FiT: Parameter Efficient Few-shot Transfer Learning

Aliaksandra Shysheya*
University of Cambridge
as2975@cam.ac.uk

John Bronskill*
University of Cambridge
jfb54@cam.ac.uk

Massimiliano Patacchiola
University of Cambridge
mp2008@cam.ac.uk

Sebastian Nowozin†
nowozin@gmail.com

Richard E. Turner
University of Cambridge
ret26@cam.ac.uk

Abstract

Model parameter efficiency is key for enabling few-shot learning, inexpensive model updates for personalization, and communication efficient federated learning. In this work, we develop FiLM Transfer (FiT) which combines ideas from transfer learning (fixed pretrained backbones and fine-tuned FiLM adapter layers) and meta-learning (automatically configured Naive Bayes classifiers and episodic training) to yield parameter efficient models with superior classification accuracy at low-shot. We experiment with FiT on a range of downstream datasets and show that it achieves better classification accuracy than the leading Big Transfer (BiT) algorithm at low-shot and achieves state-of-the art accuracy on the challenging VTAB-1k benchmark, with fewer than 1% of the updateable parameters.

1 Introduction

Model parameter efficiency is key for enabling few-shot learning, inexpensive model updates for personalization, and communication efficient federated learning. In order to develop data-efficient and parameter-efficient learning systems, we draw on ideas developed by the few-shot learning community. Few-shot learning approaches can be characterized in terms of shared and updateable parameters. From a statistical perspective, shared parameters capture similarities between datasets, while updateable parameters capture the differences. Updateable parameters are those that are either recomputed or learned as the model is updated or retrained, whereas shared parameters are fixed. In personalized or federated settings, it is key to minimize the number of updateable parameters, while still retaining the capacity to adapt.

Broadly, there are two different approaches to few-shot learning: meta-learning [Hospedales et al., 2020] and transfer learning (fine-tuning) [Yosinski et al., 2014]. Meta-learning approaches provide methods that have a small number of updatable parameters [Requeima et al., 2019]. However, while meta-learners can perform strongly on datasets that are similar to those they are meta-trained on, their accuracy suffers when tested on datasets that are significantly different [Dumoulin et al., 2021]. Transfer learning algorithms often outperform meta-learners, especially on diverse datasets and even at low-shot [Dumoulin et al., 2021, Tian et al., 2020]. However, the leading Big Transfer (BiT) [Dumoulin et al., 2021, Kolesnikov et al., 2019] algorithm requires every parameter in a large network to be updated. In summary, performant transfer learners are parameter-inefficient, and parameter-efficient few-shot learners perform relatively poorly.

In this work we propose FiLM Transfer or FiT, a novel method that synthesizes ideas from both the transfer learning and meta-learning communities in order to achieve the best of both worlds

*Authors contributed equally
†Work performed while at Microsoft Research

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

– parameter efficiency without sacrificing accuracy, even when there are only a small number of training examples available. From transfer learning, we take advantage of backbones pretrained on large image datasets and the use of fine-tuned parameter efficient adapters. From meta-learning, we take advantage of metric learning based final layer classifiers trained with episodic protocols that we show are more effective than the conventional linear layer classifier. Our contributions:

- A parameter and data efficient network architecture for low-shot transfer learning that (i) utilizes frozen backbones pretrained on large image datasets; (ii) augments the backbone with parameter efficient FiLM [Perez et al., 2018] layers in order to adapt to a new task; and (iii) makes novel use of an automatically configured Naive Bayes final layer classifier instead of the usual linear layer, saving a large number of updateable parameters, yet improving classification performance;
- A meta-learning inspired episodic training protocol for low-shot fine-tuning requiring no data augmentation, no regularization, and a minimal set of hyper-parameters;
- State-of-the-art results on the challenging VTAB-1k benchmark (74.9% for backbones pretrained on ImageNet-21k) while using $\approx 1\%$ of the updateable parameters when compared to the leading transfer learning method BiT;

2 FiLM Transfer (FiT)

In this section we detail the FiT algorithm focusing on the few-shot image classification scenario.

Preliminaries We denote input images $\boldsymbol{x} \in \mathbb{R}^{ch \times W \times H}$ where W is the width, H the height, ch the number of channels and image labels $y \in \{1, \dots, C\}$ where C is the number of image classes indexed by c. Assume that we have access to a model $f(\boldsymbol{x}) = h_{\boldsymbol{\phi}}(b_{\boldsymbol{\theta}}(\boldsymbol{x}))$ that outputs class-probabilities for an image $p(y = c|\boldsymbol{x}, \boldsymbol{\theta}, \boldsymbol{\phi})$ for $c = 1, \dots, C$ and is comprised of a feature extractor backbone $b_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathbb{R}^{d_b}$ with parameters θ that has been pretrained on a large upstream dataset such as Imagenet where d_b is the output feature dimension and a final layer classifier or head $h_{\boldsymbol{\phi}}(\cdot) \in \mathbb{R}^C$ with weights $\boldsymbol{\phi}$. Let $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^N$ be the downstream dataset that we wish to fine-tune the model f to.

FiT Backbone For the network backbone, we freeze the parameters θ to the values learned during upstream pretraining. To enable parameter-efficient and flexible adaptation of the backbone, we add Feature-wise Linear Modulation (FiLM) [Perez et al., 2018] layers with parameters $\boldsymbol{\psi}$ at strategic points within $b_{\boldsymbol{\theta}}$. A FiLM layer scales and shifts the activations $\boldsymbol{a}_{ij}$ arising from the j^{th} channel of a convolutional layer in the i^{th} block of the backbone as $\texttt{FiLM}(\boldsymbol{a}_{ij}, \gamma_{ij}, \beta_{ij}) = \gamma_{ij}\boldsymbol{a}_{ij} + \beta_{ij}$, where γ_{ij} and β_{ij} are scalars. The set of FiLM parameters $\boldsymbol{\psi} = \{\gamma_{ij}, \beta_{ij}\}$ is learned during fine-tuning. An advantage of FiLM layers is that they enable expressive feature adaptation while adding only a small number of parameters [Perez et al., 2018]. For example, in a ResNet50 with a FiLM layer in every block, the set of FiLM parameters $\boldsymbol{\psi}$ account for only 11648 parameters which is fewer than 0.05% of the parameters in $b_{\boldsymbol{\theta}}$.

FiT Head For the head of the network, we use a specially tailored Gaussian Naive Bayes classifier. Unlike a linear head, this head can be automatically configured directly from data and has only a small number of free parameters which must be learned, ideal for few-shot, personalization and federated learning. We will also show that this head is often more accurate than a standard linear head. The class probability for a test point $\boldsymbol{x}^*$ is:

$$p(y^* = c|b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}^*), \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{\pi_c \mathcal{N}(b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}^*)|\mu_c, \Sigma_c))}{\sum_{c'}^{C} \pi_{c'} \mathcal{N}(b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}^*)|\mu_{c'}, \Sigma_{c'})} \tag{1}$$

where $\pi_c = \dfrac{N_c}{N}$, $\mu_c = \dfrac{1}{N_c}\sum_{i=1}^{N_c} b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}_i)$, $\Sigma_c = \dfrac{1}{N_c}\sum_{i=1}^{N_c}(b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}_i) - \mu_c)(b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}_i) - \mu_c)^T$

are the maximum likelihood estimates, N_c is the number of examples of class c in $\mathcal{D}$, and $\mathcal{N}(z|\mu, \Sigma)$ is a multivariate Gaussian over z with mean μ and covariance Σ.

Estimating the mean μ_c for each class c is straightforward and incurs a total storage cost of Cd_b. However, estimating the covariance Σ_c for each class c is challenging when the number of examples per class N_c is small and the embedding dimension of the backbone d_b is large. In addition, the storage cost for the covariance matrices may be prohibitively high if d_b is large. Here, we use three different approximations to the covariance in place of Σ_c in Eq. (1) [Fisher, 1936, Duda et al., 2012]:

- **Quadratic Discriminant Analysis (QDA)**: $\Sigma_{\text{QDA}} = e_1\Sigma_{class} + e_2\Sigma_{task} + e_3\boldsymbol{I}$
- **Linear Discriminant Analysis (LDA)**: $\Sigma_{\text{LDA}} = e_2\Sigma_{task} + e_3\boldsymbol{I}$
- **ProtoNets** [Snell et al., 2017]: $\Sigma_{\text{PN}} = \boldsymbol{I}$; i.e. there is no covariance and the class representation is parameterized only by μ_c and the classifier logits are formed by computing the squared Euclidean distance between the feature representation of a test point $b_{\boldsymbol{\theta},\boldsymbol{\psi}}(\boldsymbol{x}^*)$ and each of the class means.

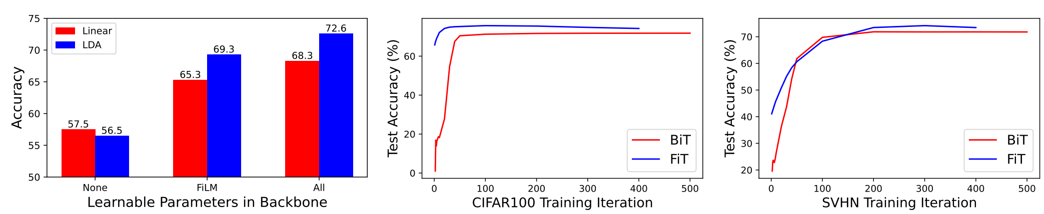
In the above, Σ_{class} is the computed covariance of the examples in class c in $\mathcal{D}$, Σ_{task} is the computed covariance of all the examples in $\mathcal{D}$ assuming they arise from a single Gaussian with a single mean, $\boldsymbol{e} = \{e_1, e_2, e_3\}$ are weights learned during training, and the identity matrix $\boldsymbol{I}$ is used as a regularizer.

QDA mainly serves as a baseline since it has a very large set of updateable parameters arising from the fact that it stores a covariance matrix for each class in the dataset. LDA is far more parameter efficient than QDA, sharing a single covariance matrix across all classes. We show that LDA leads to very similar performance to QDA. The number of model shared and updateable parameters for the three FiT variants as well as the BiT algorithm are detailed in Table 1.

Table 1: Shared and updateable parameters for the transfer learning methods considered. The Example column contains the updateable parameters for all methods using a BiT-M-R50x1 backbone with $|\boldsymbol{\theta}| = 23,500,352$, $\boldsymbol{\psi} = 11,648$, $d_b = 2048$, and $C = 10$.

Method	**Shared**	**Updateable**	**Example**
BiT	0	$\|\boldsymbol{\theta}\| + \|\boldsymbol{\phi}\| = \|\boldsymbol{\theta}\| + Cd_b$	23,520,832
FiT - QDA	$\|\boldsymbol{\theta}\|$	$\|\boldsymbol{\psi}\| + \|\boldsymbol{\mu}\| + \|\boldsymbol{\Sigma}\| + \|e\| = \|\boldsymbol{\psi}\| + Cd_b + C\frac{d_b(d_b+1)}{2} + 3$	21,013,891
FiT - LDA	$\|\boldsymbol{\theta}\|$	$\|\boldsymbol{\psi}\| + (\|\boldsymbol{\mu}\| + \|\boldsymbol{\Sigma}\|) + \|e\| = \|\boldsymbol{\psi}\| + C(d_b + 1) + 2$	32,140
FiT - ProtoNets	$\|\boldsymbol{\theta}\|$	$\|\boldsymbol{\psi}\| + \|\boldsymbol{\mu}\| = \|\boldsymbol{\psi}\| + Cd_b$	32,128

An empirical justification for the use of the FiT-LDA head is shown in Fig. 1a where it outperforms a linear head in the case of FiLM and when all the backbone parameters are learned. In Fig. 1b, we see for both datasets, FiT-LDA converges faster than BiT, which uses a linear head. The primary limitation of the Naive Bayes head is the higher (versus linear) computational cost due to having to invert a $d_b \times d_b$ covariance matrix on each training iteration.



(a) LDA outperforms linear head. (b) FiT-LDA converges more quickly than BiT.

Figure 1: (a) Average accuracy on VTAB-1k for linear and LDA heads versus learnable parameters in the backbone. (b) Test accuracy versus training iteration for CIFAR100 and SVHN on VTAB-1k.

FiT Training We learn the FiLM parameters $\boldsymbol{\psi}$ and the covariance weights $\boldsymbol{e}$ via fine-tuning (parameters $\boldsymbol{\theta}$ are fixed from pretraining). One approach would be to apply standard batch training on the downstream dataset, however it was hard to balance under- and over-fitting using this setup. Instead, we found that an approach inspired by *episodic training* [Vinyals et al., 2016] that is often used in meta-learning yielded better performance. We refer to this approach as *episodic fine tuning* and it works as follows. Note that we require 'training' data to compute $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ to configure the head, and a 'test' set to optimize $\boldsymbol{\psi}$ and $\boldsymbol{e}$ via gradient ascent. Thus, from the downstream dataset $\mathcal{D}$, we derive two sets – $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$. We randomly split $\mathcal{D}$ into $\mathcal{D}_{train}$ and $\mathcal{D}_{test}$ such that the number of examples or *shots* in each class c are roughy equal in both partitions and that there is at least one example of each class in both. Refer to Algorithm A.1 for details.

For each training iteration, we sample a task τ consisting of a *support* set $\mathcal{D}_S^\tau$ drawn from $\mathcal{D}_{train}$ with S_τ examples and a *query* $\mathcal{D}_Q^\tau$ set drawn from $\mathcal{D}_{test}$ with Q_τ examples. First, $\mathcal{D}_S^\tau$ is formed by randomly choosing a subset of classes selected from the range of available classes in $\mathcal{D}_{train}$. Second, the number of shots to use for each selected class is randomly selected from the range of available examples in each class of $\mathcal{D}_{train}$ with the goal of keeping the examples per class as equal as possible. Third, $\mathcal{D}_Q^\tau$ is formed by using the classes selected for $\mathcal{D}_S^\tau$ and all available examples from $\mathcal{D}_{test}$ in those classes up to a limit of 2000 examples. See Algorithm A.2 for details. Episodic fine-tuning is crucial to achieving the best classification accuracy with the Naive Bayes head.

The support set $\mathcal{D}_S^\tau$ is then used to compute $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$ and we then use $\mathcal{D}_Q = \{\{\boldsymbol{x}_q^{\tau *}, y_q^{\tau *}\}_{q=1}^{Q_\tau}\}_{\tau=1}^{T}$ to train $\boldsymbol{\psi}$ and $\boldsymbol{e}$ with maximum likelihood. We optimize the following:

$$\hat{\mathcal{L}}\left(\boldsymbol{\psi}, \boldsymbol{e}\right) = \sum_{\tau=1}^{T} \sum_{q=1}^{Q_\tau} \log p\left(y_q^{\tau *} | h_e(b_{\theta,\psi}(\boldsymbol{x}_q^{\tau *})), \boldsymbol{\pi}(\mathcal{D}_s^\tau), \boldsymbol{\mu}(\mathcal{D}_s^\tau), \boldsymbol{\Sigma}(\mathcal{D}_s^\tau)\right). \tag{2}$$

FiT training hyper-parameters include a learning rate, $|\mathcal{D}_S^\tau|$, and the number of training iterations. For the transfer learning experiments in Section 4 these are set to constant values across all datasets and do not need to be tuned based on a validation set. We do not augment the training data. In the 1-shot case, we do not perform episodic fine-tuning and leave the FiLM parameters at their initial value of $\boldsymbol{\gamma} = 1, \boldsymbol{\beta} = 0$ and $\boldsymbol{e} = (0.5, 0.5, 1.0)$ and predict as described next.

FiT Prediction Once the FiLM parameters $\boldsymbol{\psi}$ and covariance weights $\boldsymbol{e}$ have been learned, we use $\mathcal{D}$ for the support set to compute π_c, μ_c, and Σ_c for each class c and then Eq. (1) can be used to make a prediction for any unseen test input.

3 Related Work

We take inspiration from residual adapters [Rebuffi et al., 2017, 2018] where parameter efficient adapters are inserted into a ResNet with frozen pretrained weights. The adapter parameters and the final layer linear classifier are then learned via fine-tuning. More recently, a myriad of additional parameter efficient adapters have been proposed including FiLM, Adapter [Houlsby et al., 2019], LoRA [Hu et al., 2021], VPT [Jia et al., 2022], AdaptFormer [Chen et al., 2022], NOAH [Zhang et al., 2022], Convpass [Jie and Deng, 2022], [Mudrakarta et al., 2019], and CaSE [Patacchiola et al., 2022]. For FiT we use FiLM as it is the most parameter efficient adapter, yet it allows for expressive adaptation, and can be used in various backbone architectures including ConvNets and Transformers.

To date, transfer learning systems that employ adapters use a linear head for the final classification layer. In meta-learning systems it is common to use metric learning heads (e.g. ProtoNets [Snell et al., 2017]), which have no or few learnable parameters. Meta-Learning systems that employ a metric learning head are normally trained with an episodic training regime [Vinyals et al., 2016]. Some of these approaches (e.g. TADAM [Oreshkin et al., 2018], FLUTE [Triantafillou et al., 2021], and Simple CNAPs [Bateni et al., 2020] use both a metric head and FiLM layers to adapt the backbone.

FiT differs from all of the preceding approaches by using a powerful Naive Bayes metric head that uses episodic fine-tuning in the context of transfer learning, as opposed to the usual meta-learning. We show in Fig. 1a and Section 4 that the episodically fine-tuned Naive Bayes head consistently outperforms a conventional batch trained linear head in the low-shot transfer learning setting.

4 Experiments

In this section, we evaluate the classification accuracy and updateable parameter efficiency of FiT on the VTAB-1k [Zhai et al., 2019] benchmark. The VTAB-1k benchmark [Zhai et al., 2019] is a low to medium-shot transfer learning benchmark that consists of 19 datasets grouped into three distinct categories (natural, specialized, and structured). From each dataset, 1000 examples are drawn at random from the training split to use for the downstream dataset $\mathcal{D}$. After fine-tuning, the entire test split is used to evaluate classification performance. In all experiments, we use Big Transfer (BiT) [Kolesnikov et al., 2019], a leading, scalable, general purpose transfer learning algorithm as a point of comparison. In addition, we compare FiT to the latest vision transformer based methods that have reported the highest accuracies on VTAB-1k to date. See Appendix A.2 for experimental details.

Table 2 shows the classification accuracy and updateable parameter count for the three variants of FiT and BiT. The key observations from our results are:

- Both FiT-QDA and FiT-LDA outperform BiT on VTAB-1k.
- The FiT-QDA variant has the best overall performance, showing that the class covariance is important to achieve superior results on datasets that differ from those used in upstream pretraining (e.g. the structured category of datasets). However, the updateable parameter cost is high.
- FiT-LDA utilizes two orders of magnitude fewer updateable parameters compared to BiT, making it the preferred approach.

Table 2: **FiT outperforms BiT on VTAB-1k.** Classification accuracy and updateable parameter count (with 10 classes) for FiT variants and BiT on VTAB-1k with BiT-M-R50x1 backbone. Accuracy figures are percentages. Bold type indicates the highest scores. Green indicates summary columns.

			Natural							Specialized				Structured							
Method	Params (M) ↓	Overall Acc ↑	Caltech101	CIFAR100	Flowers102	Pets	Sun397	SVHN	DTD	EuroSAT	Resics45	Camelyon	Retinopathy	Clevr-count	Clevr-dist	dSprites-loc	dSprites-ori	sNORB-azi	sNORB-elev	DMLab	KITTI-dist
BiT	23.5	68.3	88.0	70.1	98.6	88.4	48.0	73.0	**72.7**	95.3	**85.9**	69.3	**77.2**	54.6	47.9	**91.6**	**65.9**	18.7	25.8	**47.1**	**80.1**
FiT-QDA	21.0	**70.6**	90.3	74.1	**99.1**	**91.0**	51.1	**75.1**	70.9	**95.6**	82.6	80.7	70.4	87.1	58.1	77.1	56.7	**18.9**	**40.4**	43.8	77.5
FiT-LDA	0.03	69.3	**90.4**	**74.2**	99.0	90.5	**51.6**	74.2	70.9	95.1	82.5	**82.5**	66.2	85.6	56.1	74.8	51.3	16.2	37.0	41.6	77.7
FiT-ProtoNets	0.03	65.5	89.6	73.9	98.6	90.8	51.5	50.1	68.2	93.8	77.0	79.9	57.9	**88.7**	**58.3**	68.6	34.2	13.5	35.0	39.3	75.3

Table 3 shows that FiT-LDA achieves state-of-the-art classification accuracy when compared to leading transfer learning methods pretrained on ImageNet-21k, while requiring the smallest number of updateable parameters and using the smallest backbone. All competing methods use a linear head.

Table 3: **FiT achieves SOTA on VTAB-1k**. Classification accuracy (%) for the 3 VTAB-1k categories (*Natural*, *Specialized*, and *Structured*) and mean accuracy over all 19 datasets (*Overall Acc*) and updateable parameter count (*Params*) for leading transfer learning methods using various backbones (backbone parameter count shown in parentheses) [Kolesnikov et al., 2019, Dosovitskiy et al., 2020, Tan and Le, 2021] pretrained on ImageNet-21k. ViT-Base-16 results from [Jie and Deng, 2022]. BiT results from [Kolesnikov et al., 2020]. Green indicates results summary columns.

Method	Backbone	Params (M) ↓	Overall Acc ↑	Natural ↑	Specialized ↑	Structured ↑
BiT [Kolesnikov et al., 2019]	BiT-M-R101x3 (382M)	382	72.7	80.3	**85.8**	59.4
BiT [Kolesnikov et al., 2019]	BiT-M-R152x4 (928M)	928	73.5	80.8	85.7	61.1
VPT [Jia et al., 2022]	ViT-Base-16 (85.8M)	0.5	69.4	78.5	82.4	55.0
Adapter [Houlsby et al., 2019]	ViT-Base-16 (85.8M)	0.2	71.4	79.0	84.1	58.5
AdaptFormer [Chen et al., 2022]	ViT-Base-16 (85.8M)	0.2	72.3	80.6	84.9	58.8
LoRA [Hu et al., 2021]	ViT-Base-16 (85.8M)	0.3	72.3	79.5	84.6	59.8
NOAH [Zhang et al., 2022]	ViT-Base-16 (85.8M)	0.4	73.2	80.3	84.9	61.3
Convpass [Jie and Deng, 2022]	ViT-Base-16 (85.8M)	0.3	74.4	81.7	85.3	62.7
FiT-LDA (ours)	EfficientNetV2-M (52.9M)	**0.15**	**74.9**	**82.2**	84.3	**63.7**

5 Discussion

In this work, we proposed FiT, a parameter and data efficient few-shot transfer learning system that allows image classification models to be updated with only a small subset of the total model parameters. We demonstrated that FiT can outperform BiT using fewer than 1% of the updateable parameters and achieve state-of-the-art accuracy on VTAB-1k.

6 Acknowledgements

This work has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service `https://www.hpc.cam.ac.uk` funded by EPSRC Tier-2 capital grant EP/P020259/1. Aliaksandra Shysheya, John Bronskill, Massimiliano Patacchiola and Richard E. Turner are supported by an EPSRC Prosperity Partnership EP/T005386/1 between the EPSRC, Microsoft Research and the University of Cambridge.

References

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

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Proceedings of the 28th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3320–3328, 2014.

James Requeima, Jonathan Gordon, John Bronskill, Sebastian Nowozin, and Richard E Turner. Fast and flexible multi-task classification using conditional neural adaptive processes. In *Proceedings of the 33rd Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 7957–7968, 2019.

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.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking few-shot image classification: a good embedding is all you need? *arXiv preprint arXiv:2003.11539*, 2020.

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.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7 (2):179–188, 1936.

Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons, 2012.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *Proceedings of the 31st Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 4077–4087, 2017.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. In *Proceedings of the 30th Annual Conference on Neural Information Processing Systems (NeurIPS)*, pages 3630–3638, 2016.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, pages 506–516, 2017.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8119–8127, 2018.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. *arXiv preprint arXiv:2203.12119*, 2022.

Shoufa Chen, Chongjian Ge, Zhan Tong, Jiangliu Wang, Yibing Song, Jue Wang, and Ping Luo. Adaptformer: Adapting vision transformers for scalable visual recognition. *arXiv preprint arXiv:2205.13535*, 2022.

Yuanhan Zhang, Kaiyang Zhou, and Ziwei Liu. Neural prompt search. *arXiv preprint arXiv:2206.04673*, 2022.

Shibo Jie and Zhi-Hong Deng. Convolutional bypasses are better vision transformer adapters. *arXiv preprint arXiv:2207.07039*, 2022.

Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew Howard. K for the price of 1: Parameter efficient multi-task and transfer learning. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=BJxvEh0cFQ`.

Massimiliano Patacchiola, John Bronskill, Aliaksandra Shysheya, Katja Hofmann, Sebastian Nowozin, and Richard E Turner. Contextual squeeze-and-excitation for efficient few-shot image classification. *arXiv preprint arXiv:2206.09843*, 2022.

Boris Oreshkin, Pau Rodríguez López, and Alexandre Lacoste. Tadam: Task dependent adaptive metric for improved few-shot learning. *Advances in neural information processing systems*, 31, 2018.

Eleni Triantafillou, Hugo Larochelle, Richard Zemel, and Vincent Dumoulin. Learning a universal template for few-shot dataset generalization. In *International Conference on Machine Learning*, pages 10424–10433. PMLR, 2021.

Peyman Bateni, Raghav Goyal, Vaden Masrani, Frank Wood, and Leonid Sigal. Improved few-shot visual classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14493–14502, 2020.

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.

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

Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, pages 10096–10106. PMLR, 2021.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Official repository for the "Big Transfer (BiT): General Visual Representation Learning" paper. `https://github.com/google-research/big_transfer`, 2020.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

A Appendix

A.1 FiT Training Algorithms

Algorithm A.1 and Algorithm A.2 detail how episodic fine-tuning tasks are split and sampled, respectively, for use in the FiT training protocol.

Algorithm A.1 Splitting the downstream dataset $\mathcal{D}$

Require: $\mathcal{D} = \{(\boldsymbol{x}_n, y_n)\}_{n=1}^{N} = \{\boldsymbol{x}, \boldsymbol{y}\}$: downstream dataset
Require: `unique()` $\equiv$ function that returns a list of unique classes and list of counts of each class
Require: `select_by_class()` $\equiv$ function that extracts samples of a specified class from a dataset
procedure SPLIT($\mathcal{D}$)
    $\mathcal{D}_{train} \leftarrow [\,]$ ▷ Create an empty list to hold $\mathcal{D}_{train}$
    $\mathcal{D}_{test} \leftarrow [\,]$ ▷ Create an empty list to hold $\mathcal{D}_{test}$
    `classes, class_counts` $\leftarrow$ `unique`$(\boldsymbol{y})$
    for all $c \in$ `classes` **do**
        `assert`(`class_counts`$(c) > 1)$ ▷ Require a minimum of 2 shots per class.
        `train_count` $\leftarrow$ `ceil`(`class_counts`$(c)/2)$
        $\mathcal{D}_c \leftarrow$ `select_by_class`(c) ▷ Select examples of class c from $\mathcal{D}$
        $\mathcal{D}_{train} \leftarrow \mathcal{D}_{train} + \mathcal{D}_c$[: `train_count`] ▷ Add `train_count` examples to $\mathcal{D}_{train}$
        $\mathcal{D}_{test} \leftarrow \mathcal{D}_{test} + \mathcal{D}_c$[`train_count` :] ▷ Add remaining examples to $\mathcal{D}_{test}$
    end for
    return $\mathcal{D}_{train}, \mathcal{D}_{test}$
end procedure

Algorithm A.2 Sampling a task τ

Require: $\mathcal{D}_{train} = \{(\boldsymbol{x}_s, y_s)\}_{s=1}^{S_\tau} = \{\boldsymbol{x}_S, \boldsymbol{y}_S\}$: train portion of downstream dataset
Require: $\mathcal{D}_{test} = \{(\boldsymbol{x}_q, y_q)\}_{q=1}^{Q_\tau} = \{\boldsymbol{x}_Q, \boldsymbol{y}_Q\}$: test portion of downstream dataset
Require: `support_set_size`: size of the support set $|\mathcal{D}_S^\tau|$
Require: `unique()` $\equiv$ function that returns a list of unique classes and list of counts of each class
Require: `randint`(min, max) $\equiv$ function that returns a random integer between min and max
Require: `choice`$(range, count)$ $\equiv$ function that returns a random list of $count$ integers from $range$
procedure SAMPLE_TASK($\mathcal{D}_{train}, \mathcal{D}_{test},$ `support_set_size`)
    $\mathcal{D}_S^\tau \leftarrow [\,]$ ▷ Create an empty list to hold $\mathcal{D}_S^\tau$
    $\mathcal{D}_Q^\tau \leftarrow [\,]$ ▷ Create an empty list to hold $\mathcal{D}_Q^\tau$
    `train_classes, train_class_counts` $\leftarrow$ `unique`$(\boldsymbol{y}_S)$
    `test_classes, test_class_counts` $\leftarrow$ `unique`$(\boldsymbol{y}_Q)$
    `min_way` $\leftarrow$ `min`(`len`(`train_classes`)$, 5)$
    `max_way` $\leftarrow$ `min`(`len`(`train_classes`), `support_set_size`)
    `way` $\leftarrow$ `randint`(`min_way`, `max_way`) ▷ Classification way to use for this task
    `selected_classes` $\leftarrow$ `choice`(`train_classes`, `way`) ▷ List of classes to use in this task
    `balanced_shots` = `max`(`round`(`support_set_size` / `len`(`selected_classes`)), 1)
    `max_test_shots` $\leftarrow$ `max`$(1,$ `floor`$(2000/$`way`$))$
    for all $c \in$ `selected_classes` **do**
        `class_shots` $\leftarrow$ `train_class_counts`(c)
        `shots_to_use` $\leftarrow$ `min`(`class_shots`, `balanced_shots`)
        `selected_shots` $\leftarrow$ `choice`(`class_shots`, `shots_to_use`) ▷ Support shot list
        $\mathcal{D}_S^\tau \leftarrow \mathcal{D}_S^\tau + \mathcal{D}_{train}$[`selected_shots`] ▷ Add examples to $\mathcal{D}_S^\tau$
        `class_shots` $\leftarrow$ `test_class_counts`(c)
        `shots_to_use` $\leftarrow$ `min`(`class_shots`, `max_test_shots`)
        `selected_shots` $\leftarrow$ `choice`(`class_shots`, `shots_to_use`) ▷ Query shot list
        $\mathcal{D}_Q^\tau \leftarrow \mathcal{D}_Q^\tau + \mathcal{D}_{test}$[`selected_shots`] ▷ Add examples to $\mathcal{D}_Q^\tau$
    end for
    return $\mathcal{D}_S^\tau, \mathcal{D}_Q^\tau$
end procedure

A.2 Training and Evaluation Details

In this section, we provide implementation details for the experiments in Section 4.

FiT All of the FiT versus BiT on VTAB-1k transfer learning experiments were carried out on a single NVIDIA A100 GPU with 80GB of memory. The Adam optimizer [Kingma and Ba, 2015] with a constant learning rate of 0.0035, for 400 iterations, and $|\mathcal{D}_S^\tau|$=100 was used throughout. No data augmentation was used and images were scaled to 384×384 pixels unless the image size was 32×32 pixels or less, in which case the images were scaled to 224×224 pixels.

FiT-QDA, FiT-LDA, and FiT-ProtoNets take approximately 12, 10, and 9 hours, respectively, to fine-tune on all 19 VTAB datasets.

For the FiT-LDA results using the EfficientNetV2-M backbone in Table 3, we used 1000 iterations instead of 400 and ran the experiments on 4 NVIDIA A100 GPUs, each with 80GB of memory.

BiT For the BiT VTAB-1k experiments, we used the three fine-tuned models for each of the datasets that were provided by the authors [Kolesnikov et al., 2020]. We evaluated all of the models on the respective test splits for each dataset and averaged the results of the three models. The BiT-HyperRule [Kolesnikov et al., 2019] was respected in all runs. These experiments were executed on a single NVIDIA GeForce RTX 3090 with 24GB of memory.