Ternary Feature Masks: zero-forgetting for task-incremental learning

Marc Masana
Computer Vision Center
Barcelona, Spain
mmasana@cvc.uab.es

Tinne Tuytelaars
ESAT-PSI
KU Leuven, Belgium
tinne.tuytelaars@esat.kuleuven.be

Joost van de Weijer
Computer Vision Center
Barcelona, Spain
joost@cvc.uab.es

Abstract

We propose an approach without any forgetting to continual learning for the task-aware regime, where at inference the task-label is known. By using ternary masks we can upgrade a model to new tasks, reusing knowledge from previous tasks while not forgetting anything about them. Using masks prevents both catastrophic forgetting and backward transfer. We argue – and show experimentally – that avoiding the former largely compensates for the lack of the latter, which is rarely observed in practice. In contrast to earlier works, our masks are applied to the features (activations) of each layer instead of the weights. This considerably reduces the number of mask parameters for each new task; with more than three orders of magnitude for most networks. The encoding of the ternary masks into two bits per feature creates very little overhead to the network, avoiding scalability issues. To allow already learned features to adapt to the current task without changing the behavior of these features for previous tasks, we introduce task-specific feature normalization. Extensive experiments on several finegrained datasets and ImageNet show that our method outperforms current state-of-the-art while reducing memory overhead in comparison to weight-based approaches.

1. Introduction

Fine-tuning has been established as the most common method to use when learning a new task on top of an already learned one. This works well if you no longer require the system to perform the previous task. However, in many real-world situations one is interested in learning consecutive tasks, all of which, in the end, the system should be able to perform all. This is the setting studied in lifelong learning, also referred to as sequential, incremental or continual learning. In this setting, the popular approach of fine-tuning suffers from catastrophic forgetting [10, 11, 20, 28, 30]: all network capabilities are used for learning the new task, which leads to forgetting of the previous ones.

A popular strategy to avoid this is to use importance-weight loss proxies or regularizers [1, 14, 18]. These approaches compute an importance score for each of the weight parameters of the model based on previous tasks and use this to decide which weights can be modified for the current task. A drawback of these methods is that they store an extra variable (the importance score) for each weight. This leads to an overhead of a float per weight parameter, i.e. double the number of parameters which have to be stored. Other methods work with a binary mask to select part of the model for each task [25, 26]. This leads to an overhead of one bit per task per weight parameter. Finally, some methods directly make a copy of the network [39] or rely on the storage of exemplars [4, 24, 36, 37], which again increases memory consumption and renders these methods unsuitable when privacy requirements forbid storing of data.

In this paper, we advocate computing a mask at the level of the features instead of at the level of the weights. We need the mask to be ternary, i.e. adding a third state allowing features to be used during the forward pass while being masked during the backward pass. This allows to reuse representations from previous tasks without introducing forgetting and drastically reduces the number of extra parameters that need to be stored. As an example, the popular AlexNet architecture [15] has around 60m weights, while having less than 10k features. One earlier method that builds on this idea is HAT [41], which stores an attention value for each feature for each task. Recently, SSL [3] also brings attention to the activation neurons by promoting sparsity with losses inspired by lateral inhibition in the mammalian brain. Those two recent works stress the importance of focusing on features instead of weights, not only because of the reduction in memory overhead, but also because they allow for better performance and less forgetting. However, both methods still allow some forgetting as new tasks are learned.

Over time, the forgetting typically increases with the number of tasks [1, 14, 18, 24, 36, 41]. However, for many practical systems, it is undesirable if the accuracy of the system deteriorates over time, while the system learns new tasks. Moreover, under these settings, the user typically has no control on the amount of forgetting, i.e. there are

no guarantees on the performance of the system after new tasks have been added. Even worse: to the user, it is unknown how much the system has actually forgotten or how well the system still performs on older tasks.

For these reasons, some works have studied continual learning systems without any forgetting at all. Currently, apart from the methods that make copies of the network for each task [39], mask-based approaches are the only ones that guarantee no-forgetting [25, 26]. Indeed, all methods that allow backward propagation into the parameters of previously learned tasks have no control on the amount of forgetting. Not updating the weights used for previous tasks using a binary mask prevents any forgetting. In the case of recent approaches PackNet [26] and PiggyBack [25], this is enforced by binary-masking weights or learning masks that will be binarized after the task is learned. However, both these non-forgetting methods mask weights and therefore have a larger memory overhead than methods based on feature-masks. Another drawback of [25] is that it requires a backbone network as a starting point.

We propose Ternary Feature Masks (TFM), a method for continual learning which does not suffer from any forgetting. Due to the nature of our proposed mask-based approach, we will only focus on evaluation on task-aware experimental setups. Instead of applying masks to all weights in the network, we propose to move the masks to the feature level, thereby significantly reducing memory overhead. Our initial method requires only a 2-bit mask value for each activation for each task. In addition, we introduce a task-dependent feature normalization (FN). This allows to adjust previously learned features to be of more optimal use for later tasks, without changing the performance or weights assigned to previous tasks. This introduces a further memory overhead of storing two floats more per activation per task. Nevertheless, this method still has a significantly lower memory overhead than any method which stores additional parameters per weight [1, 14, 18, 23, 25, 26, 45].

Mask-based approaches have shown to be better at overcoming catastrophic forgetting on task-incremental learning (task-IL) [5, 26, 41]. However, unlike most distillation and model-based approaches, they make use of some overhead memory during inference. In Fig. 1 we visualize the absolute memory overhead used by some approaches on an ImageNet scenario with 10 tasks. Considering that the network used is around 220Mb, we can observe that the approaches that focus on using feature-masks (HAT, TFM) have a negligible overhead in comparison to the weight-masked approaches (PackNet). It should also be noticed that mask-based approaches keep the same overhead during training, while distillation and model-based approaches usually duplicate the network size at least. In conclusion, our method has similar memory usage as HAT, however, we outperform this method on all proposed experiments, and our method

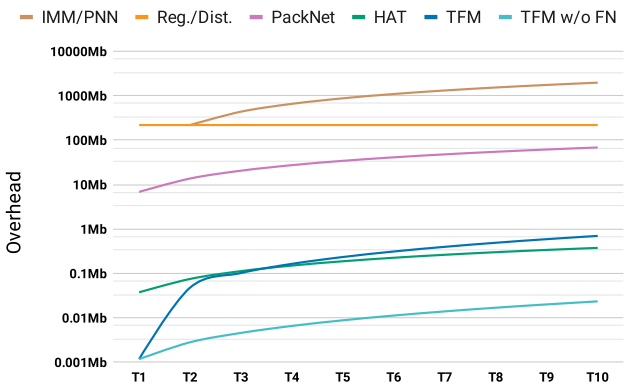

Figure 1. Log scale overhead growth for ImageNet on AlexNet. Best viewed in color.

is significantly more memory efficient than PackNet whose performance we either match or outperform.

2. Related work

Continual learning in the proposed task-IL setup has been addressed in multiple prior works [5, 19, 33, 34]. A large part of the approaches use regularization-based techniques to reduce catastrophic forgetting without having to store raw input. They can be divided into two main families: distillation approaches and model-based approaches.

Distillation approaches use teacher-student setups that aim at preserving the output of the teacher model on the new data distribution. LwF proposes to use the knowledge distillation loss [22] to preserve the performance of previous tasks. However, if the data distribution of the new task is very different from the previous tasks, performance drops drastically [2]. In order to solve that, iCaRL [36] stores a subset of each tasks' data as exemplars; while EBLL [35] solves the issue by learning undercomplete autoencoders for each task. LFL is also similar to LwF, preserving the previous tasks' performance by penalizing changes on the shared representation [13]. Expert Gate [2] learns a model for each task and an autoencoder gate which will choose the model to be used. Recently, [17, 46] propose to learn new classes separately and then learn a final model with multiple distillation and extra unlabelled data. However, most of these methods need a pre-processing step before each task. Furthermore, a main issue is also the scalability when learning many tasks, since the described methods have to store data, autoencoders, or larger models for each new task.

Most model-based approaches, when learning a new task, apply a smooth penalty for changing weights, proportional to their importance for previous tasks [1, 14, 18, 23, 45]. One of the main issues is they might over or underestimate the importance of those weights. The main difference among those methods is how that importance is cal-

culated. In EWC an approximation of the diagonal of the Fisher Information Matrix (FIM) is used [14]. R-EWC proposes a rotation of the weight space to get a better approximation of the FIM [23]. In IMM the moments of the posterior distribution are matched incrementally [18]. SI computes the importance weights in an online fashion by storing how much the loss would change for each parameter over the training [45]. MAS computes the weight importance in an online unsupervised way, connecting their approach with Hebbian learning [1].

Some more works use other underlying methods. PNN add lateral connections at each layer of the network to a duplicate of that layer [39]. Then, the new column learns the new task while the old one keeps the weights fixed, meaning that resources are duplicated each time a task is added. This approach leads to zero-forgetting while making the knowledge of previous tasks available during the learning of a new one through distillation. However, as each new task adds a column with the corresponding connections, the overhead scales drastically with the number of tasks. Progress and Compress expands the idea of PNN with the use of EWC but keeping the number of parameters constant [40]. They propose a two-component setup with a knowledge base and an active column that follows a similar setup as PNN. Recently, Learn to Grow allows for each layer to reuse existing weights, adapt them or grow the network [21]. In the worst case scenario, layers are added which makes the growth linear in the number of tasks. Finally, ACL combines an architecture growth with experience replay to learn task-specific and task-invariant features [8]. However, this comes at a quite large overhead per task, and we have not been able to obtain competitive results outside of their proposed small datasets (i.e. below Finetuning for Flowers 4 tasks).

Apart from the above mentioned families, some recent works use masks to directly influence or completely remove forgetting. We refer to this family of approaches as mask-based. PathNet uses evolutionary strategies to learn selective routing through the weights [9]. However, it is not end-to-end differentiable and computationally very expensive. PackNet trains with available weights, then prunes the less relevant ones and retrains with a smaller subset of them [26]. Those weights are then not available for further learning of new tasks, which quickly reduces the capacity of the network. This results in lower number of parameters being free and performance dropping quickly on longer sequences. Piggyback proposes to use a pretrained network as a backbone and then uses binary masks on the weights to create different sub-networks for each task [25]. Its main drawback is the backbone network itself, which is crucial to being able to learn each task on top of it and cannot have a too different distribution from them. Finally, HAT proposes a hard attention mechanism on the features after each layer [41]. The attention embeddings are non-binary and are learned together with each task and conditioned by the attentions of previous ones. This offers plasticity to the embeddings in order to learn them, but also allows the possibility to forget previous tasks during the back-propagation step. A zero-forgetting idea is discussed in the appendices of their manuscript with a note on binary masks, connecting the removal of plasticity to *inhibitory synapses* [29].

Table 1. Difference between number of weights and features for different common network architectures (without heads).

Network	#weights	#features
LeNet [16]	59,956	226
AlexNet [15]	54,547,712	9,344
VGGNet [42]	119,579,904	10,880
ResNet-50 [12]	19,330,304	22,720

In our approach we take the latter side of that balance, using rigid masks that reduce plasticity but also ensure non-forgetting of previous tasks. Our approach also focuses on a natural expansion of the capacity of the network, which is not addressed in HAT and most of the previous related work. Our approach uses masks on the features of the network to have a better control over which weights can be modified while learning new tasks. At the same time, the mask being ternary allows weights fixed for previous tasks to be used on new tasks without modifying those weights. This masking strategy allows the network to not forget anything from previous tasks and reduce the computational overhead in comparison to masking the weights. Our proposed method is unique in that it combines being expandable, having a low overhead cost and having no forgetting. All other methods have to choose only one of those three characteristics if any.

3. Learning without any Forgetting

Here we propose our zero-forgetting method for task-aware incremental learning. As discussed in the introduction, in order to enforce non-forgetting of previous tasks, the use of masks that create rigid states is an efficient way. Works which have addressed this problem have focused on *weight-masks* where an additional parameter is learned for each weight in the network [25, 26]. From a network overhead point of view, we argue that it is, however, better to work with *feature-masks* which learn an additional parameter for each feature in the network. In Table 1 we compare the number of weights and features in several popular networks. The table clearly shows that the overhead is significantly lower: on average weight-masks are a quadratic factor bigger than feature-masks.

First, we discuss binary masks and how those can easily encode the parts that we want to learn and the parts that we want to fix. Afterwards, we explore what happens when we want to learn more than one task and extending to ternary

masks. Finally, we explore the use of feature normalization to allow for less rigid learning of new tasks.

3.1. Binary feature masks

Using binary feature masks on neural networks means that the masked neuron will have one of two states (0 or 1). When the masks are directly multiplied by the neuron activations, the corresponding filters will be used or not (same for the backward pass, which will either be applied or not). Then, for each task we have a binary mask with the neurons that can be used. Since we pursue zero-forgetting, those masks will have to be disjoint. In Fig. 2 we show an example with two tasks where each of them is only allowed to use different neurons. A large amount of connections are completely unused, making the two sub-networks totally separable from one another.

Consider a fully-connected layer (the theory can easily be extended to convolutional layers). The output of the layer is $y = Wx$ where $y \in \mathbb{R}^{p\times 1}$, $x \in \mathbb{R}^{q\times 1}$ and $W \in \mathbb{R}^{p\times q}$. The binary feature mask for the forward pass is defined as:

$$y = (Wx) \odot m^{t,l} \tag{1}$$

where $m^{t,l} \in \mathbb{R}^{p\times 1}$ refers to the mask for task t at layer l and $\odot$ is an element-wise multiplication. Masks from different tasks are forced to select different features ($\left(m^{s,l}\right)^{\mathrm{T}} \cdot m^{t,l} = 0 \; \forall s \neq t$). The backward pass for training task t is defined as:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = (m_i^{t,l} \wedge m_j^{t,l-1}) x_j \frac{\partial \mathcal{L}}{\partial y_i} \tag{2}$$

where $\wedge$ is the AND logical operator and there are only non-zero gradients for those weights which join in a feature which is masked for task t.

This setup allows the associated weights to an activation to be either used-and-learnable, or neither. If used, they will contribute forward to the next layers (which is good, as it promotes forward transfer, i.e. sharing of knowledge from previous tasks). Yet at the same time this also implies that it will be possible to modify them (which is bad, as it introduces catastrophic forgetting on previous tasks). With only binary masks, you cannot have one without the other. Alternatively, one could also define two separate binary states: "used" and 'learnable". This has been used for a long time in deep learning by freezing weights [32]. Freezing weights is a mask-based way of switching on and off the learning of a layer. In this case, in both states the layer would contribute to the outcome of the network, but the update of the weights would only be done on those layers that are not masked. Here, we further explore this idea. We advocate that, in a sequential setup where the capacity of the network might increase when learning new tasks, the best way to mask the neurons is by having three states: "used", "learnable" and "unused". This can be achieved by using ternary masks on the neurons.

3.2. Ternary feature masks (TFM)

Being able to use the connections between the neurons of the previous tasks and the neurons of the newly added task is important to reuse the learned information and reduce the amount of capacity that needs to be added. By using a ternary mask we can define three states:

- **forward only**: features are used during the forward pass so that the learned information from previous tasks is used; but the backward pass step is removed in order to keep the weights and prevent forgetting. This state is used on features from previous tasks.
- **normal**: forward and backward passes are applied as usual to learn the task at hand. This state is used on new features created by the network expansion.
- **masked**: neither forward nor backward passes are allowed, the features do not contribute to the network inference and the weights associated to it are frozen. This state is used at test time only when evaluating an old task after a new task is added. When extending the capacity of the network, the new features will not be used when doing inference on the previous tasks since those did not exist at the moment of their training.

Similar as in the case of the binary mask we assign features to tasks with a mask $m^{t,l}$ (with l the corresponding layer). Again overlap in the selected features is not allowed. However, different than before, we now define a second mask $n^{t,l}$ per task t which is defined as:

$$n_i^{t,l} = \begin{cases} 1, & \text{if } \exists\, s \leq t \;:\; m_i^{s,l} = 1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The forward and backward pass are now given by:

$$y = (Wx) \odot n^{t,l}, \tag{4}$$

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = (n_j^{t,l-1} n_i^{t,l} - n_j^{t-1,l-1} n_i^{t-1,l}) x_j \frac{\partial \mathcal{L}}{\partial y_i}, \tag{5}$$

respectively. During the forward pass, features selected by previously learned tasks can be used in the current task. During the backward pass, we make sure that all new weights can be updated while forcing the existing ones from previous tasks to remain the same. In Fig. 3 we show an example with two tasks, where adding features to the layer allows for more connections to be used than in the binary case. The part of the mask corresponding to $n_j^{t,l-1} n_i^{t,l}$ corresponds to all available connections at task t. In a similar way, the part of the mask corresponding to $n_j^{t-1,l-1} n_i^{t-1,l}$ corresponds to all available connections at task $t-1$. Subtracting both terms allows us to mask the connections that contain the already learned content and apply backpropagation only on the new connections.

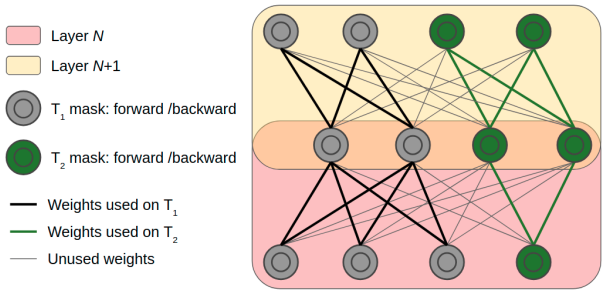

Figure 2. Binary masks encode two states: used or unused. In this case, neurons in grey are learnable for task 1 but neurons in green are not, and the opposite is true for task 2. All grey weights are unused by both tasks.

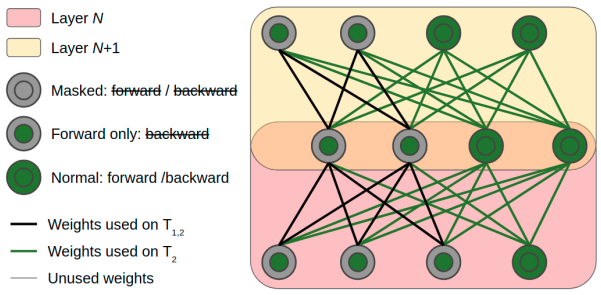

Figure 3. Ternary masks encode three states: masked (frozen), forward only or normal (forward and backward). In comparison to Fig. 2, all unused connections can now be learned without forgetting previous knowledge.

Note that this definition also allows to use the same forward and backward pass in case we would want to re-train one of the previous tasks. However, since we do not contemplate this option for our proposed setup, we can simplify equation 5 to (non-revisiting) task-IL. In this case, the forward pass remains the same as in equation 4, and the backward pass can be rewritten as:

$$\frac{\partial \mathcal{L}}{\partial W_{ij}} = (m_i^{t,l} \vee m_j^{t,l-1}) x_j \frac{\partial \mathcal{L}}{\partial y_i} \tag{6}$$

where $\vee$ is the OR logical operator which makes the mask active when either operands are active.

Since $n_i^{t,l}$ can never be 0 if one of the current or previous $m_i^{1..t,l}$ is 1, both masks $m^{t,l}$ and $n^{t,l}$ can be combined in a single ternary mask. This is because weights associated to a feature that is not used in the forward pass are never updated. With this ternary mask, the states are associated as follows: when $m_i^{t,l} = 1$ and $n_i^{t,l} = 1$ the neuron is used and learnable (normal state), when $m_i^{t,l} = 0$ and $n_i^{t,l} = 1$ the neuron is used and contributes to the forward pass but the associated weights are not updated (forward only state), and finally when $m_i^{t,l} = 0$ and $n_i^{t,l} = 0$ the neuron is unused, not taking part in the inference or the update of the network (masked state).

Allowing to use previously learned parameters in the forward pass, but only updating network parameters assigned to the current task in the backward pass is also applied in Packnet [26] and HAT [41]. However, in contrast to us, Packnet has the masks on the weights and not on the features. HAT applies a soft activation mask, which permits forgetting of previous tasks. We further distinguish from these methods by the task-specific feature normalization (discussed in the next section) which is a crucial ingredient of our method, and which allows not only to exploit previously learned features, but also to adapt them to the current task. This is not possible for neither Packnet nor HAT.

3.3. Task-specific feature normalization (FN)

Since binary or ternary masks freeze filters learned on previous tasks, those filters have no room for flexibility to small changes in the features. This means that even when being very similar to the ones needed for a new task, they tend to learn a similar version of those filters with shifted or scaled operators. This phenomenon is similar to the one observed when learning several styles for style transfer networks. A way of reusing learned filters in a more efficient way but still keep the zero-forgetting property would be to use a similar approach as *conditional instance normalization* [7], which consists in transforming a set of features x into a normalized version $\hat{x}$ depending on the task.

Let $x_{l,1..I}$ be the features of layer l, and γ_t, β_t the learnable parameters for each feature given a fixed task t. We define the task-specific feature normalization of $x_{l,i}$ as:

$$\hat{x}_{l,i} = \gamma_{t,l,i}\, x_{l,i} + \beta_{t,l,i} \tag{7}$$

where we apply a conditional normalization on the task without applying an instance normalization on the mean and standard deviation across the spatial dimensions. These parameters allow to slightly adjust the learned filters to the new tasks without modifying existing parameters (thus no forgetting happens) and with little overhead to the network capacity since the γ and β parameters are for each feature and not for all weights.

3.4. Growing Ternary Feature Masks

One of the core characteristics of our proposed method is that it can easily grow and expand the capacity of the network as is required. Given a network with L layers, any layer with the corresponding $y_{l,1..I}$ learned features can be expanded if those learned features are not enough to represent the new task. When expanding a layer by N new features, the output of the layer grows to $y_{l,1..I+N}$. That affects only the newly added forward mask values:

$$n_j^{t,l} = \begin{cases} 1, & \text{for current task, if } 1 \leq j \leq I{+}N \\ 0, & \text{for previous tasks, if } I < j \leq I{+}N \end{cases} \tag{8}$$

so that all features can be seen while learning the new task but ignored by previous tasks. Then, the backward mask:

$$m_j^{t,l} = \begin{cases} 0, & \text{for current task, if } 1 \leq j \leq I \\ 1, & \text{for current task, if } I < j \leq I{+}N \\ 0, & \text{for previous tasks, if } I < j \leq I{+}N \end{cases} \quad (9)$$

so that it only affects the new connections without modifying previous knowledge.

Training small tasks on large networks at the beginning of a continual learning setup, usually leads to overfitting or too much repetition of filters. Feature usage on the new tasks look very unbalanced in comparison to learning larger tasks [27]. We believe that learning tasks in their correct capacity and growing when more is needed is a much better approach to avoid overfitting. This observation is backed by the better results some other approaches have when pruning and retraining on smaller sub-networks than when directly pruning or learning in larger sub-networks [26, 39].

4. Experimental results

In this section we report on a range of experiments to quantify the effectiveness of our proposed approach and compare with other state-of-the-art methods and baselines. More details can be found in the Supplementary material.[1]

4.1. Experimental Setup

Datasets. We evaluate approaches on a larger lower resolution dataset (tiny ImageNet ILSVRC2012 [6]), on a large-scale dataset (ImageNet [38]) and some fine-grained classification datasets: Oxford 102 Flowers [31], CUB-200-2011 Birds [43] and Stanford Actions [44]. Statistics over those datasets are summarized in Table 2. For all experiments we take a fixed random set of 10% of images for validation. The validation set is equally distributed among the number of classes and fixed for each experiment to ensure a fair comparison. Since the test set is not labelled for ImageNet ILSVRC2012, we use the validation set for test instead.

Network architectures. For tiny ImageNet we use VGG-16, which provides high performance results [42]. Since tiny ImageNet has a low resolution, the last max-pool layer and the last three convolutional layers from the feature extractor are removed. For ImageNet and the fine-grained datasets we use AlexNet [15]. The models are trained from scratch using only samples from train. Our proposed method TFM starts with a network that is smaller than the proposed ones at each layer (reduced number of output filters). Then, it grows as explained in Sec. 3.4 as more features are added every time a new task is learned. We limit the growth of the network to the total size of the one used by all other approaches.

[1]Code: https://github.com/mmasana/TernaryFeatureMasks

Table 2. Summary of datasets used.

Dataset	#Train	#Eval	#Classes
tiny ImageNet [6]	100,000	10,000	200
ImageNet [38]	1,281,167	50,000	1000
Oxford Flowers [31]	2,040	6,149	102
CUB Birds [43]	5,994	5,794	200
Stanford Actions [44]	4,000	5,532	40

Training details. We train using backpropagation with plain Stochastic Gradient Descent following the setup of HAT [41]. With a batch size of 64, learning rate starts at 0.05, decaying by a factor of 3 when 5 consecutive epochs have no improvement on the validation loss, until either the learning rate is reduced below 10^{-4} or 200 epochs have passed. Data splits, task sequence, data loader shuffle and network initialization are fixed for all approaches given a seed. Following [5], we use dropout with $p = 0.5$.

Baselines. Finetuning uses the cross-entropy loss to learn each task as it comes, without using data from previous tasks nor avoiding catastrophic forgetting. Incremental Joint training breaks the no-revisiting data rule and learns with data from the current task as well as all the previous tasks, serving as an upper-bound to compare all approaches. Finally, we propose to use Freezing as a baseline where we learn the first task and then freeze all layers except the head for the remaining tasks.

Hyperparameters. Distillation and model-based approaches use hyperparameters to control the trade-off between forgetting and intransigence on the knowledge of previous tasks. On top of that, LwF has a temperature scaling hyperparameter for the cross-entropy loss. From the mask-based models, HAT has a trade-off hyperparameter too and a maximum for the sigmoid gate steepness. PackNet has a prune percentage of the layers.

For TFM, at each new task, several growth percentages are evaluated on the validation set without the knowledge of previous or future tasks. We pick the lowest growth rate which obtains a performance within a margin of the best performance (we set the margin to be 1.5% for tiny ImageNet and 0.1% for fine-grained). Then, we learn the task at hand on train and move to the next task. For ImageNet this scheme would be computationally demanding and we use a fixed growth schedule, starting from 55% of the weights for the first task and add 5% for all remaining tasks.

4.2. Fine-grained datasets

A common setup to evaluate task-IL over a number of learning sessions are disjoint splits (tasks) inside the same classification dataset. It should be noted that we start training from scratch resulting in lower scores than reported by papers which train from a pretrained network. However, because of the large number of classes in ImageNet (including

Table 3. Comparison with the state-of-the-art. Accuracy after learning 4 tasks on AlexNet from scratch. Number between brackets indicates forgetting.

Oxford 102 Flowers					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Finetuning	10.0 (-20.3)	5.1 (-17.1)	6.7 (-13.6)	17.3 (0.0)	9.8
Freezing	30.3 (0.0)	39.8 (0.0)	32.0 (0.0)	33.1 (0.0)	33.8
Joint	54.6 (+24.3)	58.9 (+11.5)	57.7 (+4.5)	47.0 (0.0)	54.6
EWC [14]	12.1 (-18.2)	11.6 (-38.1)	9.3 (-24.4)	25.8 (0.0)	14.7
HAT [41]	17.2 (-12.7)	19.3 (-28.5)	28.6 (+1.4)	31.6 (0.0)	24.2
PackNet [26]	32.0 (0.0)	53.7 (0.0)	43.6 (0.0)	37.9 (0.0)	41.8
TFM w/o FN	36.4 (0.0)	54.1 (0.0)	38.6 (0.0)	39.0 (0.0)	42.0
TFM	36.4 (0.0)	53.8 (0.0)	45.5 (0.0)	37.6 (0.0)	**43.3**
CUBS 200 Birds					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Finetuning	7.4 (-30.2)	2.6 (-30.0)	29.7 (-3.4)	43.1 (0.0)	20.7
Freezing	37.6 (0.0)	35.1 (0.0)	35.4 (0.0)	38.4 (0.0)	36.6
Joint	48.7 (+11.1)	52.1 (+6.0)	50.7 (+1.5)	51.9 (0.0)	50.8
EWC [14]	16.2 (-21.4)	19.0 (-21.2)	24.2 (-14.0)	41.7 (0.0)	25.3
HAT [41]	18.7 (-1.8)	19.4 (-0.4)	28.5 (-0.6)	31.2 (0.0)	24.4
PackNet [26]	35.3 (0.0)	42.8 (0.0)	44.4 (0.0)	45.9 (0.0)	42.1
TFM w/o FN	42.9 (0.0)	44.1 (0.0)	48.3 (0.0)	49.1 (0.0)	46.1
TFM	42.9 (0.0)	43.1 (0.0)	49.9 (0.0)	48.8 (0.0)	**46.2**
Stanford 40 Actions					
Method	Task 1	Task 2	Task 3	Task 4	Avg.
Finetuning	24.4 (-10.5)	26.5 (-7.7)	17.6 (-16.8)	28.9 (0.0)	24.4
Freezing	34.9 (0.0)	29.4 (0.0)	30.1 (0.0)	30.5 (0.0)	31.2
Joint	45.7 (+10.8)	40.3 (+4.8)	43.2 (-1.1)	40.2 (0.0)	42.4
EWC [14]	24.2 (-10.7)	28.2 (-2.0)	25.2 (-5.6)	34.3 (0.0)	28.0
HAT [41]	25.7 (-1.0)	25.5 (-2.7)	30.1 (-2.1)	34.4 (0.0)	28.9
PackNet [26]	32.5 (0.0)	32.9 (0.0)	36.7 (0.0)	34.3 (0.0)	34.1
TFM w/o FN	35.3 (0.0)	38.3 (0.0)	39.2 (0.0)	38.0 (0.0)	37.7
TFM	35.3 (0.0)	37.2 (0.0)	42.0 (0.0)	37.2 (0.0)	**38.0**

a subset of Birds) we consider training from scratch provides a more natural setting for continual learning.

We compare our method (TFM) and an ablation version of it without the task-specific feature normalization (TFM w/o FN) with two mask-based approaches (HAT, PackNet), a well-known model-based approach (EWC) and the baselines (Finetuning, Freezing, Joint) on three fine-grained datasets (Flowers, Birds, Actions). As can be seen in Table 3, our approach outperforms the other approaches for the three datasets. For these datasets only on Flowers a considerable performance gain is observed when adding task-specific feature normalization. Only PackNet manages to obtain competitive results, however, on both Birds and Actions, TFM does significantly better, while having a much lower memory overhead than PackNet (0.2Mb versus 27.3Mb respectively). It is also interesting to note how well Freezing works as a non-forgetting baseline.

4.3. Task-similarity effects on tiny ImageNet

Next we experiment on several ten-task splits of tiny ImageNet. We compare our approach (TFM) with two distillation methods (LFL [13], LwF [22]), two model-based methods (EWC [14], IMM [18]) and two mask-based methods (HAT [41], PackNet [26]). We also include two baselines (Finetuning, Freezing). Performance is evaluated under the same conditions on a random tiny ImageNet partition and on a semantically similar partition (see Table 4). For further information on the latter and for per-task comparison, check the Supplementary material.

Table 4. Comparison with Tiny ImageNet on VGGnet from scratch. Average accuracy after learning all tasks. Classes are randomly split and fixed for all approaches.

Tiny ImageNet – avg. acc. after 10 tasks			
Approach	Random split	Semantic split	Larger 1st task
Finetuning	47.4	28.0	57.2
Freezing	42.7	32.8	69.9
LfL [13]	47.7	27.8	60.0
LwF [22]	56.4	37.8	61.1
IMM-mode [18]	48.3	33.9	62.0
EWC [14]	47.8	27.8	56.6
HAT [41]	54.2	44.0	66.5
PackNet [26]	**56.4**	45.2	70.8
TFM w/o FN (Ours)	54.9	44.3	72.4
TFM (Ours)	56.0	**45.3**	**73.3**

For random splits most methods have quite good results on the last tasks with minor to no forgetting. LFL, IMM and EWC provide some improvement over Finetuning. LwF has a very good performance due to tasks being quite similar. All mask-based models have a very similar performance, with PackNet having the better performance. In the semantically similar splits, which has a more different distribution for each task than the random case, some approaches have difficulties avoiding catastrophic forgetting as the sequence gets longer. It is interesting to see, that the good performance of LwF on the random split is not transferred when we using semantic splits. As observed before [2], LwF fails when there exists large changes in the distributions between tasks. Mask-based models outperform all other approaches again, with TFM having the better performance. Freezing the feature extractor after the first task and learning only the classifier for the remaining tasks works better in the semantically similar splits than in the random splits.

4.4. Effect of starting-task size on tiny ImageNet

We also propose an experimental setup where the first task of tiny ImageNet uses 110 classes (55%) while the remaining 9 tasks use 10 classes (5%) each (see Table 4). This allows most of the methods to start with a rich representation after learning the first task. In this setup, comparing existing methods with the Freezing baseline is more interesting. EWC shows little forgetting, but that causes the model to become too rigid and learn the rest of the tasks with more difficulty and having a lower overall performance. Trying to

Table 5. Comparison with the state-of-the-art. ImageNet on AlexNet from scratch. Accuracy of each task after learning all tasks. Number between brackets indicates forgetting.

ImageNet - classes randomly split											
Approach	Task 1 (1-100)	Task 2 (101-200)	Task 3 (201-300)	Task 4 (301-400)	Task 5 (401-500)	Task 6 (501-600)	Task 7 (601-700)	Task 8 (701-800)	Task 9 (801-900)	Task 10 (901-1000)	Avg. all
Finetuning	25.8 (-43.0)	32.2 (-36.2)	31.4 (-35.3)	37.8 (-27.7)	39.1 (-27.7)	43.7 (-25.7)	46.0 (-22.8)	50.0 (-16.5)	53.4 (-12.1)	63.7 (0.0)	42.3
Freezing	68.8 (0.0)	53.5 (0.0)	52.0 (0.0)	51.2 (0.0)	51.3 (0.0)	53.9 (0.0)	52.2 (0.0)	53.9 (0.0)	51.7 (0.0)	51.2 (0.0)	54.0
LwF [22]	27.6 (-41.2)	37.2 (-19.9)	42.0 (-22.6)	44.4 (-20.9)	50.5 (-14.1)	56.6 (-11.3)	57.9 (-9.1)	61.2 (-5.0)	62.0 (-1.3)	62.7 (0.0)	50.2
IMM-mode [18]	68.5 (-0.3)	53.6 (0.0)	52.1 (0.0)	51.7 (-0.1)	52.5 (+0.3)	55.5 (+0.2)	54.7 (+0.1)	53.5 (0.0)	54.2 (+0.1)	51.8 (0.0)	54.8
EWC [14]	21.8 (-47.0)	26.5 (-41.7)	29.5 (-36.5)	32.9 (-32.6)	35.6 (-30.9)	40.4 (-28.1)	40.0 (-26.2)	44.7 (-20.7)	47.8 (-16.2)	61.1 (0.0)	38.0
PackNet [26]	67.5 (0.0)	65.8 (0.0)	62.2 (0.0)	58.4 (0.0)	58.6 (0.0)	58.7 (0.0)	56.0 (0.0)	56.5 (0.0)	54.1 (0.0)	53.6 (0.0)	59.1
TFM (Ours)	63.6 (0.0)	62.2 (0.0)	60.1 (0.0)	61.6 (0.0)	62.6 (0.0)	64.5 (0.0)	64.0 (0.0)	63.7 (0.0)	63.0 (0.0)	59.9 (0.0)	**62.5**

lower the trade-off hyperparameter shows a stronger forgetting of the first tasks and causes severe catastrophic forgetting. Distillation approaches try to keep representations the same as new tasks are learned. However, small changes in the weights cause forgetting later into the sequence. HAT works fine, but with a limited capacity to make changes, ends up not learning the new tasks as easily. Freezing the network after the first task seems to be one of the best options in this setup, since the rich representation of the first 110 classes is a good starting point to learn the rest of the tasks with a simple classifier. We therefore advocate for the Freezing baseline to be included in continual learning comparisons since it often provides a much harder baseline than Finetuning. Only PackNet and TFM are able to improve over that baseline even if they start from a smaller capacity, with TFM having the best results. We again refer to the Supplementary material for further per-task results.

4.5. ImageNet

Most of the compared task-aware approaches have not been evaluated using a large-scale dataset such as ImageNet. We therefore compare our proposed method (TFM) with some of those state-of-the-art approaches. In Table 5 we can see that TFM outperforms all other approaches on ImageNet split into 10 tasks of random classes. LwF does well when learning each new task with the help of the representations of previous tasks. However, as more tasks are included, older tasks start forgetting more. IMM (mode) has the opposite effect, it focuses on intransigence and tries to keep the knowledge of older tasks, running out of capacity for the newer tasks. This allows for the approach to not forget much and even have a small backward transfer, but at the cost of performing worse with newer tasks. EWC has the worst performance, possibly due to the difficulty of having a good approximation of the FIM when there is so many classes per task. HAT had problems scaling to this scenario, showing difficulties to learn new tasks. Both PackNet and TFM have a good overall zero-forgetting performance, and rely on the amount of capacity of the network more than other approaches. PackNet has a better performance during the first three tasks, taking advantage of the compression power of the pruning and finetuning. TFM has much less capacity for those tasks and therefore provides a bit lower start. However, as the remaining capacity of the network gets smaller for PackNet, TFM is capable of growing at a more scalable pace, getting a better performance on the remaining seven tasks and achieving the best results overall.

5. Conclusions

For many practical applications, it is important that network accuracy on tasks does not deteriorate when learning new tasks. Therefore, in this paper, we propose a new method for continual learning which does not suffer from any forgetting. Other than previous methods which apply masks to the weights, we propose to move the mask to the features (activations). This greatly reduces the number of extra parameters which are added per task and reduce the overhead of the network in which other approaches incur. In addition, we propose to apply a task-specific feature normalization of features, which allows adjusting previously learned features to new tasks. In ablation experiments this was found to improve results of the ternary feature masks. Furthermore, when compared to a wide range of other continual learning techniques, our method consistently outperforms these methods on a variety of datasets. However, as a limitation, the usage of mask-based approaches becomes computationally less efficient when moving from the proposed task-IL scenario to a class-incremental one. The absence of the task-label at inference time requires mask-based approaches to evaluate one forward pass per task to provide the joint prediction. We consider adapting mask-based approaches to a class-incremental setting as an interesting direction for future research.

Acknowledgments

We would like to thank B. Villalonga and M. Menta for their helpful discussion. M. Masana acknowledges grant 2019-FI_B2-00189 from Generalitat de Catalunya. We acknowledge the financial support by the Spanish project PID2019-104174GB-I00. We also thank KU Leuven C1 project Macchina.

References

[1] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3

[2] 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 (CVPR)*, 2017. 2, 7

[3] Rahaf Aljundi, Marcus Rohrbach, and Tinne Tuytelaars. Selfless sequential learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 1

[5] Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2021. 2, 6

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 6

[7] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017. 5

[8] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. Adversarial continual learning. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 3

[9] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734*, 2017. 3

[10] Robert M French. Catastrophic forgetting in connectionist networks. In *Trends in cognitive sciences*, volume 3, pages 128–135. Elsevier, 1999. 1

[11] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 1

[12] 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 (CVPR)*, 2016. 3

[13] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting learning in deep neural networks. *arXiv preprint arXiv:1607.00122*, 2016. 2, 7

[14] 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*, 2017. 1, 2, 3, 7, 8

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 1, 3, 6

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324, 1998. 3

[17] Kibok Lee, Kimin Lee, Jinwoo Shin, and Honglak Lee. Overcoming catastrophic forgetting with unlabeled data in the wild. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. 2

[18] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*, 2017. 1, 2, 3, 7, 8

[19] Timothée Lesort, Vincenzo Lomonaco, Andrei Stoian, Davide Maltoni, David Filliat, and Natalia Díaz-Rodríguez. Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges. *Information Fusion*, 2020. 2

[20] Xuhong Li, Yves Grandvalet, and Franck Davoine. A baseline regularization scheme for transfer learning with convolutional neural networks. *Pattern Recognition*, 98:107049, 2020. 1

[21] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning (ICML)*, 2019. 3

[22] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2017. 2, 7, 8

[23] Xialei Liu, Marc Masana, Luis Herranz, Joost Van de Weijer, Antonio M Lopez, and Andrew D Bagdanov. Rotate your networks: Better weight consolidation and less catastrophic forgetting. In *International Conference on Pattern Recognition (ICPR)*, 2018. 2, 3

[24] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, 2017. 1

[25] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 3

[26] 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 (CVPR)*, 2018. 1, 2, 3, 5, 6, 7, 8

[27] Marc Masana, Joost van de Weijer, Luis Herranz, Andrew D Bagdanov, and Jose M Alvarez. Domain-adaptive deep network compression. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 6

[28] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning

problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989. 1

[29] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. 3

[30] Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in psychology*, 4:504, 2013. 1

[31] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *IEEE Indian Conference on Computer Vision, Graphics & Image Processing*, 2008. 6

[32] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. 4

[33] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019. 2

[34] Benedikt Pfülb and Alexander Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in dnns. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019. 2

[35] Amal Rannen, Rahaf Aljundi, and Matthew B Blaschko Tinne Tuytelaars. Encoder based lifelong learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[36] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, and Christoph H Lampert. iCaRL: Incremental classifier and representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[37] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018. 1

[38] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. 6

[39] 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, 2, 3, 6

[40] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning (ICML)*, 2018. 3

[41] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning (ICML)*, 2018. 1, 2, 3, 5, 6, 7

[42] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. 3, 6

[43] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 6

[44] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. Human action recognition by learning bases of action attributes and parts. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2011. 6

[45] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning (ICML)*, 2017. 2, 3

[46] Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020. 2