A Step to Remember: Study of Single Model VS Dynamic Model

Liguang Zhou
School of Science and Engineering (SSE)
香港中文大學(深圳)
The Chinese University of Hong Kong, Shenzhen

Abstract

- When sequentially training a series of tasks, the deep neural networks (DNNs) will suffer from the catastrophic forgetting problem. There are many life long learning methods proposed to tackle with this problem recently. Elastic Weights Consolidation (EWC) is a method that prevents the DNN from forgetting the previous task while learning the current task by measuring the importance of parameters in DNNs with the Fisher Information Matrix that senses the second derivatives of the loss function. However, EWC also limits the learning ability of the network with only a single network. If there are new tasks required to be learned, the dynamic graph will be a more powerful option because of its capacity of preserving more information about the old tasks. LwR is one of the most typically dynamic models for learning multiple tasks sequentially. Moreover, the sampling of old tasks for new task training is also superior in preserving the memory of previous tasks.

Introduction

- In robotics area, the incremental learning of various objects is an essential problem for perception of robots.
- When there are many tasks to be trained in sequence, the DNNs will be suffering from catastrophic forgetting problem. One way to solve this catastrophic problem is called multi-task training, in which the various task will be trained concurrently in the training process. This solution is also the upper bound of the multiple-task training problem. However, in reality, if we need to train DNNs every time when new task comes, it will be low-efficiency and involves a lot of computing resources waste.
- Therefore, the alternative methods of solving this life long learning problem have been proposed, such as EWC, LwR, generative methods and so on.
- EWC is a method that utilize the Fisher Information Matrix, which is also related to the second derivative of the gradient, to preserve some important parameters of the previous tasks during the training.
- LwR is a dynamic graph based method used for preserve the memory of the previous tasks by expend the network and introducing the distillation loss.

Elastic Weights Consolidation (EWC)

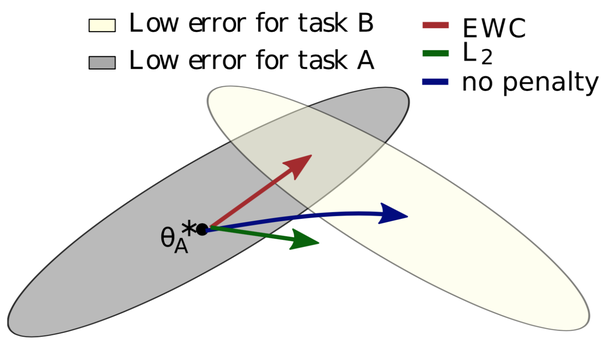

Figure: The learning sequence is from task A to task B

- We assume some parameters that are less useful and others are more valuable in DNNs. In the sequentially training, each parameter is treated equally. In EWC, we intend to ultilize the diagonal components in Fisher Information Matrix to identify the importance of parameters to task A and apply the corresponding weights to them.

Close Look at EWC

- Baye's rule

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p|(\theta) - \log p(\mathcal{D}) \tag{1}$$

$$\log p(\theta|\mathcal{D}) = \log p\left((\mathcal{D}_B|\theta) + \log p\left(\theta|\mathcal{D}_A\right) - \log p\left(\mathcal{D}_B\right)\right. \tag{2}$$

- To avoid forgetting the learned knowledge in task A, one simple trick is to minimize the distances between $\theta, \theta^*_{\mathcal{A}}$, which also can be regarded as L_2.

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_{\mathcal{B}}(\theta) + \frac{1}{2}\alpha\left(\theta - \theta^*_{\mathcal{A}}\right)^2 \tag{3}$$

- In L_2 case, each parameters is treated equally, which is not a wise solution because the sensitivity of each parameters varies a lot. The assumption is the importance of each parameters is different and varies a lot. Hence, the diagonal components in Fisher Information Matrix is used to measure the weights of importance of each parameter.

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \mathcal{L}_B(\theta) + \frac{1}{2}\alpha F_{\theta^*_A,i}\left(\theta_i - \theta^*_{\mathcal{A},i}\right)^2$$

- Fisher Information Matrix

$$F_{\theta^*_A} = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log p\left(x_{\mathcal{A},i}|\theta^*_{\mathcal{A}}\right) \nabla_\theta \log p\left(x_{\mathcal{A},i}|\theta^*_{\mathcal{A}}\right)^T$$

- Loss function

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \sum_i \frac{\lambda}{2} F_i\left(\theta_i - \theta^*_{A,i}\right)^2 \tag{4}$$

Learning without Forgetting (LwR)

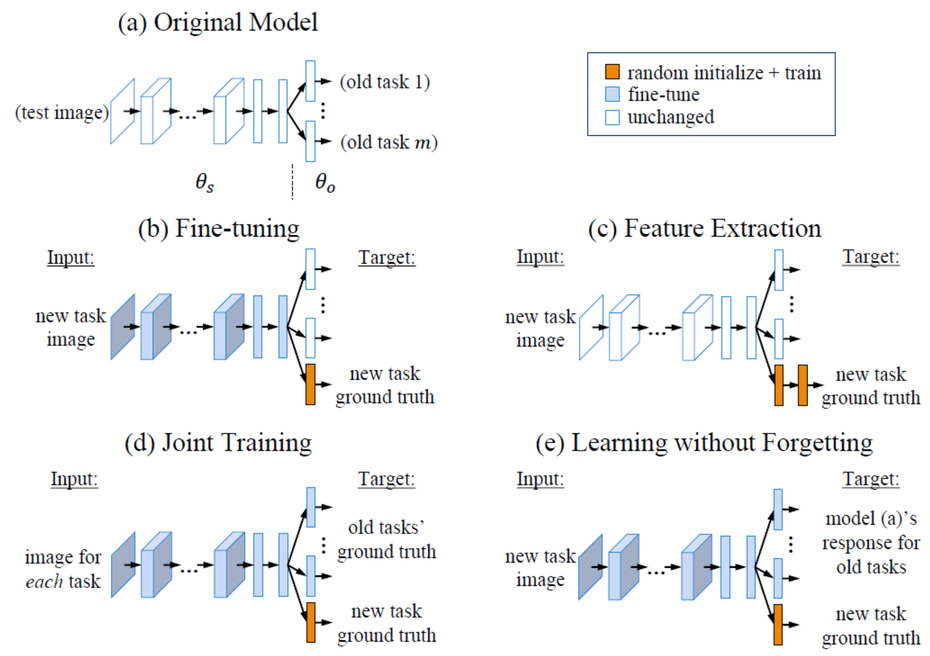

Figure: The learning sequence is from task A to task B

- θ_s: a set of shared parameters for CNNs (e.g., five convolutional layers and two fully connected layers for AlexNet [3] architecture),
- θ_0: task-specific parameters for previously learned tasks (e.g., the output layer for ImageNet [4] classification and corresponding weights),
- θ_n: randomly initialized task specific parameters for new tasks(e.g., scene classifiers).
- Loss function

$$\mathcal{L}_{new}(y_n, \hat{y}_n) = -y_n \cdot \log \hat{y}_n \tag{5}$$

- Knowledge Distillation loss

$$\mathcal{L}_{old}(y_o, \hat{y}_o) = -H\left(y'_o, \hat{y}'_o\right) = -\sum_{i=1}^{l} y'^{(i)}_o \log \hat{y}'^{(i)}_o$$

$$y^{(i)}_o = \frac{\left(y^{(i)}_o\right)^{1/T}}{\sum_j \left(y^{(j)}_o\right)^{1/T}}, \quad \hat{y}'^{(i)}_o = \frac{\left(\hat{y}^{(i)}_o\right)^{1/T}}{\sum_j \left(\hat{y}^{(j)}_o\right)^{1/T}}$$

Close Look at LwR

LEARNINGWITHOUTFORGETTING:
Start with:
θ_s: shared parameters
θ_o: task specific parameters for each old task
X_n, Y_n: training data and ground truth on the new task
Initialize:
$Y_o \leftarrow \text{CNN}(X_n, \theta_s, \theta_o)$ // *compute output of old tasks for new data*
$\theta_n \leftarrow \text{RANDINIT}(|\theta_n|)$ // *randomly initialize new parameters*
Train:
Define $\hat{Y}_o \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_o)$ // *old task output*
Define $\hat{Y}_n \equiv \text{CNN}(X_n, \hat{\theta}_s, \hat{\theta}_n)$ // *new task output*
$\theta^*_s,\ \theta^*_o,\ \theta^*_n \leftarrow \underset{\hat{\theta}_s,\hat{\theta}_o,\hat{\theta}_n}{\operatorname{argmin}} \left(\lambda_o \mathcal{L}_{old}(Y_o, \hat{Y}_o) + \mathcal{L}_{new}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n)\right)$

Figure: The details of algorithms

Experiment Setting and Results

- **Settings**: The resnet101 is used as our base model. The task is first sequentially trained on the training set. The total epoch of whole dataset is about 12*2(for each task) in total.

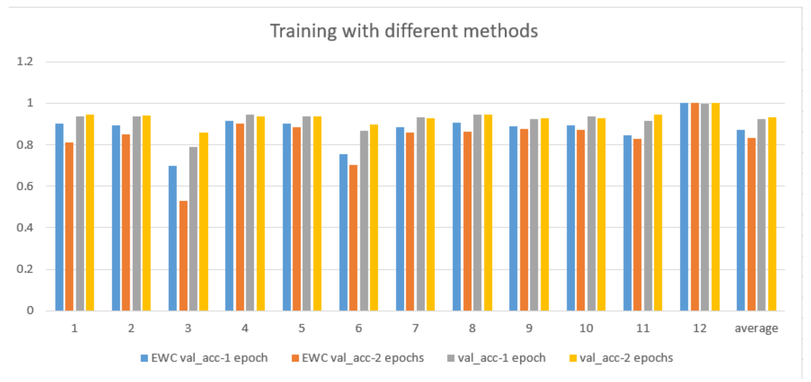

Figure: Training with different methods and configurations, X represents for task name and average accuracy, while y is the accuracy.

- We first training the task sequentially and got 93.33% average accuracy at Validation set across task 1 to task 12. However, during the training process, the accuracy test on Validation set is nearly 100%, which means the model is suffering from the catastrophic forgetting problem in sequentially training.
- EWC is then employed on the training process, however, the result is getting worse.

Conclusion

- Sequentially Training will be suffering from the catastrophic forgetting problem.
- Less training epochs out performances large training epochs.
- EWC training has a worse result due to the fact the estimation of Fisher Information Matrix might be biased estimated.
- In the future, we will focus on the dynamic graph for better preserving the memory of pervious task.

Acknowledgments

- Thanks to Robotics and Artificial Intelligence Lab (RAIL) and Shenzhen Institute of Artificial Intelligence and Robotics for Society (AIRS) for supporting our group to participate in the Life Long Recognition challenge of the IROS 2019.

References

[1] 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, 114(13):3521–3526, 2017.

[2] Zhizhong Li and Derek Hoiem.
Learning without forgetting.
IEEE transactions on pattern analysis and machine intelligence, 40(12):2935–2947, 2017.

Liguang Zhou: liguangzhou@link.cuhk.edu.cn