ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Autonomous Systems Lab
Prof. Roland Siegwart

Semester Thesis

Continual Learning for Binary Foreground and Background Segmentation

Autumn Term 2020

Supervised by:
Abel Gawel
Hermann Blum
Francesco Milano

Author:
Kaiyue Shen

Declaration of Originality

I hereby declare that the written work I have submitted entitled

Continual Learning for Binary Foreground and Background Segmentation

is original work which I alone have authored and which is written in my own words.[1]

Author(s)

Kaiyue	Shen

Student supervisor(s)

Abel	Gawel
Hermann	Blum
Francesco	Milano

Supervising lecturer

Roland	Siegwart

With the signature I declare that I have been informed regarding normal academic citation rules and that I have read and understood the information on 'Citation etiquette' (`https://www.ethz.ch/content/dam/ethz/main/education/rechtliches-abschluesse/leistungskontrollen/plagiarism-citationetiquette.pdf`). The citation conventions usual to the discipline in question here have been respected.

The above written work may be tested electronically for plagiarism.

Zürich 26.02.2021 — Place and date

Kaiyue Shen — Signature

[1] Co-authored work: The signatures of all authors are required. Each signature attests to the originality of the entire piece of written work in its final form.

Contents

Abstract

Continual learning is of great importance for autonomous agents since it is impossible to provide the artificial intelligent with all the knowledge it need in the real world. While previous work has proposed a large amount of continual learning methods, little research has been done on the segmentation task. For autonomous mobile robots, the differentiation of foreground and background matters a lot for the safety consideration. Therefore, we aim to solve the continual learning for binary foreground and background segmentation task.

To mitigate catastrophic forgetting, the biggest problem of continual learning, we totally implement five continual learning methods including fine-tuning (baseline), output distillation, feature distillation, EWC and Progress and Compress. Apart from the naive fine-tuning, the last four methods all adopt techniques to preserve the old knowledge. The first three add regularization terms on the output space, feature space and weight space respectively. The last one reuses the old parameters with a layerwise lateral connection in the model architecture. We evaluate and compare those methods on NYU and CLA dataset of different types of scene. The result demonstrates that continual learning methods can prevent catastrophic forgetting to a certain degree only when the old scene and new scene are similar. When two scenes differ a lot, the regularization-based methods may not be a good choice.

Chapter 1

Introduction

Scene understanding is the process of perceiving and analyzing the relationship between objects and 3D structure of the scene. It is of great importance for the autonomous mobile robot nowadays. The robot need to correctly tell the static and moving objects apart in order to safely move in the unknown environment. The breakthrough in deep learning field prompts the development of scene understanding, especially in the semantic segmentation tasks. Yet, an important drawback of most neural networks used for semantic segmentation is that they are trained offline using the already collected dataset. When an autonomous robot pretrained on such datasets moves in the dynamic environment, the practical 3D scene it encounters may differ a lot from its own knowledge, leading to the wrong segmentation of the foreground and background, and further, the failure of decision for the next movement. Therefore, it requires the robot to continually learn knowledge over time.

The main issue of continual learning is that the model has a tendency for catastrophic forgetting, that is, when the model keeps updating using the new data, the old knowledge stored will be gradually overwritten by the new one. Obviously, this is not what we want. Suppose our robot learns the segmentation task in the university cafeteria, then it moves to the sports center. It would be great if it continually learns how to correctly segment the scene of the sports center. And at the same time, remember how to segment the environment in the cafeteria. So if goes back to the cafeteria, it can quickly adapt to the environment. However, it is not easy. The ability to learn new knowledge and to preserve old knowledge are two key factors of continual learning.

Up to now, great work has been done on continual learning in deep learning and the proposed approaches either restrict new learning [1, 2, 3, 4, 5], or store a new set of parameters for each task [6, 7, 8], or require old training data [9, 10, 11]. However, most of them are only evaluated on image classification, object detection or reinforcement learning tasks. The goal of our project is to build a network that can continually learn the binary foreground and background segmentation task, the study on which is limited. Moreover, previous study on the segmentation task only evaluates on the well-labeled benchmark datasets (e.g., Pascal VOC2012 [12]) which differ a lot from the real industrial environment a robot will see. It is interesting to see how would the network perform on the real data.

In this work, we implement and compare five continual learning approaches for the binary segmentation task. A naive approach is to fine-tune the network with the data of the new scene based on pretrained parameters [13]. Output Distillation and Feature Distillation can be thought of as two approaches that add regularization on the output and feature space, which are realized by minimizing the prediction of the model pretrained on the old scene and prediction of the new model. Elastic Weight

Consolidation (EWC), different from above two methods, directly regularizes the parameter updating direction. By computing the fisher information which reflects the importance of every model parameter, it prevents the parameters that have great influences on the segmentation of the old scene from big changes. Progress and Compress, uses two models named knowledge base and active column. The former is used to store existed knowledge of segmentation and the latter learns how to segment in the new scene. During the progress stage, we update the active column with the help of knowledge base. During the compress stage, we distill the new knowledge from the active column back to the knowledge base. For evaluation, we set up two experiments. One is from "kitchen" scene to "bedroom" scene in NYU Depth V2 Dataset [14]. Another is from the whole NYU Depth V2 Dataset to CLA dataset which contains labeled images of real indoor scenes in CLA building.

Our contributions are as follows:

- We systematically summarize five continual learning methods, most of which have not been test on the semantic segmentation task. This may help people later who want to investigate on such topics quickly understand the advantage and disadvantage of existing methods and avoid to spare effort on already tested unsuccessful approaches.
- We evaluate all five approaches not only on well-labeled benchmark dataset (NYU Depth V2), but more importantly, on the real dataset collected from the university building (CLA). The difference between the results on two datasets indicates a potential space of improvement left for future work.

The report is structured as follows: First, Chapter 2 gives an overview about the previous work on semantic segmentation and continual lifelong learning. Next, in Chapter 3, we formulate the binary foreground and background segmentation task, and introduce the theory and training pipeline of five continual learning methodologies in the following order: fine-tuning, output distillation, feature distillation, EWC, Progress and Compress. After that, in Chapter 4, we show the experiments and results. Section 4.1 details the experiment setup, including the dataset, architecture and training process. Section 4.2 provides a complete analysis of two sets of experiments on different datasets. The performance of all implemented methods are compared both quantitatively and qualitatively. Finally, in Chapter 5, we conclude the work and list potential future work. The code is available. [1]

[1] `https://github.com/ethz-asl/background_foreground_segmentation/tree/kaiyue/all_methods`

Chapter 2

Related Work

2.1 Semantic Segmentation

Before the advent of deep learning, semantic segmentation task is usually solved by classical machine learning classifiers like SVM [15], Random Forest [16] or K-means Clustering [17]. Since convolution neural networks (CNNs) were first applied to semantic segmentation research field by Long et al. [18], now we see that state-of-the-art methods in semantic segmentation are based on CNNs trained with pixel-wise labeled images. We also see that most of the architectures have encoder-decoder structures [18, 19, 20, 21, 22]. To solve the problem of information loss, Ronneberger et al. [21] proposed the U-net architecture, i.e., to use shortcut connections to propagate information from downsampling layers in the encoder to corresponding upsampling layers in the decoder. Chen et al. [22] further improved the existing results with a multiple of techniques such as atrous convolutions, atrous spatial pyramidal pooling and conditional random fields.

In our work, since we focus on comparing the performance of different continual learning methods, in order to do a fair comparison, we only need to keep the basic encoder-decoder architecture fixed for all methods.

2.2 Continual Lifelong Learning

Continual lifelong learning aims to learn consecutive tasks without catastrophic forgetting, which, however, is of great challenges. This is because of the tendency of existing knowledge stored in the learned models to be overwritten by new information, which has been shown in [23].

Nowadays, different neural network approaches for continual learning have been proposed and we find they can be separated into three types: regularization methods, dynamic architecture methods, and memory replay methods. The first type [1, 2, 3, 4, 5] retrains the whole network with some regularization terms to prevent catastrophic forgetting. The second type retrains part of the network and expand it if needed [6, 7, 8]. For example, progressive networks [8] retains a group of pre-trained models, and generates a new model using lateral connections with the old models whenever a new task comes. The third type [9, 10, 11] uses memory replay to store the old data, which obviously requires more memory.

Our study mainly focuses on the first type which can be further divided into two categories based on what to control. One is to directly control the model parameters, e.g., Elastic Weight Consolidation (EWC) [1] adds constraints on model parameters to restrict the updating of parameters that are important for old tasks. Another is to control the layer activation such as outputs or intermediate features. Learning

without Forgetting (LwF) [2] uses the pseudo label generated by the model trained on old tasks as well as the true label to train the new model, where the pseudo label preserves the old knowledge. Similarly, incremental learning [4] uses the "pseudo" feature to regularize the learning procedure of the new model. Deep Generative Replay (GR) [3] learns a generator and a task solver at the same time, and then uses the learned generator to sample the old data when learning the new task. In certain fields such as medical imaging, since the generated data lacks pixel-level details which are important for biomedical diagnosis, this method may lead to poor performance. Keep and learn method [5] regularizes both the latent feature space and output space. It preserves the old knowledge by modeling the feature space and the output space to be mutually informative and restricting the feature in the modeled space. By minimizing both regularization terms, the paper proves, it is equal to maximize the mutual information.

Despite the success of different kinds of methods for continual learning, most previous works focus on image classification tasks or reinforcement learning tasks. The network output is either the predicted image class or the agent action, which is of limited dimension or even to be a scaler. Little research has been done on the semantic segmentation task, where a pixel-wise labelling should be considered. Michieli and Zanuttigh [4] first introduced the incremental learning problem for semantic segmentation. They use the knowledge distillation to preserve the old knowledge while updating the new one. Their task is to incrementally learn new class, belonging to multi-center multi-task learning. While we focus on binary segmentation task, which is multi-center single-task learning, where "multi-center" means data used in different tasks are different, and "single-task" means the consecutive tasks are doing the same thing.

Chapter 3

Problem and Methodology

In this chapter, we will formally formulate our task and introduce all methods we implemented in the following order: fine-tuning, output distillation, feature distillation, EWC, Progress and Compress.

3.1 Problem Formulation

As we have mentioned in the last chapter, our task belongs to the multi-center single-task learning. To be specific, we consider the setting in which (1) the system learns one task at a time, (2) the consecutive tasks are essentially the same, i.e., binary foreground and background segmentation, (3) the training data for consecutive tasks belongs to different types of scenes, (4) the system has no access to training data of previous tasks and only the validation data is stored.

The continual learning for binary foreground and background segmentation task, can be defined as the ability of our model to learn the segmentation on the new data without large performance decreasing on the old data. More concretely, we first train the model on the dataset of first scene, then we continually train the model on the second scene. Our goals are to improve the segmentation accuracy on the second scene and to prevent the accuracy decreasing a lot on the first scene. The good performance on the second scene means the model can absorb new knowledge, while the one on the first scene shows it can avoid catastrophic forgetting. The core problem is how to keep a good balance of them.

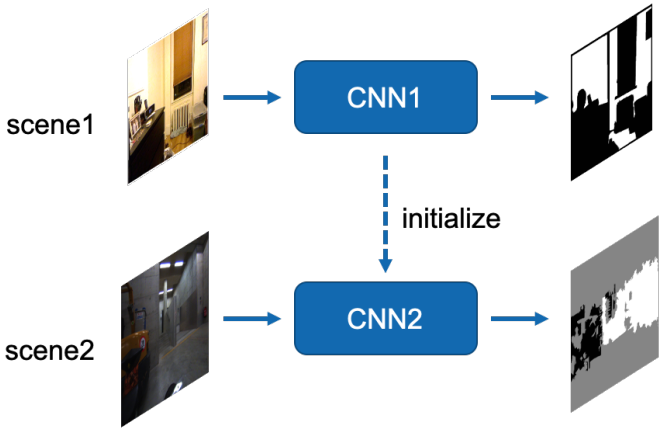

Figure 3.1: Pipeline of Fine-tuning

3.2 Fine-tuning

A natural idea to do continual learning is fine-tuning, the pipeline of which is shown in Figure 3.1. First, we train the model (CNN1) on training data of scene 1, then we initialize a new model (CNN2) with the pre-trained model (CNN1), and train on training data of scene 2. The target loss function $L_B(\theta)$ is the cross entropy loss generally used for the segmentation task. Finally, we compute the segmentation accuracy on validation data of scene 1 and scene 2.

3.3 Output Distillation

Figure 3.2 displays the pipeline of output distillation. First, we train the model (CNN1) on training data of scene 1, same as fine-tuning. When given the training data of scene 2, we create two models (CNN1 and CNN2) at the same time and both of them are initialized with the model pre-trained on scene 1. We keep one model (CNN1) fixed, using it to generate pseudo labels, and update another model (CNN2) using true labels as well as pseudo labels. So the loss for the output distillation method is described as,

$$L(\theta) = (1 - \lambda)L_B(\theta) + \lambda L_o(\theta) \tag{3.1}$$

where $L_B(\theta)$ is the cross entropy loss between the output prediction and its corresponding ground truth, $L_o(\theta)$ is the cross entropy loss between the output prediction and pseudo labels, λ is the hyperparameter used to weigh two loss terms.

By adding the extra regularization term with pseudo labels, the old knowledge is distilled to the new model from the output layer.

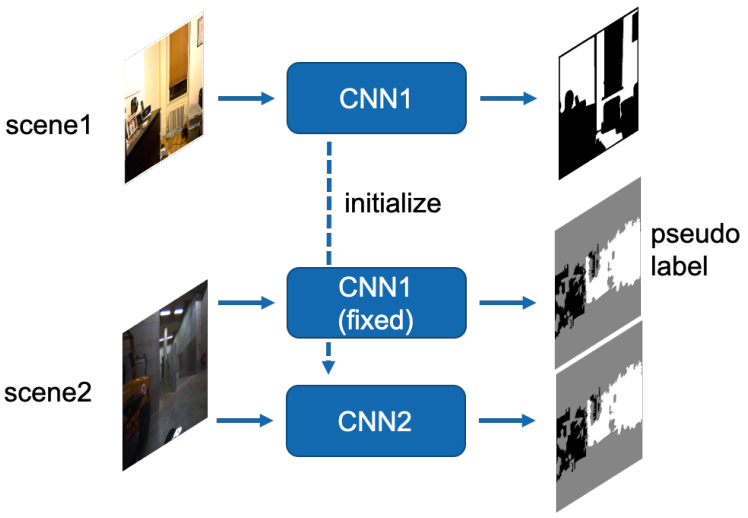

Figure 3.2: Pipeline of output distillation

3.4 Feature Distillation

The pipeline of feature distillation is quite similar to that of output distillation. Instead of distilling the old knowledge from the output layer, now we apply the knowledge distillation on the intermediate feature space, to be specific, the encoder output. Correspondingly, the cross entropy loss should be replaced with the $L2$ loss

since the intermediate feature don not have the meaning of probability as output space. So the loss for the feature distillation would become,

$$L(\theta) = (1 - \lambda)L_B(\theta) + \lambda L_f(\theta) \tag{3.2}$$

where $L_B(\theta)$ is the cross entropy loss between the model output and its corresponding ground truth, $L_f(\theta)$ is the $L2$ loss between the new encoder output and pseudo features computed by the fixed old encoder, λ is still the weighting parameter.

3.5 EWC

The above two methods separately control the output and feature activation. Another way is to directly add constraints on weights of the network. That's EWC method. This method is normally used in the classification task, but we think the idea fits into our segmentation task as well. So we extend this method to our task. The assumption of this method is that the neural network is over-parameterization, which has been proven in [24, 25] to be true in most situations. So the optimal weights of network for task A is close to that of task B, as long as these two tasks are similar. In our case, both tasks are binary segmentation tasks. When the difference between two types of scenes is small, the assumption holds.

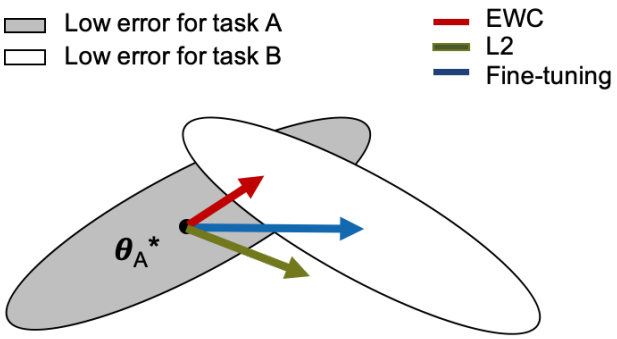

Figure 3.3: Illustration of weight updating direction in the parameter space for different training methods

Figure 3.3 illustrates the direction of weight updating in the parameter space for three different training methods: fine-tuning (without regularization), L2 (adding equal constraints on weights), EWC (adding constraints on weights selectively). The grey area is the weight space with low error for task A and the white area is for task B. After learning task A, the optimal weights θ_A^* locate at the center of the grey area. The blue arrow refers to fine-tuning, we can see the although the performance on task B increases, the performance on task A decreases a lot. The green arrow refers to adding equal constraints on every weight, we can see it limits the learning on task B. The red arrow performs well on both tasks. That's the idea of EWC. We want weights that are important for task A to change as little as possible and only update in other directions. The importance is measured by fisher information, which is defined as,

$$F_i = \mathrm{E}\left[\left(\frac{\partial}{\partial\theta_i}\log f(Y;\theta)\right)^2 \mid \theta\right] \tag{3.3}$$

where $f(Y;\theta)$ is the probability density function (pdf) of model output Y conditioned on the value of model parameters θ. If one parameter makes great contribu-

tion to the final prediction, then the partial derivative with respect to it should be large and this parameter is considered important.

However, the segmentation map output by the model only reflect the pdf of each pixel. In order to compute the fisher information, we need an extra assumption that the distribution of all pixels in the model output are independent, so that the pdf of the whole output is the product of that of all pixels. Adding "log" notation on both sides, it turns into the following form,

$$\log f(Y;\theta) = \sum_{y\in \text{ all pixels}} \log f(y;\theta) \tag{3.4}$$

Based on the above discussion, the loss for EWC is,

$$L(\theta) = (1-\lambda)L_B(\theta) + \lambda \sum_i F_i(\theta_i - \theta^*_{A,i})^2 \tag{3.5}$$

where $\theta^*_{A,i}$ is the parameter after training on the first scene. Every time we want to learn a new task, we need to first store the old parameters and compute the fisher information for each parameter. Model parameters with higher importance are restricted more.

3.6 Progress and Compress

Different from all above regularization methods that estimate a single set of weights for all tasks, progress and compress method belongs to the dynamic architecture approaches, that is, the model learns a new set of weights for each new task.

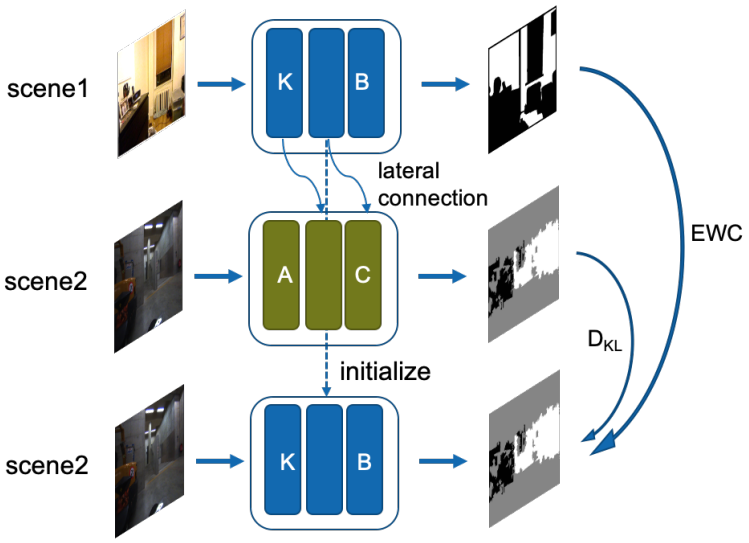

Figure 3.4: Pipeline of progress and compress

Figure 3.4 demonstrates the Progress and Compress learning process, where the blue blocks are knowledge base (KB) used to store old knowledge whilst the green blocks are active columns (AC) used to learn new tasks. We first train the knowledge base on training data of scene 1. Then in the "progress" stage, we can train the active column with data of scene 2. There exist some layerwise lateral connections between the knowledge base and the active column, which are used to reuse the old knowledge when learning the new task. Finally, in the "compress"

stage, again with training data of scene 2, we update the knowledge base with the guide of active columns.

In order to understand how "lateral connection" works, we can refer to Eq 3.6 and Figure 3.5,

$$h_i^{AC} = \sigma(W_i h_{i-1}^{AC} + \alpha_i \odot U_i \sigma(V_i h_{i-1}^{KB})) \tag{3.6}$$

where h_{i-1}^{KB} and h_{i-1}^{AC} are activation of previous layer in KB and AC, W_i and V_i

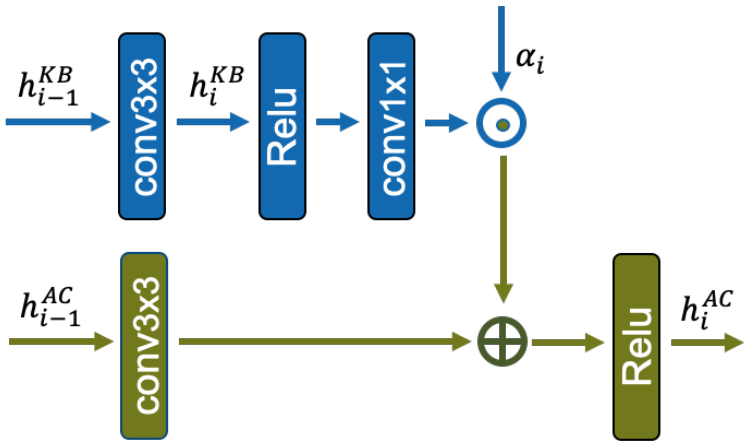

Figure 3.5: scheme of lateral connection, $\odot$ denotes element-wise multiplication, $\oplus$ denotes element-wise addition

are 3x3 convolutions, U_i is 1x1 convolution, α_i is a trainable vector of size equal to the number of filters in layer i and sampled from a uniform distribution, $\odot$ denotes element-wise multiplication, σ is the Relu activation.

Therefore, in the "progress" stage, the computation of activation of current layer in AC makes use of the activation of previous layer in both AC and KB. With the support of KB, in theory, the training of AC can be speed up.

In the "compress" stage, on one hand, new knowledge learned by the active column is consolidated into the knowledge base. This is done by minimizing the KL divergence between the prediction of AC and KB. On the other hand, we need to preserve the old knowledge, so we make use of the same trick we used before, EWC, to regulate the updating procedure of model parameters in KB. Combining these two points, the loss for Progress and Compress can be formulated as,

$$L(\theta) = (1-\lambda)\mathbb{E}\left[\mathrm{KL}\left(\pi^{AC}(\cdot \mid x) \| \pi^{\mathrm{KB}}(\cdot \mid x)\right)\right] + \lambda \sum_i F_i(\theta_i^{KB} - \theta_i^{KB*})^2 \tag{3.7}$$

where $\pi^{AC}(\cdot \mid x)$ and $\pi^{KB}(\cdot \mid x)$ are predicted pixelwise categorical probabilities of the active column and knowledge base, θ_i^{KB*} is the parameter of KB after training on scene 1.

To summarize, in the "progress" stage, we fix the parameters of KB and use KB to support the training of AC. In the "compress" step, we freeze the parameters of AC and update KB with AC. Finally, the parameters of KB store the knowledge learned so far.

Chapter 4

Experiments and Results

4.1 Experiment Setup

4.1.1 Dataset

We evaluate all methodologies on NYU Depth V2 Dataset [14] and CLA Dataset for binary background and foreground segmentation.

NYU Depth V2 Dataset is comprised of video sequences from a variety of indoor scenes as recorded by both the RGB and Depth cameras from the Microsoft Kinect. There are totally 1449 densely labeled pairs of aligned RGB and depth images, 849 object classes and 27 scene types. Since we are doing the binary foreground and background segmentation task, we combine the "ceiling", "wall" and "floor" among 849 classes to be the background and others to be the foreground. Moreover, among 27 scene types, we extracted two scenes with the most number of labeled images, i.e., "kitchen" and "bedroom", to create two sets of sub-datasets. "kitchen" dataset contains 225 labeled images and "bedroom" has 383 labeled images.

CLA Dataset is composed of 2708 auto-labeled images from the CLA building. It consists of three labels (0,1,2) where all classes that belong to the background, e.g. "floor", "wall" and "roof" are assigned the '2' label. Foreground has been assigned the '0' label and the unknowns the '1' label.

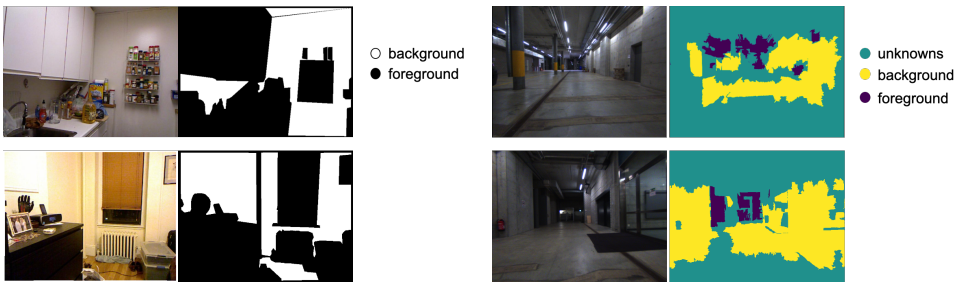

Figure 4.1: Visualization of NYU Depth V2 and CLA dataset

Table 4.1: Experiment Setting.

Experiment	Scene 1	Scene 2
Experiment 1	NYU Depth V2 (kitchen)	NYU Depth V2 (bedroom)
Experiment 2	NYU Depth V2 (whole)	CLA Dataset

4.1.2 Experiment Setting

In order to compare different methodologies, we set up two experiments as listed in Table 4.1. For experiment 1: scene 1 is the "kitchen" dataset extracted from the NYU Depth V2 dataset, and scene 2 is the "bedroom" dataset same from the NYU. For experiment 2: scene 1 is the whole NYU dataset, scene 2 is the CLA dataset. For each dataset, we use 80% for training and 20% for validation.

Figure 4.1 displays the visualization result of two datasets we used. Images on the left belong to the NYU Depth V2 dataset. Among those, the image pair at the top is drawn from the "kitchen" dataset while the one at the bottom comes from the "bedroom" dataset. We can see that, on one hand, two type of scenes are both indoor scenes and the labeling is consistent so that continual learning from "kitchen" to "bedroom" is relevant easy. On the other hand, there still exist many differences between two scenes. For example, in the top image pair, the white cupboard, which is common in the kitchen, is labeled as the foreground. If the model is only trained on "bedroom" scene, it will easily classify it as the background from the appearance. Based on these two points, experiment 1 is a easy case to study the performance of all continual learning methods.

From the right images corresponding to the CLA dataset, we see that,

1. The scene in CLA is very different from that in NYU, whether from the illumination or the category of indoor items, which makes the continual learning harder.

2. The CLA data itself is very noisy. Since the ground truth label is auto-generated by neural networks with the information of combined sensors, the segmentation of background and foreground is very coarse and sometimes may be mislabeled.

3. The CLA data contains many unknowns, which can be seen from the large green area. So we need to mask out unknowns when computing the final loss.

4. The class distributions in CLA are highly imbalanced, which can be seen from the inconsistent ratio of the purple and yellow area.

All these factors make the gap between the NYU and CLA larger, thus the experiment 2 more challenging.

4.1.3 Architecture Details

To do a fair comparison, all methods use the same model architecture except for the "lateral connection" part in Progress and Compress. To be specific, all blocks labeled with "CNN", "KB" or "AC" in Figure 3.1, Figure 3.2, Figure 3.4 are same.

For the general network architecture, we choose VGG16 [26] as the encoder. It contains 5 sub blocks, each made of 3x3 convolution layer and max-pooling layer. As for the decoder, we use the U-net structure [21] with five upsampling blocks, each has a skip connection with the corresponding sub block in the encoder. Every upsampling block contains two 3x3 convolution layers, an upsampling layer and a concatenation layer that connects the encoder feature of the same size. And each convolution layer is followed by a batch normalization layer and Relu activation. Finally, for the segmentation head, we use a 3x3 convolution layer with 2 output channels and the Softmax activation layer to output pixelwise categorical probabilities. In all, the number of output channels for all sub blocks are 64, 128, 256, 512, 512, 256, 128, 64, 32, 16.

As for the special "lateral connection" part in Progress and Compress method, taking it as a sub module as depicted in Figure 3.5, we insert this module after

each pooling layer in the encoder and each Relu activation layer in the decoder. Therefore, there are 15 lateral connection modules in total.

4.1.4 Training Process

Because the training procedure for two experiments are indeed the same, we will take experiment 1 for illustration in the following part.

The first step of training are same for all methods, i.e., to train the network on the "kitchen" dataset. The encoder is initialized with the weights pretrained on ImageNet [27] and the whole network is trained using Adam [28] with a learning rate of 0.0001. We totally train 40 epochs and the model almost converges after 20 epochs.

In the second step, we initialize a new network with the weights pretrained on "kitchen" dataset. Before continual learning, EWC needs an extra step to store both the pretrained weights and the computed fisher information for each model weight in the memory, which are then to be used when calculating the regularization for weights. For feature distillation and output distillation method, an auxiliary model is needed to generate the pseudo feature (output of the encoder) and pseudo label (output of the network). It has the same architecture as our new network and is also initialized with the pretrained weights. It is fixed, however, in order to keep the old knowledge. For Progress and Compress, we also build two models of the same architecture, one called KB, one called AC. KB is initialized with pretrained weights while AC and the convolution layer in the lateral connection are randomly initialized. Then during the training, only the weights in AC and lateral connection are updated with the ones in KB fixed. We use Adam optimizer to train the new network on "bedroom" dataset with a learning rate of 0.00001. We totally train 40 epochs for each method.

The training process is complete after above two steps for all methods except for Progress and Compress, since it only finishes the "Progress" step. To fulfill the training, we initialize the KB with weights pretrained on "kitchen" dataset (step 1) and the AC with weights pretrained on "bedroom" dataset (step 2). On the opposite of step 2, this time, we fix AC and update KB using Adam with a learning rate of 0.00001.

For each method, the hyperparameter λ is selected from {0.0001, 0.001, 0.01, 0.1}. As the evaluation metric, we use the accuracy on scene 1 and scene 2. The former is used to measure the ability to preserve the old knowledge, and the latter reflects the capability to absorb the new knowledge. In some experiments, we also use the mean intersection of union (mIoU).

4.2 Results and Analysis

In this section, we aim to answer the following questions:

- How is the capability of continual learning methods to prevent catastrophic forgetting?
- How is the capability of continual learning methods to learn new tasks?
- Why do methods work, not work, or work similarly?
- Why methods work in one situation while not in another?

4.2.1 Results of Experiment 1

In experiment 1, we first train the model on the "kitchen" dataset, then we perform continual learning on the "bedroom" dataset. Therefore, the accuracy on "kitchen" dataset reflects the capability of learning in the new scene, and the accuracy on "bedroom" dataset shows the ability to segment the old scene. The higher, the better.

We list 6 methods in total for comparison: fine-tuning, feature distillation, output distillation, EWC, fine-tuning + compress, and progress + compress. Five of them have been detailed explained in Chapter 3. Fine-tuning + compress can be seen as the simplification of Progress and Compress. It first train a model (knowledge base) on "kitchen" dataset, then directly train another model (active column) using fine-tuning on "bedroom" dataset, and finally back to use the fine-tuned model (active column) to update the knowledge base. The only difference is the second step, where it does not reuse the weights stored in the knowledge base to train the active column.

Table 4.2: Evaluation of Experiment 1, Architecture:VGG16+Unet

Method	Accuracy on scene 2 (bedroom)	Accuracy on scene 1 (kitchen)
Fine-tuning	**92.14**	88.15
Feature distillation	92.13	88.48
Output distillation	92.08	88.49
EWC	89.9	**89.19**
Fine-tuning + Compress	89.85	88.81
Progress + Compress	89.87	88.71

A summary of the evaluation of all implemented methods is reported in Table 4.2, the data is the average of three trials and is obtained after training for same epochs. From the table we can draw the following conclusions:

1. From the rightmost column, we see that the accuracy of last five methods are all higher than fine-tuning, which shows the effectiveness of knowledge preservation to some degree. Among all methods, EWC performs the best, achieving 89.19% segmentation accuracy.

2. From the middle column, we see that the fine-tuning achieves higher accuracy on scene 2 than other methods. It shows that continual learning methods will impair new task learning.

3. Combining two columns, we can find a general tendency that when the accuracy on scene 1 increases, that on scene 2 would decrease. There exists a trade off between the ability to keep knowledge of the old task and the ability to learn knowledge from the new task.

4. Comparing two columns, we can find the accuracy on scene 2 is always higher than that on scene 1, which proves the necessity to study continual learning. There is always a mismatch between the type of scene used to train the segmentation model and the practical scene it encounters. We also see that for each method, the difference is not very large. It reflects that the "kitchen" scene and "bedroom" scene are close from the appearance and this is consistent with our common sense.

5. It is noticeable that completely different approaches achieve similar results. There could be two possible explanations: (1) Taking the fine-tuning as the baseline, its accuracy on scene 2 and scene 1 provides the upper limit (92.14%) and lower limit (88.15%) respectively. They are very close to each other, leaving little space for the performance improvement. (2) Remind that the loss function for each method are in the following form:

   Output distillation: $L(\theta) = (1-\lambda)L_B(\theta) + \lambda L_o(\theta)$

   Feature distillation: $L(\theta) = (1-\lambda)L_B(\theta) + \lambda L_f(\theta)$

   EWC: $L(\theta) = (1-\lambda)L_B(\theta) + \lambda \sum F_i \left(\theta_i - \theta^*_{A,i}\right)^2$

   Progress and Compress: $L(\theta) = (1-\lambda)\mathbb{E}\left[\mathrm{KL}\left(\pi^{AC}(\cdot \mid x) \| \pi^{\mathrm{KB}}(\cdot \mid x)\right)\right] + \lambda \sum_i F_i \left(\theta_i^{KB} - \theta_i^{KB*}\right)^2$

   The first three methods differ in their regularization terms, which can be thought of different methodologies to preserve old knowledge. For output distillation, it uses the model pretrained on "kitchen" to generate the pseudo segmentation map. This map reflects what the model expect to be classified, for example, the white cupboard is supposed to be classified as foreground. So as the model continues training on "bedroom", whenever the training data contains similar white cupboards, the model will have a stronger tendency to take it as foreground. The regularization term requires the encoder-decoder architecture to update parameters cleverly. For feature distillation, it minimizes the L2 loss in the intermediate space using the pretrained model. The idea behind it is that high-level features extracted by the encoder are object-level, which should be shared across different datasets. Therefore, the encoder output should not change too much. The regularization term will force parameters of the encoder to update as little as possible. For EWC, as explained in section 3.5, the regularization will force the weights that are important for correct segmentation on "kitchen" scene to change less. For Progress and Compress, it has the same regularization term as EWC. As for the target loss, $L_B(\theta)$ is the cross entropy between the ground truth labels and predictions. In our case, the ground truth labels are provided as "logits" (0 for foreground and 1 for background). $\mathrm{KL}\left(\pi^{AC}(\cdot \mid x) \| \pi^{\mathrm{KB}}(\cdot \mid x)\right)$ is the KL divergence between the predictions of the active column and predictions of the knowledge base. First of all, cross entropy loss and KL divergence are essentially the same. Second, if the active column is well-trained in the "progress" step, the predictions of the active column can be considered as "ground truth" labels. It is in the form of probability distribution rather than "logits", thus more precise for the optimization. Therefore, two target losses have the same idea. In summary, different methods, though using apparently variant loss functions, all either directly or indirectly control the updating directions of model parameters and optimize the performance on both the old and new scene.

6. The performance of Progress and Compress is not as good as expected. Before looking for possible explanations, we need to understand what the "lateral connection" module for, in other words, the advantage of reusing the old knowledge. One thing we already familiar with is that it can avoid catastrophic forgetting, in another aspect, it can speed up the positive transfer. However, in our experiment, we did not see such advantages, which can be seen from Figure 4.2. Comparing two plots, neither the training speed nor the final performance gets improved. The problem could derive from the initialization of "lateral connection" modules. The introduction of 15 "lateral connection" modules brings a large amount of trainable parameters into the original network. Therefore, even loaded with pretrained weights, the perfor-

mance of the new network get decreased a lot when it continues training on the new scene, which can be seen from the low accuracy at the starting point in the right plot. As for the lower final performance, the reason could be that part of the network is fixed, which limits the learning on the new scene.

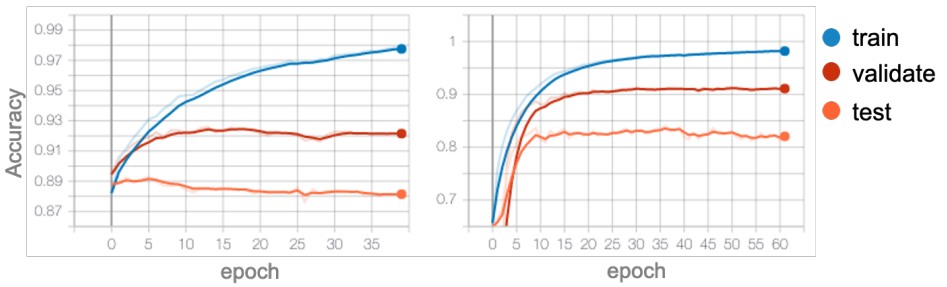

Figure 4.2: Learning curve on scene 2 ("bedroom") , left: fine-tuning, right: progress (lateral connection)

Moreover, for EWC, the best method in experiment 1, we study the influence of hyperparameter λ on its performance. Figure 4.3 displays the learning curves on scene 2 ("bedroom") with different λ setting. From left to right, λ is gradually increasing. Based on the loss function displayed in Eq 3.5, increasing λ means strengthening the weight constraints. Comparing the trend of training, validation and testing curves, we may find when λ becomes larger, the testing accuracy decreases less whilst both the training and validation accuracy increase more slowly. In the end, the validation accuracy for all λ are almost equal, and the method with the largest λ achieves the highest testing accuracy. The phenomenon reflects that, on one side, EWC could resilience against catastrophic forgetting, on the other side, fisher information over-constrain the model parameters, impairing the new task learning.

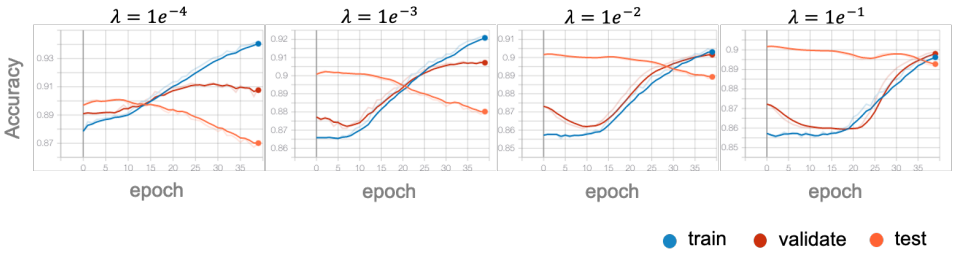

Figure 4.3: Learning curves of EWC method on scene 2 ("bedroom") with different λ setting, from left to right, λ is increasing

4.2.2 Results of Experiment 2

In experiment 2, we first train the model on NYU Depth V2 dataset, then we do continual learning on CLA dataset.

Similarly, we summarize the evaluation results for all methods in Table 4.3, from which we can find more discoveries beyond that in experiment 1:

1. From the rightmost column, we see that Output distillation replaces EWC, becoming the best continual learning method. One possible explanation is

that the assumption of EWC does not hold anymore. Remember that it states that the optimal model parameters for two tasks are close to each other if two tasks are similar. However, in experiment 2, this is not the case. We can see this from two aspects: (1) From the visualization result as shown in Figure 4.1, NYU and CLA dataset are quite different. (2) In Figure 4.4, the left and right plot separately display the pretraining process on scene 1 in experiment 1 and experiment 2. From the left image, we can see that as the training goes, both the accuracy on scene 1 ("kitchen") and scene 2 ("bedroom") increase, despite a small gap between the orange and red lines due to the difference of scene type. From the right image, however, we find while the accuracy on scene 1 (NYU) is increasing, the accuracy on scene 2 (CLA) fluctuates at a level of 50% (random guessing). Therefore, we can conclude the training of the first task does not bring any useful information to the second task. The gap between two tasks are too large.

2. Comparing two columns, we find for fine-tuning, the accuracy on scene 2 and scene 1 are 98.17% and 77.15%. According to the argument in experiment 1, there exists a large space for improvement. However, the accuracy on scene 1 for all methods are still similar. One way to understand this is that all regularization-based methods estimate a single set of parameters for different tasks, which only works when tasks are similar. This brings us back to the discussion in the last point.

Table 4.3: Evaluation of Experiment 2, Architecture:VGG16+Unet

Method	Accuracy on scene 2 (CLA)	Accuracy on scene 1 (NYU)
Fine-tuning	**98.17**	77.15
Feature distillation	98.13	76.35
Output distillation	97.87	**77.99**
EWC	95.26	77.3
Fine-tuning + Compres	91.39	71.38
Progress + Compress	88.2	76.2

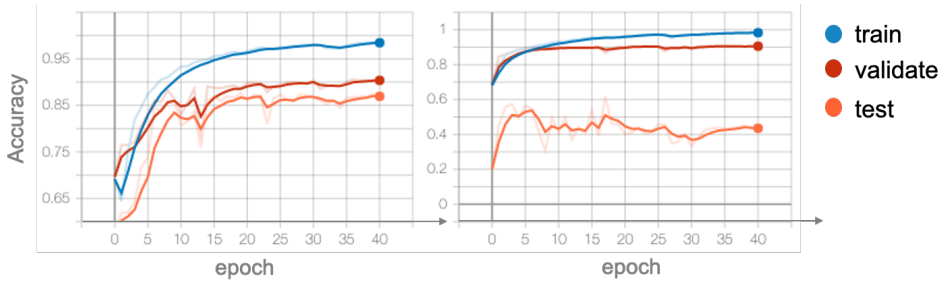

Figure 4.4: Learning curve on scene 1, left: experiment 1, right: experiment 2

4.2.3 Qualitative Comparisons

In Figure 4.5, 4.6, 4.7 and 4.8, we display some sample results of two scenes in two experiments. In each of them, the top row refers to the real image, the middle row

is the ground truth segmentation and the bottom row is the segmentation generated by our models.

Comparing Figure 4.5 with Figure 4.6, we see after continual learning, the model is able to generate reasonable segmentation on both "kitchen" and "bedroom" scene, though it may sometimes lose some information about the "kitchen" scene. For example, in the third column in Figure 4.5, part of the white refrigerator is predicted as the background.

Figure 4.5: Visualization of segmentation results of "kitchen" (scene 1 in experiment 1), top row: real image, middle row: ground truth segmentation, bottom row: segmentation generated by EWC. White: background, Black: foreground.

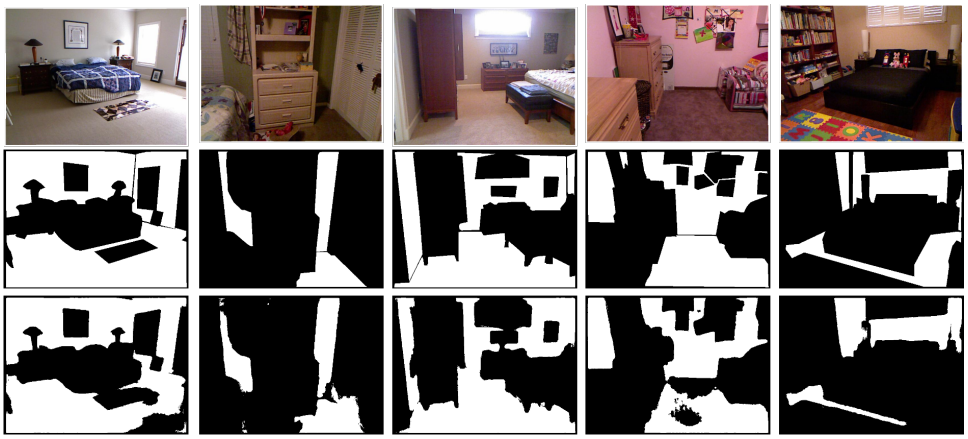

Figure 4.6: Visualization of segmentation results of "bedroom" (scene 2 in experiment 1), top row: real image, middle row: ground truth segmentation, bottom row: segmentation generated by EWC. White: background, Black: foreground.

Comparing Figure 4.7 with Figure 4.8, we will find the model has the tendency to predict the uncertainty as background. For illustration, see the second and third columns in Figure 4.7 and all five columns in Figure 4.8, the white color covers more area in the segmentation generated by our model than that in the ground truth. This may due to the imbalanced training data of CLA, where the number of pixels labeled as background is much larger than that labeled as foreground.

Comparing the visualization of experiment 1 with that of experiment 2, we observe that: (1) The segmentation generated in experiment 2 is quite coarse, while

that in experiment 1 has relatively clear and smooth object contour. One possible reason is that the ground truth provided by CLA dataset is itself noisy and not smooth. For example, the main foreground objects in the first three columns in Figure 4.8 are truck, human and car, however, we could not clearly tell from the given ground truth. (2) There exist some differences between the labeling manner in NYU and CLA. For example, in the first and last columns of Figure 4.6, the carpet is labeled as the background. However, in the last column of Figure 4.8, the carpet is considered as the foreground. It could because the labels in CLA dataset is generated using information collected by sensors while flat objects like carpets are not detected.

Figure 4.7: Visualization of segmentation results of NYU (scene 1 in experiment 2), top row: real image, middle row: ground truth segmentation, bottom row: segmentation generated by output distillation. White: background, Black: foreground.

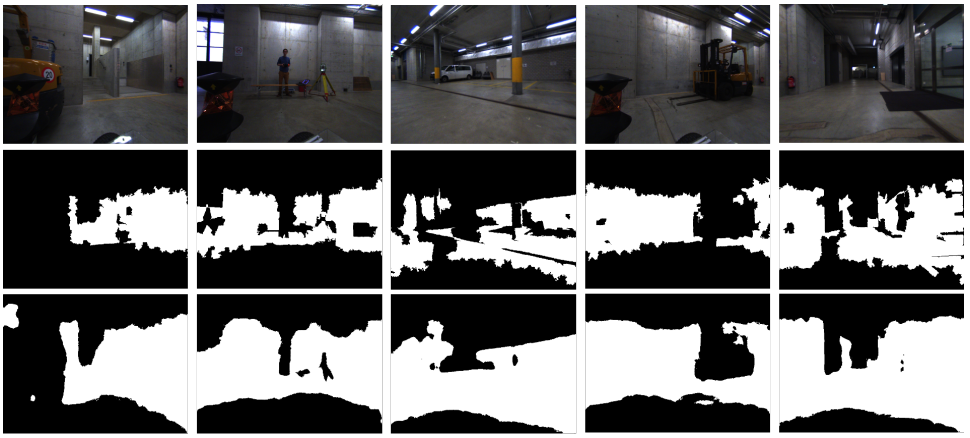

Figure 4.8: Visualization of segmentation results of CLA (scene 2 in experiment 2), top row: real image, middle row: ground truth segmentation, bottom row: segmentation generated by output distillation. White: background, Black: foreground.

4.2.4 Influence of Backbone

In the above discussion, all methods are based on the model architecture which uses VGG16 as the backbone and U-net structure as the decoder. In order to show

it works in the general case, we replace the VGG16 with EfficientNet-B1 [29], and perform the whole training pipeline as before. EfficientNet-B1 is 7.6x smaller and 5.7x faster on CPU inference than ResNet-152 [30], with similar ImageNet accuracy.

In Table 4.4 and 4.5, we summarize the evaluation results of experiment 1 and experiment 2. Since Progress and Compress method need requires the customized model and we did not spare effort on it, there are only 4 methods listed for comparison. We can see with the new architecture, in both experiment settings, output distillation perform the best in terms of accuracy and mIoU. However, a strange phenomenon is that EWC perform badly, both the metric on scene 1 and scene 2 are low. Especially in experiment 2, the accuracy on NYU dataset is 45.93% (random guessing).

Table 4.4: Evaluation of Experiment 1, Architecture:efficientnetb1+Unet

Method	scene2 (kitchen)		scene1 (bedroom)	
	Acc	mloU	Acc	mloU
Fine tuning	92.45	85.24	89.32	74.77
Feature distillation	92.57	85.45	89.82	75.75
Output distillation	92.66	85.65	**89.84**	**75.89**
EWC	85.95	73.44	89.45	75.18

Table 4.5: Evaluation of Experiment 2, Architecture:efficientnetb1+Unet

Method	scene2 (CLA)		scene1 (NYU)	
	Acc	mloU	Acc	mloU
Fine tuning	97.91	93.76	74.31	56.89
Feature distillation	97.93	93.81	71.01	52.64
Output distillation	97.83	93.53	**75.95**	**58.96**
EWC	90.08	79.69	45.93	29.65

Chapter 5

Conclusion

In this work, we study five continual learning approaches for binary foreground and background segmentation task: fine-tuning, output distillation, feature distillation, EWC, Progress and Compress. For each approach, we first train the model on scene 1, then we initialize a new model with the pretrained model, and train on scene 2 using the corresponding trick for continual learning. In order to compare different methods, we set up two experiment for evaluation. One is from "kitchen" scene to "bedroom" scene in NYU Depth V2 Dataset. Another is from NYU Depth V2 Dataset to CLA dataset. Based on both the visualization of NYU and CLA datasets and accuracy curves in Figure 4.4, we can easily see that the second experiment is much more difficult than the first one. In experiment 1, all continual learning methods with special techniques perform better than naive fine-tuning on the old scene, though at the cost of impaired learning on new scene. Among those, EWC method performs the best. It achieves the accuracy of 89.19% on old scene, 1% higher than fine-tuning, which is not easy to obtain considering that the accuracy on the new scene is 92.14%. In experiment 2, only some of continual learning techniques help to prevent catastrophic forgetting. Methods like feature distillation is even worse than fine-tuning. It indicates a potential problem of regularization-based methods, that is, they use a single set of parameters for different tasks, in our case, to segment on two apparently greatly different datasets (NYU and CLA). However, the representation capability for a model with fixed set of parameters is limited, which may explain the failure in the second experiment.

For future work, as we have discovered that the use of a single set of parameters for different tasks may lead to poor performance, we can try some other continual learning methods beyond the regularization-based methods. In the related work, we mentioned the other two types of continual learning methods: dynamic architecture method and memory replay method, which we have not fully investigated in our experiment. Memory replay method saves a sample of old training data to reuse them later. However, based on our analysis, it still uses a single model for all tasks, hence, cannot get rid of the problem existed in the regularization method. Of course, it's worth having a trial to see if such type of method works. Dynamic architecture method, which selectively trains the network and stores a new set of parameters for each new task, could be a choice. More recently, we see the success of combination of meta-learning and continual learning [31, 32]. He et al. [32] state that their proposed framework can handle a more challenging scenario where different tasks may even conflict with each other, which fits into our experiment where the labelling manner of NYU and CLA are quite different.

Bibliography

[1] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[2] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[3] H. Shin, J. K. Lee, J. Kim, and J. Kim, "Continual learning with deep generative replay," *arXiv preprint arXiv:1705.08690*, 2017.

[4] U. Michieli and P. Zanuttigh, "Incremental learning techniques for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[5] H.-E. Kim, S. Kim, and J. Lee, "Keep and learn: Continual learning by constraining the latent space for knowledge preservation in neural networks," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2018, pp. 520–528.

[6] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[7] N. Y. Masse, G. D. Grant, and D. J. Freedman, "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization," *Proceedings of the National Academy of Sciences*, vol. 115, no. 44, pp. E10 467–E10 475, 2018.

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

[9] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *arXiv preprint arXiv:1706.08840*, 2017.

[10] B. Lüders, M. Schläger, and S. Risi, "Continual learning through evolvable neural turing machines," in *Nips 2016 workshop on continual learning and deep networks (cldl 2016)*, 2016.

[11] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, "Variational continual learning," *arXiv preprint arXiv:1710.10628*, 2017.

[12] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[14] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[15] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 670–677.

[16] J. Shotton, M. Johnson, and R. Cipolla, "Semantic texton forests for image categorization and segmentation," in *2008 IEEE conference on computer vision and pattern recognition*. IEEE, 2008, pp. 1–8.

[17] N. Dhanachandra, K. Manglem, and Y. J. Chanu, "Image segmentation using k-means clustering algorithm and subtractive clustering algorithm," *Procedia Computer Science*, vol. 54, pp. 764–771, 2015.

[18] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[19] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[20] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[21] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[22] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[23] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and autonomous systems*, vol. 15, no. 1-2, pp. 25–46, 1995.

[24] H. J. Sussmann, "Uniqueness of the weights for minimal feedforward nets with a given input-output map," *Neural networks*, vol. 5, no. 4, pp. 589–593, 1992.

[25] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*. Elsevier, 1992, pp. 65–93.

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

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6105–6114.

[30] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[31] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International Conference on Machine Learning*. PMLR, 2017, pp. 1126–1135.

[32] X. He, J. Sygnowski, A. Galashov, A. A. Rusu, Y. W. Teh, and R. Pascanu, "Task agnostic continual learning via meta learning," *arXiv preprint arXiv:1906.05201*, 2019.