Learning Without Forgetting Approaches for Lifelong Robotic Vision

Zhengwei Wang\(^1\), Eoin Brophy\(^2\) and Tomás E. Ward\(^2\)

Abstract—Recent advances in deep learning have achieved exciting results in the areas such as object detection, image recognition and object localization. However, robotic vision poses new challenges for applying visual algorithms due to varying distribution of images from real world and it requires that the model is able to learn knowledge continuously. This competition is about developing lifelong learning algorithms which can be applied to the robotic vision system. This work describes the approach that we submit to this open competition.

I. INTRODUCTION

Humans have remarkable abilities to learn knowledge continuously from the real world. One of ultimate goals of a robotic vision system is to build an artificial intelligent agent which is capable of understanding the real world based on their current scenes and their previous knowledge. Object recognition in the computer vision area has achieved exciting results [1], [2], [3], [4], where some deep neural networks (DNNs) even outperform human annotators. However, these approaches still have limitations when they are applied to a robotic vision system [5], [6]. First, distributions of image datasets may vary across categories and tasks. For example, illumination can vary significantly across time (day time and night differences). A well-developed approach should have ability to recognize the object correctly with different illumination levels. Second, it is not feasible to train a model by using all images across tasks for a robotic vision system because it requires more and more computational complexity when tasks are accumulated i.e., the model needs to be pretrained by using previous tasks. Both \( \hat{\theta}_p \) and \( \theta_\text{old} \) are updated by using previous tasks in order to construct the regularization term for training new weights \( \theta_\text{new} \). It should be noted that there is no replay of previous task images in this structure and only the update \( \theta_\text{new} \) is retained after training which is going to be used during the testing session. During the experiment, we find that initializing \( \theta_p \) by using the pretrained weights for each task performs better than using \( \theta_p \) continuously trained by all tasks. Thus we load the initial pretrained weights \( \theta_p \) when processing a new task and \( \theta_p \) is going to be fine tuned during the training. Details of training scheme are included in Algorithm 1.

Algorithm 1 Training details

Inputs:
- Training images \( X \), labels \( Y \) of the new task and the pretrained parameters \( \theta_p \)

Initialize:
1: \( Y_\text{old} \leftarrow M_{\theta_p, \theta_\text{old}}(X) \) // Output labels using model trained by previous tasks. Both \( \hat{\theta}_p \) and \( \theta_\text{old} \) are updated by using previous tasks
2: \( \theta_\text{new} \leftarrow \text{Xavier-init}(\theta_\text{new}) \) // Use Xavier initialization for the bottleneck weights
3: Load the pretrained weights \( \theta_p \) to the new model

Train:
4: \( \theta_p^*, \theta_\text{new}^* \leftarrow \argmin_{\theta_p, \theta_\text{new}} (L_\text{old}(Y, Y_\text{old}) + L_\text{new}(Y, Y_\text{new})) \)
5: \( \theta_\text{old} \leftarrow \theta_\text{new} \) // Cache the updated weights which are going to be used as old weights for the next task.
It should be noted that there are some differences between our approach and the original approach. First, we did not retain old weights $\theta_{old}$ for each task. This might deteriorate the performance of the model but this is closer to the real-world situation because we do not know which task is going to be tested. Practical situation requires a unified model which is able to learn tasks continuously. Our method is also more computationally efficient during the training compared to the original method especially when the number of tasks is huge. Second, instead of fine tuning $\theta_p$ continuously for each task, we load the pretrained weights to $\theta_p$ for each new task and then fine tune it. We find this strategy will improve the performance.

III. RESULTS

We compared our model ($M_{LwF}$) to model 1 ($M_1$) which does not deploy any lifelong learning strategy and model 2 ($M_2$) which fine tunes $\theta_p$ continuously as we mentioned before. Performance of three models is included in Table I. It can be seen that $M_{LwF}$ achieves the highest accuracy and this model is our final solution submitted to the competition.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M_{LwF}$</td>
<td>76.6%</td>
</tr>
<tr>
<td>$M_1$</td>
<td>60.9%</td>
</tr>
<tr>
<td>$M_2$</td>
<td>74.3%</td>
</tr>
</tbody>
</table>

TABLE I

VALIDATION ACCURACY OF THREE MODELS.

IV. CONCLUSIONS

In this work, we described the approach we used for solving the lifelong robotic vision challenge. The core backend of our approach is LwF which was proposed to overcome the catastrophically forgetting issue arisen from the lifelong learning. We modified the original approach in order to be more suitable to deal with this challenge.

ACKNOWLEDGMENT

The authors thank organizers for organizing such a great challenge which addresses the importance of lifelong learning in our real world and providing the extraordinary datasets. We also acknowledge the financial support by Science Foundation Ireland (SFI) under the Grant Number 15/RP/2776 and 17/RC-PhD/3482.

REFERENCES