Dynamic networks

Deep neural networks are proven to produce task-specific features. For example, when the training dataset contains cars, the model tends to depict the wheels and windows. However, if the model is updated with the new classes containing cats, the features would be adapted for beards and stripes. Since the capacity of a model is limited, adapting to new features will result in the overwriting of old ones and forgetting. Hence, utilizing the extracted features for beards and strides is inefficient for recognizing a car. To this end, dynamic networks are designed to dynamically adjust the model’s representation ability to fit the evolving data stream. There are several ways to expand representation ability, and we divide them into three sub-groups, i.e., neural expansion, backbone expansion, and prompt expansion.

Early works focus on neural expansion, which adds neurons when the representation ability is not enough to capture new classes. DEN [64] formulates the adjusting process into selection, expansion, duplication, and elimination. Facing a new incremental task, the model first selectively retrains the neural that are relevant to this task. If the retrain loss is still above some threshold, DEN considers expanding new neurons top-down and eliminating the useless ones with group-sparsity regularization. Afterward, it calculates the neuron-wise drift and duplicates neurons that drift too much from the original values. Apart from heuristically expanding and shrinking the network structure, RCL[65] formulates the network expansion process into a reinforcement learning problem and searches for the best neural network architecture for each incoming task. Similarly, Neural Architecture Search (NAS) is also adopted to find the optimal structure for each of the sequential tasks.

Expanding neurons shows competitive results with expandable representation. Correspondingly, several works try to duplicate the backbone network for stronger representation ability. PNN [67] proposes learning a new backbone for each new task and fixing the formal in incremental learning. It also adds layer-wise connections between old and new models to reuse former features. Expert Gate [68] also expands the backbone per incremental task, while it requires learning an extra gate to map the instance to the most suitable pathway during inference. To release the expansion cost, P&C [69] suggests a progression-compression protocol - it first expands the network to learn representative representations. Afterward, a compression process is conducted to control the total budget. [70], [71] maintain a dual branch network for class incremental learning, one for fast adaptation and one for slow adaptation. Recently, DER [20] has been proposed to address the CIL problem. Similar to PNN, it expands a new backbone when facing new tasks and aggregates the features with a larger FC layer. Take the second incremental task for an example, where the model output is aggregated as:

,

where is the former backbone, and is the newly initialized backbone. denotes feature aggregation, and is the newly initialized FC layer. In the model updating process, the old backbone is frozen to maintain former knowledge.

where denotes the old backbone is frozen. DER also adopt an auxiliary loss to differentiate between old and new classes. Eq. 9 depicts a way to continually expand the model with new features. Under such circumstances, if the old backbone is optimized with ‘cars’, the features extracted by are then representative of wheels and windows. The new backbone trained with ‘cat’ is responsible for extracting beards and stripes. Since the old backbone is frozen in later stages, learning new classes will not overwrite the features of old ones, and forgetting is alleviated. Figure 4 depicts the model evolution of DER.

DER has achieved powerful performance with humble forgetting. However, saving a backbone per task requires numerous memory size, and many works are proposed to obtain expandable features with a limited memory budget. FOSTER formulates the learning process in Eq. 9 as a feature-boosting problem. It argues that not all expanded features are needed for incremental learning and need to be integrated to reduce redundancy. For example, suppose old classes contain ‘tigers’, and new classes contain ‘zebras’. In that case, the stripe will be a useful feature that could be extracted by bold old and new backbones. Under such circumstances, forcing the new backbones to extract the same kind of feature is less effective for recognition. Hence, FOSTER adds an extra model compression process by knowledge distillation [144]:

(10)

Eq. 10 aims to find the student model that has the same discrimination ability as the teacher model by minimizing discrepancy between them. The teacher is the *frozen* expanded model with two backbones: , and the student is newly initialized model . Hence, the number of backbones is consistently limited to a single one, and the memory will not suffer catastrophic expansion. Figure 4 depicts the model evolution of FOSTER.

Recently, MEMO[22] has been proposed to address the memory problem in CIL, aiming to enable model expansion with the least budget cost. It finds that in class-incremental learning, shallow layers of different models are similar, while deep layers are diverse. In other words, shallow layers are more generalizable, while deep layers are specific to the task, making expanding shallow layers less memory-efficient for CIL. Hence, MEMO proposes decouple the backbone at middle layers: where specialized block corresponds to the deep layers in the network, while generalized block corresponds to the rest shallow layers. Compared to DER, MEMO only expands specialized blocks , and transforms Eq. 9 into:

Which indicates that task-specific deep layerscan be built for each task upon the shared shallow layers . Figure 4 depicts the model evolution of MEMO.

Recently, Vision Transformer (ViT) has attracted the attention of the computer vision community, and many works tend to design CIL learners using ViT as the backbone. DyTox is the first work to explore ViT in CIL, which finds that model expansion in ViT is much easier than in convolutional networks. In DyTox, only task tokens are expanded for each new task, which requires much less memory than saving the whole backbone. Similarly, L2P and DualPrompt explore how to build CIL learners with pre-trained ViT. They borrow ideas from Visual Prompt Learning (VPT) to incrementally finetune the model with prompts. In L2P, the pre-trained ViT is frozen during the learning process, and the model only optimizes the prompts to fit new patterns. The prompt pool is defines as:

Where M is the total number of prompts, is a single prompt with token length and the same embedding size as the instance embedding . The prompts are organized as key-value pairs – each instances selects the most similar prompts in the prompt pool via KNN search. It obtains instance-specific predictions by adapting the input embeddings as:

are selected prompts via KNN search. The adapted embeddings are then fed into the self-attention layers to obtain instance specific representations. CODA-Prompt extends the prompt search with the attention mechanism. Apart from pre-trained ViT, S-Prompt utilizes the pre-trained language-vision model CLIP for incremental learning, which simultaneously learns language prompts and visual prompts to boost representative embeddings. [76] extends CLIP with spatial and temporal prompts for incremental video classification.

3.2.2 Parameter Regularization

Dynamic networks seek to adjust model capacity with data evolves. However, if the model structure is fixed and unchangeable, how to adjust the plasticity to resist catastrophic forgetting? Parameter regularization methods consider that the contribution of each parameter to the task is not equal. However, they seek to evaluate the importance of each parameter to the network and keep the important ones static to maintain former knowledge.

Typical works estimate a distribution over the model parameters and use it as the prior when learning new tasks. Due to the large amounts of parameters, the estimation process often assumes them to be independent. EWC is the first work addressing parameter regularization. It maintains an importance matrix with the same scale of the network, i.e., . Denote the -th model parameter as , the importance of is represented by (the larger indicates that is more important). Apart from the training loss in Eq. 4 to learn new classes, EWC builds an additional regularization term to remember old ones:

(13)

The parameter-wise regularization term is calculated based on two parts. denote the -th parameter after the learning last stage . Hence, represents the parameter drift from the last stage, and weighs it to make sure important parameters do not shift away from the last stage. Since the model at last stage represents the ‘old knowledge’, consolidating important parameters can fix the knowledge from being forgotten.

Eq.13 depicts a way to penalize essential parameters, and there are different ways to calculate the importance matrix . In EWC, Fisher information matrix is adopted to estimate However, the importance calculation in EWC is conducted at the end of each task, which ignores the optimization dynamics along the model training trajectory. To this end, SI proposes to estimate in an online manner and weigh the importance via its contribution to loss decay. RWalk combines these importance estimation techniques. [79], [80] resort to an extra unlabeled dataset for online evaluation. IMM [81] finds a maximum of the mixture of Gaussian posteriors with the estimated Fisher information matrix. IADM [82] and CE-IDM[83] analyze the capacity and sustainability of different layers in class-incremental learning and find that different layers have different characteristics in CIL. In detail, shallow layers converge faster but have limited representation ability. By contrast, deep layers converge slowly while having powerful discrimination abilities. Hence, IADM augments EWC with an ensemble of different layers and learns layer-wise importance matrix in an online manner. K-FAC extends the Fisher information matrix approximation with the Kronecker factorization technique.

3.2.3 Discussions about Model-Centric Methods.

Model-centric methods either design model structure adjustment techniques or build generalization terms to resist forgetting. Apart from these groups, there are works addressing network masks to divide a large network into sub-networks for each task [149], [150], [151], [152]. However, deciding the activation of a specific sub-network requires the task identifier or learning extra task classifiers. On the other hand, several works [153], [154], [155] propose to design specific modules for incremental new tasks, e.g., adapters [156]. However, manually handcrafting these modules requires heuristic designs or task-specific priors.

Learning dynamic networks, especially backbone expansion methods, have achieved state-of-the-art performance in recent years [22]. However, if often requires expandable memory budgets, which is unsuitable for incremental learning on edge devices. Secondly, pre-trained models are needed for the prompt expansion method. However, a pre-trained model is not always available for some specific downstream tasks, e.g., face recognition [157], speech recognition [158], etc. Therefore, how to get rid of the dependence on pre-trained model is essential for these methods in real-world applications.

On the other hand, parameter regularization methods have attracted the attention of the community in the early years. However, estimating parameter importance requires saving the matrix with the same scale as the model. It faces the same risk as dynamic networks that the memory budget is linearly increasing. On the other hand, the importance matrix may conflict at different incremental stages [92], making it hard to optimize the model.

3.3 Algorithm-Centric Class-Incremental Learning

Algorithm-centric CIL methods focus on designing algorithms to maintain the model’s knowledge in former tasks. For example, the core idea of knowledge distillation-based CIL method is to build the mapping between old and new models and reflect the characteristics of the old model in the updating process. On the other hand, model rectify-based methods aim to discover and reduce the bias in incremental models.

3.3.1 Knowledge Distillation

Training data is evolving in the incremental learning process, requiring tuning the model sequentially. We can denote the previous stage as the ‘old model’ and the current updating model as the ‘new mode’. Assuming the old model is a good classifier for all then seen classes in , how can we utilize it to resist forgetting in the new model?

To enable the old model to assist the new model, an intuitive way utilizes the concept proposed in [144], [159], [160], i.e., knowledge distillation (KD). KD enables the knowledge transfer from a teacher model to the student model, with which we can teach the new model not to forget. There are several ways to build the distillation relationship, and we divide these KD-based methods into three subgroups, i.e., logit distillation, feature distillation, and relational distillation.

LwF [85], [86] is the first success to apply knowledge distillation into CIL. Similar to Eq. 13, it builds the regularization term via knowledge distillation to resist forgetting.

(14)

where the old model is frozen during updating. The regularization term builds the mapping between the old and new models by forcing the predicted probability among old classes to be the same. Given a specific input x, the output probability of the -th class reveals the sematic similarity of the input to this class. Hence, Eq.14 forces the semantic of the old model and new model to be the same and resists forgetting. iCARL [32] extends LwF with the exemplar set, which helps to further recall former knowledge during incremental learning. Additionally, it drops the fully-connected layers and follows [161] to utilize nearest-mean-of-examplars during inference. Eq. [14] strikes a trade-off between old and new classes, where the former part aims to learn new classes and the latter one maintains old knowledge. Since the number of old and new classes may differ in different incremental stages, BiC [87] extends Eq[14] by introducing a *dynamic* trade-off term.

LwF inspires the community to build the mapping between models, making knowledge distillation a useful tool in CIL. D+R[88] suggests changing the first part in Eq.14 into a distillation loss by training an extra expert model. GD[89] proposes to select wild data for model distillation and designs a confidence-based sampling method to effectively leverage external data. Simillarly, DMC[90] proposes to train a new model in each incremental stage and then compress them into a single model via an extra unlabeled dataset. Finally, if no additional data is available for knowledge distillation, ABD [91] proposes distilling synthetic data for incremental learning. These methods only concentrate on utilizing the old model to help resist forgetting in the new model. However, COIL suggests conducting bidirectional distillation with co-transport, where semantic relationships between old and new models are both utilized.