Dense Network Expansion for Class Incremental Learning

Zhiyuan Hu[1] Yunsheng Li[2] Jiancheng Lyu[3] Dashan Gao[3] Nuno Vasconcelos[1]
[1]UC San Diego [2]Microsoft Cloud + AI [3]Qualcomm AI Research
{z8hu, nvasconcelos}@ucsd.edu, yunshengli@microsoft.com, {jianlyu, dgao}@qti.qualcomm.com

Abstract

The problem of class incremental learning (CIL) is considered. State-of-the-art approaches use a dynamic architecture based on network expansion (NE), in which a task expert is added per task. While effective from a computational standpoint, these methods lead to models that grow quickly with the number of tasks. A new NE method, dense network expansion (DNE), is proposed to achieve a better trade-off between accuracy and model complexity. This is accomplished by the introduction of dense connections between the intermediate layers of the task expert networks, that enable the transfer of knowledge from old to new tasks via feature sharing and reusing. This sharing is implemented with a cross-task attention mechanism, based on a new task attention block (TAB), that fuses information across tasks. Unlike traditional attention mechanisms, TAB operates at the level of the feature mixing and is decoupled with spatial attentions. This is shown more effective than a joint spatial-and-task attention for CIL. The proposed DNE approach can strictly maintain the feature space of old classes while growing the network and feature scale at a much slower rate than previous methods. In result, it outperforms the previous SOTA methods by a margin of 4% in terms of accuracy, with similar or even smaller model scale.

1. Introduction

Deep learning has enabled substantial progress in computer vision. However, existing systems lack the human ability for continual learning, where tasks are learned incrementally. In this setting, tasks are introduced in sequential time steps t, and the dataset used to learn task t is only available at the t^{th} step. Standard gradient-based training is not effective for this problem since it is prone to *catastrophic forgetting*: the model overfits on task t and forgets the previous tasks. This is unlike humans, who easily learn new tasks without forgetting what they know. While continual learning can be posed for any topic in computer vision, most research has addressed classification and the *class incremental* (CIL) setting [18]. In CIL, tasks consist of subsets of disjoint classes that are introduced sequentially. Most approaches also allow the learning of task t to access a small buffer memory of examples from previous tasks.

Different strategies have been proposed to solve the CIL problem. *Distillation methods* [3, 8, 9, 12, 14, 18, 23, 24, 29] and *parameter regularization* methods [15, 32] regulate the new model by the output logits, intermediate features or important parameters. Gradient methods [16, 19, 28] estimate the null space of the existing feature space and project the gradients of the next task into this null space, so that the newly learned features are orthogonal to the previous ones. These methods try to fit all tasks into a single model, preserving properties of the feature space from one task to the next. This is, however, difficult to guarantee due to the scarcity of prior task data. Furthermore, as the number of tasks grows, the model will eventually run out of capacity to accommodate new tasks.

Network expansion (NE) methods [1, 22, 27, 30, 31] address these problems by freezing the model that solves the previous tasks and adding a new subnetwork per task. As illustrated in Figure 1, a network f^1 learned for task 1 is augmented with new networks f^t, denoted as *task experts,* for each subsequent task t, and the feature spaces concatenated. *NE with cross connections* (NEwC) methods [10,20,25] further add links across tasks (red lines in Figure 1) to further transfer knowledge from old to new tasks. Since the original features are always accessible for examples from previous classes, these methods achieve the best performances on CIL benchmarks. However, the process is very inefficient in terms of both model size and complexity. The right side of the figure shows the accuracy and size of several NE models on the CIFAR100 dataset. Best accuracies are obtained with larger networks and the model grows very quickly with the number of tasks. For most practical applications, this rate of growth is unsustainable.

While NE and NEwC achieve state of the art performance among CNN-based CIL methods, we show that they do not translate well to the more recent transformer architecture [7]. Standard transformers learn spatial connections across image patches through a spatial attention mechanism. A natural CIL extension is to feed the input image to multiple heads, each corresponding to a task. Attention can

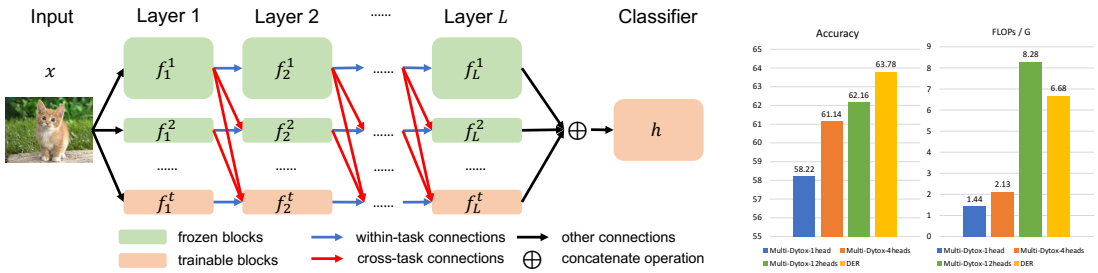


Figure 1. CIL by NE. Left: Given a new task t, a new branch, denoted the *task t expert*, is added while freezing existing experts. In classical NE, the model is simply replicated per task, originating a set of t independent models, whose outputs are concatenated into a classifier. In the proposed DNE scheme, a cross-task attention mechanism (red connections) is introduced to allow the re-use of knowledge from previous experts and smaller experts per task. Right: Comparison of model accuracy and size for various implementations of NE, using DER [31] and multi-Dytox [9] models of different sizes per task expert, on the CIFAR100 dataset. Both accuracy and FLOPs are shown for the final model, which classifies 100 classes grouped into 6 tasks.

then be computed between all image patches of all heads, leading to a Spatial-and-Task Attention (STA) mechanism. This strategy has is commonly used in multi-modal transformers [2,21]. However, in CIL, the features generated per patch by different heads are extracted from exactly the same image region. Hence, their representations are highly similar and STA is dominated by the attention between replicas of the same patch. Furthermore, because all patches are processed by all heads, the remaining attention is dispersed by a very large number of patch pairs. This leads to the *fragmentation* of attention into a large number of small-valued entries, which severely degrades performances. To overcome this problem, we propose a *Dense Network Expansion* (DNE) strategy that disentangles spatial and cross-task attention. Spatial attention is implemented by the standard transformer attention mechanism. Cross-task attention is implemented by a novel *task attention block* (TAB), which performs attention at the level of feature-mixing, by replacing the multi-layer perceptron (MLP) block with an attention module.

Overall, the paper makes four contributions. First, we point out that existing NE methods are unsustainable for most practical applications and reformulate the NE problem, to consider the trade-off between accuracy and model size. Second, we propose the DNE approach to address this trade-off, leading to a CIL solution that is both accurate and parameter efficient. Third, we introduce an implementation of DNE based on individual spatial and cross-task attentions. Finally, extensive experiments show that DNE outperforms all previous CIL methods not only in terms of accuracy, but also of the trade-off between accuracy and scale.

2. Related Works

CIL aims to learn a sequence of classification tasks, composed of non-overlapping classes, without catastrophic forgetting. The literature can be roughly divided into *single model* and *network expansion* (NE) methods.

Single model methods: Single model methods assume that catastrophic forgetting can be avoided by imposing constraints on the model as new tasks are learned. *Distillation* methods [8,9,12,14,18,23,24,29] feed the data from the current task to both old and new networks. [9, 12, 18, 24] force the new network to reproduce the logits of the old network, to guarantee stable logits for old class data. [8] and [14] further match the intermediate feature tensors of the old and new networks. [29] proposes a transformer-based method that defines an external key feature per attention block to represent old tasks and regularizes this feature. *Parameter regularization* methods [15, 32] assume that the knowledge of old classes is stored in the model parameters. For example, [15] uses the Fisher information matrix of a parameter to estimate its importance for certain classes. Important parameters are then kept stable across tasks using a L2 regularization weighted by parameter importance. *Gradient* methods [16, 19, 28] modify gradient descent training to eliminate catastrophic forgetting. [19] computes loss gradients for current and old classes. The current gradient is then forced to have a positive dot product with the old gradients, so that training of current classes does not increase the loss for old ones. Since this is typically difficult to satisfy, [28] proposes to directly estimate the null space of the gradients of old classes. The new gradients are then projected into this null space, to eliminate their influence on old classes.

NE methods: While single model methods solve the catastrophic forgetting problem to some extent, the scarcity of data from previous tasks makes it difficult to guarantee model stability across tasks. Furthermore, a single model eventually lacks the capacity to accommodate all tasks, as task cardinality grows. *Dynamic architecture* methods [1, 22, 27, 30, 31] address this problem with NE. A new network, or task expert, is learned per task, while previous experts are frozen. In a result, the features originally produced for old task classes are always available. [1, 31] train an entire backbone per task expert. [1] uses an auto-encoder task level selector, to select the expert that best suits the example to classify. [31] directly concatenates the feature vectors generated by old and new experts for classification. These methods outperform single model ones, but lead to models that grow very quickly with the number of tasks. Hence, they are unrealistic for most practical applications.

Some works have sought a better trade-off between model accuracy and complexity. [30] notes that the shallow layers of the old and new experts learn similar low-level features, and uses pre-trained shallow layers that are shared by all experts. This reduces model size and leverages the power of pretraining. [27] adds an entire backbone per task, but distills the old and new networks into a single one, with the size of a single backbone. The model size is thus kept fixed as the number of tasks grows. However, these solutions are mostly pre-defined, either breaking the model into a single model component (early layers) and a component that grows without constraints (later layers) or effectively using a single model. The proposed DNE approach instead explores the use of connections between task experts, feature reuse between tasks, and dynamic information fusion through cross-task attention to enable a significantly better trade-off between accuracy and model size.

3. Dense Network Expansion

In this section, we first revisit the mathematical definition of CIL problem and previous NE methods amd then introduce the DNE approach.

CIL: We consider the problem of image classification, where image x has category label y, and the CIL setting, where a model $g(x;\theta)$ of parameter θ learns a sequence of M tasks $T = \{T_1, T_2, \ldots, T_M\}$. Each task T_i has a dataset $D_i = \{(x_k, y_k)\}_{k=1}^{N_i}$ of N_i samples from a class set $\mathcal{Y}_i$ disjoint from the remaining task sets, i.e. $\mathcal{Y}_i \cap \mathcal{Y}_j = \varnothing, i \neq j$.

At step t, the model predicts the posterior class probabilities $g_i(x;\theta) = P_{Y|X}(i|x)$ for all the classes observed so far $i \in \mathcal{Y}_1 \cup \mathcal{Y}_2 \cup \cdots \cup \mathcal{Y}_t$. In strict incremental learning, only the dataset D_t of the current task is available for learning. However, this has proven too challenging, most incremental learning methods assume the availability of a small memory buffer $\mathcal{M} = \{(x_k, y_k)\}_{k=1}^{S}$ with data from prior tasks, where S is the buffer size and each sample (x_k, y_k) is drawn from the union of previous datasets $D_1 \cup D_2 \cup \cdots \cup D_{t-1}$.

Most deep learning models $g(x;\theta)$ consist of a series of blocks and a classifier. The network parameters can be divided into $\theta = \{\theta_1, \theta_2, \ldots, \theta_L, \phi\}$ where θ_i is the parameter vector of the i-th block, ϕ that of the classifier and L the number of blocks. The network is then implemented as

$$
\begin{aligned}
r_0 &= x && (1)\\
r_l &= f_l(r_{l-1};\theta_l), \;\; l \in \{1,\ldots,L\} && (2)\\
g(x;\theta) &= h(r_L;\phi), && (3)
\end{aligned}
$$

where f_l represents the l-th block, h is the classifier and r_l the feature tensor at the output of the l-th block.

NE methods: A popular approach to CIL is to rely on the NE procedure of Figure 1, which learns a task expert f^t, of the form of (2), per task t. To learn this branch, NE methods freeze experts $f^1, \ldots, f^{t-1}$ of the previous tasks, forcing the new expert to generate features specific to task t. The entire network can then be written as

$$
\begin{aligned}
r_0^t &= x && (4)\\
r_l^t &= f_l^t(r_{l-1}^t;\theta_l^t),, \;\; l \in \{1,\ldots,L\} && (5)\\
g(x;\theta) &= h(r_L;\phi), && (6)
\end{aligned}
$$

where l denotes block, t denotes task,

$$
\begin{aligned}
r_l &= r_l^1 \oplus r_l^2 \oplus \cdots \oplus r_l^t && (7)\\
\theta_l &= \{\theta_l^1, \theta_l^2, \ldots, \theta_l^t\} && (8)\\
\phi &= \{\phi^1, \phi^2, \ldots, \phi^t\} && (9)
\end{aligned}
$$

and $\oplus$ is the concatenation operation. The parameters $\theta_l^1, \ldots, \theta_l^{t-1}$ are frozen and only θ_l^t is learned for each block l. The classifier parameters ϕ are learned over the entire feature vector r_L, leveraging the features produced by all task experts to assign x to one of the classes from all tasks.

Challenges: While NE is popular, it can be quite inefficient. If the dimensionality of the new task parameters θ_l is small, i.e. the new task expert is a small network, the model has limited capacity to learn the new task and the recognition accuracy can be low. On the other hand, if the dimensionality is large, e.g. a full network as in the popular DER [31] method, the model size grows very quickly and there can be too much capacity, leading to overfitting.

This problem is illustrated in the right side of Figure 1, which shows both the accuracy and model size obtained by expanding the Dytox model of [9] using various parameter dimensionalities. In this example, each expert is a Dytox transformer with a different number of spatial attention heads k. The figure shows the CIL performance on CIFAR100, for a sequence of 6 tasks of $\{50, 10, 10, 10, 10, 10\}$ classes each, by networks with $k \in \{1, 4, 12\}$. While a single attention head per task is not sufficient to enable high recognition accuracy, the size of the model grows very quickly as more heads are used per task, without a large increase in recognition accuracy. In this work, we seek expansion methods with a better trade-off between model accuracy and size.

We hypothesize that the inefficiency of NE is due to its lack of ability to transfer knowledge from old tasks to new task. To address this problem we introduce *cross-task connections,* connecting the layers of the new task expert to those of the experts already learned for the previous tasks. For task t, block f_l^t takes feature vectors of all tasks as input to generate the outputs of the task,

$$
r_l^t = f_l^t(r_{l-1}^1, \ldots, r_{l-1}^{t-1}, r_{l-1}^t; \theta_l^t) \qquad (10)
$$

The question becomes how to design a task t expert capable of learning features *complimentary* to those already available. For this, the expert must integrate the features generated by the previous task experts. This issue has been

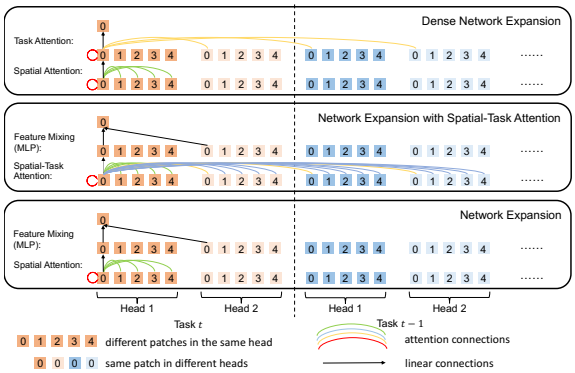


Figure 2. Different implementations of cross-task attention. Only one output patch is shown for simplicity. Bottom: network expansion has no cross-task attention. Middle: Network expansion with Spatial Attention adds spatial attention connections between task experts. Top: DNE replaces the MLP with cross-task attention connections between corresponding patches of different heads.

considered in the literature for purely convolutional models [10, 20, 25], where cross-task connections are trivially implemented as a convolution block that linearly projects the output channels of all task experts into input channels of the next block of task t,

$$r_l^t = \text{Conv}(r_{l-1}^1 \oplus r_{l-1}^2 \oplus \cdots \oplus r_{l-1}^t) \tag{11}$$

However, this architecture underperforms NE approaches such as DER [31]. In this work, we revisit the question in the context of transformer models, which have stronger ability to transfer information across the features of a network layer, via attention mechanisms. This has been used to integrate information across data streams produced by either different image regions [4, 5, 7] or different perceptual modalities [21]. However, as we will next see, cross-task connections are not trivial to implement for this model.

Independent Attention Model: We leverage the Vision Transformer (ViT) [7] as backbone used to implement all task expert branches. Under the ViT architecture [26], block f_l of layer l implements a sequence of two operations

$$s_l = r_{l-1} + \text{MHSA}_l(\text{LN}(r_{l-1})) \tag{12}$$
$$r_l = s_l + \text{MLP}_l(\text{LN}(s_l)) \tag{13}$$

where MHSA is a multi-head self-attention block, MLP a multi-layer perceptron, and LN a layer normalization. Under the NE strategy, a transformer is added independently per task. Since, as illustrated at the bottom of Figure 2, this has no connections between different task branches, we denote it as the *independent attention* (IA) model. As shown in the right part of Figure 1, IA does not achieve high CIL accuracy unless each task expert is a computationally heavy multi-headed transformer. We seek to introduce the inter-task connections of (10) to improve the performance of the simplest models, namely that with a single head per task.

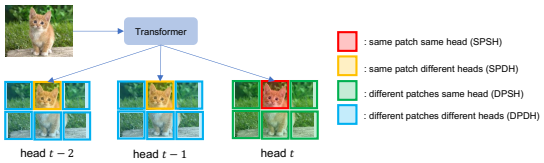


Figure 3. Four different types of tokens in STA when computing the output of top-middle patch in head t (token in red frame)

Spatial Task-Attention Model: The straightforward generalization of (11) to the transformer model is to implement a joint attention block across tokens of different tasks. This is illustrated in the middle of Figure 2 and denoted as *Spatial-Task Attention* (STA). Omitting the block index l for simplicity, (12) is replaced by

$$s_p^t = r_p^t + \text{SA}(\text{LN}(\{r_p^0\}_{p=0}^{P-1}, \{r_p^1\}_{p=0}^{P-1}, \ldots, \{r_p^{t-1}\}_{p=0}^{P-1}))$$

where p is a patch index, P the total number of patches, and t the task branch. The MLP of (13) is maintained.

Our experiments show that this approach is not effective. Unlike most other uses of transformers, say multi-modal models like CLIP [21], the different branches of the CIL network (task experts) process the *same input* image. As illustrated in Figure 3, the image to classify is fed to *all* experts. Hence, the cross-task attention connects different projections of the *same* patches. Without lack of generality, the figure assumes each task expert is implemented with a single transformer head. Consider the processing of the patch in red by expert t and note that the yellow patches processed by the prior task experts all refer to the same image region. In result, the dot-products between the projections of red and yellow patches dominate the attention matrix.

As defined in the figure, there are four types of attention dot-products: SPSH (red patch with itself), SPDH (red-yellow), DPSH (red-green) and DPDH (red-blue) . The left of figure 4 compares the percentage of attention of each of these types of the STA model to those of the IA model, which only has SPSH and DPSH connections. Connections to the same patch (either red-red or red-yellow) account for 25.8% of the STA attention strength, as opposed to only 12.7% for the IA model. Furthermore, because STA also includes the DPDH connections to the projections of all blue patches of Figure 3 (rather than just the DPSH of those in green), the spatial attention between the red and remaining patches is highly fragmented, with many small entries in the attention matrix. Even though STA includes vastly more connections to different patches, these only amount to 74.2% of the total attention, as opposed to 87.3% for the IA model. Hence, the amount of attention per connection is much smaller for STA, i.e. spatial attention is much more fragmented. This leads to a substantial degradation of accuracy for STA, which as shown on the right of Figure 4 is even lower than that of IA. The supplementary presents a more extensive discussion of these issues.

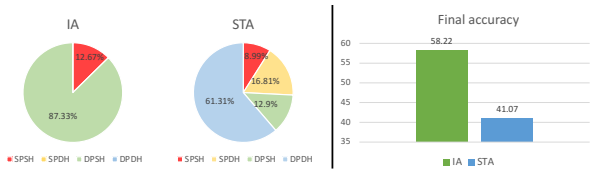


Figure 4. Left: Distribution of attention by the four groups of Figure 3, for IA and STA models. Right: Final accuracy of IA and STA.

DNE Model: To avoid this problem, the proposed DNE model decouples attention into 1) spatial attention, which continues to be performed within each task solely, using (12) (as in the IA model), and 2) cross-task attention (CTA), which is implemented at the level of the MLP block of (13). As shown at the bottom of figure 2, the standard MLP mixes the features produced, for each patch, by the different heads of *each* task expert, processing features of different tasks independently. This is implemented as

$$r_p^i = s_p^i + \text{FC}(\text{LN}(o_p^i)) \tag{14}$$
$$o_p^i = \text{GELU}(\text{FC}(\text{LN}(s_p^i))) \tag{15}$$

where $p \in \{0, 1, \ldots, P-1\}$ is the patch index, $i \in \{1, 2, \ldots, t\}$ the task index, FC is a fully connected linear layer and GELU the Gaussian Error Linear Unit, and o_p^i is the intermediate output for p-th path of task i features.

We propose to simply complement this by the yellow CTA connections of the top model of Figure 2. This transforms the MLP into a *task attention block* (TAB). Assume that the expert of task t has H_t spatial attention heads. The input of the TA block is

$$s_p^t = s_p^{t,1} \oplus s_p^{t,2} \oplus \cdots \oplus s_p^{t,H_t} \tag{16}$$

where $s_p^{t,i}$ is the tensor output for patch p of spatial attention head i of task expert t. DNE does not change the outputs $r^1, \ldots, r^{t-1}$ nor the intermediate responses $o^1, \ldots, o^{t-1}$ of the TA blocks of the experts previously learned, which are frozen at step t. The only operation used to implement (10) is, for the task t expert, (14)-(15) are replaced by

$$r_p^t = s_p^t + \text{TA}(\text{LN}(o_p^1, o_p^2, \ldots, o_p^t)) \tag{17}$$
$$o_p^t = \text{GELU}(\text{TA}(\text{LN}(s_p^1, s_p^2, \ldots, s_p^t))), \tag{18}$$

where TA is the *task attention* operation.

Figure 5 illustrates how the proposed TA block fuses features of different heads. Consider (18). TA is implemented by an attention block that recombines the spatial attention features $\{s_p^1, \ldots, s_p^{t-1}\}$ generated by previous task experts for patch p with those (s_p^t) of the current expert. For this, the features s_p^t of task t form a *query matrix*

$$Q_p = [Q_p^{t,1}, Q_p^{t,2}, \ldots, Q_p^{t,H_t}] \in \mathbb{R}^{D \times H_t}, \text{where}$$
$$Q_p^{t,i} = W_q \text{LN}(s_p^{t,i}) \in \mathbb{R}^D \tag{19}$$

is the query vector for the i-th head, p-th patch in s^t, and $W_q \in \mathbb{R}^{D \times D}$ is a learned matrix. A *key matrix*

$$K_p = [K_p^{1,1}, \ldots, K_p^{1,H_1}, K_p^{2,1}, \ldots, K_p^{t,H_t}] \in \mathbb{R}^{D \times H}$$
$$K_p^{i,j} = W_k \text{LN}(s_p^{i,j}) \in \mathbb{R}^D \tag{20}$$

where $K_p^{i,j}$ is the key vector for the j-th head p-th patch in s^i, $W_k \in \mathbb{R}^{D \times D}$ a learned matrix, and $H = \sum_{k=1}^t H_k$ the total number of heads, is then created from the features produced, for patch p, by *all* task experts. The similarity between query and key features is then captured by the dot-product matrix $C_p = Q_p^T K_p \in \mathbb{R}^{H_t \times H}$, which is normalized, with a per-row softmax, to obtain the attention weight matrix A_p, of rows

$$A_p^i = \text{SoftMax}(C_p^i / \sqrt{D}) \in \mathbb{R}^H \tag{21}$$

where C_p^i is the i-th row of C_p. Row A_p^i contains the set of H weights that determine the relevance of the features generated by each of the spatial attention heads in the model to the features computed by the i^{th} head of task t.

The *value matrix* V_p is similar to the *key matrix*,

$$V_p = [V_p^{1,1}, \ldots, V_p^{1,H_1}, V_p^{2,1}, \ldots, V_p^{t,H_t}] \in \mathbb{R}^{D' \times H}$$
$$V_p^{i,j} = W_v^{i,j} \text{LN}(s_p^{i,j}) \in \mathbb{R}^{D'} \tag{22}$$

where $V_p^{i,j}$ is the value vector for the j-th head, p^{th} patch in s^i, $D' = \gamma D$ the dimension of intermediate heads with an expansion factor γ and $W_v^{i,j} \in \mathbb{R}^{D' \times D}$ a learned matrix. Note that, for the *query* and *key*, the matrices W_q and W_k are shared by all heads. However, a separate $W_v^{i,j}$ is used per head for the *value matrix*. This is because the value vectors are the components of the TA block output. Since different heads encode knowledge of different tasks, the matrix W_v can be seen as translating knowledge across task domains. This benefits from the added flexibility of a matrix per task.

Finally, the i-th head of output o_p^t is

$$o_p^{t,i} = \text{GELU}(\lambda_i \sum_{j=1}^{H} A_p^{i,j} V_p^j) \in \mathbb{R}^{D'} \tag{23}$$

where $A_p^{i,j}$ is the j-th element of A_p^i, V_p^j the j-th column of V_p, and λ_i a learned scalar. The procedure used to implement (17) is identical to (19)-(23), but the dimension of each head changes from D' to D. Note that the entries $A_p^{i,j}$ measure the similarity between the query $Q_p^{t,i}$ generated by the i^{th} head of task expert t for patch p and the keys K_p^j generated for the patch by each of the $H = \sum_{k=1}^t H_k$ heads of all task experts, as illustrated by the yellow connections of Figure 2. If $A_p^{ij} = 1, \forall i, j$, the TA block simplifies to a generalization of the MLP of (13), whose linear layers implement projections from all tasks to the current one.

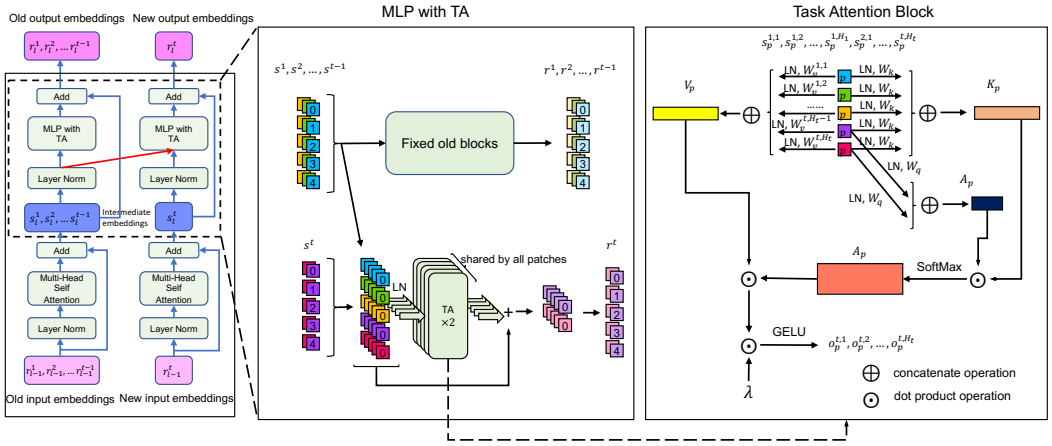


Figure 5. Illustration of the Cross-Task Attention. Left: Overview of the transformer block. Middle: MLP layer with Task Attention Block. Right: Architecture of Task Attention Block. Layer index l is omitted in the middle and right for simplicity. Patch embedding is divided into multiple heads which belong to different tasks. Heads of old tasks (old heads) are handled with fixed old blocks. Correlations between new heads and old heads (cross-task attention) and new heads (self attention) are computed through key matrix K_p, query matrix Q_p and their softmaxed dot product, the attention A_p. Each input head is projected with an individual value matrix $W_v^{i,j}$ and weighted summed by A_p to generate the output. λ is the scaling factor.

Training Objective: The training objective of DNE is the weighted average of three losses. Task t is trained with 1) a cross entropy classification loss $\mathcal{L}_{ce}$ defined over the class set of all tasks, applied at the output of the classifier g of (6), 2) a task expertise loss $\mathcal{L}_{te}$ that encourages task expert t to learn features that are more informative of the new task, and 3) a distillation loss $\mathcal{L}_{dis}$ that assures that the expert has good performance on the classes of the previous tasks. The task expertise loss was first introduced by DER [31]. It treats all previous classes as one ($\mathcal{Y}' = \bigcup_{i=1}^{t-1} \mathcal{Y}_i$) and conducts a $|\mathcal{Y}_t| + 1$-way classification. The distillation loss is a KL divergence between the common outputs of g after tasks $t-1$ and t. This is implemented by applying a softmax to the logits of classes in $\mathcal{Y}' = \bigcup_{i=1}^{t-1} \mathcal{Y}_i$ of g^{t-1} and g^t, and computing the KL divergence between the two distributions. Minimizing this KL divergence guarantees that task expert t performs similarly to task expert $t-1$ on the classes of all previous tasks. Standard procedures [13] are used to account for the data imbalance between the current task and memory buffer data.

Discussion: DNE leverages the fact that the MLP plays a critical role in the creation of new features by the transformer. Like any 1×1 convolution operator, its main function is to perform the feature mixing that transforms features of lower semantic abstraction at the bottom of the network into the features of high-level semantics at the top. Since DNE aims to *reuse* existing features and *combine* them with the information of the new task to *create* new features that account for the information not already captured by the CIL model, the MLP block is naturally suited to implement CTA. The proposed TAB simply extends feature mixing across tasks. Hence, the DNE model can reuse "old" knowledge to solve new tasks. Rather than having to relearn all features by itself, task t expert inherits the features already computed by the experts of the previous tasks.

This is similar to distillation approaches to CIL, which use the features produced by the previous models to regularize the features of task t. However, in distillation, the model is replicated and produces a new feature vector that is constrained to be similar to that of the existing model. In DNE, because the new task expert *reuses* the features computed by the previous experts *throughout the network*, it can be a small network. In result, the model is mostly frozen and only a small branch added per task, enabling a better trade-off between size and accuracy. This also provides DNE with a better trade-off between accuracy and complexity than the IA model of NE, which repeats the full network per task.

On the other hand, by implementing spatial attention with (12) and CTA with (17)-(18), DNE is immune to the fragmentation of attention of the STA model. Note that, with respect to Figure 3, spatial attention is implemented exactly as in the IA model: it relies uniquely on the dot-products of red and green patches. CTA is then performed, using (17)-(18), once the information fusion of spatial attention has been accomplished. At this point, the feature projections of the yellow patches already account for all information in the blue patches and attention only relies on dot-products between red and yellow patches. This prevents the fragmentation of attention that plagues STA.

Computation: Under the IA model of NE, the spatial attention computation of (12) has complexity $O(P^2D^2)$ per attention head, and the MLP of (13) computation $O(H^2D^2)$ per patch, where D is the patch token dimension and H the number of heads. A CIL model of T task experts and H attention heads per expert has computation $O(THP^2D^2 + TPH^2D^2) = O(THPD^2(P+H))$. Under DNE, the complexity of spatial attention is the same, but the current task

expert queries the previous task experts for the features generated per patch, using feature similarity to determine the relevance of the old knowledge to the new task. This operation has complexity $O(T^2H^2PD^2)$, for a total complexity of $O(THP^2D^2 + T^2H^2PD^2) = O(THPD^2(P+TH))$. Hence, in principle, DNE has more computation. However, the reuse of features allows a small number of heads H per task. In our implementation, we use $H = 1$, which is shown to suffice for good accuracy in the next section. This makes the effective computation of DNE $O(TPD^2(P+T))$. Hence the compute ratio between DNE and DE is $(P+T)/H(P+H)$, and DNE is more efficient if $T < H^2 + (H-1)P$. For the standard configuration of the ViT transformer ($H = 12, P = 196$), this bound is $T < 2,300$.

4. Experiments

In this section, we discuss various experiments conducted to evaluate the CIL performance of DNE.

Experimental Setup: Various experiments were performed with the following set-up.

Benchmarks. All experiments used CIFAR100 [17] or ImageNet100 [6][1], a first task with 50 classes and N_s classes per subsequent task. We denote N_s as the step size.

Methods. **DNE** was compared to multiple baselines. **Joint** trains all classes simultaneously. It is not a CIL method but an upper bound. **Dytox** [9] is a transformer based method, using a single network with trained task tokens to generate different feature vectors per task. **DER** [31] is a NE method, learning a sub-network per task and concatenating the features of all sub-networks for classification. **FOSTER** [27] is similar to DER, adding a network per task. However, the new and old networks are distilled into one compressed network to keep the total model size fixed. **iCaRL** [24] and **PODNet** [8] are distillation based methods. They introduce constraints on the logits of pooled intermediate features of the old and new networks.

Evaluation metrics. Let M be the number of tasks and A_i the average accuracy of all known classes after learning task i. Performance is measured by *Last Accuracy* $LA = A_M$ and *Average Incremental Accuracy* $AA = \frac{1}{M}\sum_{i=1}^{M} A_i$. To eliminate the effects of backbones (e.g. DER uses ResNet18 [11] and Dytox a transformer), we also consider the *Difference* of LA between joint and CIL model, $D = A_{M,joint} - A_{M,model}$. Finally, methods are compared by *floating point operations per second (FLOPs)* F.

Implementation details. DNE and Dytox use a 6-layer Vision Transformer [7] backbone. Patch size is 4 (16) on CIFAR100 (ImageNet100). DNE learns a 12-head transformer in the first task and adds $k \in \{1, 2, 4\}$ heads per subsequent task. DER, FOSTER, iCaRL and PODNet use a modified ResNet18 [31] (standard ResNet18 [11]) backbone on CIFAR100 (ImageNet100). A buffer memory $\mathcal{M}$ of $2,000$ examples is used on both datasets. We first train the backbone and classifier with all the data from the current dataset and memory. Similar to [31], we then build a class-balanced dataset by subsampling the current dataset to tune the classifier. More details are given in the supplementary.

[1] All datasets used in the paper were solely downloaded by the university.

Method	CIFAR100, N_s=10				ImageNet100, N_s=10			
	$LA\uparrow$	$AA\uparrow$	$D\downarrow$	$F\downarrow$	$LA\uparrow$	$AA\uparrow$	$D\downarrow$	$F\downarrow$
Joint (Transf.)	76.12	-	0	1.38G	79.12	-	0	1.38G
Joint (ResNet18)	80.41	-	0	1.12G	81.20	-	0	1.12G
iCaRL	44.72	59.32	31.40	1.12G	42.84	55.65	38.36	1.12G
PODNet	52.46	66.41	27.95	1.12G	63.46	73.57	17.74	1.12G
DER	63.78	71.69	16.63	6.68G	70.40	76.90	10.80	6.68G
FOSTER	63.31	72.20	17.10	1.12G	67.68	75.85	13.52	1.12G
Dytox	64.06	71.55	12.06	1.38G	68.84	75.54	10.28	1.38G
DNE-1head	68.04	73.68	8.08	2.68G	72.30	78.09	6.82	2.68G
DNE-2heads	69.73	74.61	6.39	3.10G	**73.64**	**78.88**	**5.48**	3.10G
DNE-4heads	**70.04**	**74.86**	**6.08**	4.02G	73.58	78.56	5.54	4.02G

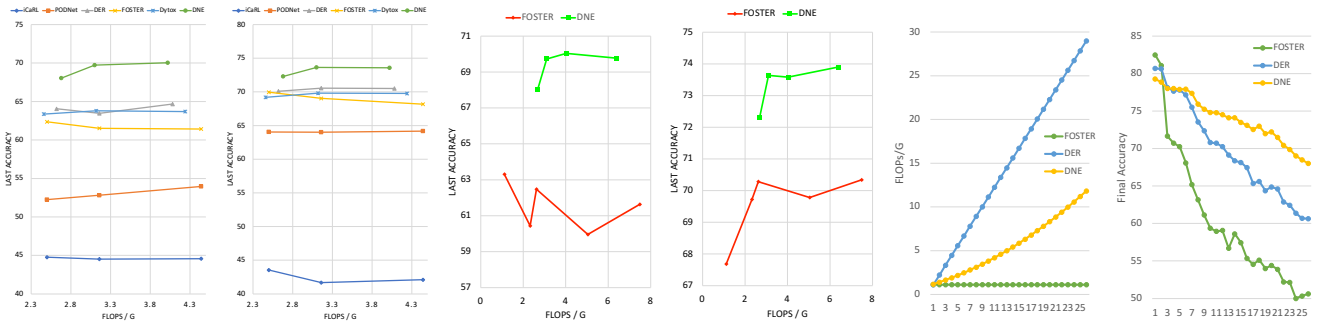
Table 1. Comparison of CIL approaches on CIFAR100 and ImageNet100, for $N_s = 10$.

Method	CIFAR100, N_s=5				CIFAR100, N_s=25			
	$LA\uparrow$	$AA\uparrow$	$D\downarrow$	$F\downarrow$	$LA\uparrow$	$AA\uparrow$	$D\downarrow$	$F\downarrow$
Joint(Transf.)	76.12	-	0	1.38G	76.12	-	0	1.38G
Joint(ResNet18)	80.41	-	0	1.12G	80.41	-	0	1.12G
iCaRL	42.89	55.23	33.23	1.12G	53.79	66.80	26.62	1.12G
PODNet	48.18	60.69	27.94	1.12G	61.54	71.45	18.87	1.12G
DER	60.73	70.42	15.39	12.19G	69.08	74.82	11.33	3.32G
FOSTER	48.18	60.69	27.94	1.12G	70.14	**76.33**	10.27	1.12G
Dytox	58.59	68.31	17.53	1.38G	69.29	74.10	6.83	1.38G
DNE-1head	69.10	74.03	7.02	5.71G	67.61	73.27	8.51	1.19G
DNE-2heads	**69.72**	**74.27**	**6.40**	7.39G	68.99	73.91	7.13	1.27G
DNE-4heads	69.43	74.20	6.69	10.75G	**71.47**	75.70	**4.65**	1.45G

Table 2. Comparison of CIL methods on CIFAR100 for different step sizes.

Accuracy Comparison to SOTA: Table 1 compares DNE to SOTA methods, for $N_s = 10$. On both datasets, DNE outperforms all methods by a clear margin. On CIFAR100, even with only 1 head per task expert, DNE reaches an LA of 68.04%, which is 3.98% higher than Dytox, the current SOTA. Regarding AA, DNE-1head is 1.48% higher than the SOTA, FOSTER. On ImageNet100, DNE-1head outperforms the SOTA, DER, by 1.9% in LA and 1.19% in AA. With more heads, DNE performs even better. Between 1 and 4 heads, LA increases by 2% (1.28%) on CIFAR100 (ImageNet100). On ImageNet100, performance saturates with 2 heads per task, due to the increased image resolution and number of patches, which allow the learning of more discriminant features and more efficient knowledge sharing between tasks.

DNE gains are even more significant for the difference D to joint model. While the transformer of DNE has 2-4% weaker joint accuracy than the ResNet18 of DER or FOSTER, DNE significantly outperforms the latter, because its difference D is 2 to 3 times smaller. When compared to the SOTA Dytox model, DNE reduces D by as much as 50%. These results illustrate the effectiveness of the TA



(a) (b) (c) (d) (e) (f)

Figure 6. Accuarcy-Scale Trade-off: (a)(b)FLOPs-Accuracy Trade-off, on CIFAR100/ImageNet100 $N_s = 10$, (c)(d) Comparison to FOSTER with different backbones, on CIFAR100/ImageNet100 $N_s = 10$, (e)(f) FLOPs/Acc vs task step, on CIFAR100 $N_s = 2$

module of Figure 5 for sharing information across tasks.

These conclusions hold qualitatively across step sizes N_s. Table 2 shows results for N_s as different as 5 and 25. The advantages of DNE are larger for smaller step sizes. When $N_s = 5$, DNE-2heads outperforms the SOTA, DER, by 8.99% in LA. For $N_s = 25$, the LA of DNE-4heads is 1.33% higher than that of the SOTA, FOSTER. This is most likely because DNE freezes the old network - only the added task experts can learn about new tasks. Hence, the number of extra heads determines the model capacity for learning. When N_s is small, 1head is enough to handle the new task classes. However, as the number of classes per task (N_s) grows, larger capacity is eventually needed. This explains the increase in accuracy from the 1head to the 4head model when $N_s = 25$. However, as the number of heads increases, DNE becomes more similar to DER or FOSTER, which uses a full network per task. Hence, the benefits of its feature sharing across tasks are smaller.

Accuracy-Scale Trade-off: A criticism of NE is the network growth with the number of tasks. However, growth is inevitable in CIL, since the ever-growing task sequence will eventually exceed the capacity of any fixed model. The real question is not *whether the network needs to grow* but *how fast it needs to grow*. For insight on this, we compared the accuracy-scale trade-off of the different methods.

We started by changing the dimension of the feature vector of all methods to match their model sizes with those of DNE for $k \in \{1, 2, 4\}$ extra heads. Experiments were conducted on CIFAR100 and ImageNet100 with $N_s = 10$. Detailed configurations are listed in the supplementary materials. Figure 6(a)(b) compares the accuracy-scale trade-off of the different methods. It is clear that the accuracy of single model methods (iCaRL, PODNet, FOSTER, Dytox) fails to improve with model size. This is unsurprising, since these methods use the same network to solve all tasks. Since the network must have enough capacity to solve all tasks, there is little benefit to a wider feature vector or a deeper architecture. This is unlike DNE, whose performance can increase with model size and FLOPs, by leveraging larger task experts as N_s increases. Overall, DNE has the best trade-off on both datasets.

A second comparison was performed by changing the network backbone. For simplicity, we limited this comparison to FOSTER. Figure 6(c)(d) compares results of FOSTER with different backbones, from ResNet18 to ResNet152, and DNE with numbers of heads $k \in \{1, 2, 4, 7\}$. Clearly, FOSTER never approaches the trade-off of DNE.

Finally, we consider the growth rate of DNE and other methods. We set $N_s = 2$ on CIFAR100, which leads to an experiment with 26 tasks, the maximum task number we can reach. The FLOPs of DNE, DER (NE based method) and FOSTER (Distillation based method), as well as their final accuracy, for different task steps are illustrated in Figure 6(e)(f). FOSTER, which maintains a roughly constant FLOPs, suffers form severe catastrophic forgetting as more tasks are added. DER performs much better than FOSTER, but consumes 26 times the number of FLOPs. As discussed above, the FLOPs of DNE are quadratic on the number of tasks. However, as the dense connections allow feature reuse and a single head per task expert, the actual FLOP growth is relatively slow. Hence, DNE has much lower FLOPs than DER (which has linear FLOPs rate with number of tasks) over a large range of task setps. On the other hand, the performance of DNE is also much better than those of DER and FOSTER. In summary, DNE is both simpler and better than the previous approaches.

5. Conclusion

In this work, we have pointed out that NE-based CIL methods have unsustainable model growth for most practical applications and reformulated the NE problem, to consider the trade-off between accuracy and model size. We then proposed the DNE approach to address this trade-off, and introduced an implementation of DNE that relies on a new TAB to perform cross-task attention. Experiments have shown that DNE outperforms all previous CIL methods both in terms of accuracy and of the trade-off between accuracy and model size.

Acknowledgment: This work was partially funded by NSF award IIS-2041009, and a gift from Qualcomm. We also acknowledge the Nautilus platform, used for the experiments.

References

[1] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3366–3375, 2017. 1, 2

[2] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1728–1738, 2021. 2

[3] Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930, 2020. 1

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 4

[5] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. *Advances in Neural Information Processing Systems*, 34:17864–17875, 2021. 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 7

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 4, 7

[8] Arthur Douillard, Matthieu Cord, Charles Ollion, Thomas Robert, and Eduardo Valle. Podnet: Pooled outputs distillation for small-tasks incremental learning. In *European Conference on Computer Vision*, pages 86–102. Springer, 2020. 1, 2, 7

[9] Arthur Douillard, Alexandre Ramé, Guillaume Couairon, and Matthieu Cord. Dytox: Transformers for continual learning with dynamic token expansion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9285–9295, 2022. 1, 2, 3, 7

[10] Siavash Golkar, Michael Kagan, and Kyunghyun Cho. Continual learning via neural pruning. *arXiv preprint arXiv:1903.04476*, 2019. 1, 4

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7

[12] Saihui Hou, Xinyu Pan, Chen Change Loy, Zilei Wang, and Dahua Lin. Learning a unified classifier incrementally via rebalancing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 831–839, 2019. 1, 2

[13] Bingyi Kang, Saining Xie, Marcus Rohrbach, Zhicheng Yan, Albert Gordo, Jiashi Feng, and Yannis Kalantidis. Decoupling representation and classifier for long-tailed recognition. *arXiv preprint arXiv:1910.09217*, 2019. 6

[14] Minsoo Kang, Jaeyoo Park, and Bohyung Han. Class-incremental learning by knowledge distillation with adaptive feature consolidation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16071–16080, 2022. 1, 2

[15] 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. 1, 2

[16] Yajing Kong, Liu Liu, Zhen Wang, and Dacheng Tao. Balancing stability and plasticity through advanced null space in continual learning. *arXiv preprint arXiv:2207.12061*, 2022. 1, 2

[17] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7

[18] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2017. 1, 2

[19] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017. 1, 2

[20] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018. 1, 4

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 2, 4

[22] Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019. 1, 2

[23] Amal Rannen, Rahaf Aljundi, Matthew B Blaschko, and Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1320–1328, 2017. 1, 2

[24] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2001–2010, 2017. 1, 2, 7

[25] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016. 1, 4

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[27] Fu-Yun Wang, Da-Wei Zhou, Han-Jia Ye, and De-Chuan Zhan. Foster: Feature boosting and compression for class-incremental learning. *arXiv preprint arXiv:2204.04662*, 2022. 1, 2, 3, 7

[28] Shipeng Wang, Xiaorong Li, Jian Sun, and Zongben Xu. Training networks in null space of feature covariance for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 184–193, 2021. 1, 2

[29] Zhen Wang, Liu Liu, Yiqun Duan, Yajing Kong, and Dacheng Tao. Continual learning with lifelong vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 171–181, 2022. 1, 2

[30] Tz-Ying Wu, Gurumurthy Swaminathan, Zhizhong Li, Avinash Ravichandran, Nuno Vasconcelos, Rahul Bhotika, and Stefano Soatto. Class-incremental learning with strong pre-trained models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2022. 1, 2, 3

[31] Shipeng Yan, Jiangwei Xie, and Xuming He. Der: Dynamically expandable representation for class incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3014–3023, 2021. 1, 2, 3, 4, 6, 7

[32] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017. 1, 2