# Meta-Learning with Latent Embedding Optimization

**Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, Raia Hadsell**
DeepMind, London, UK
{andreirusu, dushyantr, sygi, vinyals, razp, osindero, raia}@google.com

## Abstract

Gradient-based meta-learning techniques have practical difficulties when operating in high-dimensional parameter spaces and extreme low-data regimes. We bypass these limitations by learning a data-dependent latent generative representation of model parameters, and performing gradient-based meta-learning in this low-dimensional latent space. The resulting approach, *latent embedding optimization* (LEO), achieves state-of-the-art performance on the competitive *mini*ImageNet and *tiered*ImageNet few-shot classification tasks.

## 1 Introduction

*Few-shot learning tasks* challenge models to learn a new concept or behaviour with very few examples or limited experience [5, 22]. One approach to address this class of problems is *meta-learning*, a broad family of techniques focused on learning how to learn or to quickly adapt to new information. More specifically, *optimization-based* meta-learning approaches [32, 6] aim to find a single set of model parameters that can be adapted with a few steps of gradient descent to individual tasks. However, using only a few samples (typically 1 or 5) to compute gradients in a high-dimensional parameter space could make generalization difficult, especially under the constraint of a shared starting point for task-specific adaptation.

In this work we propose a new approach, named *Latent Embedding Optimization (LEO)*, which learns a low-dimensional latent embedding of model parameters and performs optimization-based meta-learning in this space. Intuitively, the approach provides two advantages. First, the initial parameters for a new task are conditioned on the training data, which enables a task-specific starting point for adaptation. By incorporating a relation network into the encoder, this initialization can better consider the joint relationship between all of the input data. Second, by optimizing in the lower-dimensional latent space, the approach can adapt the behaviour of the model more effectively. Further, by allowing this process to be stochastic, the uncertainties and ambiguities present in the few-shot data regime can be expressed.

## 2 Model

### 2.1 Problem Definition

We define the $N$-way $K$-shot problem using the episodic formulation of [40]. Each task instance $\mathcal{T}_i$ is a classification problem sampled from a task distribution $p(\mathcal{T})$. The tasks are divided into a *training meta-set* of tasks, $\mathcal{S}^{tr}$, a *validation meta-set*, $\mathcal{S}^{val}$, and a *test meta-set* $\mathcal{S}^{test}$, each with a disjoint set of target classes (i.e., a class seen during testing is not seen during training). The validation meta-set is used for model selection, and the testing meta-set is used only for final evaluation.

Each task instance $\mathcal{T}_i \sim p(\mathcal{T})$ is composed of a training set $\mathcal{D}^{tr}$ and validation set $\mathcal{D}^{val}$, and only contains $N$ classes randomly selected from the appropriate meta-set (e.g. for a task instance

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

in the training meta-set, the classes are a subset of those available in $\mathcal{S}^{tr}$). In most setups, the training set $\mathcal{D}^{tr} = \left\{(\mathbf{x}_n^k, y_n^k) \mid k = 1 \dots K; n = 1 \dots N\right\}$ contains $K$ samples for each class. The validation set $\mathcal{D}^{val}$ can contain several other samples from the same classes, providing an estimate of generalization performance on the $N$ classes for this problem instance. We note that the validation set of a problem instance $\mathcal{D}^{val}$ (used to optimize a meta-learning objective) should not be confused with the held-out validation meta-set $\mathcal{S}^{val}$ (used for model selection).

### 2.2 Latent Embedding Optimization for Meta-Learning

The primary contribution of this paper is to show that it is possible, and indeed beneficial, to decouple optimization-based meta-learning techniques from the high-dimensional space of model parameters. We achieve this by learning a stochastic latent space with an information bottleneck, conditioned on the input data, from which the high-dimensional parameters are generated.

**Algorithm 1** Latent Embedding Optimization

**Require:** Training meta-set $\mathcal{S}^{tr} \in \mathcal{T}$
**Require:** Learning rates $\alpha, \eta$
Randomly initialize $\phi_e, \phi_r, \phi_d$
Let $\phi = \{\phi_e, \phi_r, \phi_d, \alpha\}$
**while** not converged **do**
  **for** number of tasks in batch **do**
    Sample task instance $\mathcal{T}_i \sim \mathcal{S}^{tr}$
    Let $\left(\mathcal{D}^{tr}, \mathcal{D}^{val}\right) = \mathcal{T}_i$
    Encode $\mathcal{D}^{tr}$ to $\mathbf{z}$ using $g_{\phi_e}$ and $g_{\phi_r}$
    Decode $\mathbf{z}$ to initial params $\theta_i$ using $g_{\phi_d}$
    Initialize $\mathbf{z}' = \mathbf{z}$, $\theta_i' = \theta_i$
    **for** number of adaptation steps **do**
      Compute training loss $\mathcal{L}_{\mathcal{T}_i}^{tr}\left(f_{\theta_i'}\right)$
      Perform gradient step w.r.t. $\mathbf{z}'$: $\mathbf{z}' \leftarrow \mathbf{z}' - \alpha \nabla_{\mathbf{z}'} \mathcal{L}_{\mathcal{T}_i}^{tr}\left(f_{\theta_i'}\right)$
      Decode $\mathbf{z}'$ to obtain $\theta_i'$ using $g_{\phi_d}$
    **end for**
    Compute validation loss $\mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)$
  **end for**
  Perform gradient step w.r.t $\phi$: $\phi \leftarrow \phi - \eta \nabla_{\phi} \sum_{\mathcal{T}_i} \mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right)$
**end while**

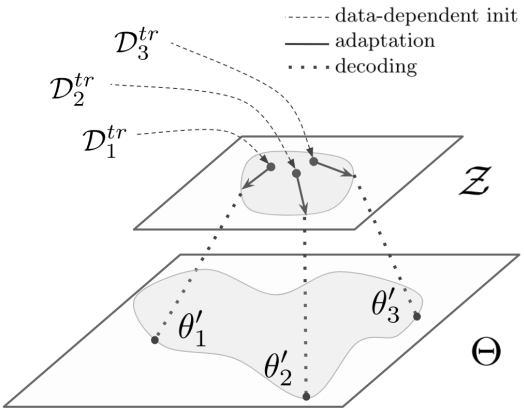


Figure 1: High-level intuition for LEO. While MAML operates directly in a high dimensional parameter space $\Theta$, LEO performs meta-learning within a low-dimensional latent space $\mathcal{Z}$, from which the parameters are generated.

Instead of explicitly instantiating and maintaining a unique set of model parameters $\theta$, as in model-agnostic meta-learning (MAML) [6], we learn a generative distribution of model parameters which serves the same purpose. This is a natural extension: we relax the requirement of finding a single optimal $\theta^* \in \Theta$ to that of approximating a data-dependent conditional probability distribution over $\Theta$, which can be more expressive, at the cost of potentially being more difficult to learn. The choice of architecture, composed of an *encoding* process, and *decoding* process (or parameter generation), enables us to perform the MAML gradient-based adaptation steps (or "inner loop") in the learned, low-dimensional embedding space of the parameter generative model (Figure 1).

The high-level operation is then as follows (Algorithm 1). First, given a task instance $\mathcal{T}_i$, the few-shot inputs $\{\mathbf{x}_n^k\}$ are passed through a stochastic encoder $g_{\phi_e}$ and relation network $g_{\phi_r}$ to produce class-conditional latent codes $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N]$[1]. A parameter generator $g_{\phi_d}$ is then used to decoded parameters $\theta_i = \left\{\mathbf{w}_n \mid n = 1 \dots N\right\}$. Given these instantiated model parameters, one or more adaptation steps are applied in the latent space, by differentiating the loss with respect to $\mathbf{z}$, taking a gradient step to get $\mathbf{z}'$, decoding new model parameters, and obtaining the new loss. Finally, optimized codes are decoded to produce the final adapted parameters $\theta_i'$, which can be used to perform the task, or compute the task-specific meta-loss. In this way, LEO incorporates aspects of model-based and optimization-based meta-learning, producing parameters that are first conditioned on the input data and then adapted by gradient descent.

[1]Note that we omit the task subscript $i$ from latent code $\mathbf{z}$ and input data $\mathbf{x}_n^k$ for clarity.

During meta-training we differentiate through the adaptation procedure and update the encoder, relation and decoder network parameters: $\phi_e$, $\phi_r$, and $\phi_d$ respectively. This process is analogous to MAML, but instead of learning a fixed set of initialization parameters we meta-learn a stochastic parameter generator. Meta-training is done by solving the following optimization problem:

$$\min_{\phi_e,\phi_r,\phi_d} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \left[ \mathcal{L}_{\mathcal{T}_i}^{val}\big(f_{\theta_i'}\big) + \beta D_{KL}\big(q(\mathbf{z}_n|\mathcal{D}_n^{tr})||p(\mathbf{z}_n)\big) + \gamma||\text{stopgrad}(\mathbf{z}_n') - \mathbf{z}_n||_2^2 \right] + \mathcal{R} \tag{1}$$

where $p(\mathbf{z}_n) = \mathcal{N}(0, \mathcal{I})$ and $\mathcal{R}$ is a regularization term. Similar to the loss defined in [13] we use a weighted KL-divergence term to encourage the generative model to learn a disentangled embedding. We also found it beneficial to encourage the encoder and relation networks to produce latent codes which are close to final adapted codes. For further details see Appendix A.2.1.

## 3 Related Work

Few-shot adaptation has been historically tackled using *fast weights* [14, 2], *learning-to-learn* methods [36, 39, 15, 1], and through *meta-learning*. The latter can be classified as metric-based [18, 40, 37], memory-based [35, 32], model-based [27, 26, 8, 9], and optimization-based [6, 7, 29].

Optimization-based methods, such as MAML [6], typically use the full, high-dimensional set of model parameters within the "inner loop", while [23] learn a layer-wise subspace in which to use gradient-based adaptation. However, it is not clear how these methods scale to large expressive models such as residual networks.

Probabilistic meta-learning approaches such as those of [3] and [11] learn Gaussian posteriors over model parameters. [16] and [7] propose probabilistic extensions to MAML that are trained using a variational approximation using simple posteriors. Other works have introduced deep parameter generators [21, 41] that can better capture a wider distribution of model parameters, but do not employ gradient-based adaptation. In contrast, our approach employs a generative model of parameters, adaptation in a low-dimensional latent space, and a data-dependent initialization.

The idea of using a neural network to produce (some fraction of) the parameters of another has a long history, with recent examples [12, 20]. In the context of few-shot adaptation, additive biases across deep networks have been conditionally generated on the few-shot training samples in [28], while in [10, 31] linear output weights for novel categories were deterministically generated using the same conditioning. None of the aforementioned approaches explicitly learn a probability distribution over model parameters, or make use of latent generative models for fast adaptation.

## 4 Few-shot Classification on *mini*ImageNet and *tiered*ImageNet

We scale up our approach to 1-shot and 5-shot classification problems defined using two commonly used ImageNet subsets. The ***mini*ImageNet** dataset [40] is a subset of 100 classes selected randomly from the ILSVRC-12 dataset [34] with 600 images sampled from each class. Following the split proposed by Ravi and Larochelle [32], the dataset is divided into training, validation, and test meta-sets, with 64, 16, and 20 classes respectively. The ***tiered*ImageNet** dataset [33] is a larger subset of ILSVRC-12 with 608 classes (779,165 images) grouped into 34 higher-level nodes in the ImageNet human-curated hierarchy [4]. This set of nodes is partitioned into 20, 6, and 8 disjoint sets of training, validation, and testing nodes, and the corresponding classes form the respective meta-sets. As argued in Ren et al. [33], this split near the root of the ImageNet hierarchy results in a more challenging, yet realistic regime with test classes that are less similar to training classes.

Two potential difficulties of using LEO to instantiate parameters with a generator network are: (1) modeling distributions over very high-dimensional parameter spaces; and (2) requiring meta-learning (and hence, gradient computation in the inner loop) to be performed with respect to a high-dimensional input space. We address these issues by pre-training a visual representation of the data and then using the generator to instantiate the final layer parameters - a linear softmax classifier operating on this representation. We train a 28-layer Wide Residual Network (WRN-28-10) [43] with supervised classification using only the 64 classes from the training meta-set and corresponding 600 images per class. Recent state-of-the-art approaches use the penultimate layer representation [31, 3, 10]; however, we choose the intermediate feature representation in layer 21, given that

higher layers tend to specialize to the training distribution [42]. For training and evaluation details, see Appendix B.1.

Following the LEO adaptation procedure (Algorithm 1) we also use fine-tuning by performing a few steps of gradient-based adaptation directly in parameter space using the few-shot set $\mathcal{D}^{tr}$; this fine-tuning is similar to the adaptation procedure of MAML, or Meta-SGD [24] when the learning rates are learned, with the important difference that starting points of fine-tuning are custom generated by LEO for every task instance $\mathcal{T}_i$. Empirically, we find that fine-tuning applies a very small change to the parameters with only a slight improvement in performance on supervised classification tasks.

### 4.1 Results

| **Model** | ***mini*ImageNet test accuracy** 1-shot | 5-shot |
|---|---|---|
| Matching networks (Vinyals et al. [40]) | $43.56 \pm 0.84\%$ | $55.31 \pm 0.73\%$ |
| Meta-learner LSTM (Ravi and Larochelle [32]) | $43.44 \pm 0.77\%$ | $60.60 \pm 0.71\%$ |
| MAML (Finn et al. [6]) | $48.70 \pm 1.84\%$ | $63.11 \pm 0.92\%$ |
| LLAMA (Grant et al. [11]) | $49.40 \pm 1.83\%$ | - |
| REPTILE (Nichol and Schulman [29]) | $49.97 \pm 0.32\%$ | $65.99 \pm 0.58\%$ |
| PLATIPUS (Finn et al. [7]) | $50.13 \pm 1.86\%$ | - |
| Meta-SGD (our features) | $54.24 \pm 0.03\%$ | $70.86 \pm 0.04\%$ |
| SNAIL (Mishra et al. [26]) | $55.71 \pm 0.99\%$ | $68.88 \pm 0.92\%$ |
| (Gidaris and Komodakis [10]) | $56.20 \pm 0.86\%$ | $73.00 \pm 0.64\%$ |
| (Bauer et al. [3]) | $56.30 \pm 0.40\%$ | $73.90 \pm 0.30\%$ |
| (Munkhdalai et al. [28]) | $57.10 \pm 0.70\%$ | $70.04 \pm 0.63\%$ |
| TADAM (Oreshkin et al. [30]) | $58.50 \pm 0.30\%$ | $76.70 \pm 0.30\%$ |
| (Qiao et al. [31]) | $59.60 \pm 0.41\%$ | $73.74 \pm 0.19\%$ |
| **LEO (ours)** | $\mathbf{61.76 \pm 0.08\%}$ | $\mathbf{77.59 \pm 0.12\%}$ |
| **Model** | ***tiered*ImageNet test accuracy** 1-shot | 5-shot |
| MAML (deeper net, evaluated in Liu et al. [25]) | $51.67 \pm 1.81\%$ | $70.30 \pm 0.08\%$ |
| Prototypical Nets (Ren et al. [33]) | $53.31 \pm 0.89\%$ | $72.69 \pm 0.74\%$ |
| Relation Net (evaluated in Liu et al. [25]) | $54.48 \pm 0.93\%$ | $71.32 \pm 0.78\%$ |
| Transductive Prop. Nets (Liu et al. [25]) | $57.41 \pm 0.94\%$ | $71.55 \pm 0.74\%$ |
| Meta-SGD (our features) | $62.95 \pm 0.03\%$ | $79.34 \pm 0.06\%$ |
| **LEO (ours)** | $\mathbf{66.33 \pm 0.05\%}$ | $\mathbf{81.44 \pm 0.09\%}$ |

Table 1: Test accuracies on *mini*ImageNet and *tiered*ImageNet. For each dataset, the first set of results use convolutional networks, while the second use much deeper residual networks, predominantly in conjuction with pre-training.

The classification accuracies for LEO and other baselines are shown in Table 1. LEO sets the new state-of-the-art performance on the 1-shot and 5-shot tasks for both *mini*ImageNet and *tiered*ImageNet datasets, with comfortable margins. We also evaluated LEO on the "multi-view" feature representation used by Qiao et al. [31] with *mini*ImageNet, which involves significant data augmentation compared to the approaches in Table 1. LEO is state-of-the-art using these features as well, with $63.97 \pm 0.20\%$ and $79.49 \pm 0.70\%$ test accuracies on the 1-shot and 5-shot tasks respectively.

## 5 Conclusions

We have introduced Latent Embedding Optimization (LEO), a meta-learning technique which uses a parameter generative model to capture the diverse range of parameters useful for a distribution over tasks, and demonstrated a new state-of-the-art result on the challenging 5-way 1- and 5-shot *mini*ImageNet and *tiered*ImageNet classification problems. LEO achieves this by learning a low-dimensional data-dependent latent embedding, and performing gradient-based adaptation in this space, which means that it allows for a task-specific parameter initialization and can perform adaptation more effectively.

## References


[1] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *Advances in Neural Information Processing Systems*, pages 3981–3989, 2016.

[2] 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*, pages 4331–4339, 2016.

[3] M. Bauer, M. Rojas-Carulla, J. Bartłomiej Świątkowski, B. Schölkopf, and R. E. Turner. Discriminative k-shot learning using probabilistic models. *ArXiv e-prints*, June 2017.

[4] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, June 2009.

[5] Li Fei-Fei, Rob Fergus, and Pietro Perona. One-shot learning of object categories. *IEEE transactions on pattern analysis and machine intelligence*, 28(4):594–611, 2006.

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

[7] Chelsea Finn, Kelvin Xu, and Sergey Levine. Probabilistic model-agnostic meta-learning. *arXiv preprint arXiv:1806.02817*, 2018.

[8] Marta Garnelo, Dan Rosenbaum, Christopher Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo Rezende, and S. M. Ali Eslami. Conditional neural processes. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1704–1713, Stockholmsmssan, Stockholm Sweden, 10–15 Jul 2018. PMLR. URL `http://proceedings.mlr.press/v80/garnelo18a.html`.

[9] Marta Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. M. A. Eslami, and Y. Whye Teh. Neural Processes. *ArXiv e-prints*, July 2018.

[10] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. *CoRR*, abs/1804.09458, 2018. URL `http://arxiv.org/abs/1804.09458`.

[11] Erin Grant, Chelsea Finn, Sergey Levine, Trevor Darrell, and Thomas Griffiths. Recasting gradient-based meta-learning as hierarchical bayes. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[12] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[13] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. Beta-VAE: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2017.

[14] 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*, pages 177–186, 1987.

[15] Sepp Hochreiter, A. Steven Younger, and Peter R. Conwell. Learning to learn using gradient descent. In *Proceedings of the International Conference on Artificial Neural Networks*, ICANN '01, pages 87–94, London, UK, UK, 2001. Springer-Verlag. ISBN 3-540-42486-5. URL `http://dl.acm.org/citation.cfm?id=646258.684281`.

[16] T. Kim, J. Yoon, O. Dia, S. Kim, Y. Bengio, and S. Ahn. Bayesian Model-Agnostic Meta-Learning. *ArXiv e-prints*, June 2018.

[17] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL `http://arxiv.org/abs/1412.6980`.

[18] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] D. Krueger, C.-W. Huang, R. Islam, R. Turner, A. Lacoste, and A. Courville. Bayesian Hypernetworks. *ArXiv e-prints*, October 2017.

[21] A. Lacoste, B. Oreshkin, W. Chung, T. Boquet, N. Rostamzadeh, and D. Krueger. Uncertainty in Multitask Transfer Learning. *ArXiv e-prints*, June 2018.

[22] Brenden Lake, Ruslan Salakhutdinov, Jason Gross, and Joshua Tenenbaum. One shot learning of simple visual concepts. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 33, 2011.

[23] Y. Lee and S. Choi. Gradient-Based Meta-Learning with Learned Layerwise Metric and Subspace. *ArXiv e-prints*, January 2018.

[24] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few shot learning. *CoRR*, abs/1707.09835, 2017. URL `http://arxiv.org/abs/1707.09835`.

[25] Yanbin Liu, Juho Lee, Minseop Park, Saehoon Kim, and Yi Yang. Transductive propagation network for few-shot learning. *arXiv preprint arXiv:1805.10002*, 2018.

[26] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *Proceedings of the 6th International Conference on Learning Representations (ICLR)*, 2018.

[27] Tsendsuren Munkhdalai and Hong Yu. Meta networks. *CoRR*, abs/1703.00837, 2017. URL `http://arxiv.org/abs/1703.00837`.

[28] Tsendsuren Munkhdalai, Xingdi Yuan, Soroush Mehri, Tong Wang, and Adam Trischler. Learning rapid-temporal adaptations. *CoRR*, abs/1712.09926, 2017. URL `http://arxiv.org/abs/1712.09926`.

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

[30] B. N. Oreshkin, P. Rodriguez, and A. Lacoste. TADAM: Task dependent adaptive metric for improved few-shot learning. *ArXiv e-prints*, May 2018.

[31] Siyuan Qiao, Chenxi Liu, Wei Shen, and Alan Yuille. Few-shot image recognition by predicting parameters from activations. *arXiv preprint arXiv:1706.03466*, 2017.

[32] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

[33] Mengye Ren, Sachin Ravi, Eleni Triantafillou, Jake Snell, Kevin Swersky, Josh B. Tenenbaum, Hugo Larochelle, and Richard S. Zemel. Meta-learning for semi-supervised few-shot classification. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=HJcSzz-CZ`.

[34] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014. URL `http://arxiv.org/abs/1409.0575`.

[35] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850, 2016.

[36] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[37] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. *CoRR*, abs/1703.05175, 2017. URL `http://arxiv.org/abs/1703.05175`.

[38] Flood Sung, Yongxin Yang, Li Zhang, Tao Xiang, Philip H. S. Torr, and Timothy M. Hospedales. Learning to compare: Relation network for few-shot learning. *CoRR*, abs/1711.06025, 2017. URL `http://arxiv.org/abs/1711.06025`.

[39] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[40] Oriol Vinyals, Charles Blundell, Tim Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in Neural Information Processing Systems*, pages 3630–3638, 2016.

[41] T. Wu, J. Peurifoy, I. L. Chuang, and M. Tegmark. Meta-learning autoencoders for few-shot prediction. *ArXiv e-prints*, July 2018.

[42] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *CoRR*, abs/1411.1792, 2014. URL `http://arxiv.org/abs/1411.1792`.

[43] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.

[44] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL `http://arxiv.org/abs/1605.07146`.

# A Implementation Details

## A.1 Model-Agnostic Meta-Learning

Model-agnostic meta-learning (MAML) [6] is an approach to optimization-based meta-learning that is related to our work. For some parametric model $f_\theta$, MAML aims to find a single set of parameters $\theta$ which, using a few optimization steps, can be successfully adapted to any novel task sampled from the same distribution. For a particular task instance $\mathcal{T}_i = \left(\mathcal{D}^{tr}, \mathcal{D}^{val}\right)$, the parameters are adapted to task-specific model parameters $\theta_i'$ by applying some differentiable function, typically an update rule of the form:

$$\theta_i' = \mathcal{G}\left(\theta, \mathcal{D}^{tr}\right), \tag{2}$$

where $\mathcal{G}$ is typically implemented as a step of gradient descent on the few-shot training set $\mathcal{D}^{tr}$, $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}^{tr}\left(f_\theta\right)$. Generally, multiple sequential adaptation steps can be applied. During meta-training, the parameters $\theta$ are updated by back-propagating through the adaptation procedure, in order to reduce errors on the validation set $\mathcal{D}^{val}$:

$$\theta \leftarrow \theta - \eta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}^{val}\left(f_{\theta_i'}\right) \tag{3}$$

The learning rate $\alpha$ can also be meta-learned concurrently, in which case we refer to this algorithm as Meta-SGD [24]. In all cases we used per-parameter learning rates.

The approach includes the main ingredients of optimization-based meta-learning with neural networks: *initialization* is done by maintaining an explicit set of model parameters $\theta$; the *adaptation procedure*, or "inner loop", takes parameters $\theta$ as input and returns parameters $\theta_i'$ adapted specifically for task instance $\mathcal{T}_i$, by using gradient descent iteratively with some learning rate schedule (Eq. 2); and *termination*, which is handled simply by choosing a fixed number of optimization steps in the "inner loop". MAML updates $\theta$ by differentiating through the "inner loop" in order to minimize errors of instance-specific adapted models $f_{\theta_i'}$ on the corresponding validation set (Eq. 3). We refer to this process as the "outer loop" of meta-learning. In the next section we use the same procedural stages to describe fast adaptation by Latent Embedding Optimization (LEO).

In the following we use the same stages to explain the LEO procedure more formally.

## A.2 Latent Embedding Optimization

### A.2.1 Initialization: Generating Parameters Conditioned on a Few Examples

The first stage is to instantiate the model parameters that will be adapted to each task instance. Whereas MAML explicitly maintains a single set of model parameters, LEO utilises a data-dependent latent encoding which is then decoded to generate the actual initial parameters. In what follows, we describe an encoding scheme which leverages the power of relation networks to map the few-shot examples into a single latent vector; this particular design choice worked very well in the experiments we considered, but other choices for this mapping are possible.

**Encoding** The encoding process involves a simple feedforward mapping of each data point, followed by a relation network that considers the joint relationship between all of the data in the problem instance. The overall encoding process is defined in Eq. 4, and proceeds as follows. First, each single example from a problem instance $\mathcal{T}_i = \left(\mathcal{D}^{tr}, \mathcal{D}^{val}\right) \sim p\left(\mathcal{T}\right)$ is processed by an encoder network $g_{\phi_e} : \mathcal{R}^{n_x} \rightarrow \mathcal{R}^{n_h}$, which maps from input space to a code in an intermediate hidden-layer code space $\mathcal{H}$. Then, codes in $\mathcal{H}$ corresponding to different training examples are concatenated pair-wise (resulting in $(NK)^2$ pairs in the case of $K$-shot classification) and processed by a relation network $g_{\phi_r}$, in a fashion similar to [30] and [38]. The $(NK)^2$ outputs are grouped by class and then averaged within each group to obtain the $(2 \times N)$ parameters of a probability distribution in a low-dimensional space $\mathcal{Z} = \mathcal{R}^{n_z}$, where $n_z \ll \dim(\theta)$, for each of the $N$ classes.

Thus, given the $K$-shot training samples corresponding to a class $n$: $\mathcal{D}_n^{tr} = \left\{\left(\mathbf{x}_n^k, y_n^k\right) \mid k = 1 \ldots K\right\}$ the encoder $g_{\phi_e}$ and relation network $g_{\phi_r}$ together parameterize a class-conditional multivariate Gaussian distribution with a diagonal covariance, which we can sample from in order to

output a class-dependent latent code $\mathbf{z}_n \in \mathcal{Z}$ as follows:

$$\begin{aligned} \boldsymbol{\mu}_n^e, \boldsymbol{\sigma}_n^e &= \frac{1}{NK^2} \sum_{k_n=1}^{K} \sum_{m=1}^{N} \sum_{k_m=1}^{K} g_{\phi_r}\Big(g_{\phi_e}\left(\mathbf{x}_n^{k_n}\right), g_{\phi_e}\left(\mathbf{x}_m^{k_m}\right)\Big) \\ \mathbf{z}_n &\sim q\left(\mathbf{z}_n | \mathcal{D}_n^{tr}\right) = \mathcal{N}\left(\boldsymbol{\mu}_n^e, diag(\boldsymbol{\sigma}_n^{e\,2})\right) \end{aligned} \tag{4}$$

Intuitively, the encoder and relation network define a stochastic mapping from one or more class examples to a single code in the latent embedding space $\mathcal{Z}$ corresponding to that class. The final latent code can be obtained as the concatenation of class-dependent codes: $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N]$.

**Decoding** Without loss of generality, for few-shot classification, we can use the class-specific latent codes to instantiate just the top layer weights of the classifier. This allows the meta-learning in latent space to modulate the important high-level parameters of the classifier, without requiring the generator to produce very high-dimensional parameters. In this case, $f_{\theta_i'}$ is a simple $N$-way linear softmax classifier, with model parameters $\theta_i' = \left\{\mathbf{w}_n \mid n = 1 \ldots N\right\}$, and each $\mathbf{x}_n^k$ can be either the raw input or some learned representation[2].

Then, given the low-dimensional latent codes $\mathbf{z}_n \in \mathcal{Z}, n = 1 \ldots N$, the decoder function $g_{\phi_d} \colon \mathcal{Z} \to \Theta$ is used to parameterize a Gaussian distribution with diagonal covariance in model parameter space $\Theta$, from which we can sample class-dependent parameters $\mathbf{w}_n$:

$$\begin{aligned} \boldsymbol{\mu}_n^d, \boldsymbol{\sigma}_n^d &= g_{\phi_d}\left(\mathbf{z}_n\right) \\ \mathbf{w}_n &\sim p\left(\mathbf{w} | \mathbf{z}_n\right) = \mathcal{N}\left(\boldsymbol{\mu}_n^d, diag(\boldsymbol{\sigma}_n^{d\,2})\right) \end{aligned} \tag{5}$$

In other words, codes $\mathbf{z}_n$ are mapped independently to the top-layer parameters $\theta_i$ of a softmax classifier using the decoder $g_{\phi_d}$, which is essentially a stochastic generator of model parameters.

### A.2.2 Adaptation by Latent Embedding Optimization (LEO) (The "Inner Loop")

Given the decoded parameters, we can then define the "inner loop" classification loss using the cross-entropy function, as follows:

$$\mathcal{L}_{\mathcal{T}_i}^{tr}\left(f_{\theta_i}\right) = \sum_{(\mathbf{x},y) \in \mathcal{D}^{tr}} \Big[ -\mathbf{w}_y \cdot \mathbf{x} + \log\Big(\sum_{j=1}^{N} e^{\mathbf{w}_j \cdot \mathbf{x}}\Big)\Big] \tag{6}$$

It is important to note that the decoder $g_{\phi_d}$ is a differentiable mapping between the latent space $\mathcal{Z}$ and the higher-dimensional model parameter space $\Theta$. Primarily, this allows gradient-based optimization of the latent codes with respect to the training loss, with $\mathbf{z}_n' = \mathbf{z}_n - \alpha \nabla_{\mathbf{z}_n} \mathcal{L}_{\mathcal{T}_i}^{tr}$. The decoder $g_{\phi_d}$ will convert adapted latent codes $\mathbf{z}_n'$ to effective model parameters $\theta_i'$ for each adaptation step, which can be repeated several times, as in Algorithm 1. In addition, by backpropagating errors through the decoder, the encoder can learn to provide a data-conditioned latent encoding $\mathbf{z}$ that produces an appropriate initialization point $\theta_i$ for the classifier model. Stochasticity in the entire process can be efficiently handled using the reparametrization trick.

### A.2.3 Meta-Training Strategy (The "Outer Loop")

For each task instance $\mathcal{T}_i$, the initialization and adaptation procedure produce a new classifier $f_{\theta_i'}$ tailored to the training set $\mathcal{D}^{tr}$ of the instance, which we can then evaluate on the validation set of that instance $\mathcal{D}^{val}$. During meta-training we differentiate through the "inner loop" and update the encoder and decoder network parameters: $\phi_e$, $\phi_r$, and $\phi_d$. This process is analogous to the MAML "outer loop", but instead of learning a fixed set of initialization parameters we learn a parameter

[2] As before, we omit the task subscript $i$ from $\mathbf{w}_n$ for clarity.

generator implicitly represented by the encoder and decoder networks. Meta-training is performed by minimizing the following objective:

$$\min_{\phi_e,\phi_r,\phi_d} \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \left[ \mathcal{L}^{val}_{\mathcal{T}_i}\left(f_{\theta'_i}\right) + \beta D_{KL}\left(q(\mathbf{z}_n|\mathcal{D}^{tr}_n)||p(\mathbf{z}_n)\right) + \gamma||\text{stopgrad}(\mathbf{z}'_n) - \mathbf{z}_n||_2^2 \right] + R \tag{7}$$

where $p(\mathbf{z}_n) = \mathcal{N}(0, \mathcal{I})$. Similar to the loss defined in [13] we use a weighted KL-divergence term to encourage the generative model to learn a disentangled embedding, which should also simplify the LEO "inner loop" by removing correlations between latent space gradient dimensions. We found it beneficial to encourage the encoder and relation networks to produce latent codes which are close to final adapted codes.

$L_2$ regularization was used with all weights of the model, as well as a soft, layer-wise orthogonality constraint on decoder network weights, which encourages the dimensions of the latent code as well as the decoder network to be maximally expressive. In the case of linear encoder, relation, and decoder networks, and assuming that $\mathcal{C}_d$ is the correlation matrix between rows of $\phi_d$, then the regularization term takes the following form:

$$R = \lambda_1 \left( ||\phi_e||_2^2 + ||\phi_r||_2^2 + ||\phi_d||_2^2 \right) + \lambda_2 ||\mathcal{C}_d - \mathcal{I}||_1 \tag{8}$$

# B Experimental setup

## B.1 Few-Shot Classification

We used the standard 5-way 1-shot and 5-shot classification setups, where each task instance involves classifying images from 5 different categories sampled randomly from one of the meta-sets, and $\mathcal{D}^{tr}_{\mathcal{T}_i}$ contains 1 or 5 training examples respectively. $\mathcal{D}^{val}_{\mathcal{T}_i}$ contains 15 samples during meta-training, as decribed in [6], and all the remaining examples during validation and testing, following [31].

As described in Section 4, we trained dataset specific feature embeddings before meta-learning, in a similar fashion to [31] and [3]. A Wide Residual Network WRN-28-10 [44] with 3 steps of dimensionality reduction was used to classify images of $80 \times 80$ pixels from only the meta-training set into the corresponding training classes (64 in case of *mini*ImageNet and 351 for *tiered*ImageNet). Activations in layer 21, with average pooling over spatial dimensions, were precomputed and saved as feature embeddings with $n_x = \dim(\mathbf{x}) = 640$, which substantially simplified the meta-learning process. We $L_2$-normalized features across the last dimension.

We did not employ any data augmentation or feature averaging during meta-learning, or any other data apart from the corresponding training and validation meta-sets. The only exception is the special case of "multi-view" [19] embedding results, where features were averaged over representations of 4 corner and central crops and their horizontal mirrored versions, which we provide for full comparison with [31]. Apart from the differences described here, the feature training pipeline closely followed that of [31].

We used the same network architecture of the parameter generator for all tasks. The encoder and decoder networks were linear with the bottleneck embedding space of size $n_s = 64$. The relation network was a 3-layer fully connected network with 128 units per layer and rectifier nonlinearities. For simplicity we did not use biases in any layer of the generator nor the relation network.

Within the LEO "inner loop" we perform 5 steps of adaptation in latent space, followed by 5 steps of fine-tuning in parameter space. We disable stochastic behavior in evaluation mode and use predicted means instead. Different per-parameter learning rates were meta-learned for both spaces in a similar fashion to Meta-SGD [24], and they were initialized to 1 and 0.001 respectively. We applied dropout independently on the feature embedding in every step, with the probability of not being dropped out $p_{keep}$ chosen (together with other hyperparameters) using random search based on the validation meta-set accuracy.

Parameters of the encoder, relation, and decoder networks as well as per-parameter learning rates in latent and parameter spaces were optimized jointly using Adam [17] to minimize the meta-learning

objective (Eq. 7) over problem instances from the training meta-set, iterating for up to $100000$ steps, with early stopping using validation accuracy. For reliable experimentation we clipped gradients to $0.1$ first by value and then by norm. We avoid $NaN$ gradients by zeroing out (only for that step) any component of the loss if its gradients have $NaNs$.

The "Meta-SGD (our features)" baseline used the same one-layer softmax classifier as base model.

### B.2 Hyper-parameters

To find the best values of hyperparameters, we performed a random grid search and we choose the set which lead to highest validation meta-set accuracy. The reported performance of our models is an average ($\pm$ a standard deviation) over $5$ independent runs (using differing random seeds) with the best hyperparameters kept fixed. The result of a single run is an average accuracy over 50000 task instances. After choosing hyperparameters (given in Table 2) we used both meta-training and meta-validation sets for training, in line with recent state-of-the-art approaches, e.g. [31].

The evaluation of each of the LEO baselines follow the same procedure; in particular, we perform a separate random search for each of them.

| **Hyperparameter** | *mini*ImageNet | | *tiered*ImageNet | |
|---|---|---|---|---|
| | 1-shot | 5-shot | 1-shot | 5-shot |
| $\eta$ (Algorithm 1) | 0.00043653954 | 0.00117573555 | 0.00040645397 | 0.00073469522 |
| $\gamma$ (Eq. (7)) | 1.33365371e−9 | 5.39245830e−6 | 1.24305386e−8 | 3.05077069e−6 |
| $\beta$ (Eq. (7)) | 0.124171967 | 0.0440372182 | 7.10800960e−6 | 0.00188644980 |
| $\lambda_1$ (Eq. (8)) | 0.000108982953 | 3.75922509e−6 | 3.10725285e−8 | 4.90658551e−8 |
| $\lambda_2$ (Eq. (8)) | 303.216647 | 0.00844225971 | 5180.09554 | 0.0081711619 |
| $p_{keep}$ | 0.711524088 | 0.755402644 | 0.644395979 | 0.628325359 |

Table 2: Values of hyperparameters chosen to maximize meta-validation accuracy during random search.