META-LEARNING INITIALIZATIONS FOR IMAGE SEGMENTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

While meta-learning approaches that utilize neural network representations have made progress in few-shot image classification, reinforcement learning, and, more recently, image semantic segmentation, the training algorithms and model architectures have become increasingly specialized to the few-shot domain. A natural question that arises is how to develop learning systems that scale from few-shot to many-shot settings while yielding human level performance in both. One scalable potential approach that does not require ensembling many models nor the computational costs of relation networks, is to meta-learn an initialization. In this work, we study first-order meta-learning of initializations for deep neural networks that must produce dense, structured predictions given an arbitrary amount of training data for a new task. Our primary contributions include (1), an extension and experimental analysis of first-order model agnostic meta-learning algorithms (including FOMAML and Reptile) to image segmentation, (2) a formalization of the generalization error of episodic meta-learning algorithms, which we leverage to decrease error on unseen tasks, (3) a novel neural network architecture built for parameter efficiency which we call EfficientLab, and (4) an empirical study of how meta-learned initializations compare to ImageNet initializations as the training set size increases. We show that meta-learned initializations for image segmentation smoothly transition from canonical few-shot learning problems to larger datasets, outperforming random and ImageNet-trained initializations. Finally, we show both theoretically and empirically that a key limitation of MAML-type algorithms is that when adapting to new tasks, a single update procedure is used that is not conditioned on the data. We find that our network, with an empirically estimated optimal update procedure yields state of the art results on the FSS-1000 dataset, while only requiring one forward pass through a single model at evaluation time.

1 INTRODUCTION

Humans have a remarkable capability to not only learn new concepts from a small number of labeled examples but also to gain expertise as more data becomes available. In recent years, there has been substantial progress towards human-level image segmentation in the high data regime (see Liu et al. (2019) and their references). While meta-learning approaches that utilize neural network representations have made progress in few-shot image classification, reinforcement learning, and, more recently, image semantic segmentation, the training algorithms and model architectures have become increasingly specialized to the low data regime. A desirable property of a learning system is one that effectively applies knowledge gained from a few *or* many examples, while reducing the generalization gap when trained on little data and not being encumbered by its own learning routines when there are many examples. A natural question that arises is how to develop learning systems that scale from few-shot to many-shot settings while yielding human level performance in both. One scalable potential approach that does not require ensembling many models nor the computational costs of relation networks, is to meta-learn an initialization.

In this work, we specifically address the problem of meta learning initializations for deep neural networks that must produce dense, structured output, such as for the semantic segmentation of images. We ask the following questions:

1. Do first-order MAML-type algorithms extend to the higher dimensional parameter spaces, dense prediction, and skewed distributions required of semantic segmentation?
2. How robust are the representations that the model has meta-learned to perturbations in the hyperparameters of the update routine?
3. Are MAML-type algorithms hindered by having a fixed update policy for training and testing tasks that is not conditioned on the data?

Through a series of theoretical and empirical analyses, we shed new light on the representations that model agnostic meta-learning algorithms learn and how they adapt to unseen tasks. We show that they extend to few shot image segmentation, yielding state of the art results when their update routine is optimized after meta-training. We will open-source our code.

2 Related Work

Learning useful models from a small number of labeled examples of a new concept has been studied for decades (Thrun, 1996) yet remains a challenging problem with no semblance of a unified solution. The advent of larger labeled datasets containing examples from many distinct concepts (Vinyals et al., 2016) has enabled progress in the field in particular by enabling approaches that leverage the representations of nonlinear neural networks. Image segmentation is a well-suited domain for advances in few-shot learning given that the labels are particularly costly to generate (Wei et al., 2019).

Recent work in few-shot learning for image segmentation has utilized three key components: (1) model ensembling (Shaban et al., 2017), (2) relation networks (Santoro et al., 2017), and (3) late fusion of representations (Rakelly et al., 2018; Wei et al., 2019). The inference procedure of ensembling models with a separately trained model for each example has been shown to produce better predictions than single shot approaches but will scale linearly in time and/or space complexity (depending on the implementation) in the number of training examples, as implemented in Shaban et al. (2017). An exciting recent approach is that of using relation networks to learn modules that can reason about the relationships of examples in a domain (Santoro et al., 2017). Relation networks have seen increased adoption in meta-learning systems (Rusu et al., 2018) and were recently employed in few-shot segmentation in Zhang et al. (2019) and Wei et al. (2019). While relation networks yield impressive results in the few-shot domain, they typically come with $O(n^2)$ training costs where n is the number of support and query examples. The use of multiple passes through subnetworks via iterative optimization modules used by Zhang et al. (2019) further exacerbate these costs. The authors in Wei et al. (2019) take an elegant approach to reducing the computational complexity by element-wise summing all feature maps of the support examples over the channel dimension, such that the encoded feature map tensors have the same depth regardless of the size of the support set, forming a type of embedded memory of the class and then relating that to the query examples. While this approach reduces computational complexity of traditional applications of relation networks, its efficacy in scaling as more training data becomes available remains unclear.

Model Agnositc Meta-Learning (MAML) is a gradient-based meta-learning approach introduced in Finn et al. (2017). First Order MAML (FOMAML) reduces the computational cost by not requiring the backpropogating the meta-gradient through inner-loop gradient and has been shown to work similarly well on classification tasks (Finn et al., 2017; Nichol & Schulman, 2018). Though learning an initialization has the potential to unify few-shot and many-shot domains, initializations learned from MAML-type algorithms have been seen to overfit in the low-shot domain when adapting sufficiently expressive models such as deep residual networks that may be more than a small number of convolutional layers [1] (Mishra et al., 2017; Rusu et al., 2018). In addition to possessing potential to unify few- and many-shot domains, MAML-type algorithms are intriguing in that they impose no constraints on model architecture, given that the output of the meta-learning process is simply an initialization. Futhermore, the meta-learning dynamics, which learn a temporary memory of a sampled task, are related to the older idea of fast weights (Hinton & Plaut, 1987; Ba et al., 2016). Despite being dataset size and model architecture agnostic, MAML-type algorithms are unproven for high dimensionality of the hypothesis spaces and the skewed distributions of image segmentation problems data (Rakelly et al., 2018). In this work, we show, for the first time, that model agnostic

[1]The original MAML and Reptile convolutional neural networks (CNNs) use four convolutional layers with 32 filters each for MiniImagenet (Finn et al., 2017; Nichol & Schulman, 2018)

meta-learning algorithms do in fact naturally extend to image segmentation, yielding state of the art results when their update procedure is optimized.

3 Preliminaries

In this section, we describe the FOMAML and Reptile meta-learning algorithms and introduce notation used throughout this paper. We assume access to a distribution over tasks $p(\tau)$ that is sampled from a generating distribution $\mathbb{P}$ and contains example tasks τ_i. Let τ_s denote a finite sample of examples from τ_i. The meta-learning algorithms introduced in Finn et al. (2017) and Nichol & Schulman (2018), work by sampling a task, τ_s, adapting a model's internal representations, θ, for a small number of gradient updates, j, to produce θ_i' after j steps of gradient descent. In first-order renditions, a finite difference is then taken to approximate the derivative of the meta-learning procedure between θ_i' and θ and a step of size α is taken in that direction. FOMAML differs from Reptile in that τ_i is split into mini-training, $\mathcal{D}^{tr}$, and validation, $\mathcal{D}^{val}$, sets. The last batch of the inner-loop is used to "correct" for overfitting to $\mathcal{D}^{tr}$ by taking a gradient step in the direction of descent pointed to by the loss, $\mathcal{L}$, on $\mathcal{D}^{val}$. Now that we are equipped with an understanding of the of the dynamics of the FOMAML and Reptile algorithms we can turn to analyzing their generalization performance.

4 Formalizing the generalization error of meta-learning algorithms

We now formalize the generalization error of arbitrary meta-learning algorithms. We do this by defining the common components that meta-learning systems utilize at test-time when adapting to a new task and how those relate to the expected risk of a learned function. Meta-learning systems have the following common components, among others:

1. A representation to store pre-existing knowledge, θ,
2. A sampled dataset to learn from, τ_s, that is typically small,
3. A loss function $\mathcal{L}(f_\theta(\mathbf{X}), \mathbf{Y})$ that returns a cost of the difference between predctions and labels, $f_\theta(\mathbf{X})$ and $\mathbf{Y}$ respectively, and
4. An update procedure, U, that is parameterized by *both* an initialization θ and hyperparameters that define its dynamics $\mathbb{H}$. U then uses τ_s to generate a new representation θ':

$$f_{\theta'} \leftarrow U(\tau_s; \theta, \mathbb{H}) \tag{1}$$

Without loss of generality, the adaptation hyperparameters, $\mathbb{H}$, can be meta-learned by a meta-learning algorithm, $\mathcal{A}$, or defined upfront as is done in the original formulations of MAML, FOMAML, and Reptile.

Maintaining this nomenclature, we now turn to the generalization error of a meta-learning system. Let $\mathbb{R}[f_\theta]$ be the expected risk of the function f parameterized by θ. For non-convex, non-smooth loss functions for which $\mathbb{R}[f_\theta]$ is non-computable, the empirical risk, $\mathbb{R}_s[f_\theta]$, is minimized, often with stochastic gradient descent (SGD), on the sample, τ_s:

$$f_\theta^{*\tau_s} = \underset{\theta}{\text{minimize}}(E_{\tau_s}[\mathcal{L}(f_\theta)]) = minimize_\theta \mathbb{R}_s[f_\theta] \tag{2}$$

where $f_\theta^{*\tau_s}$ is the function [2] that minimizes the expectation of the loss of the examples in τ_s. The generalization gap of a meta-learning system is then the difference between the expected risk of the non-computable functions, $f_\theta^{*\tau_i}$, that would minimize the risk on the generating distribution of the task $p(\tau_i)$ and the expectation of the risk of the functions $f_\theta^{*\tau_s}$ learned on samples τ_s in $p(\tau)$:

$$E_{\tau\sim\mathbb{P}}[\mathbb{R}[f_\theta^{*\tau_i}] - \mathbb{R}[f_\theta^{*\tau_s}]] \tag{3}$$

We now specialize these analyses to meta-learning algorithms that use a fixed update procedure such as a predefined learning rate, α and number of gradient steps j. This class of algorithms includes

[2]It is important to note that this function, and it's definition as the term $f_\theta^{*\tau_s}$, is different from $f_{\theta'}$, which is the function returned by the fixed update routine U and not necessarily the minimizer of the empirical loss $\mathcal{L}(f_\theta(\mathbf{X}), \mathbf{Y})$.

the implementations of MAML, FOMAML, and Reptile. These algorithms are aimed at minimizing the expectation of the loss over tasks in τ:

$$\underset{\theta}{\text{minimize}} E_{\tau\sim\mathbb{P}}\left[\mathcal{L}_{\tau}\left(U(\tau_i;\theta\mathbb{H})\right)\right] \tag{4}$$

Because the risk is defined in terms of the parameters θ' that are learned by $U(\tau_s;\theta,\mathbb{H})$, there will also be an optimal set of hyperparameters, $\mathbb{H}^*$ for every θ, which may or may not be equal to $\mathbb{H}$. Thus, the expectation of the loss with respect to $\mathbb{H}$ given a fixed θ can also be minimized:

$$\underset{\mathbb{H}}{\text{minimize}} E_{\tau\sim\mathbb{P}}[\mathcal{L}(U(\tau_s;\theta,\mathbb{H}))] \tag{5}$$

Furthermore, for a fixed update routine that is not conditioned on the data, each parameter, w_i, can travel a maximum distance from its initialization d

$$d = \sum_j \alpha_j \tag{6}$$

A corollary that follows from this is that if minima, potentially with low generalization error, are outside of the n-dimensional sphere reachable by $U(\tau_s;\theta,\mathbb{H})$, they will not be found by U given the same hyperparameters $\mathbb{H}$ used during meta-training. Thus, on many problems, it may be useful to estimate an optimal update procedure on the available data given the meta-learned initialization.

We apply this insight to developing a simple update hyperparameter optimization (UHO) algorithm. The routine samples n hyperparameter values within a predefined, broad range, evaluates each of the hyperparameters on a sample of tasks, τ^{val}. After evaluating all τ_s in τ^{val}, UHO defines a range around the $x\%$ best hyperparameter configurations and samples from that space, repeating until a predefined computational budget is exhausted or the expectation of the loss is no longer reduced. Finally, the best configuration of the hyperparameters that was seen is returned. This algorithm can be a viewed as a variant of Successive Halving (Jamieson & Talwalkar, 2016).

5 EXPERIMENTS

We evaluate the FOMAML and Reptile meta-learning algorithms on the FSS-1000 dataset. Model topology development and hyperparameter search was done on a held out set of validation tasks and not the final test tasks. For the final evaluations, we train for $\sim$200 epochs through the training and validation tasks, using a meta-batch size of 5, an inner loop batch size of 8, and 5 inner loop iterations. During training, we use SGD in the inner loop with a fixed learning rate of 0.005. After we have learned an initialization, we then applied the UHO algorithm on 100 randomly sampled tasks from the training set to search over the learning rate. Because the effects of the learning rate are intertwined with the number of gradient updates we also use early stopping (ES) to estimate the optimal number of gradient steps when adapting to a new task.

5.1 DATASETS

The first few-shot image segmentation dataset was the PASCAL-5^i presented in Shaban et al. (2017) which reimagines the PASCAL dataset (Everingham et al., 2010) as a few-shot binary segmentation problem for each of the classes in the original dataset. Unfortunately, the dataset contains relatively few distinct tasks (20 excluding background and unlabeled). The idea of a meta-learning dataset for image segmentation was further developed with the recently introduced FSS-1000 dataset, which contains 1000 classes, 240 of which are dedicated to the test-set, with 10 image-mask pairs for each class (Wei et al., 2019). This dataset is the focus of the empirical work of this paper.

For investigating how the meta-learned representations integrate new information as more data becomes a available, we put together a small dataset that we call FP-k. FP-k takes 5 tasks from FSS-1000 and 5 tasks from PASCAL-5^i for the same concept[3] Using this dataset, we train over a range of "k"-training shots from ImageNet-trained initializations[4] and our meta-learned initializations.

[3]See the Appendix for more details on the dataset construction.

[4]The encoder is trained on ImageNet, while the lightweight skip decoder and final layer weights are initialized using the Glorot uniform initialization (Glorot & Bengio, 2010)

5.2 Neural Network Architecture

To extend first-order MAML-type algorithms to more expressive models, with larger hypothesis spaces, while yielding state of the art few-shot learning results, we developed a novel neural network architecture, which we term EfficientLab. The top level hierarchy of the network's organization of computational layers is similar to Chen et al. (2018), with 4 convolutional blocks that successively halve the features in spatial resolution while increasing the number of feature maps. This is followed by a 4x bilinear upsampling which is concatenated with features from a long skip connection from the second downsampling block in the encoding part of the network. The concatenated low and high resolution features are then fed through an atrous spatial pyramid pooling (ASPP) module and finally bilinearly upsampled to original image size.

The differences between our model and the original DeepLabV3+ model are in (1) the encoder network used and (2) how the low resolution embedded features are upsampled to full resolution predictions. For the encoding subnetwork, we utilize the recently proposed EfficientNet (Tan & Le, 2019). After encoding the images, instead of feeding them directly into an atrous spatial pyramid pooling module (ASPP), we first immediately bilinearly upsample the features by 4x. The upsampled features are then concatenated with features from the second downsampling block. Moving the ASPP module to the usampled resolution provides two advantages. First, it allows us to use 1 convolutional module in place of two. Due to the high dimensionality of the features along the channel axis at the lowest resolution, the convolutional kernels are especially expensive in terms of number of parameters. Second, the ASPP module is designed to learn multiscale context which could be useful in refining the boundaries of semantic features in mid-resolution feature maps. Our ASPP module utilizes three parallel branches of a $1x1$ convolution, $3x3$ convolution with dilation rate = 2, and a simple average-pooling across spatial dimensions of the feature maps. The output of the three branches is concatenated and fed into a final $3x3$ convolutional layer with 112 filters. We call this structure a lightweight skip decoder (LSD). A residual connection wraps around the LSD to ease gradient flow. Before the final $1x1$ convolution that produces the unnormalized heatmap of class scores, we use a single layer of dropout with a drop rate probability = 0.2 [5]. We use the standard softmax to produce the normalized predicted probabilities.

We use batch normalization layers following convolutional layers (Ioffe & Szegedy, 2015). We meta-learn the β and γ parameters, adapt them at test time to test tasks, and use running averages as estimates for the population mean and variance, $E[x]$ and $Var[x]$, at inference time as suggested in Antoniou et al. (2018). *All* parameters at the end of an evaluation call are reset to their pre-adaptation values to stop information leakage between the training and validation sets. The network is trained with the binary cross entropy minus the log of the dice score:

$$\mathcal{L} = H - log(\mathcal{J}) \tag{7}$$

where H is binary cross entropy loss:

$$H = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1 - y_i)\log\left(1 - \hat{y}_i\right)\right) \tag{8}$$

J is the modified Dice score:

$$J = \frac{2IoU}{IoU + 1} \tag{9}$$

and IoU is the intersection over union metric:

$$IoU = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i\hat{y}_i + \epsilon}{y_i + \hat{y}_i - y_i\hat{y}_i + \epsilon}\right) \tag{10}$$

5.3 Results

The results of our model with an initialization meta-learned using Reptile and FOMAML are shown in Table 1. We find that our model trained with FOMAML and importantly with improved use of batch normalization yields near state of the art results. After optimizing the update hyperparameters,

[5]As described in Li et al. (2019) and used in Tan & Le (2019) the dropout layer is applied after all batch norm layers.

Table 1: FSS-1000 5-shot Results

Shots	**Method**	**MeanIoU**
1	FSS-1000 Baseline(Wei et al., 2019)	73.47%
5	FSS-1000 Baseline(Wei et al., 2019)	80.12%
1	FOMAML	73.26%
1	**FOMAML+UHO**	**73.86%**
5	Reptile	69.63%
5	FOMAML	79.02%
5	**FOMAML+UHO**	**81.39%**

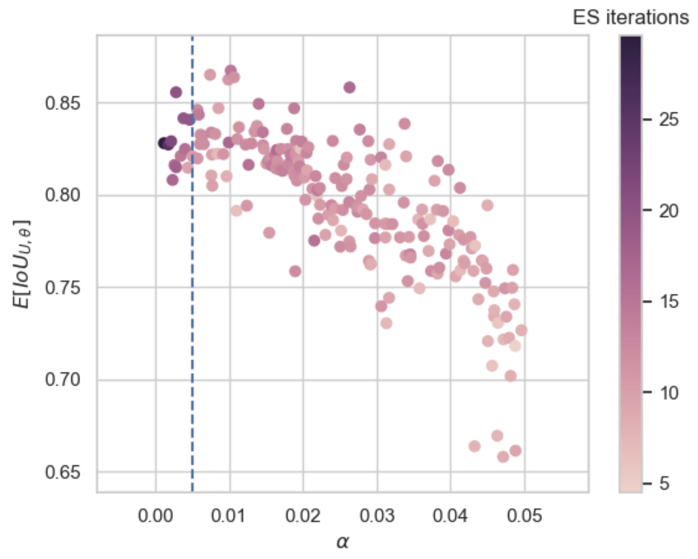

Figure 1: Each point represents the mean IoU for the validation tasks with a sampled learning rate α. The blue dashed line indicates the learning rate used by SGD in the inner loop during meta-training. Points are colored by how many iterations they were trained before stopped by early stopping (ES) with a patience of 3 iterations.

our approach sets the new state of the art for the FSS-1000 dataset by a small margin. We also find that by searching through a range of learning rates that are $10x$ less to $10x$ greater than the learning rate, that the learned representations are not robust to such large variations in the hyperparameter 1.

In this work we posit that a fixed update procedure that is used at test time and not conditioned on the training data is one of major hinderances of MAML-type algorithms. In section 4, we show that minima with low generalization error may be unreachable via a single update routine from a single parameter initialization θ. We find this analysis to be supported empirically as well. We find that: (1) the estimated optimal hyperparameters for the update routine even on the *training* tasks are not the same as those specified a priori during meta-training, as illustrated in Figure 1. One may expect that MAML-type algorithms would converge to a point in parameter space from which optimal minima for each of the training tasks are reachable. We find that even after 200 epochs through the training set, this was not the case. The best learning rate and number of iterations we found via the random search UHO algorithm were 0.007475495476045889 and 8, respectively, compared to a learning rate of 0.005 and 5 iterations used during training. (2) Optimizing the hyperparameters (even on the training set) improves test-time results on unseen tasks.

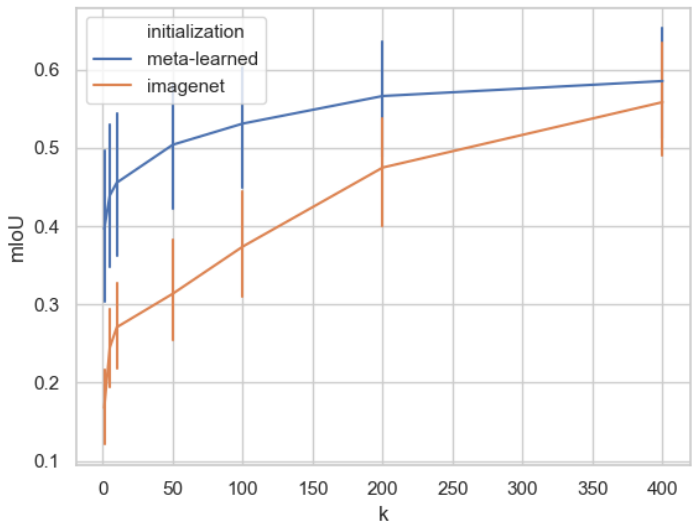

Figure 2: Mean IoU results as a function of the training set size of our EfficientLab model trained on the FP-k dataset. The meta-learned initialization outperformed EfficientLab initialized with an ImageNet-trained encoder and a randomly initialized decoder for all numbers of labeled training examples that we evaluated.

By training our model on the FP-k dataset, we also found that our meta-learned initializations outperformed an ImageNet-trained encoder and a randomly initialized decoder for up to 400 training examples[6].

6 Discussion

In this work, we showed that gradient-based first order model agnostic meta-learning algorithms do in fact extend to the high dimensionality of the hypothesis spaces and the skewed distributions of few-shot image segmentation problems. Furthermore, we find that the representations that are meta-learned smoothly transition as more data becomes available, unifying few- and many-shot regimes.

Future work could look more critically at learned update procedures, forms of meta-regularization, and second order methods for image segmentation. It would also be useful in future work to take a more critical look at the interplay between batch normalization and meta-learning. While single task deep neural networks in large data regimes apply batch normalization with a consistent pattern, different groups working in few-shot meta-learning have incorporated batch norm in completely different ways such as by: (1) not using it at all for the meta-learning components (Rusu et al., 2018), (2) not using learned β and γ parameters at all while still using estimated population means and variances during inference Zhang et al. (2019), or (3) meta-learning β and γ while only using batch statistics for the normalization (Finn et al., 2017; Nichol & Schulman, 2018), or (4) meta-learning β and γ and also using population estimates of the mean and variance, as done conventionally when training deep neural networks in the large data regime, which is the approach that we adopt and find to be most useful.

We showed that the optimal hyperparameter configuration for the update procedure may not be the same configuration used during meta-learning. These findings are supported by our theoretical

[6]The examples in the PASCAL dataset are known to be more challenging than the FSS-1000 dataset (Wei et al., 2019). From visual inspection of the two datasets, it is also clear that the PASCAL dataset contains more label noise than the FSS-1000 dataset. For these reasons, the mean IoU values shown in Figure 1, which contain examples from both datasets, are not directly comparable to the results shown in Table 1, which contain examples only from FSS-1000.

analyses which show that MAML-type algorithms minimize the empirical risk on the training set of the update procedure and the initialization θ. We suspect that improvements realized by relation networks (Wei et al., 2019; Zhang et al., 2019; Rusu et al., 2018), models that learn to generate parameters conditioned on the training data (Rusu et al., 2018; Shaban et al., 2017), and models with learned learning rates (Antoniou et al., 2018) directly leverage information on *how* to adapt given a few-shot sample. It is our hope that our empirical analyses and formalization of the generalization error of meta-learning systems leads to better explanations of why some meta-learning systems work better than others in different problem spaces. Lastly, we hope that this work draws, what we argue is necessary, attention to the open problem of building learning systems that can unify low and high data regimes by gaining expertise and smoothly integrating new information as more data becomes available, much as people do.

REFERENCES

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

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. In *Advances in Neural Information Processing Systems*, pp. 4331–4339, 2016.

Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 801–818, 2018.

Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338, 2010.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

Geoffrey E Hinton and David C Plaut. Using fast weights to deblur old memories. In *Proceedings of the ninth annual conference of the Cognitive Science Society*, pp. 177–186, 1987.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Artificial Intelligence and Statistics*, pp. 240–248, 2016.

Xiang Li, Shuo Chen, Xiaolin Hu, and Jian Yang. Understanding the disharmony between dropout and batch normalization by variance shift. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2682–2690, 2019.

Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 82–92, 2019.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alyosha Efros, and Sergey Levine. Conditional networks for few-shot semantic segmentation. 2018.

Andrei A Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization. *arXiv preprint arXiv:1807.05960*, 2018.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pp. 4967–4976, 2017.

Amirreza Shaban, Shray Bansal, Zhen Liu, Irfan Essa, and Byron Boots. One-shot learning for semantic segmentation. *arXiv preprint arXiv:1709.03410*, 2017.

Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

Sebastian Thrun. Is learning the n-th thing any easier than learning the first? In *Advances in neural information processing systems*, pp. 640–646, 1996.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pp. 3630–3638, 2016.

Tianhan Wei, Xiang Li, Yau Pun Chen, Yu-Wing Tai, and Chi-Keung Tang. Fss-1000: A 1000-class dataset for few-shot segmentation. *arXiv preprint arXiv:1907.12347*, 2019.

Chi Zhang, Guosheng Lin, Fayao Liu, Rui Yao, and Chunhua Shen. Canet: Class-agnostic segmentation networks with iterative refinement and attentive few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5217–5226, 2019.

A FP-K DATASET

The 5 tasks in PASCAL-5^i that have direct analogs in FSS-1000, which we use, are: `[{"airliner", "aeroplane"}, {"bus"}, {"motorbike"}, {"potted_plant", "potted plant"}, {"television", "tvmonitor"}]`
Each entry in this list is a set of 1 or 2 strings (if they differ between the two parent datasets) for the names of tasks in FSS-1000 and PASCAL-5^i, respectively. We combine all examples for synonymous tasks. During evaluation, we simply randomly sample 20 test examples, and sample a training set of k examples over the range: `[1, 5, 10, 50, 100, 200, 400]`

B FSS-1000 TEST TASKS

At the time of this writing, the authors of Wei et al. (2019) had not released the train-val-test splits of their work, noting that “The train/validation/test split used in the experiments consists of 5,200/2,400/2,400 image and label pairs.” For our test set, we randomly sampled the following 240 tasks (which contain 2,400 image-label pairs in total):

```
["chess_bishop", "lifeboat", "african_grey", "kobe_logo", "ab_wheel",
"face_powder", "gourd", "ski_mask", "stool", "vine_snake", "ladle",
"wandering_albatross", "agama", "school_bus", "ballpoint", "radio",
"television", "orange", "mitten", "pizza", "sealion", "toaster",
"window_shade", "corn", "albatross", "african_elephant", "pidan", "taxi",
"cardoon", "indian_elephant", "moon", "pistachio", "brambling",
"fire_screen", "quail_egg", "yawl", "pencil_box", "rhinoceros",
"bighorn_sheep", "diver", "bison", "golden_retriever", "stretcher",
"prairie_chicken", "zebra", "water_heater", "espresso_maker", "sundial",
"stop_sign", "microsd", "mite_predator", "sports_car", "mount_fuji",
"pyramid", "truss_bridge", "giant_panda", "walnut", "hippo",
"loggerhead_turtle", "avocado", "partridge", "telescope", "fan",
"egyptian_cat", "tile_roof", "potato_chips", "chinese_knot", "cradle",
"garbage_can", "guitar", "airship", "spoonbill", "pencil_sharpener2",
"acorn", "turnstile", "cn_tower", "typewriter", "microscope",
"hornbill", "ashtray", "scorpion", "vestment", "combination_lock",
"hare", "nintendo_gba", "titi_monkey", "golfcart", "garbage_truck",
"candle", "pyraminx", "nagoya_castle", "beaker", "scissors",
"feather_clothes", "american_alligator", "fish_eagle", "abe's_flyingfish",
"streetcar", "jacko_lantern", "taj_mahal", "stole", "sunscreen",
"poached_egg", "aubergine", "calculator", "lobster", "swimming_trunk",
"power_drill", "windsor_tie", "sandwich_cookies", "kazoo",
"indian_cobra", "soap_dispenser", "spotted_salamander", "strawberry",
"warplane", "flying_squirrel", "tennis_racket", "dragonfly", "crepe",
"skull", "starfish", "shovel", "electronic_toothbrush", "bra",
"chess_knight", "white_shark", "red_fox", "smoothing_iron", "colubus",
"flying_disc", "excavator", "toothbrush", "peregine_falcon",
"polo_shirt", "magpie_bird", "apple_icon", "triumphal_arch", "bottle_cap",
"great_wall", "ac_wall", "manatee", "pinwheel", "nintendo_3ds",
"pinecone", "melon_seed", "ruler", "wrench", "throne", "pomegranate",
"killer_whale", "tiger", "shih-tzu", "croquet_ball", "conch",
"oscilloscope", "frog", "coho", "golf_ball", "monitor", "sparrow",
"paper_plane", "tiger_cat", "seal", "buckingham_palace",
"waffle_iron", "sewing_machine", "paper_crane", "dumbbell", "quail",
"band-aid", "beer_glass", "ox", "hawk", "spider_monkey",
"pufferfish", "cuckoo", "carrot", "sweatshirt", "terrapin_turtle",
"arctic_fox", "eletrical_switch", "african_crocodile",
"electronic_stove", "timber_wolf", "hammer", "wagtail",
"cushion", "angora", "mario", "paper_towel", "kit_fox", "snail",
"snowball", "gazelle", "fig", "cloud", "nike_logo", "scarerow",
"lacewing", "letter_opener", "mcdonald_sign", "eft_newt", "black_swan",
"file_cabinet", "turtle", "assult_rifle", "sandbar", "monkey",
"common_newt", "rocket", "triceratops", "siamese_cat", "panda",
"mountain_tent", "mortar", "adhensive_tape", "afghan_hound", "mule",
"tomb", "air_strip", "orang", "space_shuttle", "aircraft_carrier",
"tofu", "cello", "water_buffalo", "sulphur_crested", "garlic",
"besom", "baby", "beer_bottle", "pingpong_ball", "coucal", "grey_whale",
"drumstick", "banjo", "flamingo", "plate", "hover_board"]
```