# Memory Efficient Meta-Learning with Large Images

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Meta learning approaches to few-shot classification are computationally efficient at test time requiring just a few optimization steps or single forward pass to learn a new task, but they remain highly memory-intensive to train. This limitation arises because a task's entire support set, which can contain up to 1000 images, must be processed before an optimization step can be taken. Harnessing the performance gains offered by large images thus requires either parallelizing the meta-learner across multiple GPUs, which may not be available, or trade-offs between task and image size when memory constraints apply. We improve on both options by proposing LITE, a general and memory efficient episodic training scheme that enables meta-training on large tasks composed of large images on a single GPU. We achieve this by observing that the gradients for a task can be decomposed into a sum of gradients over the task's training images. This enables us to perform a forward pass on a task's entire training set but realize significant memory savings by back-propagating only a random subset of these images which we show is an unbiased approximation of the full gradient. We use LITE to train meta-learners and demonstrate new state-of-the-art accuracy on the real-world ORBIT benchmark and 3 of the 4 parts of the challenging VTAB+MD benchmark relative to leading meta-learners. LITE also enables meta-learners to be competitive with transfer learning approaches but at a fraction of the test-time computational cost, thus serving as a counterpoint to the recent narrative that transfer learning is all you need for few-shot classification.

## 1 Introduction

Meta-learning approaches to few-shot classification are very computationally efficient. Once meta-trained, they can learn a new task at test time with as few as 1-5 optimization steps [1, 2] or a single forward pass through the model [3–5] and with minimal or no hyper-parameter tuning. In contrast, transfer learning approaches based on fine-tuning typically rely on a large pre-trained feature extractor, and instead take 100s-1000s of optimization steps at test time in order to learn a task [6], thus incurring a high computational cost for each new task encountered. This makes meta-learned solutions attractive in compute-constrained deployments, or scenarios where the model must learn multiple different tasks or update on-the-fly (e.g. in continual and online learning settings [7–10]).

However, a crucial barrier to progress is that meta-learning approaches are memory-intensive to train and thus cannot easily leverage large images for a performance boost, as recent fine-tuning approaches have done. This limitation arises because a meta-learner must back-propagate through *all* the examples in a task's support set (i.e. the task's training set) that contribute to a prediction on a query example. In some cases, this can be as many as 1000 images [11]. As a result, the amount of memory required for the computational graph grows linearly with the number of support images, and quadratically with their dimension. In contrast, transfer learning approaches can employ standard

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

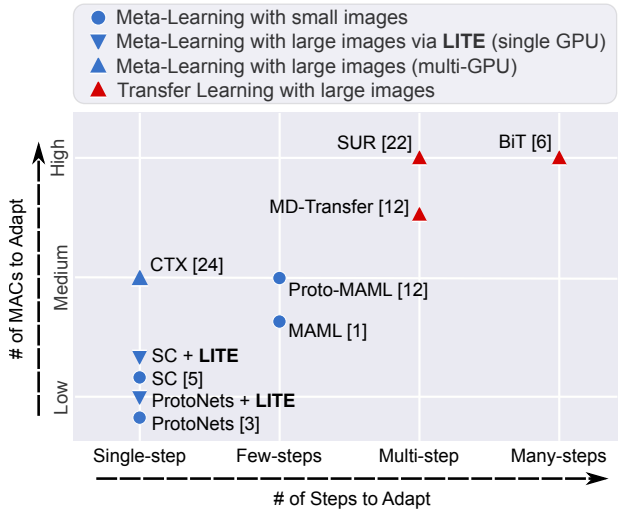


Figure 1: **LITE enables meta-learners to be trained on large images with one GPU thereby significantly improving performance while retaining their test-time computational efficiency.** The schematic shows test-time efficiency (the number of steps and number of Multiply-Accumulate operations (MACs) needed to learn a new task at test time) and whether the method can be trained on large images (required for good performance). Existing meta-learners are cheap to adapt but trained on small images (multiple GPUs are required for large images), transfer learning methods are expensive to adapt but trainable on large images. Meta-learners with LITE get the best of both worlds. Note: SC + LITE is Simple CNAPS [5] trained with LITE.

batch processing techniques to scale to larger images when under memory constraints — a feature which has contributed significantly to their recent success on few-shot benchmarks [11].

Current solutions for training meta-learners on large images include 1) parallelizing the model across multiple GPUs, which may not be available or convenient, or 2) considering tasks with only a small number of training images. Instead, most existing work [1, 3, 2, 12, 4, 13] has opted for training on large tasks but small images which translates poorly into real-world applications and limits competitiveness on few-shot benchmarks.

In this work, we improve on these alternatives by proposing LITE, a Large Image *and* Task Episodic training scheme for meta-learning models that enables training with large images and large tasks, on a single GPU. We achieve this through the simple observation that meta-learners typically aggregate a task's support examples using a permutation invariant sum operation. This structure ensures invariance to the ordering of the support set. Consequently, the gradients for a task can be decomposed as a sum of gradient contributions from the task's support examples. This enables us to perform a forward pass on a task's entire support set, but realize significant savings in memory by back-propagating only a random subset of these images which we show is an unbiased approximation of the true gradient. Fig. 1 illustrates the key trade-offs, with LITE enabling meta-learners to benefit from large images for improved classification accuracy but still remain computationally efficient at test time.

We use LITE to train key meta-learning methods and show that our best performing instantiation – Simple CNAPS [5] with LITE – achieves state-of-the-art results relative to all meta-learners on two challenging few-shot benchmarks: VTAB+MD [11], an extensive suite of both meta-learning and transfer learning tasks, and ORBIT [13], an object recognition benchmark of high-variation real-world videos. Our results showcase the unique advantage of meta-learning methods – that when properly trained they can be competitive with transfer learning approaches in terms of accuracy for a fraction of the computational cost at test time — and they serve as a counterpoint to the recent narrative that transfer learning is all you need for few-shot classification.

### Our contributions

1. LITE, a general and memory-efficient episodic training scheme which enables meta-learning models to be trained on large images and large tasks on a single GPU.
2. A mathematical justification for approximating the true gradient with a random subset of a task's support examples which applies to common classes of meta-learning methods.
3. Instantiations of LITE on key classes of meta-learners to demonstrate its versatility.
4. State-of-the-art performance using Simple CNAPS with LITE compared to other leading meta-learners on two challenging few-shot benchmarks, VTAB+MD and ORBIT[1].

[1] Source code for all experiments is included in the supplementary material.

## 2 Why Meta-Learning with Large Images and Tasks is Difficult

**Meta-learning preliminaries** In few-shot image classification, the goal is to recognize new classes when given only a few training (or support) images of each class. Meta-learners typically achieve this through *episodic* training [14]. Here, an episode or task $\tau$ contains a support set $\mathcal{D}_S^\tau = \{(\boldsymbol{x}_n^\tau, y_n^\tau)\}_{n=1}^{N_\tau}$ and a query set $\mathcal{D}_Q^\tau = \{(\boldsymbol{x}_m^{\tau *}, y_m^{\tau *})\}_{m=1}^{M_\tau}$, where $(\boldsymbol{x}, y)$ is an image-label pair, $N_\tau$ is the number of (labeled) support elements given to learn the new classes, and $M_\tau$ is the number of query elements requiring predictions. Note that in a given task, elements in $\mathcal{D}_Q^\tau$ are drawn from the same set of classes as the elements in $\mathcal{D}_S^\tau$. For brevity we may use the shorthand $\mathcal{D}_S = \{\boldsymbol{x}, y\}$ and $\mathcal{D}_Q = \{\boldsymbol{x}^*, y^*\}$.

During meta-training, a meta-learner is exposed to a large number of training tasks $\{\tau\}$. For each task $\tau$, the meta-learner takes as input the support set $\mathcal{D}_S$ and outputs the parameters of a classifier that have been adapted to the current task $\boldsymbol{\theta}^\tau = \boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S)$. The classifier can now make task-specific probabilistic predictions $f(\boldsymbol{x}^*, \boldsymbol{\theta}^\tau = \boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S))$ for any query input $\boldsymbol{x}^*$ (see Fig. 2). A function $\mathcal{L}(y^*, f(\boldsymbol{x}^*, \boldsymbol{\theta}^\tau))$ computes the loss between the adapted classifier's predictions for the query input and the true label $y^*$ which is observed during meta-training. Assuming that $\mathcal{L}$, $f$, and $\boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S)$ are differentiable, the meta-learner can then be trained with stochastic gradient descent by back-propagating the loss and updating the parameters $\boldsymbol{\phi}$.

At meta-test time, the trained meta-learner is given a set of unseen test tasks, which typically contain classes that have *not* been seen during meta-training. For each task, the meta-learner is given its support set $\mathcal{D}_S$, and is then evaluated on its predictions for all the query inputs $\boldsymbol{x}^*$ (Fig. 2, left).

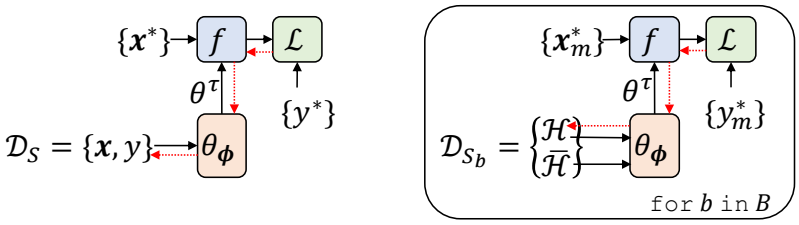


Figure 2: Left: Canonical meta-learner. Right: Meta-learner with LITE. The red dotted line shows the back-propagated gradients. Refer to Algorithm 1 for nomenclature.

**Large memory requirements for meta-training** The primary bottleneck to using large images (i.e. $\geq 224 \times 224$ pixels) in meta-learning approaches is the large amount of (GPU) memory required to process a task's support set $\mathcal{D}_S$ during meta-training. Specifically, the meta-learner $\theta_{\boldsymbol{\phi}}(\mathcal{D}_S)$ must perform a forward pass with a task's *entire* support set before it can back-propagate the loss for query elements $(\boldsymbol{x}^*, y) \in D_Q$ (and release the computation graph), thus preventing the use of conventional batch processing. The amount of memory required scales linearly with the number of support images $N^\tau$ and quadratically with their dimensions. If $N^\tau$ is large (e.g. the recent VTAB+MD benchmark [11] requires a task's support set to be as large as 1000 images), memory on a single GPU is thus quickly exceeded for large images.

Note, the number of query elements in the task $M^\tau$ is not a bottleneck when using large images as the loss decomposes over elements of the query set $\mathcal{D}_Q$ and is therefore amenable to mini-batching. By contrast, as the classifier itself is a non-linear function of the *support set*, the loss does not decompose and so it is not obvious how to apply similar ideas to allow scaling of $\mathcal{D}_S$ in a principled way.

Current ad hoc solutions to this problem are: (i) train on tasks with smaller images (e.g. $84 \times 84$ pixels in *mini*ImageNet [15]) which limits performance and translates poorly to many real-world applications; (ii) train on tasks with smaller (or sub-sampled) support sets which may adversely affect performance on test tasks with more classes and/or large numbers of samples per class; or (iii) parallelize the meta-learner across multiple GPUs which may not be convenient or available and can involve significant engineering effort.

Although training meta-learners has large memory requirements, meta-testing is generally memory efficient requiring only a small number of gradient operations, or none at all, compared to transfer learning approaches that would perform large numbers of gradient-based updates at test-time.

**Algorithm 1** LITE for a meta-training task $\tau$

**Require:** $\mathcal{D}_S$: task support set; $\mathcal{D}_Q$: task query set; $N$: number of support examples in $\mathcal{D}_S$; $H$: number of elements in $\mathcal{D}_S$ to back-propagate; $M$: number of query examples in $\mathcal{D}_Q$; $M_b$: batch size for $\mathcal{D}_Q$; `backward()` $\equiv$ function to back-propagate a loss; `step()` $\equiv$ function to update parameters with a gradient step.

$B \leftarrow \text{ceil}(M/M_b)$ ▷ number of query batches
**for all** $b \in 1, \ldots, B$ **do**
  $\mathcal{D}_{Q_b} \leftarrow \{\boldsymbol{x}_m^*, y_m^*\}_{m=1}^{M_b}$ ▷ get query batch from $\mathcal{D}_Q$
  $\mathcal{H} \leftarrow \{(\boldsymbol{x}_{n_h}, y_{n_h})\}_{h=1}^{H}$ where $\{n_h\}_{h=1}^{H} \sim \mathcal{U}(1, N)$ ▷ $\mathcal{H}$ to back-propagate
  $\overline{\mathcal{H}} \leftarrow \overline{\mathcal{D}_S \cap \mathcal{H}}$ ▷ $\overline{\mathcal{H}}$ to not back-propagate
  $\mathcal{D}_{S_b} \leftarrow \mathcal{H} \cup \overline{\mathcal{H}}$
  $\boldsymbol{\theta}^\tau \leftarrow \boldsymbol{\theta}(\mathcal{D}_{S_b})$
  $L_b \leftarrow \frac{1}{M_b} \Sigma_{m=1}^{M_b} \mathcal{L}(y_m^*, f(\boldsymbol{x}_m^*, \boldsymbol{\theta}^\tau))$ ▷ get loss of query batch
  `backward`$(L_b)$ ▷ back-propagate loss on query batch
**end for**
$\boldsymbol{\phi} \leftarrow$ `step`$(\boldsymbol{\phi}, N/H)$ ▷ update $\boldsymbol{\phi}$ using weighting factor $N/H$

# 3 Large Image and Task Episodic (LITE) training

In this section, we introduce our general and memory-efficient solution for training meta-learners episodically on tasks with large support sets and large images. We call our approach *Large Image and Task Episodic* training or LITE. In Section 3.1, we describe how LITE can be applied to key classes of meta-learners.

**Approach** The fundamental idea underlying LITE is to perform a forward pass using the entire support set $\mathcal{D}_S$, but to compute the gradient contribution on only a small random subset of the examples in the support set. By doing this, we realize large savings in memory that includes gradients, activations, and the computation graph for the elements of $\mathcal{D}_S$ that are not back-propagated. This is an approximation of the true gradient that would result if back-propagation was performed on all of the examples in $\mathcal{D}_S$. In the following section, we show that this approximation is an unbiased estimate of the true gradient. The approach for a general meta-learner is detailed in Algorithm 1 and shown diagrammatically in Fig. 2.

**Mathematical justification** The parameters of the meta-learner $\boldsymbol{\phi}$ are found by minimizing the expected loss over all tasks.

$$\underset{\boldsymbol{\phi}}{\text{argmin}} \sum_{\tau=1}^{T} \sum_{m=1}^{M_\tau} \mathcal{L}\left(y_m^{\tau *}, f\left(\boldsymbol{x}_m^{\tau *}, \boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S^\tau)\right)\right). \tag{1}$$

In most meta-learning approaches, the support set enters into the loss through a sum over the $N$ individual contributions from each data point it contains. This structure enables the meta-learners to be invariant to the ordering of the support set and allows all members of the support set to contribute to the adapted parameters (unlike alternative permutation invariant operators like *max* or *min*). Below we show in blue how this sum arises in popular brands of meta-learners.

In amortization methods (e.g. CNAPS [4] and VERSA [16]), the aggregation of support set points is built in directly via a deep set encoder $e_{\boldsymbol{\phi}_1}(\cdot)$. This encodes the support set into an embedding vector which is mapped to the classifier parameters by a hyper-network $t_{\boldsymbol{\phi}_0}(\cdot)$.

$$\boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S) = t_{\boldsymbol{\phi}_0}\left(\sum_{n=1}^{N} e_{\boldsymbol{\phi}_1}(\boldsymbol{x}_n, y_n)\right). \tag{2}$$

In gradient-based methods (e.g. MAML [1]), the classifier parameters are adapted from an initial value $\phi_0$ using a sum of derivatives of an inner-loop loss computed for each data point in the support set. The derivatives play the role of the deep set encoder in amortization methods.

$$\boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S) = \boldsymbol{\phi}_0 + \boldsymbol{\phi}_1 \sum_{n=1}^{N} \frac{d}{d\boldsymbol{\phi}} \mathcal{L}_{\text{inner}}(y_n, f(\boldsymbol{x}_n, \boldsymbol{\phi}))\Big|_{\boldsymbol{\phi}=\boldsymbol{\phi}_0} \tag{3}$$

Metric-based methods (e.g. ProtoNets [3]) comprise a body formed of a feature extractor and a head formed from a distance-based classifier. The classifier's body parameters are not adapted in a task specific way $\boldsymbol{\theta}^{(\text{body})}_{\phi,c}(\mathcal{D}_S) = \phi_0$. The classifier's head is adapted by averaging the activations for each class in the support set to form prototypes. Letting $k_c$ denote the number of support examples of class $c$, the adapted head parameters are given by

$$\boldsymbol{\theta}^{(\text{head})}_{\phi,c}(\mathcal{D}_S) = \frac{1}{k_c}\sum_{i=1}^{k_c} f(\boldsymbol{x}_i^{(c)}, \phi_0) = \frac{1}{k_c}\sum_{n=1}^{N} \mathbb{1}(y_n = c) f(\boldsymbol{x}_n, \phi_0). \tag{4}$$

Query points can then be classified using their distance from these prototypes $d(f(\boldsymbol{x}_n^*, \phi_0), \boldsymbol{\theta}^{(\text{head})}_{\phi,c})$.

We have established that in many meta-learners, each support set affects the classifier parameters and therefore the loss through a sum of contributions from each of its elements. We now focus on the consequences of this structure on the gradients of the loss with respect to the meta-learner's parameters. To reduce clutter, we consider the contribution from just a single query point from a single task and suppress the dependence of the loss on the classifier and the query data point, writing

$$\mathcal{L}\left(y^*, f(\boldsymbol{x}^*, \boldsymbol{\theta}_{\boldsymbol{\phi}}(\mathcal{D}_S))\right) = \mathcal{L}\left(e_\phi(\mathcal{D}_S)\right) \text{ where } e_\phi(\mathcal{D}_S) = \sum_{n=1}^{N} e_{\boldsymbol{\phi}}(\boldsymbol{x}_n, y_n) = \sum_{n=1}^{N} e_{\boldsymbol{\phi}}^{(n)}. \tag{5}$$

As a consequence of the summation, the derivative of the loss is given by

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\phi}}\mathcal{L}(e_\phi(\mathcal{D}_S)) = \mathcal{L}'(e_\phi(\mathcal{D}_S)) \times \left(\sum_{n=1}^{N} \frac{\mathrm{d}e_{\boldsymbol{\phi}}^{(n)}}{\mathrm{d}\boldsymbol{\phi}}\right) \text{ where } \mathcal{L}'(e_\phi(\mathcal{D}_S)) = \frac{\mathrm{d}\mathcal{L}(e))}{\mathrm{d}e}\Big|_{e=e_\phi(\mathcal{D}_S)} \tag{6}$$

which is a product of the sensitivity of the loss to the encoding of the data points and the sensitivity of the contribution to the encoding from each data point w.r.t. the meta-learner's parameters. This second term is the source of the memory overhead when training meta-learners, but importantly, it can be rewritten as an expectation w.r.t. a uniform distribution over the support set data-point indices,

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\phi}}\mathcal{L}(e_\phi(\mathcal{D}_S)) = N\ \mathcal{L}'(e_\phi(\mathcal{D}_S))\ \mathbb{E}_{n\sim\mathcal{U}(1,N)}\left[\frac{\mathrm{d}e_{\boldsymbol{\phi}}^{(n)}}{\mathrm{d}\boldsymbol{\phi}}\right]. \tag{7}$$

We can now define the LITE estimator of the loss-derivative by approximating the expectation by Monte Carlo sampling $H$ times,

$$\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\phi}}\mathcal{L}(e_\phi(\mathcal{D}_S)) \approx \frac{N}{H}\mathcal{L}'(e_\phi(\mathcal{D}_S))\sum_{h=1}^{H}\frac{\mathrm{d}e_{\boldsymbol{\phi}}^{(n_h)}}{\mathrm{d}\boldsymbol{\phi}} = \frac{\mathrm{d}}{\mathrm{d}\boldsymbol{\phi}}\hat{\mathcal{L}}(e_\phi(\mathcal{D}_S)) \text{ where } \{n_h\}_{h=1}^{H} \sim \mathcal{U}(1,N). \tag{8}$$

This estimator is unbiased, converging to the true gradient as $H \to \infty$. The estimator does not simply involve subsampling of the support set – parts of it depend on all the support set data points $\mathcal{D}_S$ – and this is essential for it to be unbiased. The expectation and variance of this estimator are

$$\mathbb{E}_{\{n_h\}\sim\mathcal{U}(1,N)}\left[\frac{\mathrm{d}\hat{\mathcal{L}}}{\mathrm{d}\boldsymbol{\phi}}\right] = \frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\boldsymbol{\phi}} \text{ and } \mathbb{V}_{\{n_h\}\sim\mathcal{U}(1,N)}\left[\frac{\mathrm{d}\hat{\mathcal{L}}}{\mathrm{d}\boldsymbol{\phi}}\right] = \frac{N^2}{H}(\mathcal{L}')^2\ \mathbb{V}_{\{n_h\}\sim\mathcal{U}(1,N)}\left[\frac{\mathrm{d}e_{\boldsymbol{\phi}}^{(n_h)}}{\mathrm{d}\boldsymbol{\phi}}\right].$$

LITE provides memory savings by subsampling $H$ examples from the support set, with $H < N$, and back-propagating only them. Crucially, a forward pass is still performed with the complementary set of points, with cardinality $N - H$, but these are not back-propagated.

### 3.1 Applying LITE to key meta-learning approaches

To demonstrate its versatility, we now describe how to apply LITE to models within the 3 main classes of meta-learners: CNAPS [4] and Simple CNAPS [5] for amortization-based methods, ProtoNets [3] for metric-based methods, and MAML [1] for gradient-based methods. Note, these are a few possible instantiations. LITE can be applied to other meta-learning methods in a straightforward manner.

In the descriptions below, we consider just one query batch $\mathcal{D}_{Q_b}$ (i.e. one iteration of the for-loop in Algorithm 1). Note, in practice, whenever $\mathcal{H}$ is passed through a module, back-propagation is

enabled, while for $\overline{\mathcal{H}}$, back-propagation is disabled.[2] Furthermore, since typically $|\mathcal{H}| \ll |\overline{\mathcal{H}}|$, we can forward $\mathcal{H}$ in a single batch, however, we need to split $\overline{\mathcal{H}}$ into smaller batches. Since $\overline{\mathcal{H}}$ is excluded from the computation graph, this can be done without a significant impact on memory.

**CNAPS [4], Simple CNAPS [5] + LITE (Appendix A.1)** CNAPS variants are amortization-based methods whose hyper-networks take a task's support set as input and generate FiLM layer [17] parameters which modulate a fixed feature extractor. The classifier head can also be generated (CNAPS [4]) or adopt a metric-based approach (Simple CNAPS [5]), thus both variants can be adapted with just a single forward pass of the support set at test time. Meta-training them with LITE involves passing $\mathcal{H}$ and then $\overline{\mathcal{H}}$ through their set-encoder $e_{\phi_1}$, and then averaging all the low-dimensional embeddings to get an embedding for the task. The task embedding is then input into a set of MLPs which generates FiLM layer parameters. $\mathcal{H}$ is passed through this configured feature extractor, followed by $\overline{\mathcal{H}}$, to get the task-adapted features for all support examples in $\mathcal{D}_{S_b}$. For CNAPS, the task-adapted features of $\mathcal{H}$ and $\overline{\mathcal{H}}$ are pooled by class and fed into a second MLP which generates the parameters of the fully-connected classification layer. For Simple CNAPS, the task-adapted features of $\mathcal{H}$ and $\overline{\mathcal{H}}$ are instead used to compute class-wise distributions (i.e. class mean and covariance matrices). With back-propagation enabled, the query batch $\mathcal{D}_{Q_b}$ is then passed through the task-configured feature extractor and classified with the task-configured classifier (for CNAPS), or with the Mahalanobis distance [18] to the class-wise distributions (for Simple CNAPS). The query batch loss is computed, and only back-propagated for $\mathcal{H}$. Note that the feature extractor is pre-trained and frozen, and only the parameters of the set-encoder and generator MLPs are learned.

**ProtoNets [3] + LITE (Appendix A.2)** ProtoNets [3] is a metric-based approach which computes a set of class prototypes from the support set and then classifies query examples by their (e.g. Euclidean) distance to these prototypes. Like CNAPS variants, it requires only a single forward pass to learn a new task. Meta-training ProtoNets with LITE involves passing $\mathcal{H}$ through the feature extractor with back-propagation enabled, followed by $\overline{\mathcal{H}}$ with back-propagation disabled, to obtain features for all support examples $\mathcal{D}_{S_b}$. These features are averaged by class to compute the prototypes such that (with back-propagation enabled) the query batch $\mathcal{D}_{S_b}$ can be passed through the feature extractor and classified based on the Euclidean distance. The loss of the query batch is computed and only back-propagated for $\mathcal{H}$. Note that here all the parameters of the feature extractor are learned.

**MAML [1] + LITE (Appendix A.3)** MAML is an optimization-based approach which optimizes the model's parameters such that it can learn a new task at test time with just 1-15 optimization steps. It thus improves on transfer learning approaches by about an order of magnitude in the number of steps to adapt. LITE does not provide an advantage when training first-order approximations of MAML as this variant drops all of the costly derivatives coming from the support set (i.e. the query loss is not back-propagated through the inner-loop gradient steps). Second-order MAML [19], however, requires Hessians to be computed for each example in the support set which leads to a high memory footprint when adapting all parameters in the model. LITE can, therefore, offer significant memory savings as Hessians would only need to be computed for elements in $\mathcal{H}$. Note, because of the memory required to retain the inner loop's computation graph, training may still need to be scaled to smaller images or smaller tasks. For this reason, we do not apply LITE to MAML in our experiments, but provide a description of a possible implementation in Appendix A.3.

# 4 Related work

We review the two main approaches to few-shot learning: transfer learning methods which are easy to scale to large images but are costly to adapt at test time, and meta-learning methods which are harder to scale but cheap to adapt (see Fig. 1). Note, we do not cover methods already described above.

Transfer learning approaches have demonstrated state-of-the-art performance on challenging few-shot benchmarks [20, 11]. However, they incur a high computational cost at test time as they rely on large (pre-trained) feature extractors which are fine-tuned with many optimization steps. MD-Transfer [12] fine-tunes all the parameters in a ResNet18 feature extractor with a cosine classifier head for 200 optimization steps. BiT [6] fine-tunes a feature extractor (pre-trained on 300M images in the JFT-300M dataset [21]) with a linear head, in some cases for up to 20,000 optimization steps to achieve

[2] In PyTorch, this can be achieved by setting `torch.grad.enabled = True` when passing $\mathcal{H}$, and `torch.grad.enabled = False` when passing $\overline{\mathcal{H}}$.

its state-of-the-art results on the VTAB [20] benchmark. SUR [22] instead trains 7 ResNet-50 feature extractors, one for each training dataset. At test time, the predictions from each are concatenated and fine-tuned to minimize a cosine-similarity loss. All of these approaches involve on the order of teras to petas of Mulitply-Accumulate operations (MACs) to learn a single new test task, and this must be repeated for each new task encountered. Furthermore, for each new task type, transfer learning approaches may need to be tuned on a validation set to obtain the optimal hyper-parameters.

In comparison, meta-learning approaches [23] generally require orders of magnitude fewer MACs and steps to learn a new task at test time. Popular approaches include CNAPS [4], Simple CNAPS [5], ProtoNets [3], and MAML [1] and are discussed in Section 3.1. Others include ProtoMAML [12] which fuses ProtoNets and MAML by initializing the classifier weights with the prototypes and then, like MAML, takes a few optimization steps to tune the weights to the task. It therefore incurs a similar cost to adapt as MAML, except it must additionally compute the prototypes. Finally, CTX [24] replaces the final average pooling layer of ProtoNets with a transformer layer that generates a series of prototypes which are spatially aware and aligned with the task. Like CNAPS, it requires just a single forward pass, however, requires more MACs to adapt since it uses a larger feature extractor (ResNet-34) and an attention module. CTX is one of the few meta-learning approaches that has been meta-trained on $224 \times 224$ images, but requires 7 days of training on 8 GPUs.

# 5 Experiments

In this section, we demonstrate that meta-learners trained with LITE achieve state-of-the-art performance among meta-learners on two challenging few-shot classification benchmarks: (i) ORBIT [13] which is a real-world few-shot object recognition dataset for teachable object recognizers; and (ii) VTAB+MD [11] which is composed of the Visual Task Adaptation Benchmark (VTAB) [20] and Meta-Dataset (MD) [12] and combines both few-shot and transfer learning tasks. We compare LITE meta-learners with state-of-the-art meta-learning and transfer learning methods in terms of classification accuracy, computational cost to learn a new task, and number of model parameters.

## 5.1 ORBIT Teachable Object Recognition Benchmark

Table 1: **Training meta-learners on large images with LITE achieves state-of-the-art accuracy with low test time adaption cost on the ORBIT Teachable Object Recognition Benchmark.** Results are reported as the average (95% confidence interval) over 85 test tasks (5 tasks per test user, 17 test users). $I$ is image size. $f_{\boldsymbol{\theta}}$ is model trained with/without LITE. RN-18 is ResNet-18. EN-B0 is EfficientNet-B0. T is $\times 10^{12}$ MACs. F is forward pass. FB is forward-backward pass.

| | | | Clean Videos | | Clutter Videos | | Test time adaption | | |
|---|---|---|---|---|---|---|---|---|---|
| MODEL | $I$ | $f_{\boldsymbol{\theta}}$ | FRAME ACC ↑ | VIDEO ACC ↑ | FRAME ACC ↑ | VIDEO ACC ↑ | MACS | STEPS | PARAMS |
| FineTuner [25] | 84 | RN-18 | 68.3 (2.3) | 77.9 (2.8) | 52.5 (1.9) | 58.0 (2.5) | 253.57T | 50FB | 11.18M |
| | 224 | RN-18 | 72.7 (2.1) | 81.7 (2.5) | 57.5 (1.9) | 61.3 (2.5) | 546.56T | 50FB | 11.18M |
| | 224 | EN-B0 | 78.4 (2.0) | 86.7 (2.2) | **65.6 (1.8)** | **71.8 (2.3)** | 121.02T | 50FB | 4.01M |
| ProtoNets [3] | 84 | RN-18 | 62.2 (2.1) | 76.7 (2.9) | 51.1 (1.7) | 60.7 (2.5) | 3.21T | 1F | 11.18M |
| | 224 | RN-18 + LITE | 76.0 (1.9) | 84.3 (2.4) | 60.5 (1.8) | 66.5 (2.4) | 5.47T | 1F | 11.18M |
| | 224 | EN-B0 + LITE | **81.4 (1.8)** | 89.2 (2.0) | **66.9 (1.8)** | **72.8 (2.3)** | **1.21T** | 1F | **4.01M** |
| CNAPs [4] | 84 | RN-18 | 66.4 (2.2) | 78.4 (2.8) | 49.8 (1.9) | 55.7 (2.5) | 3.52T | 1F | 12.76M |
| | 224 | RN-18 + LITE | 75.3 (2.0) | 85.2 (2.3) | 58.1 (1.9) | 62.3 (2.5) | 7.64T | 1F | 12.76M |
| | 224 | EN-B0 + LITE | 79.8 (1.9) | 87.7 (2.1) | 62.3 (1.8) | 68.5 (2.4) | 3.38T | 1F | 10.59M |
| Simple CNAPs [5] | 84 | RN-18 | 70.2 (2.1) | 82.0 (2.5) | 53.0 (1.8) | 61.4 (2.5) | 3.48T | 1F | 11.97M |
| | 224 | RN-18 + LITE | 76.5 (2.0) | 86.0 (2.3) | 59.3 (1.9) | 64.3 (2.4) | 7.64T | 1F | 11.97M |
| | 224 | EN-B0 + LITE | **82.2** (1.8) | **91.6 (1.8)** | **66.6 (1.9)** | **72.3 (2.3)** | 3.39T | 1F | 5.67M |

ORBIT [13] is a highly realistic few-shot video dataset collected by people who are blind/low-vision. It presents an object recognition benchmark task which involves personalizing (i.e. adapting) a recognizer to each individual user with just a few (support) videos they have recorded of their objects. To achieve this, the benchmark splits data collectors into disjoint train, validation, and test user sets along with their corresponding objects and videos. Models are then meta-trained on the train users, and meta-tested on how well they can learn a test user's objects given just their videos (on a user-by-user basis). The benchmark has two evaluation modes: how well the meta-trained model can

recognize a test user's objects in *clean* videos where there is only that object present, and in *clutter* videos where that object appears within a realistic, multi-object scene.

**Experiments** We meta-train ProtoNets [3], CNAPs [4] and Simple CNAPs [5] with LITE on tasks composed of large ($224 \times 224$) images and compare this to meta-training on tasks of small ($84 \times 84$) images (i.e. the original baselines [13]). We also compare against a transfer learning approach, FineTuner [25], which freezes a pre-trained feature extractor and fine-tunes just the linear classifier for 50 optimization steps. For each model, we consider a ResNet-18 (RN-18) and EfficientNet-B0 (EN-B0) feature extractor, both pre-trained on ImageNet [26]. We follow the task sampling protocols described in [13] (see Appendices B and C.1 for further implementation and experimental details). We also include analyses on meta-training with small tasks of large images in Appendix D.3.

**Results** In Table 1, we report frame accuracy and video accuracy, averaged over all the query videos from all tasks across all test users (17 test users, 85 tasks in total), along with its corresponding 95% confidence interval. We also report the computational cost to learn a new task at test time in terms of the number of Multiply-Accumulate operations (MACs) and number of steps to adapt. See Appendix C.1 for results on additional metrics. The key observations from our results are:

- Training on larger ($224 \times 224$) images leads to better performance compared to smaller ($84 \times 84$) images. The boost is significant for both clean and clutter videos, though absolute performance remains lower on clutter videos. This suggests that object detection or other attention-based mechanisms may be required to further exploit large images in more complex/cluttered scenes.
- Simple CNAPs + LITE sets a new state-of-the-art on clean videos, and performs competitively with ProtoNets + LITE and the FineTuner on clutter videos, using an EfficientNet-B0 backbone.
- Meta-learners are competitive with transfer learning approaches in accuracy but are almost two orders of magnitude more efficient in terms of MACs to learn a new task, and 1 order of magnitude in the number of steps to adapt.

### 5.2 VTAB+MD

VTAB+MD [11] combines revised versions of the Meta-Dataset (MD) and VTAB datasets and is one of the largest, most comprehensive, and most challenging benchmarks for few-shot learning systems. The MD-v2 part of the benchmark involves testing on 8 diverse datasets while VTAB-v2 involves testing on 18 datasets grouped into three different cataegories (natural, specialized, and structured).

**Results** Fig. 3 compares our best meta-learner, Simple CNAPS + LITE, with 6 other competitive approaches on VTAB+MD: BiT, MD-Transfer, and SUR are transfer learning based methods while ProtoMAML, ProtoNets, and CTX are meta-learning based. Note, comparisons are not always like-for-like as methods use differing backbones, image sizes, and pre-training datasets. Appendix D.2 provides this information along with the precise numbers and 95% confidence intervals. Appendices B and C.2 summarize all implementation and experimental details, and Appendix D.3 includes further analyses on the impact of task size on meta-training. The key observations from our results are:

- On MD-v2, Simple CNAPS + LITE has the highest average score and sets a new state-of-the-art.
- On VTAB-v2, BiT scores highest overall, but among meta-learners, Simple CNAPS + LITE is the best overall and on the natural and specialized sections. It falls short of CTX on the structured section due to poor performance on dSprites which involves predicting the position and orientation of small white shapes on a black background – a task quite different to image classification.
- These results are significant given that CTX takes 7 days to train on 8 GPUs, whereas Simple CNAPS + LITE trains in about 20 hours on a single 16GB GPU. In addition, Simple CNAPS + LITE uses a relatively small pre-trained backbone (4.0M parameters) compared to SUR's 7 pre-trained ResNet-50s (one for each MD-v2 training set, plus ImageNet).
- Simple CNAPS + LITE using $224 \times 224$ images significantly outperforms Simple CNAPS using $84 \times 84$ images, except for when the dataset images are small (e.g. Omniglot, QuickDraw, dSprites).
- MD-Transfer and BiT perform strongly across VTAB+MD, however, Simple CNAPS + LITE is significantly faster to adapt to a new task requiring only a forward pass of the support set with no hyper-parameter tuning whatsoever. The transfer learners instead perform 100s of optimization steps to adapt to a new task and may require a human-in the-loop to tune hyper-parameters such as the learning rate and number of optimization steps.

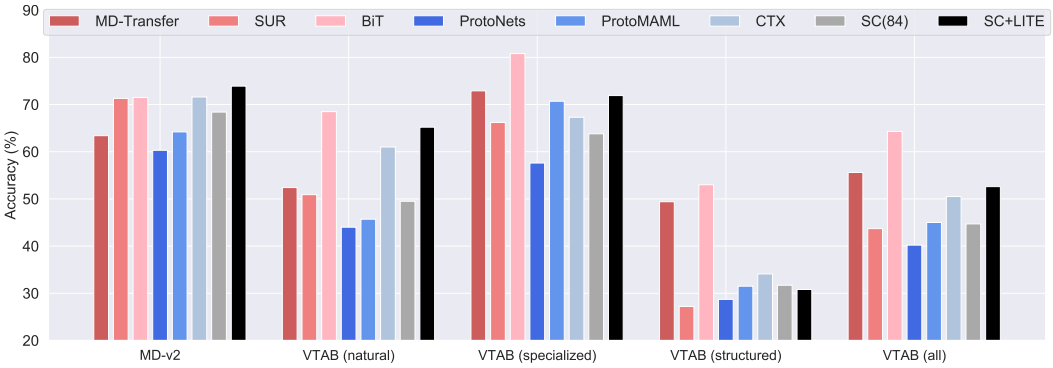


Figure 3: Summary of results on VTAB+MD. **Simple CNAPs with LITE (SC + LITE, black bar) trained on $224 \times 224$ images achieves state-of-the-art accuracy on Meta-Dataset (MD-v2), and state-of-the-art accuracy among meta-learners on 3 of 4 parts of VTAB**. As reference, we have included transfer learning methods (red bars), other meta-learning methods (blue bars), and Simple CNAPs without LITE trained on small images ($84 \times 84$, gray bar). Competitive results from [11]. See Table D.2 for tabular results on individual datasets.

## 6 Discussion

We propose LITE, a general and memory-efficient episodic training scheme for meta-learners that enables them to exploit large images for higher performance with limited compute resources. LITE's significant memory savings come from performing a forward pass on a task's full support set, but back-propagating only a random subset which we show is an unbiased estimate of the full gradient. We demonstrate that meta-learners trained with LITE are state-of-the-art among meta-learners on two challenging benchmarks, ORBIT and VTAB+MD, and are competitive with transfer learning approaches at a fraction of the test time computational cost.

This offers a counterpoint to the recent narrative that transfer learning approaches are all you need for few-shot classification. Both classes of approach are worthy pursuits (and will need to exploit large images in real-world deployments) but careful consideration should be given to the data and compute available at test time to determine which class is best suited to the application under consideration. If it involves learning just a single task type (e.g. classifying natural images) with ample data and no compute or time constraints, then a fine-tuning approach would suffice and perform well. However, if a multitude of task types will be encountered at test time, each with minimal data, and new tasks need to be learned on resource-constrained devices (e.g. mobile phone, robot) or quickly/repeatedly (e.g. in continual or online learning settings), then a meta-learning solution will be better suited.

Finally, as the machine learning community grapples with greener solutions for training deep neural networks, LITE offers a step in the right direction by allowing meta-learners to exploit large images without an accompanying increase in compute. Future work may look toward applying the basic concept of LITE to other types of training algorithms to realize similar memory savings.

**Limitations** As discussed in Section 3, LITE can be applied to a wide range of meta-learners provided that they aggregate the contributions from a task's support set via a permutation-invariant operation like a sum. Because only a subset of the support set is back-propagated, however, the gradients can be more noisy and meta-training may require lower learning rates. Furthermore, LITE is a memory-efficient scheme for training meta-learners episodically and has not been tried with meta-learners trained in other ways (e.g. with standard supervised learning).

**Societal impact** Few-shot learning systems hold much positive potential – from personalizing object recognizers for people who are blind [13] to rendering personalized avatars [27] (see [23] for a full review). These systems, however, also have the potential to be used in adverse ways - for example, in few-shot recognition in military/surveillance applications. Meta-trained few-shot systems may also pose risks in decision making applications as uncertainty calibration in meta-learning models has not yet been extensively explored. Careful consideration of the intended application, and further study of uncertainty quantification in meta-learning approaches will be essential in order to minimize any negative societal consequences of LITE if deployed real-world applications.

# References


[1] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning (ICML)*, pages 1126–1135, 2017.

[2] Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning (ICML)*, 2019.

[3] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4077–4087, 2017.

[4] James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 7957–7968, 2019.

[5] Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14493–14502, 2020.

[6] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. *arXiv preprint arXiv:1912.11370*, 6(2):8, 2019.

[7] Mark B Ring. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.

[8] David Saad. *On-line learning in neural networks*, volume 17. Cambridge University Press, 2009.

[9] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871*, 2018.

[10] Antreas Antoniou, Massimiliano Patacchiola, Mateusz Ochal, and Amos Storkey. Defining benchmarks for continual few-shot learning. *arXiv preprint arXiv:2004.11967*, 2020.

[11] Vincent Dumoulin, Neil Houlsby, Utku Evci, Xiaohua Zhai, Ross Goroshin, Sylvain Gelly, and Hugo Larochelle. Comparing transfer and meta learning approaches on a unified few-shot classification benchmark. *arXiv preprint arXiv:2104.02638*, 2021.

[12] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-dataset: A dataset of datasets for learning to learn from few examples. In *International Conference on Learning Representations (ICLR)*, 2020.

[13] Daniela Massiceti, Luisa Zintgraf, John Bronskill, Lida Theodorou, Matthew Tobias Harris, Edward Cutrell, Cecily Morrison, Katja Hofmann, and Simone Stumpf. Orbit: A real-world few-shot dataset for teachable object recognition. *arXiv preprint arXiv:2104.03841*, 2021.

[14] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, koray kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3630–3638, 2016.

[15] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations (ICLR)*, 2017.

[16] Jonathan Gordon, John Bronskill, Matthias Bauer, Sebastian Nowozin, and Richard Turner. Meta-learning probabilistic inference for prediction. In *International Conference on Learning Representations*, 2019.

[17] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Association for the Advancement of Artificial Inteligence (NAACL)*, 2018.

[18] Prasanta Chandra Mahalanobis. On the generalized distance in statistics. In *National Institute of Science of India*, 1936.

[19] Antreas Antoniou, Harrison Edwards, and Amos Storkey. How to train your maml. *arXiv preprint arXiv:1810.09502*, 2018.

[20] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[21] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *International Conference on Computer Vision (ICCV)*, pages 843–852, 2017.

[22] Nikita Dvornik, Cordelia Schmid, and Julien Mairal. Selecting relevant features from a multi-domain representation for few-shot classification. In *European Conference on Computer Vision (ECCV)*, pages 769–786, 2020.

[23] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[24] Carl Doersch, Ankush Gupta, and Andrew Zisserman. CrossTransformers: spatially-aware few-shot transfer. *arXiv preprint arXiv:2007.11498*, 2020.

[25] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3320–3328, 2014.

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

[27] Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *International Conference on Computer Vision (ICCV)*, pages 9459–9468, 2019.

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

[29] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, pages 6105–6114, 2019.

[30] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[31] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[32] Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Code for "meta-dataset: A dataset of datasets for learning to learn from few examples". `https://github.com/google-research/meta-dataset`, 2019.

[33] TensorFlow Datasets, a collection of ready-to-use datasets. `https://www.tensorflow.org/datasets`, 2020.

# Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Section 3.1 for limitations when applying LITE to key meta-learners. See Section 5 for limitations observed in the experimental results. See Section 6 for a broader treatment of LITE's limitations.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] See Section 6
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our code with a `README` is included in the supplementary. Both benchmark datasets are open-source.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix C.1 for ORBIT and Appendix C.2 for VTAB+MD.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the average and 95% confidence interval over multiple test tasks sampling from the ORBIT benchmark (see Table 1) and the VTAB+MD benchmark (see Appendix D.2).
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendices C.1 and C.2
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] All data and code that was used in our experiments is cited.
   (b) Did you mention the license of the assets? [Yes] All data and code that was used in our experiments are under open-source licenses (e.g. MIT license).
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] All code required to run LITE on key meta-learners on VTAB+MD and ORBIT is included in the supplementary and will be publicly released under an open-source license after the review period.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] The ORBIT dataset collection was approved by the City, University of London Research Ethics Committee and consent was obtained in writing from all data contributors. VTAB+MD is a collection of public (and anonymous) web-scraped image data.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] The ORBIT and VTAB+MD datasets are anonymized and do not contain any offensive content.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]