# META-LEARNING INITIALIZATIONS FOR IMAGE SEGMENTATION

**Sean M. Hendryx**
School of Information, University of Arizona
seanmhendryx@email.arizona.edu

**Andrew B. Leach**
Program in Applied Mathematics, University of Arizona *
ableach@email.arizona.edu

**Paul D. Hein**
School of Information, University of Arizona
pauldhein@email.arizona.edu

**Clayton T. Morrison**
School of Information, University of Arizona
claytonm@email.arizona.edu

## 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 competitive 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 novel neural network architecture built for parameter efficiency and fast learning which we call EfficientLab, (3) a formalization of the generalization error of meta-learning algorithms, which we leverage to decrease error on unseen tasks, and (4) a small benchmark dataset, FP-k, for the empirical study of how meta-learning systems perform in both few- and many-shot settings. We show that meta-learned initializations for image segmentation provide value for both canonical few-shot learning problems and larger datasets, outperforming ImageNet-trained initializations for up to 400 densely labeled examples. 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

In recent years, there has been substantial progress in high accuracy 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. This property is desirable because training and maintaining multiple models is more cumbersome than training and maintaining one model. A natural question that arises is how to develop learning systems that scale from few-shot to many-shot settings while yielding competitive accuracy 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 such as via Model Agnostic Meta-Learning (MAML) (Finn et al., 2017).

*Now at Google

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 sensitive is the test-time performance of gradient-based meta-learning to the hyperparameters of the update routine used to adapt the initialization to new tasks?
3. What is the number of labeled examples beyond which a conventional approach to training deep neural networks outperforms the meta-learned initializations?

To address the third question, we put together a small benchmark dataset, which we call FP-k, that contains 400 training examples for 5 tasks each. In recent works (Li et al., 2017; Shaban et al., 2017; Rusu et al., 2018; Zhang et al., 2019; Lee et al., 2019), few-shot learning approaches have become increasingly complex and appear to be specialized to the few-shot domain. Such specialization leaves many open questions, such as: What is the accuracy of a few-shot learning system when more labeled examples become available? After a certain number of labeled examples, will the few-shot learning system have the same accuracy as a simpler training approach such as conventional training via SGD? If so, what is the number of labeled examples beyond which a conventional approach to training deep neural networks outperforms a meta-learning system? We address these questions in 5.4, providing empirical justification for our meta-learning approach for up to 400 densely labeled examples.

In summary, we address the above research questions as follows: We show that MAML-type algorithms do extend to few shot image segmentation, yielding state of the art results when their update routine is optimized after meta-training and when the model is regularized. Addressing question 2, we find that the meta-learned initialization's performance when being evaluated on a task is particularly sensitive to changes in the update routine's hyperparameters (see Figure 2). We show theoretically in section 3.3 and empirically in our results (see Table 2) that a single update routine used both during meta-training and meta-testing may not have optimal generalization. Finally, we address question 3 by showing that our meta-learned initializations are competitive with ImageNet (Deng et al., 2009) trained initializations for up to 400 labeled examples.[1]

## 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) the relation networks of Sung et al. (2018)[2], and (3) late fusion of representations (Rakelly et al., 2018; Zhang et al., 2019; 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). The use of multiple passes through subnetworks via iterative optimization modules was shown by Zhang et al. (2019) to yield improved segmentation results but comes at the expense of additional time complexity during inference. The relation networks proposed in Sung et al. (2018) have seen increased adoption in meta-learning systems and were recently extended to the modality of dense prediction by the authors in Zhang et al. (2019) and Wei et al. (2019). While this extension of the relation networks of Sung et al. (2018) to image segmentation yield impressive results in the few-shot domain, their efficacy in scaling as more training data becomes available is untested.

[1] We will release our code and the FP-k dataset upon acceptance.

[2] Not to be confused with the relation networks of Santoro et al. (2017).

Model Agnostic 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 backpropogating the meta-gradient through the 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 [3] (Mishra et al., 2018; Rusu et al., 2018). The Meta-SGD learning framework added additional capacity to the same network architecture used in MAML with improved generalization by meta-learning a learning rate for each parameter in the network (Li et al., 2017), but lacks a first order approximation. 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).

## 3 PRELIMINARIES

### 3.1 GENERALIZATION ERROR IN META-LEARNING

In the context of image segmentation, an example from a task $\tau$ is comprised of an image $x$ and its corresponding binary mask $y$, which assigns each pixel membership to the target (ex. black bear) or background class. Examples $(x, y)$ from the domain $\mathcal{D}_\tau$ are distributed according to $q_\tau(x, y)$, and we measure the loss $\mathcal{L}$ of predictions $\hat{y}$ generated from parameters $\theta$ and an update routine $U$. For a distribution $p(\tau)$ over the domain of tasks $\mathcal{T}$, the parameters that minimize the expected loss are

$$\theta^* = \arg\min_\theta \mathbb{E}_p \left[ \mathbb{E}_{q_\tau} \left[ \mathcal{L} \left( U(\theta) \right) \right] \right] \tag{1}$$

In practice, we only have access to a finite subset of the tasks, which we divide into the training $\mathcal{T}^{tr}$, validation $\mathcal{T}^{val}$, and test tasks $\mathcal{T}^{test}$, and instead optimize over an empirical distribution $\hat{p}(\tau) := p(\tau|\tau \in \mathcal{T}^{tr})$. For examples within each available task, we can similarly define $\mathcal{D}_\tau^{tr}$, $\mathcal{D}_\tau^{val}$, $\mathcal{D}_\tau^{test}$, and the corresponding empirical distribution $\hat{q}_\tau(x, y) := q_\tau(x, y|(x, y) \in \mathcal{D}_\tau^{tr})$. The empirically optimal initialization

$$\hat{\theta}^* = \arg\min_\theta \mathbb{E}_{\hat{p}} \left[ \mathbb{E}_{\hat{q}_\tau} \left[ \mathcal{L} \left( U(\theta) \right) \right] \right] \tag{2}$$

has a generalization error can then be expressed as

$$\mathbb{E}_p \left[ \mathbb{E}_{q_\tau} \left[ \mathcal{L} \left( U(\hat{\theta}^*) \right) \right] \right] - \mathbb{E}_{\hat{p}} \left[ \mathbb{E}_{\hat{q}_\tau} \left[ \mathcal{L} \left( U(\hat{\theta}^*) \right) \right] \right] \tag{3}$$

The generalization gap between the actual and empirical error in meta-learning is twofold: from the domain of all tasks $\mathcal{T}$ to the sample $\mathcal{T}^{tr}$, and within that, from all examples in $\mathcal{D}_\tau$ to $\mathcal{D}_\tau^{tr}$.

### 3.2 MODEL AGNOSTIC META-LEARNING

The model agnostic meta-learning (MAML) algorithm introduced in Finn et al. (2017) uses a gradient based update procedure $U$ with hyperparameters $\omega$, which applies a limited number of training steps with a few-shot subset of $\mathcal{D}_\tau^{tr}$ to adapt a meta-learned initialization $\theta$ to each task. To minimize

[3] 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)

the loss incurred in the update routine, we first take the derivative with respect to the initialization

$$\frac{\partial}{\partial\theta}\mathcal{L}(U(\theta)) = U'(\theta)\cdot\mathcal{L}'(U(\theta)) \tag{4}$$

where the resulting term $U'$ is the derivative of a gradient based update procedure, and hence, contains second order derivatives. In first-order renditions explored in Nichol & Schulman (2018), FOMAML and Reptile, finite differences are used to approximate the gradient of the meta-update $\nabla\theta$. The difference between the two approximations can be summarized by how they make use of $\mathcal{D}_\tau^{tr}$ and $\mathcal{D}_\tau^{val}$:

$$\theta^{tr} \leftarrow U(\theta \ ; \mathcal{D}_\tau^{tr}, \omega^{tr}) \tag{5}$$

$$\theta^{val} \leftarrow U(\theta^{tr} ; \mathcal{D}_\tau^{val}, \omega^{val}) \tag{6}$$

$$\theta^{both} \leftarrow U(\theta \ ; \mathcal{D}_\tau^{tr} \cup \mathcal{D}_\tau^{val}, \omega^{tr}) \tag{7}$$

Reptile trains jointly on both, while FOMAML trains on the two sets separately in sequence, favoring initializations that differ less between the splits.

$$\text{Reptile: } \nabla\theta \propto \theta^{both} - \theta \tag{8}$$

$$\text{FOMAML: } \nabla\theta \propto \theta^{val} - \theta^{tr} \tag{9}$$

The gradient approximation $\nabla\theta$ can then be used to optimize the initialization by stochastic gradient descent or any other gradient-based update procedure.

### 3.3 Tuning Meta-Learning Hyperparameters for Inference

To address research question 2 in section 1, we leverage the flexibility to choose hyperparameters $\omega^{test}$ used for inference, separately from the hyperparameters $\omega^{tr}$ used in meta-training. The optimal choice of $\omega^{test}$ can be determined by minimizing the expected loss in eq. 1 with respect to the hyperparameters, treating $\hat{\theta}^*$ and $\mathcal{D}_\tau^{tr}$ as parameters of the update routine:

$$\hat{\omega}^* = \arg\min_{\omega} \mathbb{E}_{\hat{p}}\left[\mathbb{E}_{\hat{q}_\tau}\left[\mathcal{L}\left(U(\omega\ ; \hat{\theta}^*, \mathcal{D}_\tau^{tr})\right)\right]\right] \tag{10}$$

Empirical estimations of the optimal initialization $\hat{\theta}^*$ have an implicit dependence on $\mathcal{T}^{tr}$ and $\omega^{tr}$ (eq. 2), and the optimal hyperparameters $\hat{\omega}^*$ depend on the $\hat{\theta}^*$ in turn (eq. 10).

## 4 EfficientLab Architecture for Image Segmentation

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 DeepLab 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 lower resolution feature maps, 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

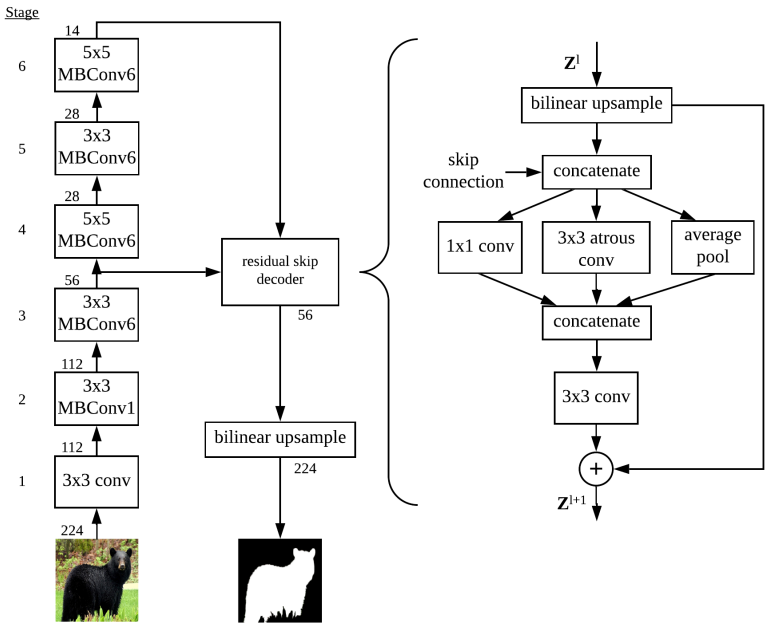


Figure 1: Diagram of the computations performed by the EfficientLab neural network. Nodes represent functions and edges represent output tensors. Output spatial resolutions are written next to the output edge. The high level architecture shows the EfficientNet feature extractor on the left with mobile inverted bottleneck convolutional blocks (see Tan & Le (2019); Sandler et al. (2018) for more details). On the right is the residual skip decoder module that we utilize in the upsampling branch of EfficientLab.

utilizes three parallel branches of a $1 \times 1$ convolution, $3 \times 3$ 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 $3 \times 3$ convolutional layer with 112 filters. A residual connection wraps around the convolutional layers to ease gradient flow [4]. We call this structure a residual skip decoder (RSD) and its computational graph of operations is shown in Figure 1. Before the final $1 \times 1$ 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 $\beta$ and $\gamma$ 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 (Dice, 1945) , which we adapt from the loss function of (Iglovikov et al., 2017), plus an $L_2$ regularization on the weights:

$$\mathcal{L} = H - log(J) + \lambda \left\|\theta\right\|_2^2 \tag{11}$$

where $H$ is binary cross entropy loss:

$$H = -\frac{1}{n}\sum_{i=1}^{n}\left(y_i \log \hat{y}_i + (1-y_i)\log\left(1-\hat{y}_i\right)\right) \tag{12}$$

[4] Residual connections have been suggested to make the loss landscape of deep neural networks more convex (Li et al., 2018). If this is the case, it could be especially helpful in finding low-error minima via gradient-based update routines such as those used by MAML, FOMAML, and Reptile.

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

$J$ is the modified Dice score:

$$J = \frac{2IoU}{IoU + 1} \tag{13}$$

and $IoU$ is the intersection over union metric:

$$IoU = \frac{1}{n} \sum_{i=1}^{n} \left( \frac{y_i \hat{y}_i + \epsilon}{y_i + \hat{y}_i - y_i \hat{y}_i + \epsilon} \right) \tag{14}$$

# 5 EXPERIMENTS

We evaluate the FOMAML and Reptile meta-learning algorithms on the FSS-1000 dataset. Model topology development and meta-training hyperparameter search was done on a held out set of validation tasks and not the final test tasks. For the final evaluations reported in Table , we meta-train for $\sim$200 epochs through the training and validation tasks, $\mathcal{T}^{tr} \cup \mathcal{T}^{val}$, using a meta-batch size of 5, an inner loop batch size of 8, and 5 inner loop iterations. During training, we use stochastic gradient descent (SGD) in the inner loop with a fixed learning rate of 0.005. During training and evaluation, we apply simple augmentations to the few-shot examples including random translation, horizontal flips, additive Gaussian noise, brightness, and random eraser (Zhong et al., 2017). We use $L_2$ regularization on all weights with a coefficient $\lambda = 5\text{e}{-4}$.

## 5.1 FSS-1000 DATASET

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 meta-test set $\mathcal{T}^{test}$, with 10 image-mask pairs for each class (Wei et al., 2019). For each of the rows in the results table 2, we evaluate the network on the 240 test tasks, sampling two random splits into training and testing sets for each task, yielding 480 data points per meta-learning approach for which the mean intersection over union (IoU) (eq. 14) and 95% confidence interval are reported. The FSS-1000 dataset is the focus of the empirical comparisons of network ablations and meta-learning approaches that we experiment with in this paper.

## 5.2 FP-K DATASET

For investigating how the meta-learned representations integrate new information as more data becomes a available, we put together a small benchmark 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[6]. Using this dataset, we train over a range of "k"-training shots from ImageNet-trained initializations[7] and our meta-learned initializations. We report the performance of our EfficientLab network meta-trained with FOMAML over a range of k examples as a benchmark which we hope will inspire future empirical research into studying how meta-learning approaches scale in accuracy and computational complexity as more labeled data become available. These results are show in Figure 3 and discussed in 5.4.

## 5.3 UPDATE HYPERPARAMETER TUNING METHODOLOGY

Generalization in meta-learning requires both the ability to learn representations for new tasks efficiently ($\mathcal{T}^{tr}$ to $\mathcal{T}^{test}$), and to select representations that are able to capture unseen test examples effectively ($\mathcal{D}_\tau^{tr}$ to $\mathcal{D}_\tau^{test}$). The approximation scheme of FOMAML addresses the latter by taking the finite difference between updates using the train and validation sets (as shown in eq. 9), favoring initializations that differ less between splits of $\mathcal{D}_\tau^{tr} \cup \mathcal{D}_\tau^{val}$. In investigation of research question 2 in section 1 and to further improve generalization within task to $\mathcal{D}_\tau^{test}$, we tune $\omega$ after meta-learning

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

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

$\hat{\theta}^*$ to find $\omega^{test}$ (as shown in eq. 10). We use $\omega^{test}$ at meta-test time when adapting the initialization to new tasks. To this end, we developed a simple update hyperparameter optimization (UHO) algorithm which can be viewed as an extension of Sequential Halving (Jamieson & Talwalkar, 2016) to meta-learning. This algorithm is discussed further and shown in pseudocode in the appendix in C. We apply the UHO algorithm to estimate the optimal update routine's hyperparameters on 100 randomly sampled tasks from the meta-training and -val sets $\mathcal{T}^{tr} \cup \mathcal{T}^{val}$. We specifically search over the learning rate and the number of gradient updates that are applied when adapting to a new task $\tau$. We report results with and without optimized update hyperparameters in table 2. We find that tuning $\omega$ significantly improves adaptation performance on the meta-test tasks $\mathcal{T}^{test}$.

### 5.4 RESULTS

We show the results of experimenting with different decoder architectures for EfficientLab in Table 1 Each network topology is meta-trained with FOMAML and the same meta-training hyperparameters defined in 5.

| **Network Architecture** | $\overline{\textbf{IoU}}$ |
|---|---|
| EfficientNet w/o decoder | $75.66 \pm 1.01\%$ |
| EfficientNet + Auto-DeepLab decoder | $73.054 \pm 01.09\%$ |
| EfficientNet + RSD at Stage 3 w/o residual | $78.17 \pm 1.02\%$ |
| EfficientNet + RSD at Stages 3 & 6 | $80.11 \pm 0.94\%$ |
| EfficientNet + RSD at Stage 3 | $\mathbf{80.60 \pm 0.93\%}$ |

Table 1: EfficientLab architecture ablations. Each network is meta-trained in the same way following 5 and tested on the set of test tasks from FSS-1000 (Wei et al., 2019). The 3rd row contains results of removing the short-range residual connection from our proposed RSD module. The final row is the best network we find for few-shot performance via model agnostic meta-learning. We call this network EfficientLab in reference to the encoder of EfficientNet (Tan & Le, 2019) and the decoder of Auto-DeepLab (Chen et al., 2018), which it is inspired by.

The results of our model with an initialization meta-learned using Reptile and FOMAML are shown in Table 2. We find that our model trained with FOMAML and importantly with regularization and improved use of batch normalization yields state of the art results. Given that previous works have used regularization minimally or not at all during meta-training, we also conducted an ablation of removing regularization on the model. We find, unsurprisingly, that the combination of an $L2$ loss on the weights, with simple augmentations, and a final layer of dropout does significantly increase generalization performance. After optimizing the update hyperparameters, our approach sets the new state of the art for the FSS-1000 dataset. We have included a visualization of example predictions for a small set of randomly sampled test tasks in B.

To address research question 2 in section 1, we also searched through a range of update routine learning rates, $\alpha$, that were $10\times$ less to $10\times$ greater than the learning rate used during meta-training. As clearly shown in Figure 2, the learned representations are **not** robust to such large variations in the hyperparameter.

In this work we posit that a fixed update procedure that is used at test time and not conditioned on the labeled examples for a new task $\mathcal{D}^{tr} \in \tau$ is one of the major hinderances of applying MAML-type algorithms to unseen tasks. In section 3, we show that the hyperparameters $\omega$ that are used to adapt the networks weights $\theta$ to a new task $\mathcal{D}_\tau$ can be optimized. 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 2. 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 search algorithm UHO were 0.007475 and 8, respectively, compared to a learning rate of 0.005 and 5 iterations used during training. (2) Optimizing the hyperparameters (even on the set of training tasks $\mathcal{T}^{tr}$) after meta-training improves test-time results on unseen tasks. Furthermore, we find that *meta-training* from scratch and evaluating with the UHO-selected hyperparameters learning rate = 0.007475 and inner-iterations = 8 yields nearly identical

| Method | $\overline{\text{IoU}}$ |
|---|---|
| FSS-1000 Baseline | 73.47% |
| **FOMAML** | $\mathbf{75.19 \pm 01.28}\%$ |

(a) FSS-1000 1-shot

| Method | $\overline{\text{IoU}}$ |
|---|---|
| FSS-1000 Baseline | 80.12% |
| Reptile | $62.36 \pm 2.12\%$ |
| FOMAML | $77.89 \pm 1.03\%$ |
| FOMAML + regularization | $80.60 \pm 0.93\%$ |
| **FOMAML + regularization + UHO** | $\mathbf{82.19 \pm 0.91}\%$ |

(b) FSS-1000 5-shot

Table 2: Mean IoU scores of the EfficientLab network evaluated on FSS-1000 test set of tasks for 1-shot and 5-shot learning. We report the FSS-1000 baseline from (Wei et al., 2019). Our best found model combined FOMAML, EfficientLab, regularization, and the UHO algorithm.

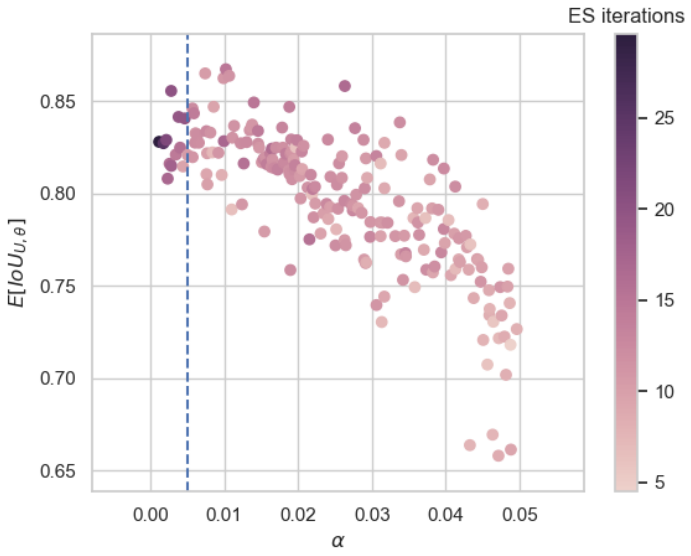


Figure 2: Each point represents the mean IoU for the validation tasks with a sampled learning rate $\alpha$. The blue dashed line indicates the learning rate used by SGD in the inner loop during meta-training. Points are colored by how many iterations they were trained before stopped by early stopping (ES) with a patience of 3 iterations.

results to meta-training with the initial hyper parameters learning rate = 0.005 and inner-iterations = 5. This further suggests that it may be useful to tune the hyperparameters $\omega$ *after* meta-training to improve the generalization performance of the gradient-based adaptation routine $U$.

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[8].

## 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

[8]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 3, which contain examples from both datasets, are not directly comparable to the results shown in Table 2, which contain examples only from FSS-1000. Furthermore, as discussed in the appendix in E, we did not tune $\omega$ in these experiments.

| Initialization | k | $\overline{\text{IoU}}$ |
|---|---|---|
| ImageNet | 1 | $16.76 \pm 4.62\%$ |
| ImageNet | 5 | $24.50 \pm 5.09\%$ |
| ImageNet | 10 | $27.08 \pm 5.58\%$ |
| ImageNet | 50 | $31.37 \pm 6.43\%$ |
| ImageNet | 100 | $37.39 \pm 6.66\%$ |
| ImageNet | 200 | $47.50 \pm 6.82\%$ |
| ImageNet | 400 | $55.86 \pm 7.15\%$ |
| Meta-learned | 1 | $38.30 \pm 9.42\%$ |
| Meta-learned | 5 | $42.59 \pm 9.16\%$ |
| Meta-learned | 10 | $43.94 \pm 9.86\%$ |
| Meta-learned | 50 | $50.12 \pm 8.62\%$ |
| Meta-learned | 100 | $53.37 \pm 7.90\%$ |
| Meta-learned | 200 | $55.55 \pm 7.73\%$ |
| Meta-learned | 400 | $58.68 \pm 7.48\%$ |

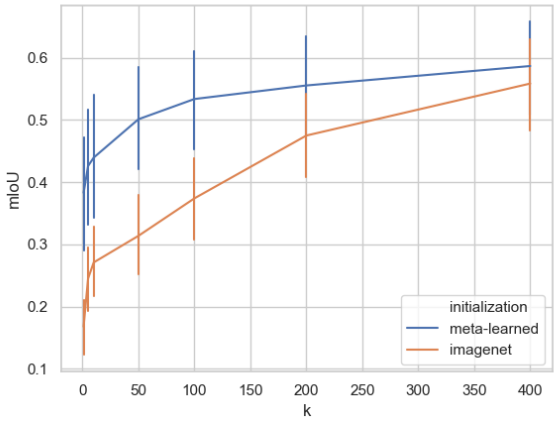


Figure 3: Mean IoU results as a function of the training set size of our EfficientLab model adapted to tasks of the FP-k dataset. Error bars represent 95% confidence intervals. The meta-learned initialization utilizes the top performing learning algorithm in Table 2. The meta-learned initialization outperformed EfficientLab initialized with an ImageNet-trained encoder and a randomly initialized decoder for all numbers of labeled training examples that we evaluated. For additional experimental details see E in the appendix.

few-shot image segmentation problems, but are sensitive to the hyperparameters, $\omega$, of the update routine, $U$. Furthermore, we find that the representations that are meta-learned are valuable as more data becomes available, unifying few- and many-shot regimes. We have also presented a novel neural network architecture, EfficientLab, for semantic image segmentation.

Future work should investigate more critically, both empirically and theoretically, the efficacy of few-shot learning systems as more labeled data becomes available. To this end, we have reported baseline results on a small meta-test benchmark dataset, FP-k, which contains 5 tasks with 400 training and 20 test examples per task. Additionally, future work could investigate more deeply 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 $\beta$ and $\gamma$ parameters at all while still using estimated population means and variances during inference Zhang et al. (2019), or (3) meta-learning $\beta$ and $\gamma$ while only using batch statistics for the normalization (Finn et al., 2017; Nichol & Schulman, 2018), or (4) meta-learning $\beta$ and $\gamma$ 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. Finally, another interesting question to address would be to evaluate how EfficientLab performs on more standard many-shot multi-class image segmentation problems such as the CityScapes dataset (Cordts et al., 2016).

In conclusion, we have shown in this work that the optimal hyperparameter configuration for the update routine may not be the same configuration used during meta-learning. These findings are supported by our theoretical analyses which show that MAML-type algorithms minimize the empirical risk on the training set of a fixed update routine and the initialization $\theta$, but do not natively guarantee that the update routine's hyperparameters are optimal. 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 (Li et al., 2017; Antoniou et al., 2018) directly leverage information on *how* to adapt given a few-shot sample of labeled examples. We also suspect that the previous work in Mishra et al. (2018) may have found MAML-type algorithms to overfit when applied to high dimensional parameter spaces due to lack of regularization and lack of an empirical risk minimization of the update routine's hyperparameters. It is our hope that our empirical analyses

and formalization of the generalization error of meta-learning systems lead 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 small and large data regimes by gaining expertise and 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.

Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3213–3223, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Lee R Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3): 297–302, 1945.

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.

Vladimir Iglovikov, Sergey Mushinskiy, and Vladimir Osin. Satellite imagery feature detection using deep convolutional neural network: A kaggle competition. *arXiv preprint arXiv:1706.06169*, 2017.

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.

Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 10657–10665, 2019.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *Advances in Neural Information Processing Systems*, pp. 6389–6399, 2018.

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.

Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

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. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=B1DmUzWAW`.

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. URL `https://openreview.net/forum?id=SkMjFKJwG`.

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.

Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4510–4520, 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.

Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip HS Torr, and Timothy M Hospedales. Learning to compare: Relation network for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1199–1208, 2018.

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.

Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. *arXiv preprint arXiv:1708.04896*, 2017.

## A FSS-1000 Test Tasks

The set of test tasks that the meta-learning systems are evaluated on in this paper are available in the FSS-1000 github repository https://github.com/HKUSTCV/FSS-1000.

## B Example predictions

We have included here a visualization of a small random sample of predictions on test examples $\mathcal{D}^{test}$ from test tasks $\mathcal{T}^{test}$ that were never seen during meta-training.

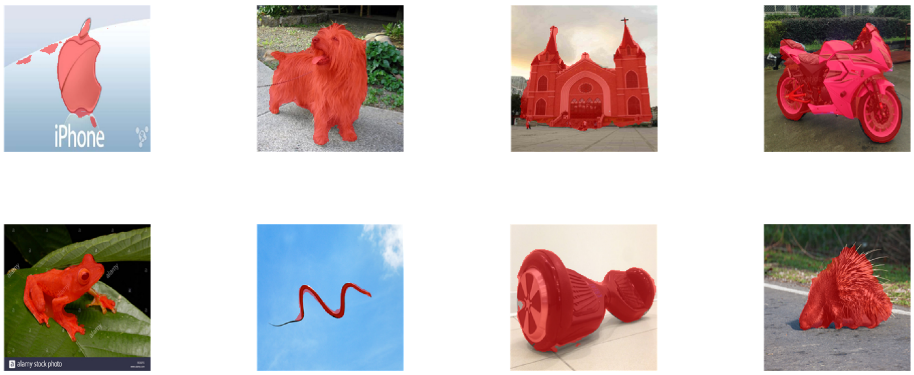


Figure 4: Randomly sampled example 5-shot predictions on the test images from test tasks. Positive class prediction is overlaid in red. From left to right, top to bottom, the classes are `apple_icon`, `australian_terrier`, `church`, `motorbike`, `flying_frog`, `flying_snakes`, `hover_board`, `porcupine`

## C Hyperparameter Tuning Methodology

Motivated by the insight discussed in section 3 that the loss of a meta-learning algorithm with a fixed initialization $\theta$ is a function of the update routine $U$ *and* its hyperparameters $\omega$ to develop a simple update hyperparameter optimization (UHO) algorithm. This algorithm is outlined in pseudocode in 1. The routine samples $n$ hyperparameter values within a predefined, broad range, evaluates each of the hyperparameters on a sample of tasks, $\mathcal{T}^{val}$. After evaluating all $\tau$ in $\mathcal{T}^{val}$, 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 viewed as a variant of Sequential Halving (Jamieson & Talwalkar, 2016).

Because the effects of the learning rate are intertwined with the number of gradient updates, we also leverage early stopping (ES) to decrease runtime and to estimate the optimal number of gradient steps, $\hat{j^*}$, when adapting to a new task. The use of early stopping in this way is purely an implementation optimization that reduces the search space that is explored when tuning the hyperparameters $\omega$. We could have simply randomly sampled the number of iterations, but this would have lead to many wasted tests of combinations of learning rates and gradient steps such as when both the learning rate and the number of gradient updates are large. We use a patience of 3 steps when using ES, meaning that if the mean IoU on $D^{test}$ did not improve for 3 gradient updates, we stop running SGD and get the number of updates with the best performance on $D^{test}$, as shown on line 6 of the UHO pseudocode.

**Algorithm 1** Update hyperparameter optimization (UHO) via random search and successive decrease in search space. Returns the estimated optimal configuration of hyperparameters $\hat{\omega^*}$, such as the learning rate $\hat{\alpha^*}$ and number of gradient steps $\hat{j^*}$, that minimize the empirical loss on $p(\mathcal{T})$.

**Require:** $\theta$: an initialization
**Require:** $f$: a model parameterized by $\theta$
**Require:** $p(\mathcal{T})$: a distribution over tasks
**Require:** $U$: an update routine with hyperparameters $\omega$
**Require:** $\omega^{min}$: vector of minimums for each hyperparameter $\omega_i$
**Require:** $\omega^{max}$: vector of maximums for each hyperparameter $\omega_i$
**Require:** $x$: a floating point value in $(0, 1)$ to decrease the search ranges by
**Require:** $b$: the number of iterations to refine the search space, which defines a computational budget
**Require:** $n$: the number of configurations to sample per refinement

Initialize list of losses, `losses`
Initialize list of sampled hyperparameter configurations, `parameters`
**for** $i = 0$ **to** $b$ **do**
    **for** $j = 0$ **to** $n$ **do**
        $\omega_s \leftarrow$ `sample`$(\omega^{min}, \omega^{max})$
        $\mathcal{L}, \hat{j^*} \leftarrow$ `evaluate`$(U, \omega_s, f, \theta, p(\mathcal{T}))$ ▷ Evaluate the expectation of the loss
        `losses.append`$(\mathcal{L})$
        $\omega_s \leftarrow \omega_s \cup \hat{j^*}$ ▷ Add number of steps returned by ES to the set $\omega_s$
        `parameters.append`$(\omega_s)$
    **end for**
    $\omega^{min}, \omega^{max} \leftarrow$ `refine_search_space`$(\mathtt{x}, \mathtt{losses}, \mathtt{parameters}, \omega^{max}, \omega^{min})$
**end for**
**return** $\underset{\omega}{argmin}(\mathtt{losses})$

**Algorithm 2** `refine_search_space` returns a new search space by computing min and max for each hyperparameter in `parameters` from the best $x\%$ of `losses`.

**Require:** $x$: a floating point value in $(0, 1)$ to decrease the search ranges by
**Require:** `losses`: a list of losses ordered corresponding to `parameters` that generated the losses.
**Require:** $\Omega$ a list of hyperparameter configurations that generated each element in `losses`.
**Require:** $\omega^{min}$: vector of minimums for each hyperparameter $\omega_i$
**Require:** $\omega^{max}$: vector of maximums for each hyperparameter $\omega_i$

$\Omega' \leftarrow$ Get the x% of the best hyperparameter configurations in $\Omega$ identified by `losses`.
**for** $i = 0$ **to** `len`$(\omega^{min})$ **do**
    $\omega_i^{min} \leftarrow \min_{\omega_i} \Omega'_i$ ▷ Get the smallest value for each hyperparameter (e.g. the learning rate)
    $\omega_i^{max} \leftarrow \max_{\omega_i} \Omega'_i$
**end for**
**return** $\omega^{min}, \omega^{max}$

## D FP-K DATASET

Table 3 contains the five tasks in PASCAL-$5^i$ that have direct analogs in FSS-1000. Each row contains the name of a task 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: $[1,\ 5,\ 10,\ 50,\ 100,\ 200,\ 400]$. For more details on our training and evaluation procedures see E.

| PASCAL-$5^i$ Task | FSS-1000 Task |
|---|---|
| aeroplane | airliner |
| bus | bus |
| motorbike | motorbike |
| potted_plant | potted_plant |
| tvmonitor | television |

Table 3: PASCAL-$5^i$ tasks with FSS-1000 analog.

## E FP-K EXPERIMENTAL DETAILS

In this section, we describe our testing protocol and simple hyperparameter choices when training the ImageNet and randomly initialized and the FOMAML initialized EfficientLab network. For each tuple of (initialization, k-training shots) we randomly sample 20 examples for a test set, $\mathcal{D}^{test}$ for the task and train on k labeled examples $\mathcal{D}^{tr}$. We repeat this random sampling and training process 4 times for each of the 5 tasks, yielding 20 evaluation samples per (initialization, k-training shots) tuple. For training both networks on the available labeled examples, we simply use the number of iterations $[1,\ 5,\ 10,\ 25,\ 50,\ 100,\ 200]$ for each k in the set of k-shot tasks $[1,\ 5,\ 10,\ 50,\ 100,\ 200,\ 400]$, respectively. We acknowledge that it is likely possible that both networks could achieve better performance if their update routine's hyperparameters are empirically optimized or conditioned on the k examples in $\mathcal{D}^{tr}$.