approach. Our approach is mainly composed of two components described in previous sections. The first one is the replacement of randomized images by recollective images. The second one is the usage of pseudo neurons. To demonstrate the effectiveness of both components, we individually analyze them with different parameter settings, and to be clear, we only report the results of *testset2*. Note that the experiments conducted are performed using the training order of scene datasets as shown in Figure 5.1. Also note that if we use randomized data and no pseudo neurons, our approach is the same as the original pseudorehearsal [40].

5.3.1 Imaging Recollection

we first show the results of using randomized images and using recollective images with varying number, all without pseudo neurons. The results shown in Table 5.4 demonstrate the advantage of using recollective images, which improves around %10 compared to the randomized one. It meets our expectations that even though the images are not recognizable for human (see Figure 5.6 for example), they are the associations reflecting some important concepts the networks learned in early stages, hence serving as a good source for pseudorehearsal. On the other hand, the improvements by augmenting the amount of pseudo data are minor for both cases. This implies that although the appearances of large recollective images representing the same concept are different from each other, the trig-

doi:10.6342/NTU201703298

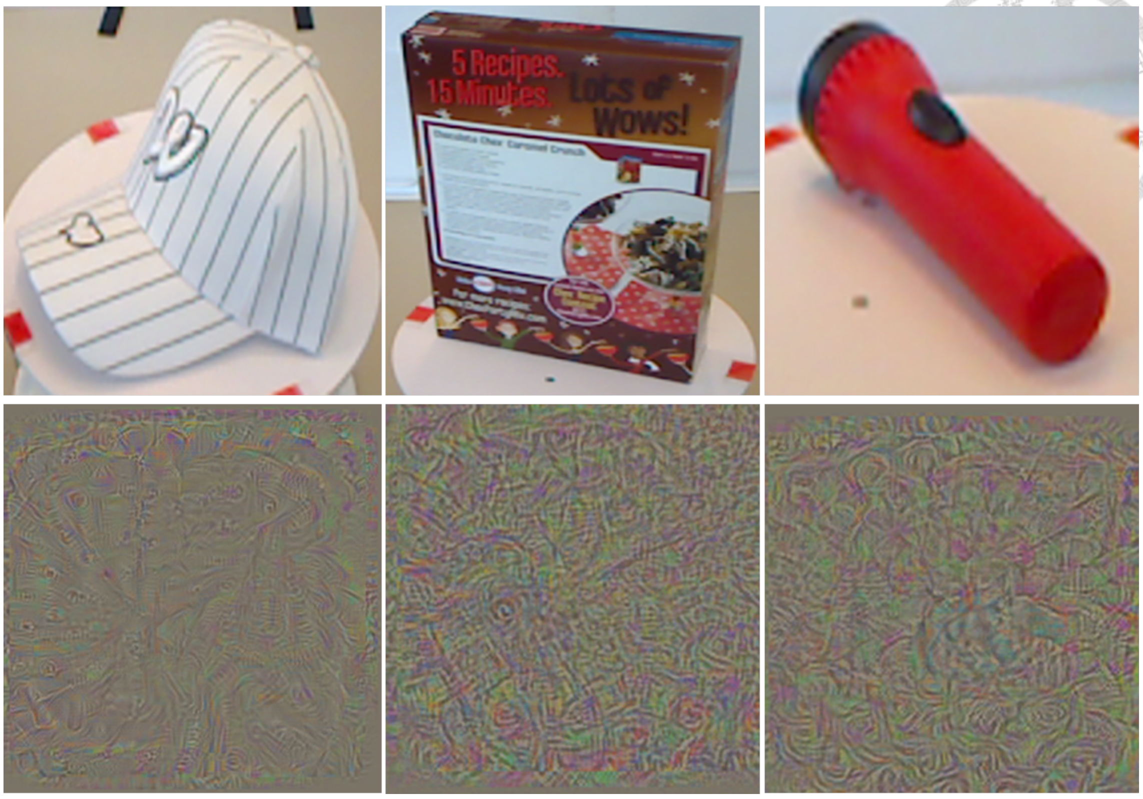


Figure 5.6: Examples of the recollective images (bottom row) that represent the concept of `cap_1`, `cereal_box_2` and `flashlight_2`, and the truly instances from RGB-D Object Dataset (top row).

gered distributions of representations inside the networks are similar. Consequently, in training time the weights will be updated toward a similar direction. The diversity provided by large amount of data is thus limited and the performance is quickly saturated.

To inspect this property, we project feature representations of recollective images from layer `conv5`, `fc6`, and `fc7` into lower dimension by applying t-SNE [52]. Figure 5.7 (a), (b), and (c) show the resulting clusters of 10 selected classes from each layer. The phenomenon of assembling already emerge in `conv5` layer, and is more convergent in `fc7` layer. Different from inference time, where an early convergent phenomenon may be desired, we wish the resulting feature maps of `conv5`[5] are diverse distributions and

[5]Representations from `conv5` are ”raw data” to CNNs as they are unchangeable due to fixed weights in our experiments. If we instead fintune this layer, then ”raw data” for CNNs is representations from `conv4` layer.

doi:10.6342/NTU201703298

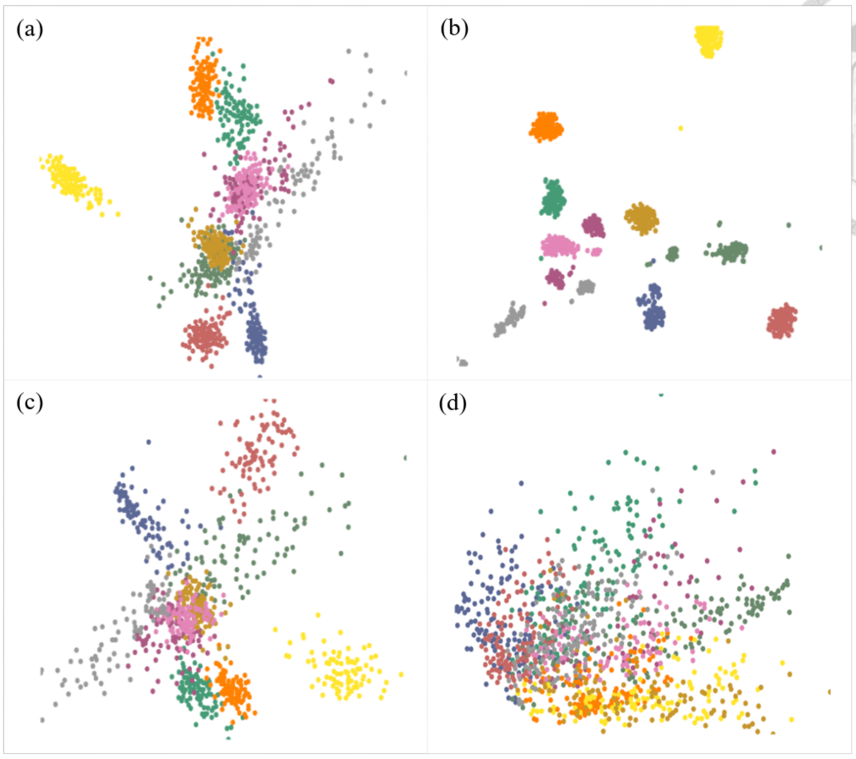


Figure 5.7: t-SNE results of (a) `fc6`, (b) `fc7`, and (c) `conv5` from recollective images and (d) `conv5` from real training data. Different color indicates different class.

can properly represent data of old classes. As a comparison with pseudo data, we show the feature projection (see Figure 5.7 (d)) from `conv5` using original training images of the 10 classes. We can see from real training data, distributions of encoded representations are diverse. This means that although recollective data encode useful information, it is lack of diversity to represent all possible data of old class. We leave the possibility to enhance diversity of data for distilling knowledge in Chapter 5.5.

5.3.2 Pseudo Neurons

Table 5.5 shows the improvements obtained by the addition of pseudo neurons. Firstly, pseudo neurons improve the original pseudorehearsal from 45.52% to 51.26%, and maximally to 57.26%. This demonstrates the generality that the advantage of pseudo neurons is not a special case for recollective images. Moreover, combining recollective images with

doi:10.6342/NTU201703298

Table 5.4: The effects of using recollective images

Settings	*testset2*
# *randomized* images per class	
16	48.97
32	45.52
80	47.93
# *recollective* images per class	
16	57.18
32	58.68
80	59.48
160	59.77
320	59.71

Table 5.5: The effects of pseudo neurons

Settings	*testset2*
# pseudo neurons with 32 *randomized* images per class	
0	45.52
5	51.26
10	53.62
20	54.54
30	57.29
50	57.18
# pseudo neurons with 32 *recollective* images per class	
0	58.68
5	72.48
10	71.95
20	73.97
30	73.90

5 pseudo neurons surprisingly boosts the accuracy from 58.68% to 72.48%, a considerable improvement of 13.8%. The results conform our thoughts in Chapter 4.4, where the confusing situation is alleviated by adopting pseudo neurons. We also report the resulting accuracies corresponding to Figure 4.6 in Table 5.6. These results reflect the impact as we mentioned in Chapter 4.4. Without pseudo neurons, the accuracies degenerate to different extents for some classes. On the contrary, with pseudo neurons only the accuracy of one

doi:10.6342/NTU201703298

class declines slightly.

Table 5.6: % Accuracy of 10 instances with/without Pseudo Neurons (PN)

without PN:

class	ins0	ins1	ins2	ins3	ins4	ins5	ins6	ins7	ins8	ins9
init.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	-
incre.	**77.08**	**0.00**	100.00	**67.57**	**98.31**	100.00	100.00	**97.22**	**98.08**	100.00

with PN:

class	ins0	ins1	ins2	ins3	ins4	ins5	ins6	ins7	ins8	ins9
init.	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	-
incre.	100.00	100.00	100.00	**97.30**	100.00	100.00	100.00	100.00	100.00	100.00

init.: After initially learn 9 instances.
incre.: After incrementally learn 1 new instance.

5.4 Extended Experiment: Various Pseudo Resource

In this extended experiment, we seek other alternative type of data to serve as pseudo data. Randomized images utilized by original pseudorehearsal [40] lack insufficient information and thus perform poor preservation of previous knowledge. Our proposed recollective image data although addresses this issue and gain significant improvements, generation of activation maximized images takes too much time to form pseudo dataset in each incremental training stage. To this end, we are interested in other possible kind of image that can save time for generation and maintain informative. We turn our focus to the option of using natural images. Though we are not allow to use previous real data, in real world we can always access other realistic and available datasets such as ImageNet data [6] or PASCAL VOC data [7]. We are curious whether the sample pair composed of arbitrary natural image and the corresponding targets which are soft outputs from previous model can also serve as a good resource to recall old knowledge. Such image data leave out the time needed for generation and we thus only need to see whether it also encode

doi:10.6342/NTU201703298

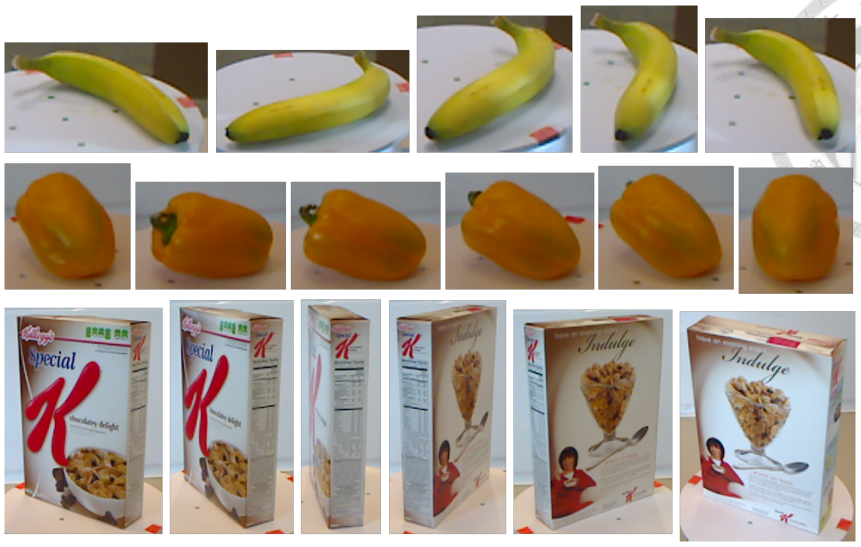


Figure 5.8: Example of recorded instance on a turntable in RGB-D Object Dataset.

useful information like recollective image or not.

We simply choose two kinds of natural image data to investigate this option, one using data from RGB-D Object Dataset [45] (different from RGB-D Scene Dataset), which is form by recording instances on a turn table[6] (Figure 5.8), and the other using 2008 PASCAL VOC images for classification/detection task[7] (Figure 5.9). Some partial instances from RGB-D Object Dataset are actually instances from RGB-D Scene Dataset. The reason we apply this alternative is that although we do not have previous data that offers diverse conditions, we may be allowed to use minimum images that represent one certain instance. Once CNN model learns an instance from data containing diverse conditions, the model may be able to recall knowledge by only looking at a small amount of images of this instance when conducting incremental learning. On the other side, the usage of PASCAL VOC dataset is to examine the utility of a very diverse dataset as resource to preserve old knowledge.

Experiment is conducted using the setting similar to the one in Chapter 5.1 (see Figure 5.5) and we show the accuracy curves of *testset1* by altering data resource for knowl-

[6]So the data contains multi view video frames
[7]The dataset contains about ten thousands images

doi:10.6342/NTU201703298

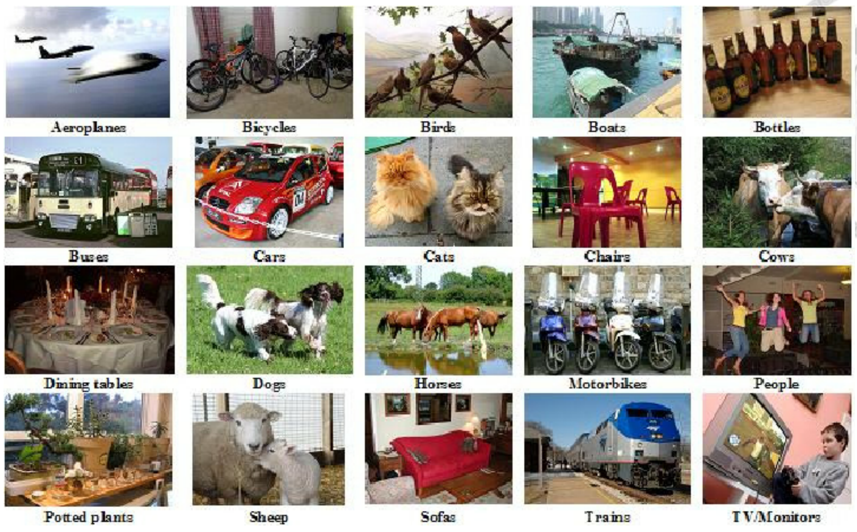


Figure 5.9: Examples images from 2008 PASCAL VOC dataset.

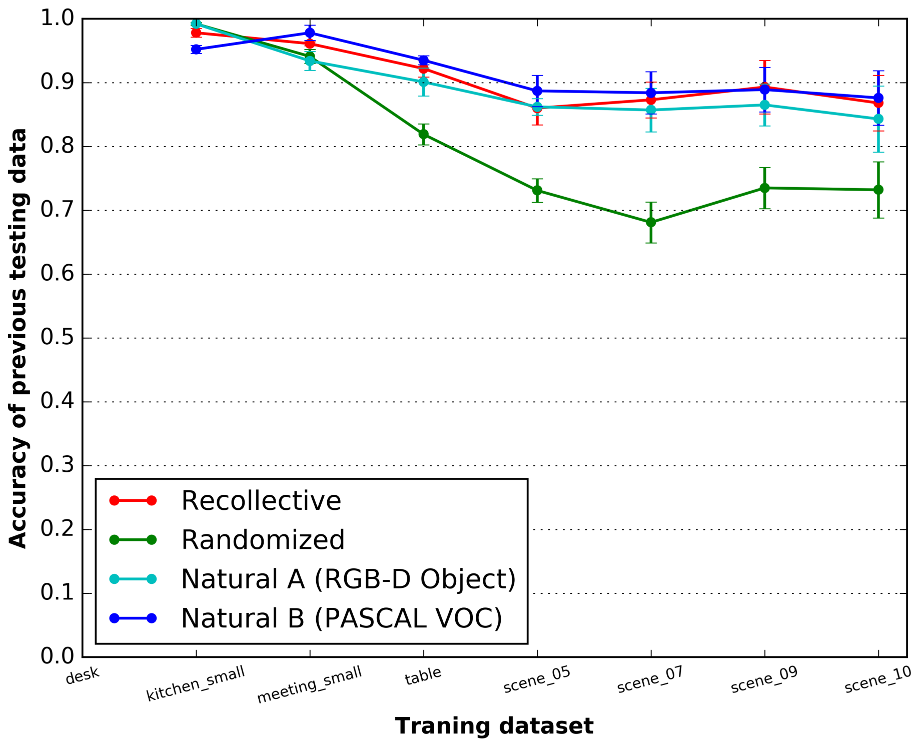


Figure 5.10: Accurcy curve by altering data resource for knowledge distillation

edge distillation in Figure 5.10.

The results are surprising as we observe that using cheap natural image data has comparable affect for preserving old knowledge, especially for the alternative using PASCAL VOC images. Natural data not only discard the time needed for generation but maintains promising performance same as using recollective pseudo image data. We infer this is

doi:10.6342/NTU201703298

because natural images containing realistic and diverse patterns and textures can provide a large ”information pool” for model. CNN model trained with the combination of distillation loss and classification loss is able to both extract useful information from such a pool.

doi:10.6342/NTU201703298

Chapter 6

Conclusion and future work

In the thesis we propose an incremental learning approach in which data with unseen classes or existing classes but with different imaging conditions can be well learned in incremental manner without access to previous data. We develop our approach based on pseudorehearsal with two contributed ideas. We show that images generated by activation maximization encode useful information that can serve as a good data source to review old knowledge. Such a generative procedure we interpret as a behaviour of ”recollection”. Furthermore, to distil the knowledge from every pseudo sample while simultaneously gain knowledge from new data, we introduce pseudo neurons as a bridge connecting both new and old information. In this way neurons belonging to new classes are trained more reasonably. Our experiments demonstrate the effectiveness of the combination of both ideas which shows superior performance compared to other works. We then further show insights of our approach by individually ablating the components, which proves our concept of learning behaviour. Finally, we have raised some problems or phenomenons in the context, which will be left as future works.

doi:10.6342/NTU201703298

Appendix A

Complete training pipeline

Figure A.1 displays step-by-step illustration of our incremental learning approach.

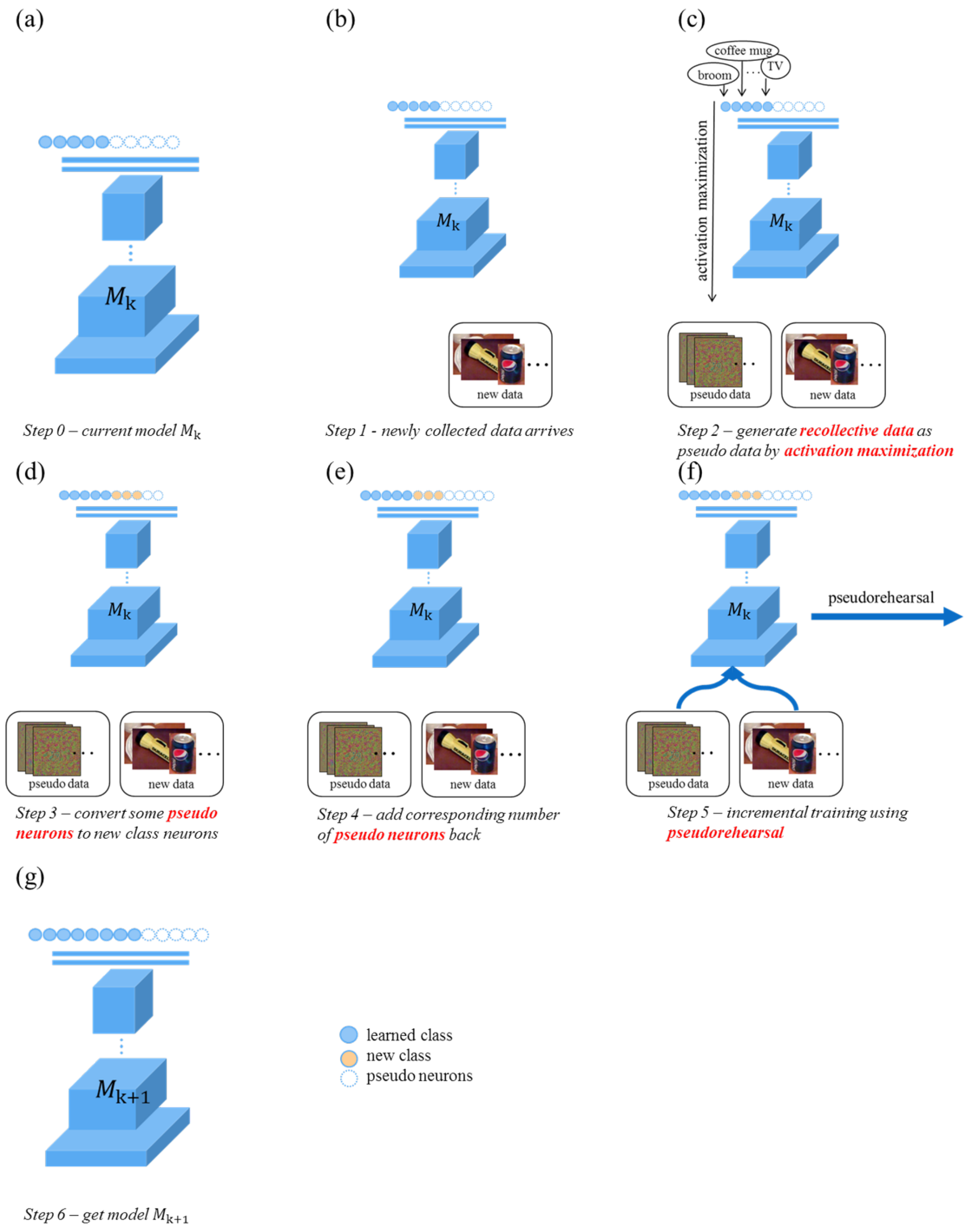

(a)
M_k
Step 0 – current model M_k
(b)
M_k
new data
Step 1 - newly collected data arrives
(c)
coffee mug
broom
TV
activation maximization
M_k
pseudo data
new data
Step 2 – generate recollective data as pseudo data by activation maximization
(d)
M_k
pseudo data
new data
Step 3 – convert some pseudo neurons to new class neurons
(e)
M_k
pseudo data
new data
Step 4 – add corresponding number of pseudo neurons back
(f)
M_k
pseudorehearsal
pseudo data
new data
Step 5 – incremental training using pseudorehearsal
(g)
M_{k+1}
Step 6 – get model M_{k+1}
learned class
new class
pseudo neurons

Figure A.1: Step-by-step illustration of our incremental learning approach.

doi:10.6342/NTU201703298

Reference

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer vision–ECCV 2006*, pages 404–417, 2006.

[2] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[3] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012.

[4] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[7] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

doi:10.6342/NTU201703298

[8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[9] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[12] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Hossein Azizpour, Ali Sharif Razavian, Josephine Sullivan, Atsuto Maki, and Stefan Carlsson. From generic to specific deep representations for visual recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 36–45, 2015.

[14] George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on audio, speech, and language processing*, 20(1):30–42, 2012.

doi:10.6342/NTU201703298

[15] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[16] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, page 3, 2010.

[17] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[18] Ruslan Salakhutdinov and Geoffrey Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[20] David Held, Sebastian Thrun, and Silvio Savarese. Robust single-view instance recognition. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 2152–2159. IEEE, 2016.

[21] Giulia Pasquale, Carlo Ciliberto, Lorenzo Rosasco, and Lorenzo Natale. Object identification from few examples by improving the invariance of a deep convolutional neural network. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pages 4904–4911. IEEE, 2016.

[22] Robi Polikar, Lalita Upda, Satish S Upda, and Vasant Honavar. Learn++: An incremental learning algorithm for supervised neural networks. *IEEE transactions on*

doi:10.6342/NTU201703298

systems, man, and cybernetics, part C (applications and reviews), 31(4):497–508, 2001.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] Suhang Wang, Yilin Wang, Jiliang Tang, Kai Shu, Suhas Ranganath, and Huan Liu. What your images reveal: Exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web*, pages 391–400. International World Wide Web Conferences Steering Committee, 2017.

[25] Ning Qian. On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151, 1999.

[26] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[27] Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological review*, 97(2):285–308, 1990.

[28] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. icarl: Incremental classifier and representation learning. *arXiv preprint arXiv:1611.07725*, 2016.

[29] Robert M French. Semi-distributed representations and catastrophic forgetting in connectionist networks. *Connection Science*, 4(3-4):365–377, 1992.

doi:10.6342/NTU201703298

[30] Robert M French. Dynamically constraining connectionist networks to produce distributed, orthogonal representations to reduce catastrophic interference. *network*, 1111:00001, 1994.

[31] Rupesh K Srivastava, Jonathan Masci, Sohrob Kazerounian, Faustino Gomez, and Jürgen Schmidhuber. Compete to compute. In *Advances in neural information processing systems*, pages 2310–2318, 2013.

[32] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, page 201611835, 2017.

[33] Michael D Muhlbaier, Apostolos Topalis, and Robi Polikar. Learn++. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes. *IEEE transactions on neural networks*, 20(1): 152–168, 2009.

[34] Lee Sang Woo, Heo Min Oh, Kim Jiwon, Kim Jeonghee, and Zhang Byoung Tak. Dual memory architectures for fast deep learning of stream data via an online-incremental-transfer strategy. *arXiv preprint arXiv:1506.04477*, 2015.

[35] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[36] Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in neural information processing systems*, pages 2654–2662, 2014.

doi:10.6342/NTU201703298

[37] Zhizhong Li and Derek Hoiem. Learning without forgetting. In *European Conference on Computer Vision*, pages 614–629. Springer, 2016.

[38] Saurabh Gupta, Judy Hoffman, and Jitendra Malik. Cross modal distillation for supervision transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2827–2836, 2016.

[39] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016.

[40] Anthony Robins. Catastrophic forgetting, rehearsal and pseudorehearsal. *Connection Science*, 7(2):123–146, 1995.

[41] Robert M French, Bernard Ans, and Stephane Rousset. Pseudopatterns and dual-network memory models: Advantages and shortcomings. In *Connectionist models of learning, development and evolution*, pages 13–22. Springer, 2001.

[42] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341:3, 2009.

[43] Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv: 1506.06579*, 2015.

[44] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[45] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

doi:10.6342/NTU201703298

[46] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[47] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3d scene labeling. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3050–3057. IEEE, 2014.

[48] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.

[49] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[50] Georgios Georgakis, Md Alimoor Reza, Arsalan Mousavian, Phi-Hung Le, and Jana Košecká. Multiview rgb-d dataset for object instance detection. In *3D Vision (3DV), 2016 Fourth International Conference on*, pages 426–434. IEEE, 2016.

[51] Ali Borji, Saeed Izadi, and Laurent Itti. ilab-20m: A large-scale controlled object dataset to investigate deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2221–2230, 2016.

[52] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

doi:10.6342/NTU201703298

VITA

姓名: 柯達方

性別: 男

生日: 民國 81 年 1 月 13 日 (1992/01/13)

籍貫: 中華民國台北市

學歷:

1. 民國 106 年 國立台灣大學電機工程學研究所畢業
2. 民國 103 年 國立台灣大學工程科學與海洋工程學系畢業
3. 民國 99 年 國立師範大學附屬高級中學畢業

發表著作:

1. Ren C. Luo, Da-Fang Ke, ”Mitigate Catastrophic Forgetting in CNNs for Effective Instance Recognition.” 49th International Symposium on Robotics (ISR 2017Asia), Shanghai, China, July 5-8, 2017.

榮譽事蹟:

2016 年	「2016 長庚醫療財團法人醫療機器人大賽」榮獲團體組冠軍
2016 年	「2016 上銀智慧機器手第九屆實作競賽」榮獲團體組亞軍
2016 年	「2016 智慧機器人創意競賽國產工業機器人組」榮獲團體組季軍

doi:10.6342/NTU201703298