Meta-Learning with Shared Amortized Variational Inference

Ekaterina Iakovleva, Jakob Verbeek, Karteek Alahari

▶ **To cite this version:**

Ekaterina Iakovleva, Jakob Verbeek, Karteek Alahari. Meta-Learning with Shared Amortized Variational Inference. ICML 2020 - 37th International Conference on Machine Learning, Jul 2020, Vienna (Online), Austria. pp.4572-4582. hal-02925830

HAL Id: hal-02925830
https://inria.hal.science/hal-02925830

Submitted on 31 Aug 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Meta-Learning with Shared Amortized Variational Inference

Ekaterina Iakovleva [1] **Jakob Verbeek** [2] **Karteek Alahari** [1]

Abstract

We propose a novel amortized variational inference scheme for an empirical Bayes meta-learning model, where model parameters are treated as latent variables. We learn the prior distribution over model parameters conditioned on limited training data using a variational autoencoder approach. Our framework proposes sharing the same amortized inference network between the conditional prior and variational posterior distributions over the model parameters. While the posterior leverages both the labeled support and query data, the conditional prior is based only on the labeled support data. We show that in earlier work, relying on Monte-Carlo approximation, the conditional prior collapses to a Dirac delta function. In contrast, our variational approach prevents this collapse and preserves uncertainty over the model parameters. We evaluate our approach on the miniImageNet, CIFAR-FS and FC100 datasets, and present results demonstrating its advantages over previous work.

1. Introduction

While people have an outstanding ability to learn from just a few examples, generalization from small sample sizes has been one of the long-standing goals of machine learning. Meta-learning, or "learning to learn" (Schmidhuber, 1999), aims to improve generalization in small sample-size settings by leveraging the experience of having learned to solve related tasks in the past. The core idea is to learn a meta model that, for any given task, maps a small set of training samples for a new task to a model that generalizes well.

A recent surge of interest in meta-learning has explored a wide spectrum of approaches. This includes nearest neighbor based methods (Guillaumin et al., 2009; Vinyals et al., 2016), nearest class-mean approaches (Dvornik et al., 2019; Mensink et al., 2012; Ren et al., 2018; Snell et al., 2017), optimization based methods (Finn et al., 2017; Ravi & Larochelle, 2017), adversarial approaches (Zhang et al., 2018), and Bayesian models (Gordon et al., 2019; Grant et al., 2018). The Bayesian approach is particularly interesting, since it provides a coherent framework to reason about model uncertainty, not only in small sample-size settings, but also others such as incremental learning (Kochurov et al., 2018), and ensemble learning (Gal & Ghahramani, 2016). Despite its attractive properties, intractable integrals over model parameters or other latent variables, which are at the heart of the Bayesian framework, make it often necessary to turn to stochastic Monte Carlo or analytic approximations for practical implementations.

In our work, we follow the Bayesian latent variable approach, and learn a prior on the parameters of the classification model conditioned on a small training sample set for the task. We use a variational inference framework to approximate the intractable marginal likelihood function during training. The variational distribution approximates the posterior on the parameters of the classification model, given training and test data. Both the prior and posterior are parameterized as deep neural networks that take a set of labeled data points as input. By sharing the inference network across these two distributions, we leverage more data to learn these conditionals and avoid overfitting. Figure 1 illustrates the overall structure of our model, SAMOVAR.

We compare the variational training approach with the Monte Carlo approach followed by Gordon et al. (2019) on synthetic data. We find that when using a small number of samples for stochastic back-propagation in the Monte Carlo approach, which results in faster training, the prior collapses to a Dirac delta, and the model degenerates to a deterministic parameter generating network. In contrast, our variational training approach does not suffer from this deficiency, and leads to an accurate estimation of the variance. Experiments on few-shot image classification using the miniImageNet, CIFAR-FS and FC100 datasets confirm these findings, and we observe improved accuracy using the variational approach to train the VERSA model (Gordon et al., 2019). Moreover, we use the same variational framework to train a stochastic version of the TADAM few-shot

[1]Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK, 38000 Grenoble, France. [2]Facebook Artificial Intelligence Research, Work done while Jakob Verbeek was at Inria. Correspondence to: Ekaterina Iakovleva <ekaterina.iakovleva@inria.fr>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

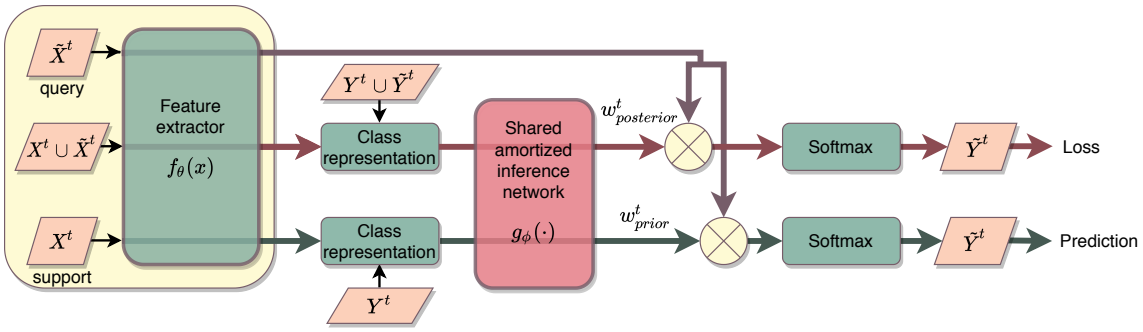


Figure 1. SAMOVAR, our meta-learning model for few-shot image classification. For task t, query data $\tilde{X}^t$ and support data X^t are put through a task-agnostic feature extractor $f_\theta(x)$. The features are then averaged class-wise, and mapped by the shared amortized inference network into prior and posterior over the task-specific classifier weight vectors. Classifiers $w^t_{posterior}$ and w^t_{prior} sampled from these distributions map query features $f_\theta(\tilde{X}^t)$ to predictions on the query labels $\tilde{Y}^t$ used in training and testing, respectively.

image classification model (Oreshkin et al., 2018), replacing the deterministic prototype classifier with a scaled cosine classifier with stochastic weights. Our stochastic formulation significantly improves performance over the base architecture, and yields results competitive with the state of the art on the miniImageNet, CIFAR-FS and FC100 datasets.

2. Related Work

Distance-based classifiers. A straightforward approach to handle small training sets is to use nearest neighbor (Weinberger et al., 2006; Guillaumin et al., 2009; Vinyals et al., 2016), or nearest prototype (Mensink et al., 2012; Snell et al., 2017; Dvornik et al., 2019; Ren et al., 2018; Oreshkin et al., 2018) classification methods. In a "meta" training phase, a metric – or, more generally, a data representation – is learned using samples from a large number of classes. At test time, the learned metric can then be used to classify samples across a set of classes not seen during training, by relying on distances to individual samples or "prototypes," i.e., per-class averages. Alternatively, it is also possible to learn a network that takes two samples as input and predicts whether they belong to the same class (Sung et al., 2018). Other work has explored the use of task-adaptive metrics, by conditioning the feature extractor on the class prototypes for the task at hand (Oreshkin et al., 2018). We show that our latent variable approach is complementary and improves the effectiveness of the latter task conditioning scheme.

Optimization-based approaches. Deep neural networks are typically learned from large datasets using SGD. To adapt to the regime of (very) small training datasets, optimization-based meta-learning techniques replace the vanilla SGD approach by a trainable update mechanism (Bertinetto et al., 2019; Finn et al., 2017; Ravi & Larochelle, 2017), e.g., by learning a parameter initialization, such that a small number of SGD updates yields good performance (Finn et al., 2017). In addition to parameter initialization, the use of an LSTM model to control the influence of the gradient for updating the current parameters has also been explored (Ravi & Larochelle, 2017). In our work, the amortized inference network makes a single feed-forward pass through data to estimate a distribution on the parameters, instead of multiple passes to update the parameters.

Latent variable models. Gradient-based estimators of the parameters have a high variance in the case of small sample sizes. It is natural to explicitly model this variance by treating the parameters as latent variables in a Bayesian framework (Garnelo et al., 2018; Gordon et al., 2019; Grant et al., 2018; Kim et al., 2019; MacKay, 1991; Neal, 1995). The marginal likelihood of the test labels given the training set is then obtained by integrating out the latent model parameters. This typically intractable marginal likelihood, required for training and prediction, can be approximated using (amortized) variational inference (Garnelo et al., 2018; Kim et al., 2019), Monte Carlo sampling (Gordon et al., 2019), or a Laplace approximation (Grant et al., 2018). Neural processes (Garnelo et al., 2018; Kim et al., 2019) are also related to our work in their structure, and the use of shared inference network between the prior and variational posterior. Where neural processes use the task-specific latent variable as an additional input to the classifier network, we explicitly model the parameters of a linear classifier as the latent variable. This increases interpretability of the latent space, and allows for a flexible number of classes.

Interestingly, some optimization-based approaches can be viewed as approximate inference methods in latent variable models (Grant et al., 2018; Rusu et al., 2019). Semi-amortized inference techniques (Marino et al., 2018; Kim et al., 2018), which combine feed-forward parameter initialization and iterative gradient-based refinement of the approximate posterior, can be seen as a hybrid of optimization-based and Bayesian approaches. Deterministic approaches

that generate a single parameter vector for the task model, given a set of training samples (Bertinetto et al., 2016; Ha et al., 2017; Qiao et al., 2018), can be seen as a special case of the latent variable model with Dirac delta conditional distributions on the parameters.

3. Our Meta-Learning Approach

We follow the common meta-learning setting of episodic training of K-shot N-way classification on the *meta-train* set with C classes (Finn et al., 2017; Gordon et al., 2019; Ravi & Larochelle, 2017). For each classification task t sampled from a distribution over tasks $p(\mathcal{T})$, the training data $D^t = \{(\boldsymbol{x}^t_{k,n}, \boldsymbol{y}^t_{k,n})\}^{K,N}_{k,n=1}$ (*support* set) consists of K pairs of samples $\boldsymbol{x}^t_{k,n}$ and their labels $\boldsymbol{y}^t_{k,n}$ from each of N classes. The meta-learner takes the KN labeled samples as input, and outputs a classifier across these N classes to classify MN unlabeled samples from the testing data $\tilde{D}^t = \{(\tilde{\boldsymbol{x}}^t_{m,n}, \tilde{\boldsymbol{y}}^t_{m,n})\}^{M,N}_{m,n=1}$ (*query* set). During the *meta-train* stage, the meta-learner iterates over T episodes where each episode corresponds to a particular task t. During the *meta-test* stage, the model is presented with new tasks where the support and query sets are sampled from the meta-test set, which consists of previously unseen classes C'. The support set is used as input to the trained meta-learner, and the classifier produced by meta-learning is used to evaluate the performance on the query set. Results are averaged over a large set of meta-test tasks.

In this section, we propose a probabilistic framework for meta-learning. In Section 3.1, we start with a description of the multi-task graphical model that we adopt. We then derive an amortized variational inference with learnable prior for this generative model in Section 3.2, and propose to share the amortized networks for prior and approximate posterior. Finally, in Section 3.3 we describe the design of our model, SAMOVAR, which is trained with the proposed shared variational inference method.

3.1. Generative Meta-Learning Model

We employ a hierarchical graphical model shown in Figure 2. This multi-task model includes latent parameters θ, shared across all the T tasks, and task-specific latent parameters $\{w^t\}^T_{t=1}$. The marginal likelihood of the query labels $\tilde{Y} = \{\tilde{Y}^t\}^T_{t=1}$, given the query samples $\tilde{X} = \{\tilde{X}^t\}^T_{t=1}$ and the support sets $D = \{D^t\}^T_{t=1}$, is obtained as

$$p(\tilde{Y}|\tilde{X}, D) = \int p(\theta) \prod_{t=1}^{T} \int p(\tilde{Y}^t|\tilde{X}^t, w^t) p(w^t|D^t, \theta) \mathrm{d}w^t \mathrm{d}\theta. \quad (1)$$

The first term, $p(\theta)$, is the prior over the global task-independent parameters θ. The second term, $p(\tilde{Y}^t|\tilde{X}^t, w^t)$, is the likelihood of query labels $\tilde{Y}^t$, given query samples $\tilde{X}^t$ and task-specific parameters w^t. For example, this could be a linear classifier with weights w^t over features computed by a network with parameters θ. The third term, $p(w^t|D^t, \theta)$ is the conditional distribution on the task parameters w^t given the support set D^t and global parameters θ. We parameterize this distribution with a deep neural network with parameters ϕ as $p_\phi(w^t|D^t, \theta)$.

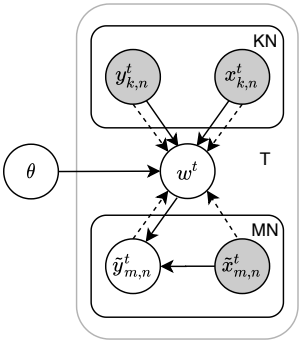


Figure 2. Hierarchical graphical model. The solid lines correspond to the generative process, while the dashed lines correspond to the variational inference procedure. Shaded nodes represent observed variables, non-shaded ones correspond to latent variables.

Following Gordon et al. (2019); Grant et al. (2018); Hu et al. (2020), we consider a point estimate for θ to simplify the model. The per-task marginal likelihood is then

$$p(\tilde{Y}^t|\tilde{X}^t, D^t, \theta) = \int p(\tilde{Y}^t|\tilde{X}^t, w^t) p_\phi(w^t|D^t, \theta) \mathrm{d}w^t, \quad (2)$$

$$p(\tilde{Y}^t|\tilde{X}^t, w^t) = \prod_{m=1}^{M} p(\tilde{\boldsymbol{y}}^t_m|\tilde{\boldsymbol{x}}^t_m, w^t). \quad (3)$$

To train the model, a Monte Carlo approximation of the integral in Eq. (2) was used in Gordon et al. (2019):

$$\mathcal{L}(\theta, \phi) = \frac{1}{TM} \sum_{t=1}^{T} \sum_{m=1}^{M} \log \frac{1}{L} \sum_{l=1}^{L} p(\tilde{\boldsymbol{y}}^t_m|\tilde{\boldsymbol{x}}^t_m, w^t_l), \quad (4)$$

where $w^t_l \sim p_\phi(w^t|D^t, \theta)$. In our experiments in Section 4, we show that training with this approximation tends to severely underestimate the variance in $p_\phi(w^t|D^t, \theta)$, effectively reducing the model to a deterministic one, and defying the use of a stochastic latent variable model.

3.2. Shared Amortized Variational Inference

To prevent the conditional prior $p_\phi(w^t|D^t, \theta)$ from degenerating, we use amortized variational inference (Kingma & Welling, 2014; Rezende et al., 2014) to approximate the intractable true posterior $p(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. Using the approximate posterior $q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$ parameterized by ψ, we obtain the variational evidence lower bound (ELBO)

of Eq. (2) as

$$\begin{aligned}\log p(\tilde{Y}^t|\tilde{X}^t, D^t, \theta) \geq \mathbb{E}_{q_\psi}\left[\log p(\tilde{Y}^t|\tilde{X}^t, w^t)\right] \\ - \mathcal{D}_{\text{KL}}\left(q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)||p_\phi(w^t|D^t, \theta)\right).\end{aligned} \tag{5}$$

The first term can be interpreted as a reconstruction loss, that reconstructs the labels of the query set using latent variables w^t sampled from the approximate posterior, and the second term as a regularizer that encourages the approximate posterior to remain close to the conditional prior $p_\phi(w^t|D^t, \theta)$. We approximate the reconstruction term using L Monte Carlo samples, and add a regularization coefficient β to weigh the KL term (Higgins et al., 2017). With this, our optimization objective is:

$$\begin{aligned}\hat{\mathcal{L}}(\Theta) = \frac{1}{T}\sum_{t=1}^{T}\Bigg[\sum_{m=1}^{M}\frac{1}{L}\sum_{l=1}^{L}\log p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, w_l^t) \\ - \beta\mathcal{D}_{\text{KL}}\left(q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)||p_\phi(w^t|D^t, \theta)\right)\Bigg],\end{aligned} \tag{6}$$

where $w_l^t \sim q_\psi(w|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. We maximize the ELBO w.r.t. $\Theta = \{\theta, \phi, \psi\}$ to jointly train the model parameters θ, ϕ, and the variational parameters ψ.

We use Monte Carlo sampling from the learned model to make predictions at test time as:

$$p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, D^t, \theta) \approx \frac{1}{L}\sum_{l=1}^{L} p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, w_l^t), \tag{7}$$

where $w_l^t \sim p_\phi(w^t|D^t, \theta)$. In this manner, we leverage the stochasticity of our model by averaging predictions over multiple realizations of w^t.

The approach presented above suggests to train separate networks to parameterize the conditional prior $p_\phi(w^t|D^t, \theta)$ and the approximate posterior $q_\psi(w^t|\tilde{Y}^t, \tilde{X}^t, D^t, \theta)$. Since in both cases the conditioning data consists of labeled samples, it is possible to share the network for both distributions, and simply change the input of the network to obtain one distribution or the other. Sharing has two advantages: (i) It reduces the number of parameters to train, decreasing the memory footprint of the model and the risk of overfitting. (ii) It facilitates the learning of a non-degenerate prior.

Let us elaborate on the second point. Omitting all dependencies for brevity, the KL divergence $\mathcal{D}_{\text{KL}}(q||p) = \int q(w)\left[\log q(w) - \log p(w)\right]$ in Eq. (5) compares the posterior $q(w)$ and the prior $p(w)$. Consider the case when the prior converges to a Dirac delta, while the posterior does not. Then, there exist points in the support of the posterior for which $p(w) \approx 0$, therefore, the KL divergence tends to infinity. The only alternative in this case is for the posterior to converge to the same Dirac delta. This would mean that for different inputs the inference network produces the same (degenerate) distribution. In particular, the additional conditioning data available in the posterior would leave the distribution unchanged, failing to learn from the additional data. While in theory this is possible, we do not observe it in practice.

We coin our approach "SAMOVAR", short for Shared AMOrtized VARiational inference.

3.3. Implementing SAMOVAR: Architectural Designs

The key properties we expect SAMOVAR to have are: (i) the ability to perform the inference in a feed-forward way (unlike gradient-based models), and (ii) the ability to handle a variable number of classes within the tasks. We build upon the work of Gordon et al. (2019); Qiao et al. (2018), to meet both these requirements. We start with VERSA (Gordon et al., 2019) where the feature extractor is followed by an amortized inference network, which returns a linear classifier with stochastic weights. SAMOVAR-base, our baseline architecture built this way on VERSA, consists of the following components.

Task-independent feature extractor. We use a deep convolutional neural network (CNN), f_θ, shared across all tasks, to embed input images $\boldsymbol{x}$ in $\mathbb{R}^d$. The extracted features are the only information from the samples used in the rest of the model. The CNN architectures used for different datasets are detailed in Section 4.2.

Task-specific linear classifier. Given the features, we use multi-class logistic discriminant classifier, with task-specific weight matrix $w^t \in \mathbb{R}^{N\times d}$. That is, for the query samples $\tilde{\boldsymbol{x}}$ we obtain a distribution over the labels as:

$$p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, w^t) = \text{softmax}\left(w^t f_\theta(\tilde{\boldsymbol{x}}_m^t)\right). \tag{8}$$

Shared amortized inference network. We use a deep permutation invariant network g_ϕ to parameterize the prior over the task-specific weight matrix w^t, given a set of labeled samples. The distribution on w^t is factorized over its rows $w_1^t, \ldots, w_N^t$ to allow for variable number of classes, and to simplify the structure of the model. For any class n, the inference network g_ϕ maps the corresponding set of support feature embeddings $\{f_\theta(\boldsymbol{x}_{k,n}^t)\}_{k=1}^K$ to the parameters of a distribution over w_n^t. We use a Gaussian with diagonal covariance to model these distributions on the weight vectors, i.e.,

$$p_\phi(w_n^t|D^t, \theta) = \mathcal{N}(\boldsymbol{\mu}_n^t, \text{diag}\left(\boldsymbol{\sigma}_n^t\right)), \tag{9}$$

where the mean and the variance are computed by the inference network as:

$$\begin{bmatrix}\boldsymbol{\mu}_n^t \\ \boldsymbol{\sigma}_n^t\end{bmatrix} = g_\phi\left(\frac{1}{K}\sum_{k=1}^{K} f_\theta(\boldsymbol{x}_{k,n}^t)\right). \tag{10}$$

To achieve permutation invariance among the samples, we average the feature vectors within each class before feeding them into the inference network g_ϕ. The approximate variational posterior is obtained in the same manner, but in this case the feature average that is used as input to the inference network is computed over the union of labeled support and query samples.

To further improve the model, we employ techniques commonly used in meta-learning classification models: scaled cosine similarity, task conditioning, and auxiliary co-training.

Scaled cosine similarity. Cosine similarity based classifiers have recently been widely adopted in few-shot classification (Dvornik et al., 2019; Gidaris et al., 2019; Lee et al., 2019; Oreshkin et al., 2018; Ye et al., 2018). Here, the linear classifier is replaced with a classifier based on the cosine similarity with the weight vectors w_n^t, scaled with a temperature parameter α:

$$p(\tilde{\boldsymbol{y}}_m^t|\tilde{\boldsymbol{x}}_m^t, w_n^t) = \text{softmax}\left(\alpha \frac{f_\theta(\tilde{\boldsymbol{x}}_m^t)^\top w_n^t}{||f_\theta(\tilde{\boldsymbol{x}}_m^t)|| \cdot ||w_n^t||}\right) \quad (11)$$

We refer this version of our model as SAMOVAR-SC.

Task conditioning. A limitation of the above models is that the weight vectors $\boldsymbol{w}_n^t$ depend only on the samples of class n. To leverage the full context of the task, we adopt the task embedding network (TEN) of Oreshkin et al. (2018). For each feature dimension of f_θ, TEN provides an affine transformation conditioned on the task data, similar to FiLM conditioning layers (Perez et al., 2018) and conditional batch normalization (Munkhdalai et al., 2018; Dumoulin et al., 2017). In particular, input to TEN is the average $\boldsymbol{c} = \frac{1}{N}\sum_n \boldsymbol{c}_n$, of the per-class prototypes, $\boldsymbol{c}_n = \frac{1}{K}\sum_k f_\theta(\boldsymbol{x}_{kn}^t)$ in the task t, and outputs are translation and scale parameters for all feature channels in the feature extractor layers. In SAMOVAR, we use TEN to modify both the support and query features f_θ before they enter the inference network g_ϕ. The query features that enter into the linear/cosine classifiers are left unchanged.

Auxiliary co-training. Large feature extractors can benefit from auxiliary co-training to prevent overfitting, stabilize the training, and boost the performance (Oreshkin et al., 2018). We leverage this by sharing the feature extractor f_θ of the meta-learner with an auxiliary classification task across all the classes in the meta-train set, using the cross-entropy loss for a linear logistic classifier over f_θ.

4. Experiments

We analyze the differences between training with Monte Carlo estimation and variational inference with a controlled synthetic data experiment in Section 4.1. Then, we present the few-shot image classification experimental setup in Section 4.2, followed by results, and a comparison to related work in Section 4.3.

4.1. Synthetic Data Experiments

We consider the same hierarchical generative process as Gordon et al. (2019), which allows for exact inference:

$$p(\psi^t) = \mathcal{N}(0, 1), \qquad p(y^t|\psi^t) = \mathcal{N}(\psi^t, \sigma_y^2). \quad (12)$$

We sample $T = 250$ tasks, each with $K = 5$ support observations $D^t = \{y_k^t\}_{k=1}^K$, and $M = 15$ query observations $\tilde{D}^t = \{\tilde{y}_m^t\}_{m=1}^M$. We use an inference network $q_\phi(\psi|D^t) = \mathcal{N}(\mu_q, \sigma_q^2)$, where

$$\begin{bmatrix}\mu_q \\ \log\sigma_q^2\end{bmatrix} = W\sum_{k=1}^K y_k^t + \boldsymbol{b}, \quad (13)$$

with trainable parameters W and $\boldsymbol{b}$. The inference network is used to define the predictive distribution

$$p(\tilde{D}^t|D^t) = \int p(\tilde{D}^t|\psi) q_\phi(\psi|D^t)\, \mathrm{d}\psi. \quad (14)$$

Since the prior is conjugate to the Gaussian likelihood $p(y^t|\psi^t)$ in Eq. (12), we can analytically compute the marginal $p(\tilde{D}^t|D^t)$ in Eq. (14) and the true posterior $p(\psi|D^t)$, which are both Gaussian.

We train the inference network by optimizing Eq. (14) in the following three ways.

1. **Exact marginal log-likelihood.** For T tasks, with M query samples each, we obtain

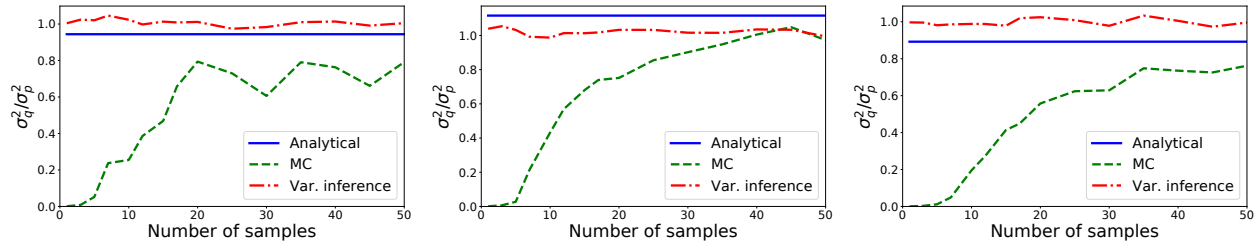
$$\mathcal{L}(\phi) = -\frac{1}{MT}\sum_{t=1}^T\sum_{m=1}^M \log\mathcal{N}(y_m^t; \mu_q(D^t), \sigma_q^2(D^t)+\sigma_y^2). \quad (15)$$

2. **Monte Carlo estimation.** Using L samples $\psi_l^t \sim q_\phi(\psi|D^t)$ we obtain

$$\mathcal{L}(\phi) = -\frac{1}{MT}\sum_{t=1}^T\sum_{m=1}^M \log\frac{1}{L}\sum_{l=1}^L \mathcal{N}(y_m^t; \psi_l^t, \sigma_y^2). \quad (16)$$

3. **Variational inference.** We use the inference network, with a second set of parameters ϕ', as variational posterior given both $\tilde{D}^t$ and D^t. Using L samples $\psi_l^t \sim q_{\phi'}(\psi|\tilde{D}^t, D^t)$, we obtain

$$\mathcal{L}(\phi) = -\frac{1}{T}\sum_{t=1}^T\left[\sum_{m=1}^M\frac{1}{L}\sum_{l=1}^L \log\mathcal{N}(y_m^t; \psi_l^t, \sigma_y^2) - \mathcal{D}_{\text{KL}}(q_{\phi'}(\psi|\tilde{D}^t, D^t)||q_\phi(\psi|D^t))\right]. \quad (17)$$



(a) $\sigma_y = 0.1$ (b) $\sigma_y = 0.5$ (c) $\sigma_y = 1.0$

Figure 3. Ratio between the variance in ψ estimated by the trained inference network $q_\phi(\psi|D^t)$ and σ_p^2 in true posterior $p(\psi|D^t)$, for different number of samples L from the inference network during training.

We trained with these three approaches for $\sigma_y \in \{0.1, 0.5, 1.0\}$. For Monte Carlo and variational methods, we used the re-parameterization trick to differentiate through sampling ψ (Kingma & Welling, 2014; Rezende et al., 2014). We evaluate the quality of the trained inference network by sampling data D^t for a new task from the data generating process Eq. (12). For new data, we compare the true posterior $p(\psi|D^t)$ with the distribution $q_\phi(\psi|D^t)$ produced by the trained inference network.

Results in Figure 3 show that both the analytic and variational approaches recover true posterior very well, including variational training with a single sample. Monte Carlo training, on the other hand, requires the use of significantly larger sets of samples to produce results comparable to other two approaches. Optimization with a small number of samples leads to significant underestimation of the target variance. This makes the Monte Carlo training approach either computationally expensive, or inaccurate in modeling the uncertainty in the latent variable.

4.2. Experimental Setup for Image Classification

MiniImageNet (Vinyals et al., 2016) consists of 100 classes selected from ILSVRC-12 (Russakovsky et al., 2015). We follow the split from Ravi & Larochelle (2017) with 64 meta-train, 16 meta-validation and 20 meta-test classes, and 600 images in each class. Following Oreshkin et al. (2018), we use a central square crop, and resize it to 84×84 pixels.

FC100 (Oreshkin et al., 2018) was derived from CIFAR-100 (Krizhevsky, 2009), which consists of 100 classes, with 600 32×32 images per class. All classes are grouped into 20 superclasses. The data is split by superclass to minimize the information overlap. There are 60 meta-train classes from 12 superclasses, 20 meta-validation, and meta-test classes, each from four corresponding superclasses.

CIFAR-FS (Bertinetto et al., 2019) is another meta-learning dataset derived from CIFAR-100. It was created by a random split into 64 meta-train, 16 meta-validation and 20 meta-test classes. For each class, there are 600 images of size 32×32.

Network architectures and training specifications. For a fair comparison with VERSA (Gordon et al., 2019), we follow the same experimental setup, including the network architectures, optimization procedure, and episode sampling. In particular, we use the shallow **CONV-5** feature extractor. In other experiments we use **ResNet-12** backbone feature extractor (Oreshkin et al., 2018; Mishra et al., 2018). The cosine classifier is scaled by setting α to 25 when data augmentation is not used, and 50 otherwise. The hyperparameters were chosen through cross-validation. The TEN network used for task conditioning is the same as in Oreshkin et al. (2018). The main and auxiliary tasks are trained concurrently: in episode t out of T, the auxiliary task is sampled with probability $\rho = 0.9^{\lfloor 12t/T \rfloor}$. The choice of β, as well as other details about the architecture and training procedure can be found in the supplementary material. We provide implementaion of our method at: `https://github.com/katafeya/samovar`.

Unless explicitly mentioned, we do not use data augmentation. In cases where we do use augmentation, it is performed with random horizontal flips, random crops, and color jitter (brightness, contrast and saturation).

Evaluation. We evaluate classification accuracy by randomly sampling 5,000 episodes, and 15 queries per class in each test episode. We also report 95% confidence intervals computed over these 5,000 tasks. We draw $d = 1,000$ samples for each class n from the corresponding prior to make a prediction, and average the resulting probabilities for the final classification.

4.3. Few-Shot Image Classification Results

Comparison with VERSA. In our first experiment, we compare SAMOVAR-base with VERSA (Gordon et al., 2019). Both use the same model, but differ only in their training procedure. We used the code provided by Gordon et al. (2019) to implement both approaches, making one important change: we avoid compression artefacts by

Table 2. Accuracy and 95% confidence intervals of TADAM and SAMOVAR on the 5-way classification task on miniImageNet. The first columns indicate the use of: cosine scaling (α), auxiliary co-training (AT), and task embedding network (TEN).

			5-SHOT		1-SHOT	
α	AT	TEN	TADAM	SAMOVAR	TADAM	SAMOVAR
			73.5 ± 0.2	75.3 ± 0.2	58.2 ± 0.3	59.3 ± 0.3
✓			74.9 ± 0.2	76.9 ± 0.2	57.4 ± 0.3	58.2 ± 0.3
	✓		74.6 ± 0.2	76.4 ± 0.2	58.7 ± 0.3	59.8 ± 0.3
		✓	72.9 ± 0.2	74.9 ± 0.2	58.2 ± 0.3	58.8 ± 0.3
✓	✓		75.7 ± 0.2	77.2 ± 0.2	57.3 ± 0.3	60.4 ± 0.3
	✓	✓	74.1 ± 0.2	77.3 ± 0.2	57.5 ± 0.3	59.5 ± 0.3
✓		✓	74.9 ± 0.2	76.8 ± 0.2	57.3 ± 0.3	58.5 ± 0.3
✓	✓	✓	75.9 ± 0.2	77.5 ± 0.2	57.6 ± 0.3	60.7 ± 0.3

Table 1. Accuracy and 95% confidence intervals of VERSA and SAMOVAR on the 5-way classification task on miniImageNet. Both approaches train the same meta-learning model.

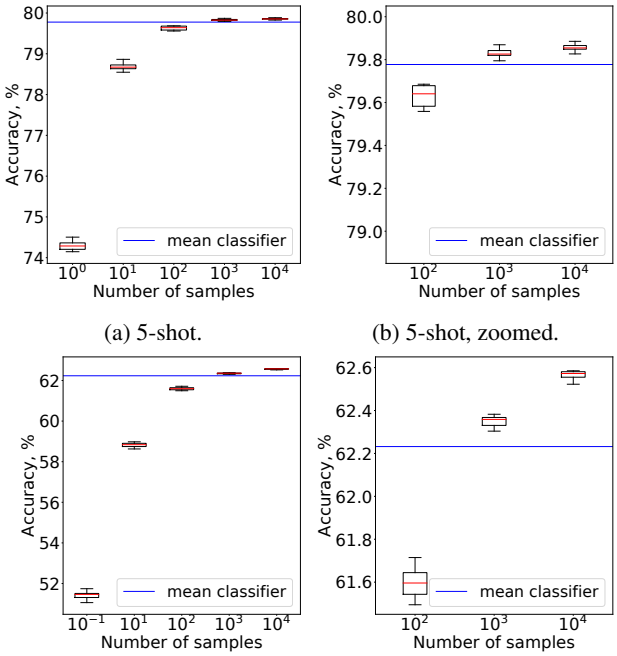
	5-SHOT	1-SHOT
VERSA (OUR IMPLEM.)	68.0 ± 0.2	52.5 ± 0.3
SAMOVAR-BASE	69.8 ± 0.2	52.4 ± 0.3
SAMOVAR-BASE (SEPARATE)	66.6 ± 0.2	50.8 ± 0.3

storing image crops in PNG rather than JPG format, which improves results noticeably.

In Table 1 we report the accuracy on miniImageNet for both the models. In the 1-shot setup, both the approaches lead to similar results, while SAMOVAR yields considerably better performance in the 5-shot setup. When training VERSA we keep track of the largest variance predicted for model parameters, and observe that it quickly deteriorates from the beginning of training. We do not observe this collapse in SAMOVAR. This is consistent with the results obtained on synthetic data. More details about distribution collapse in VERSA are presented in the supplementary material.

To evaluate the effect of sharing the inference network between prior and posterior, we run SAMOVAR-base with separate neural networks for prior and posterior, and with the reduced number of hidden units to even out the total number of parameters. From the results in the last two lines of Table 1, it can be seen that for both 1-shot and 5-shot classification sharing the inference network has a positive impact on the performance.

Comparison with TADAM. In our second experiment, we use SAMOVAR in combination with the architecture of TADAM (Oreshkin et al., 2018). To fit our framework, we replace the prototype classifier of TADAM with a linear classifier with latent weights. We compare TADAM and SAMOVAR with metric scaling (α), auxiliary co-training (AT) and the task embedding network (TEN) included or not. When the metric is not scaled, we use SAMOVAR-base with the linear classifier, otherwise we use SAMOVAR-SC with the scaled cosine classifier. For this ablative study we fix the random seed to generate the same series of meta train, meta validation and meta test tasks for both models, and for all configurations. The results in Table 2 show that SAMOVAR provides a consistent improvement over TADAM across all the tested ablations of the TADAM architecture.



(a) 5-shot. (b) 5-shot, zoomed.

(c) 1-shot. (d) 1-shot, zoomed.

Figure 4. Accuracy on miniImageNet as a function of the number of samples drawn from the learned prior over the classifier weights, compared to using the mean of the distribution.

Effect of sampling classifier weights. To assess the effect of the stochasticity of the model, we evaluate the prediction accuracy obtained with the mean of the distribution on clas-

Table 3. Accuracy and 95% confidence intervals of state-of-the-art models on the 5-way task on miniImageNet. Versions of the models that use additional data during training are not included. Exception is made only if this is the sole result provided by the authors. $*$: Results obtained with data augmentation. †: Transductive methods. ○: Validation set is included into training. △: Based on a 1.25×wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
MATCHING NETS(VINYALS ET AL., 2016)	CONV-4	60.0	46.6	
META LSTM(RAVI & LAROCHELLE, 2017)	CONV-4	60.6 ± 0.7	43.4 ± 0.8	600 EP. / 5×15
MAML (FINN ET AL., 2017)	CONV-4	63.1 ± 0.9	48.7 ± 1.8	600 EP. / 5 × SHOT
RELATIONNET (SUNG ET AL., 2018)	CONV-4	65.3 ± 0.7	50.4 ± 0.8	600 EP. / 5 × 15
PROTOTYPICAL NETS (SNELL ET AL., 2017)	CONV-4	65.8 ± 0.7	46.6 ± 0.8	600 EP. / 5 × 15
VERSA (GORDON ET AL., 2019)	CONV-5	67.4 ± 0.9	53.4 ± 1.8	600 EP. / 5 × SHOT
TPN (LIU ET AL., 2019)	CONV-4†	69.9	55.5	2000 EP. / 5 × 15
SIB(HU ET AL., 2020)	CONV-4†	70.7 ± 0.4	58.0 ± 0.6	2000 EP. / 5 × 15
GIDARIS ET AL. (2019)	CONV-4	71.9 ± 0.3	54.8 ± 0.4	2000 EP. / 5 × 15
SAMOVAR-BASE (OURS)	CONV-5	69.8 ± 0.2	52.4 ± 0.3	5000 EP. / 5 × 15
QIAO ET AL. (2018)	WRN-28-10	73.7 ± 0.2	59.6 ± 0.4	1000 EP. / 5 × 15
MTL HT (SUN ET AL., 2019)	RESNET-12	75.5 ± 0.8	61.2 ± 1.8	600 EP. / 5 × SHOT
TADAM (ORESHKIN ET AL., 2018)	RESNET-12	76.7 ± 0.3	58.5 ± 0.3	5000 EP. / 100
LEO (RUSU ET AL., 2019)	WRN-28-10*○	77.6 ± 0.1	61.8 ± 0.1	10000 EP. / 5 × 15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*	78.2 ± 0.5	57.7 ± 0.6	1000 EP. / 5 × 15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*†	78.4 ± 0.5	65.7 ± 0.7	1000 EP. / 5 × 15
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12*△	78.6 ± 0.5	62.6 ± 0.6	2000 EP. / 5 × 15
SIB (HU ET AL., 2020)	WRN-28-10*†	79.2 ± 0.4	70.0 ± 0.6	2000 EP. / 5 × 15
GIDARIS ET AL. (2019)	WRN-28-10*	79.9 ± 0.3	62.9 ± 0.5	2000 EP. / 5 × 15
CTM (LI ET AL., 2019)	RESNET-18*†	80.5 ± 0.1	64.1 ± 0.8	600 EP. / 5 × 15
DVORNIK ET AL. (2019)	WRN-28-10*	80.6 ± 0.4	63.1 ± 0.6	1000 EP. / 5 × 15
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12	77.5 ± 0.2	60.7 ± 0.3	5000 EP. / 100
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12*	79.5 ± 0.2	63.3 ± 0.3	5000 EP. / 5 × 15

sifier weights, and approximating the predictive distribution of Eq. (7) with a varying number of samples of the classifier weights. For both the 5-shot and 1-shot setups, we fix the random seed and evaluate SAMOVAR-SC-AT-TEN on the same 1,000 random 5-way tasks. We compute accuracy 10 times for each number of samples.

Results of these experiments for 5-shot and 1-shot tasks are shown in Figure 4. It can be seen that for both setups the mean classification accuracy is positively correlated with the number of samples. This is expected as a larger sample size corresponds to a better estimation of the predictive posterior distribution. The dispersion of accuracy for a fixed n is slightly bigger for the 1-shot setup compared to the 5-shot setup, and in both cases it decreases as we use more samples. This difference is also expected, as the 1-shot task is much harder than the 5-shot task, so the model retains more uncertainty in the inference in the former case. The results also show that the predicted classifier mean demonstrates good results on both classification tasks, and it can be used instead of classifier samples in cases where computational budget is critical. At the same time we can see that sampling of a large number of classifiers leads to a better performance compared to the classifier mean. While on the 5-shot setup the gain from classifier sampling over using the mean is small, around 0.1% with 10K samples, on the 1-shot setup the model benefits more from the stochasticity yielding additional 0.4% accuracy with 10K samples.

Comparison to the state of the art. In Table 3, we compare SAMOVAR to the state of the art on miniImageNet. For a fair comparison, we report results with and without data augmentation. SAMOVAR yields competitive results, notably outperforming other approaches using ResNet-12 features. The only approaches reporting better results explore techniques that are complementary to ours. Self-supervised co-training was used by Gidaris et al. (2019), which can be used as an alternative to the auxiliary 64-class classification task we used. CTM (Li et al., 2019) is a recent transductive extension to distance-based models, it identifies task-relevant features using inter- and intra-class relations. This module can also be used in conjunction with SAMOVAR, in particular, as an input to the inference network instead of the prototypes. Finally, knowledge distillation on an ensemble of 20 metric-based classifiers was used by Dvornik et al. (2019), which can be used as an alternative feature extractor in our work.

In Table 4, we compare to the state of the art on the FC100 dataset. We train our model using data augmentation. SAMOVAR yields the best results on the 5-shot

Table 4. Accuracy and 95% confidence intervals of state-of-the-art models on the 5-way task on FC100. Versions of the models that use additional data during training are not included. $*$: Results obtained with data augmentation. $\square$: Results from Lee et al. (2019). †: Transductive methods. $\triangle$: Based on a 1.25×wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
PROTOTYPICAL NETS (SNELL ET AL., 2017)	RESNET-12$^{*\square\triangle}$	52.5 ± 0.6	37.5 ± 0.6	2000 EP. / 5 × 15
TADAM (ORESHKIN ET AL., 2018)	RESNET-12	56.1 ± 0.4	40.1 ± 0.4	5000 EP. / 100
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12$^{*\triangle}$	55.5 ± 0.6	41.1 ± 0.6	2000 EP. / 5 × 15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*	57.2 ± 0.6	38.3 ± 0.5	1000 EP. / 5 × 15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10*†	57.6 ± 0.6	43.2 ± 0.6	1000 EP. / 5 × 15
MTL HT (SUN ET AL., 2019)	RESNET-12*	57.6 ± 0.9	45.1 ± 1.8	600 EP. / 5 × SHOT
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12*	57.9 ± 0.3	42.1 ± 0.3	5000 EP. / 5 × 15

Table 5. Accuracy and 95% confidence intervals of state-of-the-art models on the 5-way task on CIFAR-FS. Versions of the models that use additional data during training are not included. All models use data augmentation. $\square$: Results from Lee et al. (2019). †: Transductive methods. $\triangle$: Based on a 1.25×wider ResNet-12 architecture.

METHOD	FEATURES	5-SHOT	1-SHOT	TEST PROTOCOL
PROTOTYPICAL NETS (SNELL ET AL., 2017)	RESNET-12$^{\square\triangle}$	83.5 ± 0.5	72.2 ± 0.7	2000 EP. / 5 × 15
METAOPTNET-SVM (LEE ET AL., 2019)	RESNET-12$^{\triangle}$	84.2 ± 0.5	72.0 ± 0.7	2000 EP. / 5 × 15
FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10	86.1 ± 0.5	68.7 ± 0.7	1000 EP. / 5 × 15
TRANSDUCTIVE FINE-TUNING (DHILLON ET AL., 2020)	WRN-28-10†	85.8 ± 0.6	76.6 ± 0.7	1000 EP. / 5 × 15
SIB (HU ET AL., 2020)	WRN-28-10†	85.3 ± 0.4	80.0 ± 0.6	2000 EP. / 5 × 15
GIDARIS ET AL. (2019)	WRN-28-10	86.1 ± 0.2	73.6 ± 0.3	2000 EP. / 5 × 15
SAMOVAR-SC-AT-TEN (OURS)	RESNET-12	85.3 ± 0.2	72.5 ± 0.3	5000 EP. / 5 × 15

classification task. Transductive fine-tuning (Dhillon et al., 2020) reports a higher accuracy for the 1-shot setting, but is not directly comparable due to the transductive nature of their approach. MTL HT (Sun et al., 2019) reports the best results (with large 95% confidence intervals due to the small amount of data used in their evaluation) in the 1-shot setting. It samples hard tasks after each meta-batch update by taking its m hardest classes, and makes additional updates of the optimizer on these tasks. This is complementary, and can be used in combination with our approach to further improve the results.

In Table 5, we compare our model to the state of the art on CIFAR-FS. Data augmentation is used during training. Similar to the aforementioned datasets, SAMOVAR yields competitive results on both tasks. On the 5-shot task, higher accuracy is reported by Dhillon et al. (2020) and Gidaris et al. (2019), while transductive SIB (Hu et al., 2020) is comparable to SAMOVAR. On the 1-shot task, SIB (Hu et al., 2020), transductive version by Dhillon et al. (2020) and Gidaris et al. (2019) report better results. Overall, the observations are consistent with those on miniImageNet.

5. Conclusion

We proposed SAMOVAR, a meta-learning model for few-shot image classification that treats classifier weight vectors as latent variables, and uses a shared amortized variational inference network for the prior and variational posterior. Through experiments on synthetic data and few-shot image classification, we show that our variational approach avoids the severe under-estimation of the variance in the classifier weights observed for training with direct Monte Carlo approximation (Gordon et al., 2019). We integrate SAMOVAR with the deterministic TADAM architecture (Oreshkin et al., 2018), and find that our stochastic formulation leads to significantly improved performance, competitive with the state of the art on the miniImageNet, CIFAR-FS and FC100 datasets.

Acknowledgements

We would like to thank the reviewers for their time and constructive comments. This work was supported in part by the AVENUE project (grant ANR-18-CE23-0011).

References

Bertinetto, L., Henriques, J., Valmadre, J., Torr, P., and Vedaldi, A. Learning feed-forward one-shot learners. In *NeurIPS*, 2016.

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. Meta-learning with differentiable closed-form solvers. In *ICLR*, 2019.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and accurate deep network learning by exponential linear units (ELUs). In *ICLR*, 2016.

Dhillon, G. S., Chaudhari, P., Ravichandran, A., and Soatto, S. A baseline for few-shot image classification. In *ICLR*, 2020.

Dumoulin, V., Shlens, J., and Kudlur, M. A learned representation for artistic style. In *ICLR*, 2017.

Dvornik, N., Schmid, C., and Mairal, J. Diversity with cooperation: Ensemble methods for few-shot classification. In *ICCV*, 2019.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, 2017.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D., Eslami, S., and Teh, Y. Neural processes. In *ICML workshop on theoretical foundations and applications of deep generative models*, 2018.

Gidaris, S., Bursuc, A., Komodakis, N., Pérez, P., and Cord, M. Boosting few-shot visual learning with self-supervision. In *ICCV*, 2019.

Gordon, J., Bronskill, J., Bauer, M., Nowozin, S., and Turner, R. Meta-learning probabilistic inference for prediction. In *ICLR*, 2019.

Grant, E., Finn, C., Levine, S., Darrell, T., and Griffiths, T. Recasting gradient-based meta-learning as hierarchical bayes. In *ICLR*, 2018.

Guillaumin, M., Mensink, T., Verbeek, J., and Schmid, C. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

Ha, D., Dai, A., and Le, Q. HyperNetworks. In *ICLR*, 2017.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. Beta-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017.

Hu, S. X., Moreno, P., Xiao, Y., Shen, X., Obozinski, G., Lawrence, N., and Damianou, A. Empirical Bayes transductive meta-learning with synthetic gradients. In *ICLR*, 2020.

Kim, H., Mnih, A., Schwarz, J., Garnelo, M., Eslami, A., Rosenbaum, D., Vinyals, O., and Teh, Y. Attentive neural processes. In *ICLR*, 2019.

Kim, Y., Wiseman, S., Miller, A., Sontag, D., and Rush, A. Semi-amortized variational autoencoders. In *ICML*, 2018.

Kingma, D. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014.

Kochurov, M., Garipov, T., Podoprikhin, D., Molchanov, D., Ashukha, A., and Vetrov, D. Bayesian incremental learning for deep neural networks. In *ICLR*, 2018.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, 2009.

Lee, K., Maji, S., Ravichandran, A., and Soatto, S. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

Li, H., Eigen, D., Dodge, S., Zeiler, M., and Wang, X. Finding Task-Relevant Features for Few-Shot Learning by Category Traversal. In *CVPR*, 2019.

Liu, Y., Lee, J., Park, M., Kim, S., Yang, E., Hwang, S., and Yang, Y. Learning to propagate labels: Transductive propagation network for few-shot learning. In *ICLR*, 2019.

MacKay, D. *Bayesian Methods for Adaptive Models*. PhD thesis, California Institute of Technology, 1991.

Marino, J., Yue, Y., and Mandt, S. Iterative amortized inference. In *ICML*, 2018.

Mensink, T., Verbeek, J., Perronnin, F., and Csurka, G. Metric learning for large scale image classification: Generalizing to new classes at near-zero cost. In *ECCV*, 2012.

Mishra, N., Rohaninejad, M., Chen, X., and Abbeel, P. A simple neural attentive meta-learner. In *ICLR*, 2018.

Munkhdalai, T., Yuan, X., Mehri, S., and Trischler, A. Rapid adaptation with conditionally shifted neurons. In *ICML*, 2018.

Neal, R. *Bayesian Learning for Neural Networks*. PhD thesis, University of Toronto, 1995.

Oreshkin, B., López, P. R., and Lacoste, A. TADAM: Task dependent adaptive metric for improved few-shot learning. In *NeurIPS*, 2018.

Perez, E., Strub, F., de Vries, H., Dumoulin, V., and Courville, A. C. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

Qiao, S., Liu, C., Shen, W., and Yuille, A. L. Few-shot image recognition by predicting parameters from activations. In *CVPR*, 2018.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *ICLR*, 2017.

Ren, M., Triantafillou, E., Ravi, S., Snell, J., Swersky, K., Tenenbaum, J. B., Larochelle, H., and Zemel, R. S. Meta-learning for semi-supervised few-shot classification. In *ICLR*, 2018.

Rezende, D., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *ICML*, 2014.

Russakovsky, O., Deng, J., Su, H., J.Krause, Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., and Bernstein, M. Imagenet large scale visual recognition challenge. *IJCV*, 115(3):211–252, 2015.

Rusu, A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. Meta-learning with latent embedding optimization. In *ICLR*, 2019.

Schmidhuber, J. *Evolutionary Computation: Theory and Applications*, chapter A general method for incremental self-improvement and multiagent learning, pp. 81–123. 1999.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

Sun, Q., Liu, Y., Chua, T., and Schiele, B. Meta-transfer learning for few-shot learning. In *CVPR*, 2019.

Sung, F., Yang, Y., Zhang, L., Xiang, T., Torr, P., and Hospedales, T. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

Vinyals, O., Blundell, C., Lillicrap, T., Kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. In *NeurIPS*, 2016.

Weinberger, K., Blitzer, J., and Saul, L. Distance metric learning for large margin nearest neighbor classification. In *NeurIPS*, 2006.

Ye, H.-J., Hu, H., Zhan, D.-C., and Sha, F. Learning embedding adaptation for few-shot learning. *CoRR*, 2018.

Zhang, R., Che, T., Ghahramani, Z., Bengio, Y., and Song, Y. Metagan: An adversarial approach to few-shot learning. In *NeurIPS*, 2018.

A. Network Architectures

We learn separate amortized inference networks to predict the mean μ and log-variance $\ln \sigma^2$ of the latent classification weight vectors w^t. Both networks have the same architecture, which depends on the feature extractor that is used. The inference networks are shared between the prior and approximate posterior distributions.

A.1. CONV-5 Feature Extractor

The embedding of the image returned by the CONV-5 feature extractor is a 256-dimensional vector. Each of the inference networks for the mean and log variance of the classifier weights w^t consists of three fully connected layers with 256 input and output features, and ELU non-linearity (Clevert et al., 2016) between the layers. There are two additional inference networks that predict the mean and log variance of the classifier biases b^t. Both of them consist of two fully connected layers with 256 input and output features followed by ELU non-linearity, and a fully connected layer with 256 input and a single output feature. The design is the same as used by Gordon et al. (2019) to ensure comparability.

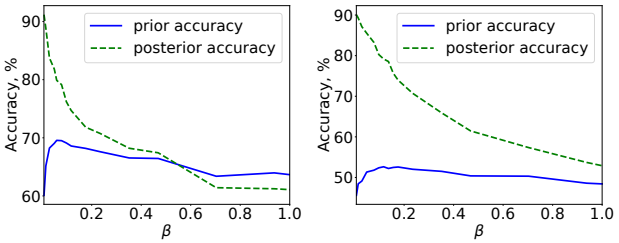
A.2. ResNet-12 Feature Extractor

With the ResNet-12 feature extractor, every image is embedded into a 512-dimensional feature vector. Each of the two inference networks consists of three fully connected layers with 512 input and output features, with skip connections and swish-1 non-linearity (Ramachandran et al., 2017) applied before addition in the first two dense layers.

B. Training Details for ResNet-12

For comparison with TADAM (Oreshkin et al., 2018) we use the same optimization procedure, number of SGD updates, and weight decay parameters for common parts of the architecture as in the paper. For experiments with data augmentation on miniImageNet we use 40k SGD updates with momentum 0.9, and early stopping based on meta-validation performance. We set the initial learning rate to 0.1, and decrease it by a factor ten after 20k, 25k and 30k updates. On FC100 and CIFAR-FS, we use 30k SGD updates with the same momentum and initial learning rate, and the latter is decreased after 15k, 20k and 25k updates. We clip gradients at 0.1, and set separate weight decay rates for the feature extractor, TEN, fully connected layer in the auxiliary task, and inference networks. For the feature extractor and TEN the weight decay is 0.0005. For the fully connected layer in the auxiliary task the weight decay is 0.00001 on miniImageNet, and 0.0005 on FC100 and CIFAR-FS. In the 1-shot setup, the inference networks are regularized with the weight decay equal to 0.0005, regardless of the dataset. In the 5-shot setup, the weight decay parameter in the inference networks is 0.00001 on miniImageNet, and 0.00005 on FC100 and CIFAR-FS. We empirically find that the regularization coefficient $\beta = \frac{K}{Nd}$ produces good results, and it can be used as a starting point for further parameter tuning. Here d is the dimensionality of the feature vector f_θ, N is the number of classes in the task, and K is the total number of query samples in the task. On CONV-5, we set β to 0.0586 for the 5-shot setup, and we multiply it by two for the 1-shot setup. On ResNet-12, we set β to 0.0125 for both setups, and we use a value of β twice as large for the 1-shot setup without auxiliary co-training.

For the 5-shot setup, mini-batches consist of two episodes, each with 32 query images. For the 1-shot setup, we sample 5 episodes per mini-batch, and 12 query images per episode. In both cases query images are sampled uniformly across classes, without any restriction on the number per class. The auxiliary 64-way classification task is trained with the batch size 64.

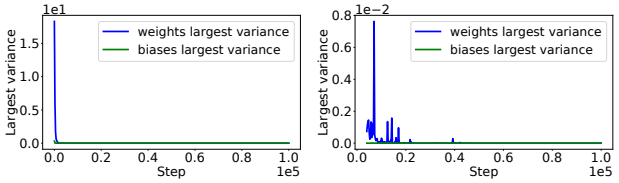


(a) 5-shot setup (b) 1-shot setup

Figure 5. Mean accuracy of the SAMOVAR-base classifiers sampled from the prior and posterior as a function of β. While training, we fix the random seed of the data to generate the same series of miniImageNet tasks. The evaluation is performed over 5000 random tasks.

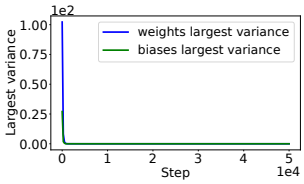
C. Impact of β-scaling

Typically, in autoencoders the dimensionality of the latent space is smaller than of the observed. This is not the case in the meta learning classification task where the output is merely a one-hot-encoded label of the class, while the latent space is of the same size as the output of the feature extractor. In our experiments we observe that the large KL term suppresses the reconstruction term resulting in a weaker performance. In particular, there is a trade off between these parts of the objective function $\hat{\mathcal{L}}(\Theta)$ which can be regulated by β-scaling of the KL term. Figure 5 shows the accuracy of SAMOVAR-base with CONV-5 feature extractor as a function of β. Even though in both setups there is a clear maximum, overall, the model is relatively robust to the setting of β. Let's denote the optimum β as β_{opt}. Then for the 5-shot setup the range at least from $0.83\beta_{\text{opt}}$ to $2\beta_{\text{opt}}$

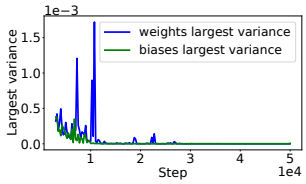


(a) 5-shot setup.

(b) 5-shot setup zoomed in.



(c) 1-shot setup.



(d) 1-shot setup zoomed in.

Figure 6. Largest variance in VERSA as a function of the optimization step. Results for optimization steps from Figure 6a and Figure 6c that follow the first encounter of variance below 0.001 are zoomed in Figure 6b Figure 6d respectively.

produces results that are within the 1% interval from the maximum accuracy at β_{opt}. For the 1-shot setup, the same holds true for the range at least from $0.66\beta_{\text{opt}}$ to $2\beta_{\text{opt}}$.

D. Posterior Collapse in VERSA

While training VERSA, every 250 optimization steps we keep track of the largest variance of the weights and biases of the predicted classifier. Figure 6 shows how this variance decreases with time. For example, the largest variance of the weights first falls below 0.001 at the step 4000 in the 5-shot setup, and at the step 3000 in the 1-shot setup.