Learning without Forgetting

Abstract:

When adding new capabilities to a system, the usual assumption is that training data for all tasks is always available. However, as the number of tasks grows, storing and retraining on such data becomes unavailable. Due to this, when we add new capabilities to Convolution Neural Network (CNN), the existing capabilities are unavailable. To overcome this problem, we implement Learning without Forgetting method, which uses only new task data to tarin the network while preserving the original capabilities.

A diagram of a model

Description automatically generated with medium confidence

Learning without Forgetting:

The purpose of Learning without Forgetting (LWF) is to learn a network that can perform well on both old tasks and new tasks when only new-task data is present. The figure above shows the working principle of LWF compared with other methods.

The key idea of LWF is inspired by knowledge Distillation.

Knowledge Distillation:

It is a method to distill the knowledge in an ensemble of cumbersome models and compress into a single model in order to make possible deployments to real-life applications.

Knowledge Distillation refers to the transfer of the learning behaviour of a model (teacher) to a student, in which, the output produced by the teacher is used as the targets for training the student. By applying this method, we can achieve results and an improvement can be obtained by distilling the knowledge in a number of models into a single model.

Note:

1. We can install MATLAB Deep Learning Toolbox: It provides functions and tools necessary for working with CNNs.
2. We can load pre-trained CNN model. MATLAB provides pre-trained models like AlexNet.
3. We can install MatConvnet – It is MATLAB toolbox implementing Convolution Neural Networks (CNNs)

θs: Five convolution layers and two fully connected layers

θo: Task-specific parameters for previously learned tasks

θn: Task-specific parameters for new tasks

Feature Extraction: θs and θo are unchanged, and the outputs of one or more layers are used as features for the new task in training θn.

Fine-tuning: θs and θn are optimized for the new task, while θo is fixed. A low learning rate is typically used to prevent large drift in θs. Potentially, the original network could be duplicated and fine-tuned for each new task to create a set of specialized networks.

Joint Training: All parameters θs, θo and θn are jointly optimized, for example by interleaving samples from each task.

Each of these methods has a major drawback. Feature extraction typically underperforms on the new task because the shared parameters fail to represent some information that is discriminative for the new task. Fine-tuning degrades performance on previously learned tasks because the shared parameters change without guidance for the original task-specific prediction parameters. Joint training becomes increasingly in training as more tasks are learned and is not possible if the training data for previously learned tasks is unavailable.

To overcome all these problems with the methods above we can use Learning without Forgetting method.

The Learning without Forgetting method does not need the old task’s images and labels. Clearly, if the network is preserved such that θo produces exactly the same outputs on all relevant images, the old task accuracy will be the same as the original network.

A white paper with black text

Description automatically generated

Given a CNN with shared parameters θs and task-specific parameters θo. The goal is to add task-specific parameters θn for a new task and to learn parameters that work well on old and new tasks, using images and labels from only the new task.

First, we record responses yo on each new task image from the original network for outputs on the old tasks (defined θs and θo). The responses are the set of label probabilities for each training image. Nodes for each new class are added to the output layer, fully connected to the layer beneath, with randomly initialized weights θn. The number of new parameters is equal to the number of new classes times the number of nodes in the last shared layer.

\*\*\*\*\*To Train the CNN with new tasks I got online resources for datasets\*\*\*\*\*\*

1. PASACAL VOC (PASCAL Visual Object Classes)
2. MIT indoor scene
3. MNIST Database

For Image classification we can get datasets from CIFAR-10 and CIFAR-100, ImageNet

Note: The above implementation can be done in MATLAB

But we can implement in Python as well,

1. We need to choose deep learning framework such as TensorFlow or PyTorch.
2. Load the pre-trained CNN model.
3. Modify the pre-trained CNN to adapt the new tasks.
4. Load the dataset and start training the model.