iTAML : An Incremental Task-Agnostic Meta-learning Approach

Jathushan Rajasegaran*, Salman Khan*, Munawar Hayat*, Fahad Shahbaz Khan*, Mubarak Shah†

*Inception Institute of Artificial Intelligence, UAE, †University of Central Florida, USA

{first.lastname}@inceptioniai.org, mshah@ucf.edu

Abstract

Humans can continuously learn new knowledge as their experience grows. In contrast, previous learning in deep neural networks can quickly fade out when they are trained on a new task. In this paper, we hypothesize this problem can be avoided by learning a set of generalized parameters, that are neither specific to old nor new tasks. In this pursuit, we introduce a novel meta-learning approach that seeks to maintain an equilibrium between all the encountered tasks. This is ensured by a new meta-update rule which avoids catastrophic forgetting. In comparison to previous meta-learning techniques, our approach is task-agnostic. When presented with a continuum of data, our model automatically identifies the task and quickly adapts to it with just a single update. We perform extensive experiments on five datasets in a class-incremental setting, leading to significant improvements over the state of the art methods (e.g., a 21.3% boost on CIFAR100 with 10 incremental tasks). Specifically, on large-scale datasets that generally prove difficult cases for incremental learning, our approach delivers absolute gains as high as 19.1% and 7.4% on ImageNet and MS-Celeb datasets, respectively. Our codes are available at: https://github.com/brjathu/iTAML.

1. Introduction

Visual content is ever-evolving and its volume is rapidly increasing each day. High-dimensionality and mass volume of visual media makes it impractical to store ephemeral or streaming data and process it all at once. Incremental Learning (IL) addresses this issue, and requires an agent to continually learn new tasks, while preserving old knowledge with limited or no access to previous data. The goal is to end-up with a single model, that performs well for all the tasks. In this manner, incremental learning models offer adaptation, gradual development and scalability capabilities that are very central to human-inspired learning.

This paper studies class-incremental learning where groups of classes are sequentially observed. This case is fundamentally different from conventional classification task due to two key factors, *plasticity* and *stability*. A *plastic* network can quickly forget old tasks while performing well on new ones (forgetting), and a *stable* network can impede
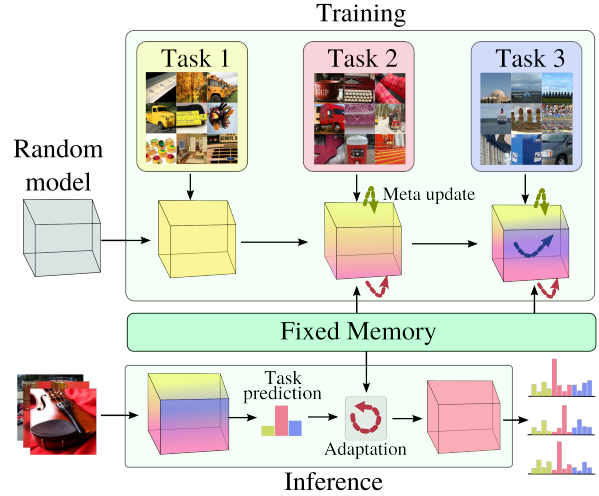

Figure 1: We propose a task-agnostic meta-learning approach for class-incremental learning. In this setting, tasks are observed sequentially where each task is a set of classes. During training, *iTAML* incrementally learns new tasks with meta-updates and tries to retain previous knowledge to learn a generic model. At inference, given a data continuum, *iTAML* first predicts the task and then quickly adapts to it. A fixed-size memory buffer is kept to support the adaptation.

learning new tasks in an effort to keep old information (interference). Despite several attempts, catastrophic forgetting and interference are still challenging problems for deep learning [19]. This paper takes a different approach to IL based on the following principle: instead of a 'one size fits all' approach that learns a single model well-suited for all tasks, we propose to learn a generic meta-model which can be quickly adapted to the desired task. The meta-learning framework focuses on learning 'how to rapidly learn?' The generality of the model arises from the '*learning to learn*' training strategy, that focuses on figuring out the shared parametric space common to all hitherto observed tasks.

Adapting meta-learning for IL involves numerous challenges. *First*, an incremental classifier keeps observing new tasks, therefore it must maintain a balance between plasticity and stability. Current meta-learning algorithms such as MAML (Model Agnostic Meta-Learning) [6], FOMAML

(First Order MAML) [17] and Reptile [18] do not offer the ability to adapt to new tasks or remember old tasks. *Secondly*, without any task-specific feedback, meta-learning algorithms are not suitable for classification problems. Although meta-learned models can quickly adapt to new tasks, they require task-information to update their parameters which restricts their applicability. *Finally*, most IL methods use a fixed-size memory buffer for old task replay. This makes the training distribution imbalanced due to majority samples from newer task, thereby learning a biased model.

To overcome these challenges, we propose *iTAML*, a task-agnostic meta-learning algorithm specifically designed for IL settings. To minimize catastrophic forgetting, *iTAML* applies a meta-update rule that maintains an equilibrium between current and old knowledge. Further, it separates the generic feature extraction module from the task-specific classifier, similar to [11], thereby minimizing interference and promoting a shared feature space amongst tasks. To avoid the limitation of knowing task information beforehand, *iTAML* learns a task-agnostic model that predicts the task automatically and subsequently adapts to the predicted task. This makes *iTAML* unique compared to other meta-learning strategies. Finally, *iTAML* mitigates the data imbalance problem by tuning the task specific parameters separately for each task. Thus, the task-specific parameters are not influenced by the majority samples of the current task.

The major contributions of this work are: **(i)** A new task-agnostic meta-learning algorithm, *iTAML*, which automatically predicts the task as well as the final class, **(ii)** A momentum based strategy for meta-update which effectively avoids forgetting, **(iii)** A new sampling rate selection approach that provides the lower-bound for the number of samples in the data continuum required for meta-update during inference, and **(iv)** Significant performance gains demonstrated by extensive experiments on ImageNet, CIFAR100, MNIST, SVHN and MS-Celeb datasets.

2. Related Work

Existing IL methods propose architectural modifications for deep CNNs e.g., dynamic networks [20, 21], dual-memory modules [7], and network expansion [24]. Rehearsal based methods have also been proposed that replay the old task by using an exemplar set [22, 4] or synthesize samples using generative models [26, 27]. IL approaches that work fundamentally on *algorithmic level* can be grouped into regularization and meta-learning based methods. We discuss these two sets of approaches next.

Regularization Strategies for Incremental Learning: The regularization based methods impose constraints during learning that seek to retain past knowledge. For example, learning without forgetting [14] adds a distillation loss to preserve the old knowledge while sequentially learning new tasks. Different from the 'task-incremental' setting

explored in [14], [22, 2] apply distillation loss in 'class-incremental' setting to reduce forgetting. A distillation loss on the attention maps of the deep network is proposed in [5] that minimizes overriding the old-task information. Recently, [29] advocates for a simple bias correction strategy that promotes re-balancing the final classifier layer to give equal importance to current and older classes.

A broad set of regularization approaches introduce synaptic intelligence [30, 12, 1] which estimate the importance of each neuron and selectively overwrite the less-important weights as the learning progresses. Lee *et al*. [13] propose to incrementally learn new tasks by merging the old and new parameters via first and second order moment matching of posterior distributions. Elastic weight consolidation (EWC) [12] computes synaptic importance offline with a Fisher information matrix. It is used to slow down the learning for weights highly relevant to previous tasks. However, EWC can also cause intransigence towards new tasks, for which [3] proposes exemplar rehearsal alongwith a Riemannian manifold distance measure for regularization.

Meta-learning for Incremental Learning: The overarching goal of meta-learning is to learn a model on a series of tasks, such that a new task can be quickly learned with minimal supervision. Meta-learning is thus ideally suited for IL since tasks are progressively introduced. A number of IL methods inspired by meta-learning have recently been proposed. Riemer *et al*. [23] learn the network updates that are well-aligned and avoid moving in directions that can cause interference. However, [23] uses a fixed loss function to align the gradients, that cannot be used in customized applications. Javed *et al*. [11] propose a meta-learning approach that disentangles generic representations from task-specific learning. Specifically, as new tasks are introduced, the task head is learned with both the inner and outer (meta) updates, while the representation learning backbone is only adapted with the outer (more-generic) updates. However, [11] assumes the training data is a correlated data stream, and all task samples are concurrently present during inference. Unlike [11], we do not impose such strict constraints. Also, both of these methods [23, 11] update the inner loop by using a single sample at one time, which is not suitable for large-scale IL. Moreover, [16] assumes that the task is known for the data continuum which limits its applicability to practical scenarios. Jamal *et al*. [10] present a task-agnostic meta-learning approach applicable only to few-shot learning. In contrast, *iTAML* is task-agnostic and well suited for large-scale settings. Our proposed meta-update is unbiased towards majority class samples and simultaneously minimizes forgetting. At inference, our model automatically adapts to the predicted task and uses task specific weights for class estimates. Besides, for the first time, we show the promise of meta-learning for large-scale incremental object recognition on five popular datasets.

3. Proposed Method

Our proposed *iTAML* is a class IL approach that is model & task-agnostic (i.e. independent of the network architecture and does not require task information). During training, we find a shared set of parameters that can work well for new tasks with minor local changes. *iTAML* therefore learns generic meaningful representations that are transferable across tasks. In other words, meta-learning process forces the model to understand the inherent relationship between sequential tasks. At inference, given a data continuum with all samples belonging to the same task, our approach follows a two stage prediction mechanism. First, we predict the task using our generic model, then, we quickly adapt to the predicted task and find the class labels.

3.1. Incremental Task Agnostic Meta-learning

We progressively learn a total of T tasks, with U number of classes per task. Consider a classification model divided into two sub-nets, a feature mapping network f_θ and a classifier f_ϕ. The function of both networks is given by,

$$f_\theta : \boldsymbol{x} \in \mathbb{R}^{C \times H \times W} \mapsto \boldsymbol{v} \in \mathbb{R}^{1 \times D}$$

$$f_\phi : \boldsymbol{v} \in \mathbb{R}^{1 \times D} \mapsto \boldsymbol{p} \in \mathbb{R}^{1 \times (UT)},$$

where, f_θ maps an input image $\boldsymbol{x}$ to a feature vector $\boldsymbol{v}$, and f_ϕ maps $\boldsymbol{v}$ to output predictions $\boldsymbol{p}$. We start with a set of randomly initialized parameters $\Phi = \{\theta, \phi\}$, where $\phi = \left[\phi_1^\top, \ldots, \phi_T^\top\right]^\top$ and $\phi_i \in \mathbb{R}^{U \times D}$ are the task-specific classification weights. Training the first task is straightforward, however, when we get a new task $t \in [1, T]$, the old parameters Φ^{t-1} should generalize to all t tasks.

Our proposed meta-learning approach (Algorithm 1) involves two updates, an *'inner loop'* which generates task-specific models for each task, and an *'outer loop'* which combines task-specific models into a final generic model.

Inner loop: To train the inner loop, we randomly sample a mini-batch with K triplets $\mathcal{B}_m = \{(\boldsymbol{x}_k, y_k, \ell_k)\}_{k=1}^K$ from the union set of current task training data $\mathcal{D}(t)$ and the exemplar memory $\mathcal{M}(t-1)$ containing a small number of samples for old tasks. Here, $\boldsymbol{x}_k, y_k$ and ℓ_k are the training images, class labels and task labels, respectively. This randomly sampled mini-batch contains training samples from multiple tasks. To train the task-specific model, we first group the training samples according to the tasks to form a micro-batch $\mathcal{B}_\mu^i = \{(\boldsymbol{x}_j, y_j, \ell_j)'\}_{j=1}^J$ per each task $i \in [1, t]$, where all ℓ_j in a micro-batch are identical. Inner loop parameters $\Phi_i = \{\theta, \phi_i\}$ are updated such that Binary Cross-entropy (BCE) loss is minimized on each micro-batch. Here, θ is updated in the inner loop for all tasks, but ϕ_i is only updated for i^{th} task. Also for each task, ϕ_i are updated for r iterations using the same micro-batch. This helps in obtaining task-specific models closer to their original task-manifold, thereby providing a better estimate for gradient update in the outer-loop to obtain a generic model.
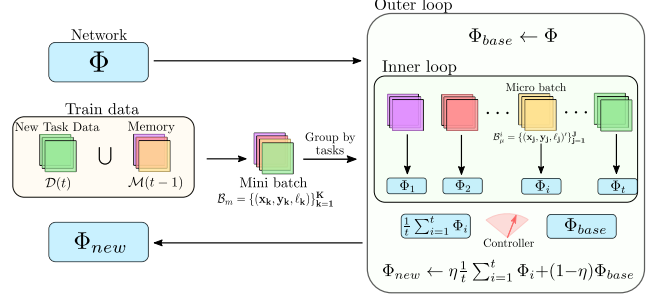

Figure 2: *iTAML Meta-update*: Mini-batches are randomly sampled from a union set of new task training data and exemplar memory. Then, we group the samples according to the task, and create micro-batches which are used to generate task-specific models. Finally, in the outer loop all the task-specific models are combined.

Outer loop: In the outer loop of *iTAML*, we combine the task specific models generated during the inner loop to form a more generic model. Let, Φ_{base} is the model parameter set before inner loop updates. Then, we treat the combined effect of all $(\Phi_{base} - \Phi_i)$ as the gradient update for the outer loop [18]. Simply put, we move the meta-model from Φ_{base} towards the average direction of all task-specific updates from Φ_{base} in the inner loop using a dynamic controller η,

$$\Phi = \Phi_{base} - \eta \frac{1}{t} \sum_{i=1}^t (\Phi_{base} - \Phi_i) = \eta \frac{1}{t} \sum_{i=1}^t \Phi_i + (1-\eta)\Phi_{base}.$$

As the training progresses, the model must learn new tasks while simultaneously preserving previous information. In an ideal case, the model should adapt quickly at the early stage of the learning, while during the later tasks, it must avoid any drastic changes since a generic set of features is already learned. To impose this, we use a simple momentum-based dynamic controller η, which speeds

---

Algorithm 1 Meta-training in *iTAML*
_____
 1:    **Require:** $\Phi^{t-1}, \mathcal{D}(t), \mathcal{M}(t-1), t, T$ and U
 2:       $\Phi \leftarrow \Phi^{t-1}$
 3:       **for** e iterations **do**
 4:          $\Phi_{base} \leftarrow \Phi$
 5:          $\mathcal{B}_m \sim \{\mathcal{D}(t) \cup \mathcal{M}(t-1)\}$
 6:          **for** $i \in [1, t]$ **do**
 7:             $\Phi_i \leftarrow \{\theta, \phi_i\}$
 8:             $\mathcal{B}_\mu^i \leftarrow \texttt{filter}(\mathcal{B}_m, i)$
 9:             **for** r steps **do**
10:                $\{\hat{y}_j\}_{j=1}^J \leftarrow \Phi_i(\{\boldsymbol{x}_j\}_{j=1}^J)$
11:                $loss \leftarrow \sum_j \texttt{BCE}(y_j, \hat{y}_j)$
12:                $\Phi_i \leftarrow \texttt{Optimizer}(\Phi_i, loss)$
13:          $\eta \leftarrow \exp(-\beta \cdot \frac{i}{t})$
14:          $\Phi \leftarrow \eta \cdot \frac{1}{t}\Sigma_i \Phi_i + (1-\eta) \cdot \Phi_{base}$
15:       **return** $\Phi^t \leftarrow \Phi$
_____

up the learning at the beginning and slows it down towards the end. This momentum-based controller is given by $\eta = \exp(-\beta \frac{t}{T})$, where β is the decay rate set using a validation set. As an example, in the last task, model parameters move $e^{-\beta}$ times slower than the first task in the outer loop. Controller is similar to having an adaptive learning rate or an adaptive optimizer [18], however, our controller depends on the number of tasks seen previously. This allows us to keep the right balance between plasticity and stability.

3.2. *iTAML* vs. Other Meta Algorithms

iTAML is close to Reptile [18] meta-learning algorithm. However, *iTAML* fundamentally differs from Reptile [18] in two aspects. First, our meta-update rule is different from Reptile, and incorporates a balancing factor that stabilizes the contribution from old and new tasks. Further, Reptile requires multiple inner loop updates ($r > 1$), whereas *iTAML* works well for $r \geq 1$. We elaborate these properties below.

Lemma 1. *Given a set of feature space parameters θ and task classification parameters $\phi = \{\phi_1, \phi_2, \ldots, \phi_T\}$, after r inner loop updates,* iTAML*'s meta-update gradient for task i is given by, $g_{itaml}(i) = g_{i,0} + \cdots + g_{i,r-1}$, where, $g_{i,j}$ is the jth gradient update with respect to $\{\theta, \phi_i\}$ on a single micro-batch. (see proof in the supplementary material)*

Compared to Reptile algorithm, which favors multiple batches to update inner loop, *iTAML* requires only one batch through all updates in the inner loop. As mentioned in [18], $g_{reptile} = g_{i,0}^{(0)} + g_{i,1}^{(1)} + \cdots + g_{i,r-1}^{(r-1)}$. Here, $g_{i,m}^{(m)}$ is the gradient calculated on m^{th} disjoint micro-batch. This differs from our meta-update rule which relies on one micro-batch per task in the inner loop as compared to r disjoint micro-batches per task in Reptile. We empirically found that a Reptile style meta-update is not useful for IL while our proposed update rule helps in finding task-specific weights useful for an optimal outer-loop update. This because, in an exemplar based IL setting, the memory limit per task decreases with new tasks. Hence, in a random sample of a mini-batch, old classes are under-represented compared to the new task. To do multiple micro-batch updates per task as in Reptile, we need to break a micro-batch further, which results in more noisy gradient updates. Therefore, *iTAML* efficiently uses a single micro-batch per task.

Lemma 2. *Given a set of feature space parameters θ and task classification parameters $\phi = \{\phi_1, \phi_2, \ldots \phi_T\}$,* iTAML *allows to keep the number of inner loop updates $r \geq 1$. (see proof in the supplementary material)*

The above property shows that a single inner-loop update does not result in normal joint training for *iTAML*. Thus, unlike Reptile, we can quickly meta-update with $r=1$.

Since *iTAML* is task agnostic, even after outer loop update, it can predict tasks without requiring any external in-
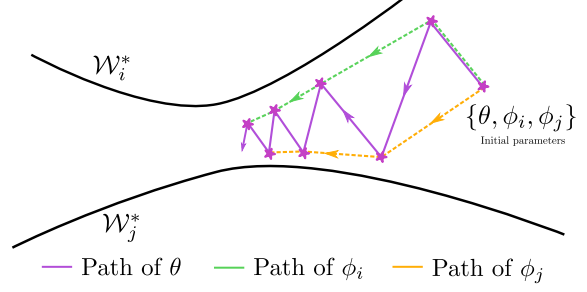

Figure 3: An illustration of how *iTAML* gradient updates move the model parameters. Let $\mathcal{W}_i^*, \mathcal{W}_j^*$ be the optimal set of parameters for tasks i and j. *iTAML* moves the feature space parameters θ towards the closet point between two optimal solution manifolds (solid line), while the classification parameters ϕ_i, ϕ_j move only if θ moves towards its corresponding manifold (dashed lines). Therefore, the task specific classification parameters stay close to their optimal solution manifolds, which allows the model to predict tasks even without any gradient updates after meta-training.

puts. This is in contrast to existing meta-learning algorithms which can not be employed in supervised classification tasks, without requiring at least some fine-tuning. For example, in few-shot learning, meta-model parameters can classify a new task only after they are updated for the given support set. Therefore, for a generalized meta-model, such as Reptile [18] and FOMAML [17], without any task information or support set, the meta-model parameters are less useful. This is because all the model parameters are updated in the inner loop for these methods. In comparison, for the case of joint training, model parameters are optimized using the current data available (i.e., exemplars and new task data) in a normal fashion. In terms of meta-learning, this is equivalent to a single gradient update using all task samples in a mini-batch. In contrast, for our proposed *iTAML*, the classification parameters $\phi = \{\phi_0, \phi_1, \ldots \phi_T\}$ are updated individually in the inner loop per task, and they remain task-specific even after the meta-update. This can be further explained from an optimal solution manifolds perspective (Fig. 3). Reptile and FOMAML move all the parameters towards a point on $\mathcal{W}^*$ which is close to all the task-specific optimal solution manifolds. In contrast, *iTAML* only moves the feature space parameters θ towards $\mathcal{W}^*$ and keeps the classification parameters ϕ_t close to their corresponding optimal solution manifolds. Further, the fixed-sized exemplar memory results in an imbalanced data distribution. Due to this, Reptile, FOMAML and Joint training methods become more biased towards the later tasks. Since, *iTAML* updates the classification parameters separately, it inherently overcomes the bias due to imbalance in tasks.

The above properties empower *iTAML* to accurately predict class labels (close to joint training) without requiring any gradient updates at inference. Further, with a given
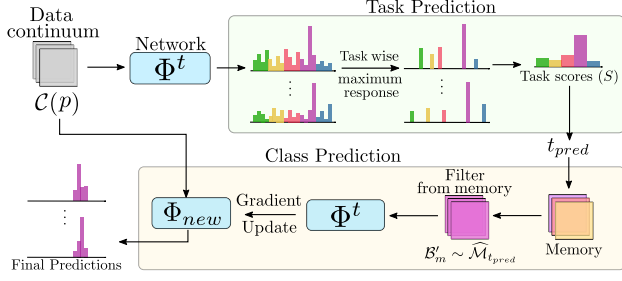
Figure 4: *Task and Class prediction*: Given the data continuum $\mathcal{C}(p)$, for all samples the maximum responses per task are accumulated into a task score to get task prediction t_{pred}. For the class prediction, exemplars of task t_{pred} from memory $\mathcal{M}$ are used to update the generic model Φ^t to task-specific model Φ_{new}. The data continuum is then fed through the Φ_{new} to get the sub-class predictions.

data continuum, *iTAML* can predict tasks with up to 100% accuracy. This allows us to design a two-stage classifier, where we first predict the task of the data continuum without any additional external knowledge, and once the task is found, we apply a gradient update to convert the generalized weights to task-specific weights, using a small exemplar set.

3.3. *iTAML* Inference

At inference time, we receive data as a continuum $\mathcal{C}(p) = \{\boldsymbol{x}_j : \ell_j = m\}_{j=1}^p$ for an unknown task m with p samples. A data continuum is simply a group of samples of an identical task bundled together. Given $\mathcal{C}(p)$, inference happens in two stages. First, the task is predicted using generalized model parameters, and then these generalized parameters are updated to the task-specific parameters to predict classes belonging to the respective task. Fig. 4 outlines the flow of task and class prediction.

Task Prediction: Consider the model Φ^t trained for t tasks, with U classes in each task. Task prediction is straightforward. First, for each sample in the continuum, we get the final classification layer response. Then, for each response vector, a maximum response per task is recorded. An average of the maximum responses per task is used as the task score. A task with a maximum score is finally predicted (t_{pred}). Algorithm 2 outlines the task prediction steps.

Algorithm 2 Task Prediction

1:   **Require:** Φ^t, $\mathcal{C}(p) = \{\boldsymbol{x}_j\}_{j=1}^p$ and U, T
2:     $S \leftarrow [0, 0, 0, ..., 0]$ ▷ initialize scores
3:     **for** $j = [1, 2, \ldots, p]$ **do**
4:       $\hat{y}^j \leftarrow \Phi^t(\boldsymbol{x}_j)$
5:       **for** $i = [1, 2, \ldots, t]$ **do**
6:          $S[i] \leftarrow S[i] + \max(\hat{y}^j[i \cdot U : (i+1) \cdot U])$
7:     **return** $t_{pred} \leftarrow \operatorname{argmax}(S)$

Class Prediction: Class prediction involves updating the generalized parameters Φ^t using exemplars. To correctly predict the classes in a task, we move the generalized parameters towards task-specific parameters Φ_{new}. To do so, we select samples from exemplar memory corresponding to classes of the predicted task t_{pred}. We use these labelled samples to do a single gradient update on the generalized parameters, which results in a task-specific parameters set. The data continuum is then fed through these parameters to find the sub-classes of the predicted task. The final class is then derived as $(U \cdot t_{pred}) + subclass$. Without losing generality, this can be extended to the case of uneven classes in different tasks. Algorithm 3 summarizes the main steps of the class prediction on data continuum.

Algorithm 3 Class Prediction

1:   **Require:** Φ^t, $\mathcal{C}(p)$, t_{pred} and memory $\mathcal{M}(t)$
2:     $\widehat{\mathcal{M}}_{t_{pred}} \leftarrow \texttt{filter}(\mathcal{M}(t), t_{pred})$
3:     $\Phi_{new} \leftarrow \{\theta, \phi_{t_{pred}}\}$
4:     **for** b iterations **do**
5:       $\mathcal{B}'_m \sim \widehat{\mathcal{M}}_{t_{pred}}$ ▷ mini-batch with Q samples
6:       $\{\hat{y}_q\}_{j=1}^Q \leftarrow \Phi_{new}(\mathcal{B}'_m)$
7:       $loss \leftarrow \sum_j \texttt{BCE}(y_j, \hat{y}_j)$
8:       $\Phi_{new} \leftarrow \texttt{Optimizer}(\Phi_{new}, loss)$
9:     **for** $j \in [1, p]$ **do**
10:       $\hat{y}_j \leftarrow \Phi_{new}(\boldsymbol{x}_j)$
11:       $subclass \leftarrow \operatorname{argmax}(\hat{y}_j[U \cdot t_{pred} : U(t_{pred} + 1)])$
12:       $C_{pred}^j \leftarrow t_{pred} \cdot U + subclass$
13:     **return** $\boldsymbol{C}_{pred} = \{C_{pred}^j\}_{j=1}^p$

3.4. Limits on the Data Continuum Size

At inference, the model is fed with a data continuum, which is used to identify the task. The model then adapts itself for the predicted task. The number of samples in the continuum plays a key role in task prediction. A higher number of samples attenuates the noisy predictions and results in a higher task accuracy. However, in practical settings, continuum sizes are limited. Therefore, it is necessary to know a lower bound on the continuum size to keep the task prediction accuracy at a certain (desired) level.

Let the model's minimum prediction accuracy be P_0 after learning t tasks, each with U classes. P_0 can be interpreted as, $P_0 = \mathcal{P}(\mathcal{Z} = i | y_{true} = i)$, where event $\mathcal{Z} = i$ denotes the case when the maximum response occurs at class i, and y_{true} is the true label. If $\bar{\mathcal{Z}} = i$ denotes the event when the maximum response occurs anywhere but at class i, then $\mathcal{P}(\bar{\mathcal{Z}} = i | y_{true} = i) = 1 - P_0$. Since *iTAML* promotes a model to be unbiased, we assume that incorrect predictions are uniformly distributed. Thus, the probability that the maximum response happens at the correct task is,

$$\mathcal{P}(\mathcal{Z} = S_\ell | y_{true} = i) = P_0 + \frac{1 - P_0}{U \cdot t - 1} \cdot (U - 1).$$

Here, S_ℓ is the set containing the classes corresponding to the ℓth task (including class i), where $\ell = \texttt{floor}(\frac{i}{U})$. Hence, $\mathcal{P}(\mathcal{Z} = S_\ell | y_{true} = i)$ is the probability that the maximum response falls at any class of the corresponding task. Let $\hat{P}_0 = \mathcal{P}(\mathcal{Z} = S_\ell | y_{true} = i)$ be the probability of correctly predicting the task. Now, if we have n samples in a continuum then the probability of overall task prediction follows a binomial distribution. In n samples correspond to any one of the t tasks, the random prediction would be $\texttt{round}(\frac{n}{t})$ samples belonging to any single task. Therefore, we require at least $\texttt{round}(\frac{n}{t}) + 1$ correctly predicted samples to find the task of a given continuum,

$$\mathcal{P}(\text{correct task}) = \sum_{k=\texttt{round}(\frac{n}{t})+1}^{n} \binom{n}{k} (\hat{P}_0)^k (1 - \hat{P}_0)^{n-k}$$

Our goal is to find a minimum value of n such that, $\mathcal{P}(\text{correct task}) > \gamma$, where, γ is the required task accuracy level. Algorithm 4 explains the main steps of finding minimum value of n. Note that Algorithm 4 can easily be solved using any simple brute-force method.

Algorithm 4 Find Lower bound on Data Continuum Size

1:   **Require:** Minimum class prediction accuracy P_0, required task accuracy γ, U and t
2:     $N \leftarrow t \cdot U$ $\triangleright$ number of classes seen
3:     $\hat{P}_0 \leftarrow \frac{U-1}{N-1} \cdot (1 - P_0)$
4:     $\min(n)$ subject to,
5:         $\sum_{k=\texttt{round}(\frac{n}{t})+1}^{n} \binom{n}{k} (\hat{P}_0)^k (1 - \hat{P}_0)^{n-k} > \gamma$
6:     **return** n

---

4. Experiments and Results

4.1. Implementation Details

Datasets: We evaluate our method on a wide spectrum of incremental learning benchmarks. These include small scale datasets i.e., split MNIST [30] and split SVHN, where each task is assigned with two classes. For medium scale datasets, we use CIFAR100 and ImageNet-100, each divided into 10 tasks (with 10 classes per task). For large scale datasets, ImageNet-1K and MS-Celeb-10K with 1000 and 10000 classes respectively are used. MS-Celeb-10K is a subset of MS-Celeb-1M dataset [8]. We consider 10 tasks, each having 100 and 1000 classes per task respectively for ImageNet-1K and Celeb-10K. To be consistent with [9], we keep a randomly sampled exemplar memory of 2000 samples for MNIST, SVHN, CIFAR100 and ImageNet-100, 20K for ImageNet-1K [22] and 50K for Celeb-10K [29].
Network Architectures: For Split-MNIST, a simple two layer MLP (400 neurons each) similar to [9] is used. For

SVHN and CIFAR100, a reduced version of ResNet-18 (*ResNet-18(1/3)*) is used, with the number of filters in all layers reduced by three times [16], resulting in $1.25\ million$ parameters. For ImageNet-100 and ImageNet-1K, we use standard ResNet-18. For training, we use RAdam [15] with initial learning rate of 0.01 for 70 epochs. Learning rate is multiplied by $\frac{1}{5}$ after 20, 40 and 60 epochs. All the models are trained on a single Tesla-V100 GPU.

4.2. Results and Comparisons

Comparison with meta-learning Algorithms: Fig. 5 compares different first-order meta-learning algorithms i.e., FOMAML and Reptile with our *iTAML* and joint training on task-agnostic, task-aware, and no inference-update settings on CIFAR100 for 10 tasks. In the task-agnostic setting, only the data continuum is present at inference, whereas for the task-aware setting, both data continuum and the task label are available. For the task-agnostic settings, the classification accuracy of Reptile and FOMAML drops drastically with more tasks since they are unable to precisely predict the tasks. Apart from that, joint training performance also drops after the first few tasks, mainly caused by highly imbalanced data distribution. However, *iTAML* is able to consistently predict the tasks with above 95% accuracy, while keeping an average classification accuracy of 77.79% even after 10 tasks. *iTAML* inherently trains task-agnostic models, thus even after the meta-updates, it can predict the tasks independently. However, other first-order meta-learning algorithms require additional information (e.g. feedback from the task or the task label). Compared with task-agnostic settings, the performance of FOMAML and Reptile improves when the task label is known (task-aware settings). Nevertheless, for the task-aware settings, *iTAML* still shows 6.9% improvement over Reptile, since it effectively tackles stability-plasticity dilemma and imbalance problem. Fig. 5 also compares different algorithms without performing any task-specific meta-updates. *iTAML* performs similar to the joint training under these settings.

Comparison with existing methods: We extensively compare *iTAML* with several popular incremental learning methods, including Elastic Weight Consolidation [12], Riemannian Walk (RWalk) [3], Learning without Forgetting (LwF) [14], Synaptic Intelligence (SI) [30], Memory Aware Synapses (MAS) [1], Deep Model Consolidation (DMC) [31], Incremental Classifier and Representation Learning (iCARL) [22], Random Path Selection network (RPS-net) [20] and Bias Correction Method (BiC) [29]. We also compare against Fixed representations (*FixedRep*) and Fine tuning (*FineTune*). *iTAML*, achieves state-of-the-art incremental learning performance on the wide range of datasets. For MNIST and SVHN, results in Table 1 show that *iTAML* achieves classification accuracy of 97.95% and
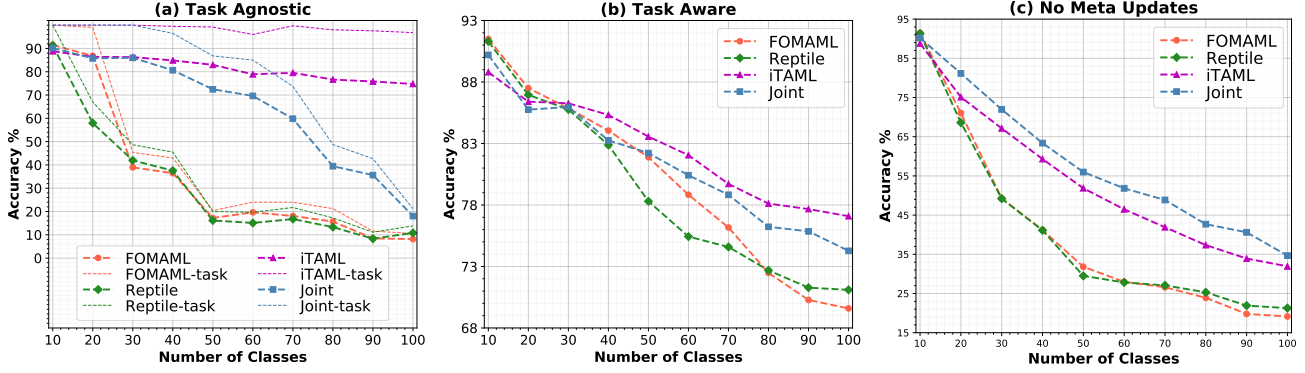
Figure 5: *Comparison between different meta-learning algorithms:* a) task-agnostic, b) task-aware, and c) with no inference updates. *iTAML* performs the best not only in task-agnostic case, but also on task-aware and no meta-update cases.

Methods	MNIST(A_5)	SVHN(A_5)
EWC [12]	19.80%	18.21%
Online-EWC [25]	19.77%	18.50%
SI [30]	19.67%	17.33%
MAS [1]	19.52%	17.32%
LwF [14]	24.17%	-
GEM* [16]	92.20%	75.61%
DGR* [26]	91.24%	-
RtF* [28]	92.56%	-
RPS-net*[20]	96.16%	88.91%
Ours*	**97.95%**	**93.97%**

Table 1: *Comparison on MNIST and SVHN datasets.* '∗' denotes memory based methods. *iTAML* outperforms state-of-the-art and performs quite close to oracle case.

93.97% respectively. This is an absolute improvement of 1.79%, 5.06% over the second best method.

Fig. 6 compares different methods on CIFAR100, for incrementally adding tasks with 10, 5 and 20 classes at once, by keeping $p = 20$. *iTAML* consistently achieves state-of-the-art results across all settings and outperforms the existing methods by a large margin. For incrementally learning 10 tasks, *iTAML* surpasses the current state-of-the-art RPS-net [20] by a margin of 21.3%. Similarly, we achieve gains of 23.6 and 18.2% on incrementally learning 5 and 20 classes respectively. Results on large scale datasets are shown in Table 2. For ImageNet-100, ImageNet-1K and MS-Celeb-10K datasets, we keep data continuum size as 50, 100 and 20 respectively. We achieve 89.8% on ImageNet-100 and surpass the current best method by a margin of 15.7%. Similarly, on ImageNet-1K, *iTAML* achieves 63.2% with an absolute gain of 19.1%. On MS-Celeb-10K dataset with 10,000 classes, the proposed *iTAML* achieves 95.02% accuracy and retains its performance with addition of new classes. These experiments strongly demonstrate the suitability and effectiveness of *iTAML* for large scale incremental learning tasks.

4.3. Ablation Analysis

We perform extensive ablation studies for different aspects of *iTAML* using CIFAR100 with 10 tasks.

Data Continuum Size (p)**:** At inference, given a data continuum, *iTAML* predicts the task, and individual sample class labels. Here, we study the impact of number of samples in the continuum on task prediction accuracy. Fig. 7(a) shows that performance improves with p. This is because, with a higher number of samples, the noise in the average response is attenuated, thereby improving task accuracy. However, the gain in the task prediction accuracy increases logarithmically. Thus, sufficient value for p ranges from 20 for CIFAR100 to as low as 3 for MS-Celeb-10K.

Variations in r**:** For higher number of inner gradient updates r, the model in the inner loop goes close to the task-specific optimal solution manifold, and the meta-model becomes more "diverse". We can see this pattern in Fig. 7(b). With $r = 5$, *iTAML* achieves 81.57% while with $r = 1$, it achieves 77.79%. However, for a model which has seen T tasks, the number of gradient updates in a batch will be $\mathcal{O}(T \cdot r)$. This slows down the training with new incoming tasks. Therefore, we keep $r = 1$, as a good trade-off between performance and computational complexity.

Variations in β**:** The parameter β controls the speed of learning new information i.e., for higher β the model does not learn any new information, and with smaller β it only learns the new information and forgets the old one. Fig. 7(c) shows that the performance improves as we vary $\beta = 0$ to $\beta = 2$, since this enhances model's ability to remember old knowledge. However, for larger β, the performance drops as model's stability increase and it is unable to learn new knowledge. We keep $\beta = 1$ in our experiments.

Lower bound on Data Continuum: We set $\gamma = 95\%$ and find the required value for n in Algorithm 4. This value is used as the data continuum size during inference for task prediction. As shown in Fig. 7(d), all the datasets achieve task accuracy around 95%, varying from $n = 17$ for CIFAR100 to $n = 3$ for MS-Celeb.
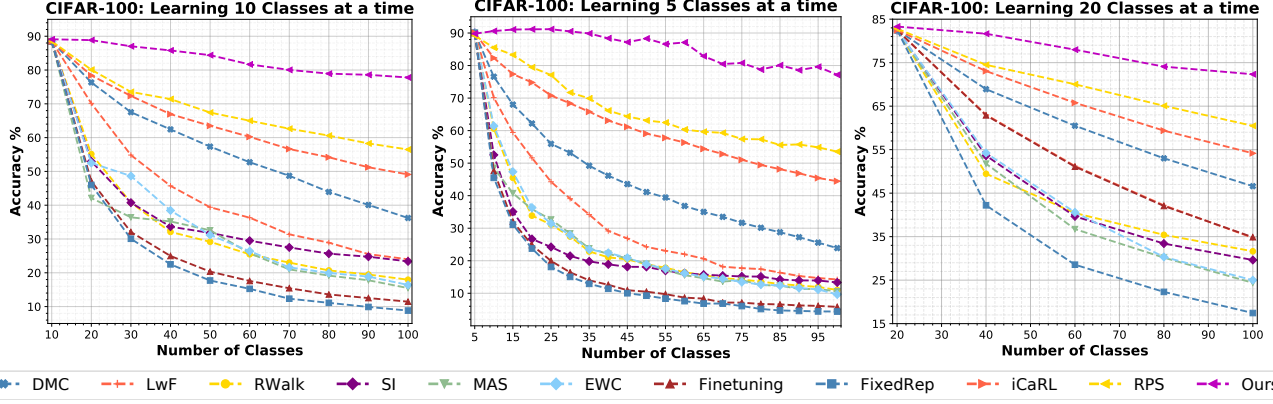
Figure 6: Classification accuracy on CIFAR100, with 10, 20 and 5 tasks from left to right. *iTAML* consistently outperforms the existing state-of-the-art across all settings.

Datasets	Methods	1	2	3	4	5	6	7	8	9	Final
ImageNet-100/10	Finetuning	99.3	49.4	32.6	24.7	20.0	16.7	13.9	12.3	11.1	9.9
	FixedRep	99.3	88.1	73.7	62.6	55.7	50.2	42.9	41.3	39.2	35.3
	LwF(TPAMI'18)[14]	99.3	95.2	85.9	73.9	63.7	54.8	50.1	44.5	40.7	36.7
	iCaRL(CVPR'17)[22]	99.3	97.2	93.5	91.0	87.5	82.1	77.1	72.8	67.1	63.5
	RPSnet(NeurIPS'19)[20]	100.0	97.4	94.3	92.7	89.4	86.6	83.9	82.4	79.4	74.1
	Ours	99.4	96.4	94.4	93.0	92.4	90.6	89.9	90.3	90.3	**89.8**+15.7
ImageNet-1K/10	Finetuning	90.2	43.1	27.9	18.9	15.6	14.0	11.7	10.0	8.9	8.2
	FixedRep	90.1	76.1	66.9	58.8	52.9	48.9	46.1	43.1	41.2	38.5
	LwF(TPAMI'18)[14]	90.2	77.6	63.6	51.6	42.8	35.5	31.5	28.4	26.1	24.2
	iCaRL(CVPR'17)[22]	90.1	82.8	76.1	69.8	63.3	57.2	53.5	49.8	46.7	44.1
	Ours	91.5	89.0	85.7	84.0	80.1	76.7	70.2	71.0	67.9	**63.2**+19.1
MS-Celeb-10K/10	iCaRL(CVPR'17)[22]	94.2	93.7	90.8	86.5	80.8	77.2	74.9	71.1	68.5	65.5
	RPSnet(NeurIPS'19)[20]	92.8	92.0	92.3	90.8	86.3	83.6	80.0	76.4	71.8	65.0
	BiC(CVPR'19)[29]	95.7	96.5	96.5	95.7	95.1	94.2	93.2	91.7	90.0	87.6
	Ours	94.0	95.6	96.0	95.8	95.5	95.4	95.2	95.1	95.0	**95.0**+7.4

Table 2: Large-scale experiments on ImageNet-1K and and MS-Celeb-10K show that *iTAML* outperforms all the state-of-the-art methods by a significant margin. Note that reported task t accuracy is an average of all $1, 2, .., t$ tasks.
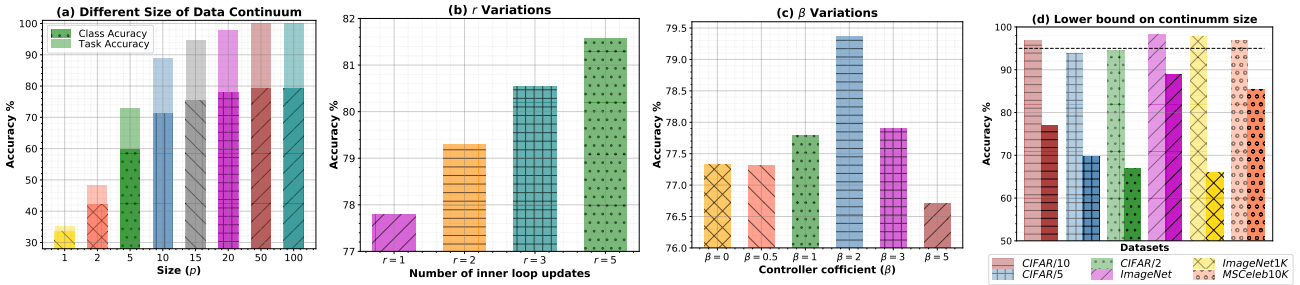

Figure 7: *Ablation studies:* We study impact of different settings on *iTAML*'s performance: **a)** Size of the data continuum at inference time vs task and class accuracies, **b)** Number of inner loop updates r, **c)** Variation in hyper parameters β, and **d)** Task and class accuracy of the model on various datasets with a data continuum size taken from Algorithm 4 with $\gamma = 95\%$.

5. Conclusion

Incremental learning aims to learn a single model that can continuously adapt itself as the new information becomes available, without overriding existing knowledge. To this end, this work proposes to update the model such that a common set of parameters is optimized on all so-far-seen tasks, without being specific to a single learning task. We develop a meta-learning approach to train a generic model that can be fast updated for a specific task. In our design, we ensure a balanced update strategy that keeps an equilibrium between old and new task information. Our approach is task-agnostic, and can automatically detect the task at hand, consequently updating itself to perform well on the given inputs. Our experiments demonstrate consistent improvements across a range of classification datasets including ImageNet, CIFAR100, MNIST, SVHN and MS-Celeb.

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)*, pages 139–154, 2018. 2, 6, 7

[2] Francisco M. Castro, Manuel J. Marin-Jimenez, Nicolas Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *The European Conference on Computer Vision (ECCV)*, September 2018. 2

[3] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018. 2, 6

[4] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. In *ICLR*, 2019. 2

[5] Prithviraj Dhar, Rajat Vikram Singh, Kuan-Chuan Peng, Ziyan Wu, and Rama Chellappa. Learning without memorizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5138–5146, 2019. 2

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017. 1

[7] Alexander Gepperth and Cem Karaoguz. A bio-inspired incremental learning architecture for applied perceptual problems. *Cognitive Computation*, 8(5):924–934, 2016. 2

[8] Yandong Guo and Lei Zhang. One-shot face recognition by promoting underrepresented classes. *arXiv preprint arXiv:1707.05574*, 2017. 6

[9] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *NeurIPS Continual learning Workshop*, 2018. 6

[10] Muhammad Abdullah Jamal and Guo-Jun Qi. Task agnostic meta-learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[11] Khurram Javed and Martha White. Meta-learning representations for continual learning. *Advances in Neural Information Processing Systems*, 2019. 2

[12] 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. In *Proceedings of the national academy of sciences*, volume 114, pages 3521–3526. National Acad Sciences, 2017. 2, 6, 7

[13] 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*, pages 4652–4662, 2017. 2

[14] Zhizhong Li and Derek Hoiem. Learning without forgettin. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947, 2018. 2, 6, 7, 8

[15] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. *arXiv preprint arXiv:1908.03265*, 2019. 6

[16] David Lopez-Paz et al. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pages 6467–6476, 2017. 2, 6, 7

[17] Alex Nichol, Joshua Achiam, and John Schulman. On first-order meta-learning algorithms. *arXiv preprint arXiv:1803.02999*, 2018. 2, 4

[18] Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2, 2018. 2, 3, 4

[19] B Pfülb and A Gepperth. A comprehensive, application-oriented study of catastrophic forgetting in dnns. *ICLR*, 2019. 1

[20] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for incremental learning. *Advances in Neural Information Processing Systems*, 2019. 2, 6, 7, 8

[21] Jathushan Rajasegaran, Munawar Hayat, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Ming-Hsuan Yang. An adaptive random path selection approach for incremental learning, 2019. 2

[22] 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. 2, 6, 8

[23] 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 *International Conference on Learning Representations*, 2019. 2

[24] 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. 2

[25] 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. *arXiv preprint arXiv:1805.06370*, 2018. 7

[26] Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pages 2990–2999, 2017. 2, 7

[27] Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine Learning Proceedings 1990*, pages 216–224. Elsevier, 1990. 2

[28] Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018. 7

[29] Yue Wu, Yinpeng Chen, Lijuan Wang, Yuancheng Ye, Zicheng Liu, Yandong Guo, and Yun Fu. Large scale incremental learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2, 6, 8

[30] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3987–3995. JMLR. org, 2017. 2, 6, 7

[31] 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. *arXiv preprint arXiv:1903.07864*, 2019. 6