Meta-Learning via Classifier(-free) Guidance

Anonymous Author(s)
Affiliation
Address
email

Abstract

We aim to develop meta-learning techniques that achieve higher zero-shot performance than the state of the art on unseen tasks. To do so, we take inspiration from recent advances in generative modeling and language-conditioned image synthesis to propose meta-learning techniques that use natural language guidance for zero-shot task adaptation. We first train an unconditional generative hypernetwork model to produce neural network weights; then we train a second "guidance" model that, given a natural language task description, traverses the hypernetwork latent space to find high-performance task-adapted weights in a zero-shot manner. We explore two alternative approaches for latent space guidance: "HyperCLIP"-based classifier guidance and a conditional Hypernetwork Latent Diffusion Model ("HyperLDM"), which we show to benefit from the classifier-free guidance technique common in image generation. Finally, we demonstrate that our approaches outperform existing meta-learning methods with zero-shot learning experiments on our Meta-VQA dataset.

1 Introduction

State-of-the-art machine learning algorithms often lack the ability to quickly generalize in a sample efficient manner to new unseen tasks. In contrast, humans show remarkable capabilities in leveraging previous knowledge for learning a new task from just a few examples. Often, not even a single example is needed, as all relevant task information can be conveyed in the form of natural language instructions.

Inspired by recent advances in conditional image generation (Ramesh et al., 2022; Rombach et al., 2022), we reframe meta-learning as a multi-modal generative modeling problem such that, given a task, its adapted neural network weights and its natural language description are considered equivalent multi-modal task representations. We show that popular techniques for the image domain, such as CLIP-based guidance (Gal et al., 2021; Patashnik et al., 2021), denoising diffusion models (Ho et al., 2020), and classifier-free guidance (Dhariwal and Nichol, 2021; Ho and Salimans, 2021; Nichol et al., 2022) can be repurposed for the meta-learning setting to generate adapted neural network weights instead of images. We point to Appendix A.1 for other related work.

Specifically, we approach the generation of neural network weights in two separate phases. In the *unconditional pre-training* phase, we train a generative hypernetwork (Ha et al., 2016) to map from its latent space to the weight space of a base model (Figure 1.A). In the *guidance* phase, we learn language-conditioned models that can be used to traverse the hypernetwork latent space and find zero-shot adapted weights with high performance on our task (Figure 1.B and 1.C).

Submitted to the 6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans. Do not distribute.

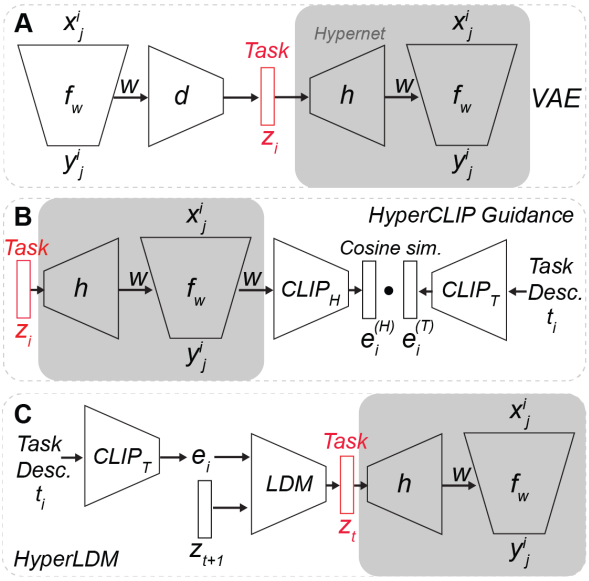

Figure 1: Schematic of the three main components of our proposed meta-learning approach. **A.** An unconditional variational autoencoder (VAE) models the latent space of adapted network weights W. Its generator hypernetwork h (highlighted in gray) can be re-used in the conditional setting with our guidance techniques. **B.** Our HyperCLIP encoder CLIP_H is contrastively trained to map network weights W to the space of CLIP embeddings e_i. Then, given a new task with descriptor t_i, we can use CLIP guidance to find a VAE latent vector z_i with embedding $e_i^{(H)}$ that has a high cosine similarity to a given task embedding $e_i^{(T)}$. **C.** Alternatively, our Hypernetwork Latent Diffusion Model (HyperLDM) learns, conditional on the task embedding e_i, to iteratively denoise a VAE latent vector $z_i^T, \ldots, z_i^0$ over T iterations.

2 Meta-Learning with Multi-Modal Task Embeddings

The setting we investigate is similar to the classic meta-learning framework, where we operate within a distribution of tasks $\mathcal{T}_i \sim p(\mathcal{T})$, each associated with a loss function $\mathcal{L}_{\mathcal{T}_i}$. Using a set of training tasks drawn from this distribution, our goal is to train a model $f(x, W_i) = y$ such that it generally performs well on new unseen tasks drawn from $p(\mathcal{T})$. In this work, we assume to have access to a natural language description t_i for each task $\mathcal{T}_i$, which can be encoded into a task embedding e_i using a pre-trained language model.

The fundamental building block of our unconditional generative model is the hypernetwork (Ha et al., 2016) $h(z, \theta) = W$, which can be trained either with classic MAML or with a VAE setup, paired to an encoder $d(W, \omega) = (\mu_z, \Sigma_z)$. For more details, we point to Appendix A.3.

3 HyperCLIP: Training a CLIP Encoder for the "Meta-Learning Modality"

Our first approach builds on top of Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021). In the original CLIP formulation, separate text and image encoders are trained such that, given a bi-modal sample (x_i, t_i) of an image and its corresponding language caption, their representations ($\text{CLIP}_I(x_i) = e_i^{(I)}$ and $\text{CLIP}_T(t_i) = e_i^{(T)}$) are aligned across modalities. In our work, we consider multi-modal representations of meta-learning *tasks* $\mathcal{T}_i$, which may be presented in the form of language as task descriptions t_i, but potentially also in the form of images, videos, and audio. We fine-tune a base machine learning model $f(x, W_i) = y$ for task $\mathcal{T}_i$ and consider the base model as part of an alternative *meta-learning modality* for task $\mathcal{T}_i$. The *meta-learning modality* can then be paired in contrastive learning with the other multi-modal descriptions of $\mathcal{T}_i$. We can thus define our new **HyperCLIP** encoder as a "reverse hypernetwork" $\mathit{CLIP}_H(W_i) = e_i^{(H)}$.

We then introduce **HyperCLIP guidance**, the first algorithm for classifier guidance in the meta-learning setting (Figure 1.B). Given a previously unseen validation task $\mathcal{T}_i$ and an unconditional generative hypernetwork model $h(z, \theta) = W$, we can use a trained HyperCLIP encoder $\text{CLIP}_H(W) = e^{(H)}$ to guide the exploration of the hypernetwork's latent space and find a set of base weights W_i with high zero-shot performance for $\mathcal{T}_i$. Specifically, as long as we are given a starting hypernetwork latent vector z^0 and a textual description t_i of the task, we can update z^0 with gradient

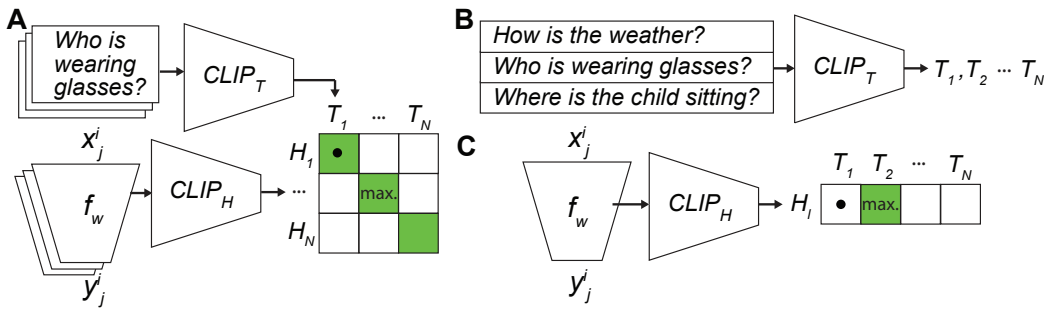

Figure 2: Our HyperCLIP encoder CLIP_H is contrastively trained to map neural network weights W to the latent space of a pre-trained language encoder CLIP_T, which we use to embed the natural language questions associated to the tasks (see **A**). To perform task inference given an already fine-tuned network, we encode all candidate task questions using the language CLIP encoder (see **B**), then encode the fine-tuned network weights with HyperCLIP (see **C**), and finally infer the correct task with a softmax operation over cosine similarities between HyperCLIP and language CLIP embeddings.

descent over the guidance loss

$$\mathcal{L}_{\text{guidance}}(z) = -\frac{\text{CLIP}_H\left(h(z,\theta)\right)^\top \text{CLIP}_T(t_i)}{\|\text{CLIP}_H\left(h(z,\theta)\right)\| \|\text{CLIP}_T(t_i)\|} + \lambda \|z - z^0\|, \tag{1}$$

and then run the optimized latent vectors $\hat{z}_i$ through the generative hypernetwork to find adapted zero-shot base network weights $h(\hat{z}_i, \theta) = \hat{W}_i$ that perform well for the task.

4 HyperLDM: Task-conditional Diffusion of Hypernetwork Latents

Recent advances in image generation involve the use of classifier guidance and classifier-free guidance during the sampling process of a Diffusion Model (Dhariwal and Nichol, 2021; Ho and Salimans, 2021; Kim et al., 2022; Crowson, 2022; Nichol et al., 2022; Rombach et al., 2022). To paint a more complete picture, we also investigate this setting in the meta-learning domain.

In our meta-learning setting, we aim to train a diffusion model which generates adapted zero-shot base network weights $\hat{W}_i$ that perform well for task $\mathcal{T}_i$. Thus, our diffusion model has to be conditional on a task embedding e_i. Moreover, in order to speed up training and leverage our previously trained generative hypernetwork $h(z, \psi)$, we define the diffusion process on latent vectors instead of doing so in weight space, emulating the Latent Diffusion technique from Rombach et al. (2022). We propose Hypernetwork Latent Diffusion Models (**HyperLDM**), which learn to sample from the conditional distribution of fine-tuned latent vectors $p(z^0|e_i)$ given a language CLIP embedding corresponding to the task. The HyperLDM neural network models the noise function $\epsilon_\psi(z^t, t, e_i)$, and is learned by optimizing the reweighted variational lower bound, which in this setting is

$$\mathcal{L}_{\text{LDM}}(\psi) = \mathbb{E}_{\mathcal{T}_i, h_{\text{enc}}(W_i), \epsilon \sim \mathcal{N}(0,1), t}\left[\|\epsilon - \epsilon_\psi(z^t, t, e_i)\|_2^2\right]. \tag{2}$$

The classifier guidance technique presented in Section 3 can be also adopted together with diffusion models. Even in the case of conditional diffusion models, the gradient of an auxiliary classifier (or CLIP encoder) can be added during sampling to induce an effect similar to GAN truncation (Brock et al., 2018), producing samples that are less diverse but of higher quality. The classifier-free guidance technique (Ho and Salimans, 2021; Nichol et al., 2022) allows us to leverage a conditional diffusion model to perform the same tempered sampling as above, without the auxiliary classifier. To do so, we train the conditional network $\epsilon_\psi(z^t, t, e_i)$ to also model the unconditional case $\epsilon_\psi(z^t, t)$. One way of doing this is with *conditioning dropout*, simply dropping the conditional input e_i for a certain percentage of training samples, inputting zeros instead. We can then sample at each diffusion iteration with

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = (1-\gamma)\,\epsilon_\psi(z^t, t, 0) + \gamma \epsilon_\psi(z^t, t, e_i). \tag{3}$$

For $\gamma = 0$, this recovers the unconditional diffusion model, while for $\gamma = 1$ it recovers the standard task-conditional model. For $\gamma > 1$, we instead obtain the classifier-free guidance effect, which we show results in the sampling of latent vectors $\hat{z}_i$ corresponding to higher-performing task-conditional network weights $h(\hat{z}_i, \psi) = \hat{W}_i$. We point to a more in-depth discussion on classifier-free guidance and its connection to score matching in Appendix A.6.

5 Experimental Setup and Results

In this section, we demonstrate the soundness of our two approaches with zero-shot and few-shot image classification experiments against a series of traditional meta-learning baseline techniques. Throughout our experiments, we fix the choice of base network model (see Appendix A.7), only varying the meta-learning techniques employed to obtain adapted base model weights.

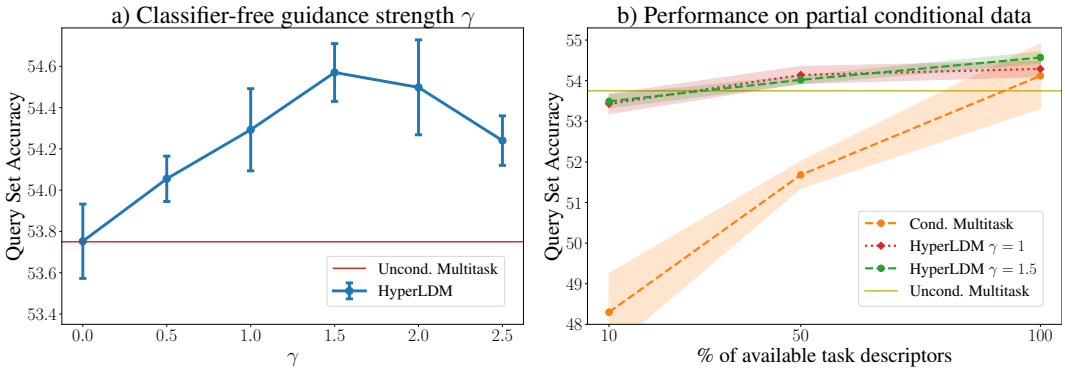

Figure 3: **a)** Performance of HyperLDM over different classifier-free guidance parameters γ. **b)** Performance of HyperLDM against baselines in the setting where only a fraction of natural language task labels are given.

In Table 1 from Appendix A.9 we show how well our methods fare on the **Meta-VQA** dataset (see Appendix A.8 for discussion on the dataset). We test **HyperCLIP Guidance** and **HyperLDM** when trained on top of either a hypernetwork or a VAE generator (see Appendix A.7 and A.11 for more details). **HyperCLIP Guidance** allows for faster sampling than **HyperLDM** but is generally less performant (53.98% acc.), still, it improves upon all other zero-shot baselines except for conditional multi-task learning (**Cond. Multitask**, 54.12%). The best performing model for the zero-shot setting is **HVAE + HyperLDM**, with an accuracy of 54.57%, and specifically for classifier-free guidance with $\gamma = 1.5$. As illustrated in Figure 3.a, to further show the effectiveness of the classifier-free guidance technique, we sweep over several candidate γ parameters to find that the optimum occurs for $\gamma > 1$. In another experiment, as shown in Figure 3.b, we train our model while only keeping 50% or 10% of task descriptors, and show that while this strongly impacts traditional **Cond. Multitask** learning, it does not affect **HyperLDM** as strongly due to its two-phased training regime based on an uncondtional VAE. In fact, even when dropping half of task descriptors, HyperLDM still performs better than the **Uncond. Multitask** baseline.

6 Conclusion

In this work we introduced a framework that re-interprets meta-learning as a multi-modal generative modeling problem. Our HyperCLIP guidance and HyperLDM methods leverage this insight to generate task-adapted neural network weights in a zero-shot manner given natural language instructions, and constitute the first application of the CLIP guidance and classifier-free guidance techniques from image generation to the meta-learning domain. Our experiments show that our methods successfully make use of external task descriptors to produce high-performance adapted networks in the zero-shot setting.

References

Bertinetto, L., Henriques, J. F., Torr, P., and Vedaldi, A. (2018). Meta-learning with differentiable closed-form solvers.

Brock, A., Donahue, J., and Simonyan, K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis.

Crowson, K. (2022). v-diffusion. original-date: 2021-12-16T17:04:35Z.

Dhariwal, P. and Nichol, A. (2021). Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pages 8780–8794. Curran Associates, Inc.

Finn, C., Abbeel, P., and Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135. PMLR. ISSN: 2640-3498.

Flennerhag, S., Rusu, A. A., Pascanu, R., Visin, F., Yin, H., and Hadsell, R. (2020). Meta-Learning with Warped Gradient Descent. arXiv:1909.00025 [cs, stat].

Gal, R., Patashnik, O., Maron, H., Chechik, G., and Cohen-Or, D. (2021). StyleGAN-NADA: CLIP-Guided Domain Adaptation of Image Generators. Technical Report arXiv:2108.00946, arXiv. arXiv:2108.00946 [cs] type: article.

Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., and Qiao, Y. (2021). CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *arXiv:2110.04544 [cs]*. arXiv: 2110.04544.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Goyal, Y., Khot, T., Summers-Stay, D., Batra, D., and Parikh, D. (2017). Making the v in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. pages 6904–6913.

Ha, D., Dai, A., and Le, Q. V. (2016). HyperNetworks. Technical Report arXiv:1609.09106, arXiv. arXiv:1609.09106 [cs] type: article.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. arXiv:1502.01852 [cs].

Henning, C., Cervera, M. R., D'Angelo, F., von Oswald, J., Traber, R., Ehret, B., Kobayashi, S., Sacramento, J., and Grewe, B. F. (2021). Posterior Meta-Replay for Continual Learning. *arXiv:2103.01133 [cs]*. arXiv: 2103.01133.

Henning, C., von Oswald, J., Sacramento, J., Surace, S. C., Pfister, J.-P., and Grewe, B. F. (2018). Approximating the Predictive Distribution via Adversarially-Trained Hypernetworks. In *Bayesian Deep Learning Workshop, NeurIPS 2018*, Montréal, Canada. Yarin.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851. Curran Associates, Inc.

Ho, J. and Salimans, T. (2021). Classifier-Free Diffusion Guidance.

Hu, J., Shen, L., and Sun, G. (2018). Squeeze-and-Excitation Networks. pages 7132–7141.

Jacot, A., Gabriel, F., and Hongler, C. (2020). Neural Tangent Kernel: Convergence and Generalization in Neural Networks. arXiv:1806.07572 [cs, math, stat].

Janner, M., Du, Y., Tenenbaum, J. B., and Levine, S. (2022). Planning with Diffusion for Flexible Behavior Synthesis. Technical Report arXiv:2205.09991, arXiv. arXiv:2205.09991 [cs] type: article.

Jayakumar, S. M., Czarnecki, W. M., Menick, J., Schwarz, J., Rae, J., Osindero, S., Teh, Y. W., Harley, T., and Pascanu, R. (2019). Multiplicative Interactions and Where to Find Them.

Kim, H., Kim, S., and Yoon, S. (2022). Guided-TTS: A Diffusion Model for Text-to-Speech via Classifier Guidance. In *Proceedings of the 39th International Conference on Machine Learning*, pages 11119–11133. PMLR. ISSN: 2640-3498.

Kingma, D. P. and Ba, J. (2017). Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs].

Kingma, D. P. and Welling, M. (2014). Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*. arXiv: 1312.6114.

Krueger, D., Huang, C.-W., Islam, R., Turner, R., Lacoste, A., and Courville, A. (2018). Bayesian Hypernetworks. arXiv:1710.04759 [cs, stat].

Lee, Y. and Choi, S. (2018). Gradient-Based Meta-Learning with Learned Layerwise Metric and Subspace. arXiv:1801.05558 [cs, stat].

Nichol, A., Achiam, J., and Schulman, J. (2018). On First-Order Meta-Learning Algorithms. arXiv:1803.02999 [cs].

Nichol, A., Dhariwal, P., Ramesh, A., Shyam, P., Mishkin, P., McGrew, B., Sutskever, I., and Chen, M. (2022). GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. Technical Report arXiv:2112.10741, arXiv. arXiv:2112.10741 [cs] type: article.

Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., and Lischinski, D. (2021). StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. pages 2085–2094.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020 [cs]*. arXiv: 2103.00020.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. (2022). Hierarchical Text-Conditional Image Generation with CLIP Latents. Technical Report arXiv:2204.06125, arXiv. arXiv:2204.06125 [cs] type: article.

Ratzlaff, N. and Fuxin, L. (2020). HyperGAN: A Generative Model for Diverse, Performant Neural Networks. Technical Report arXiv:1901.11058, arXiv. arXiv:1901.11058 [cs, stat] type: article.

Ravi, S. and Larochelle, H. (2016). Optimization as a Model for Few-Shot Learning.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-Resolution Image Synthesis With Latent Diffusion Models. pages 10684–10695.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pages 234–241, Cham. Springer International Publishing.

Rusu, A. A., Rao, D., Sygnowski, J., Vinyals, O., Pascanu, R., Osindero, S., and Hadsell, R. (2018). Meta-Learning with Latent Embedding Optimization.

Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 2256–2265. PMLR. ISSN: 1938-7228.

Song, Y. and Ermon, S. (2019). Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2020). Score-Based Generative Modeling through Stochastic Differential Equations.

von Oswald, J., Henning, C., Grewe, B. F., and Sacramento, J. (2020). Continual learning with hypernetworks. In *International Conference on Learning Representations (ICLR 2020)*. arXiv:1906.00695 [cs, stat].

von Oswald, J., Kobayashi, S., Sacramento, J., Meulemans, A., Henning, C., and Grewe, B. F. (2021a). Neural networks with late-phase weights. *arXiv:2007.12927 [cs, stat]*. arXiv: 2007.12927.

von Oswald, J., Zhao, D., Kobayashi, S., Schug, S., Caccia, M., Zucchet, N., and Sacramento, J. (2021b). Learning where to learn: Gradient sparsity in meta and continual learning. *arXiv:2110.14402 [cs]*. arXiv: 2110.14402.

Wang, K.-C., Vicol, P., Lucas, J., Gu, L., Grosse, R., and Zemel, R. (2018). Adversarial Distillation of Bayesian Neural Network Posteriors. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5190–5199. PMLR. ISSN: 2640-3498.

Zhao, D., Kobayashi, S., Sacramento, J., and von Oswald, J. (2020). Meta-Learning via Hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*, Virtual Conference. IEEE.

Zhmoginov, A., Sandler, M., and Vladymyrov, M. (2022). HyperTransformer: Model Generation for Supervised and Semi-Supervised Few-Shot Learning. arXiv:2201.04182 [cs].

Zintgraf, L. M., Shiarlis, K., Kurin, V., Hofmann, K., and Whiteson, S. (2019). Fast Context Adaptation via Meta-Learning. arXiv:1810.03642 [cs, stat].

A Appendix

A.1 Related Work

Hypernetworks By introducing multiplicative interactions within neural networks (Jayakumar et al., 2019), hypernetworks (Ha et al., 2016) have been shown to allow the modeling of diverse target network weights in, *e.g.*, continual learning, even in the compressive regime (von Oswald et al., 2021a, 2020) without loss of performance. For a given supervised problem, hypernetworks have been used to model the complex Bayesian posterior of the weights in conjunction with variational inference (Henning et al., 2018; Krueger et al., 2018). Similar approaches have been used for continual learning, whereby task-specific weight posteriors are generated by a task-specific hypernetwork, itself generated by hyper-hypernetwork conditioned on a learned task embedding (Henning et al., 2021). Another vein of work consists in using hypernetworks to distill ensembles of diverse networks (Wang et al., 2018; Ratzlaff and Fuxin, 2020; von Oswald et al., 2021a).

Meta learning In the context of meta-learning, hypernetworks have been successfully used in combination with popular gradient-based meta-learning methods (Finn et al., 2017), especially benefiting in few-shot classification and regression problems (Zintgraf et al., 2019; Zhao et al., 2020; Flennerhag et al., 2020). More generally, related works have shown the usefulness of learning a low dimensional manifold in which to perform task-specific gradient-based adaptation at meta test time (Rusu et al., 2018; von Oswald et al., 2021b; Lee and Choi, 2018), instead of directly adapting in weight space. Recent works bypasses the formal bi-level formulation of meta-learning by, *e.g.*, using transformers to directly output the weights of the target network using both the few-shot labeled data and transductive data as input (Zhmoginov et al., 2022).

Generative Modeling and Classifier(-free) guidance A plethora of techniques have been proposed for the generation of samples from high-dimensional domains such as images, such as Generative Adversarial Networks (Goodfellow et al., 2014; Brock et al., 2018, GANs) and Variational Autoencoders (Kingma and Welling, 2014, VAEs). Denoising Diffusion Probabilistic Models (Sohl-Dickstein et al., 2015; Ho et al., 2020, DDPM) overcome common issues in generative modeling using a simple likelihood-based reconstruction loss for iterative denoising, and have been shown to achieve state-of-the-art results in high resolution image generation (Dhariwal and Nichol, 2021; Rombach et al., 2022). Several techniques have been proposed for effective conditional sampling in generative and diffusion models, such as classifier/CLIP guidance (Dhariwal and Nichol, 2021; Gal et al., 2021; Patashnik et al., 2021) and classifier-free guidance (Ho and Salimans, 2021; Crowson, 2022; Nichol et al., 2022). Diffusion models with classifier-free guidance have also been successfully applied in non-visual domains, such as audio generation (Kim et al., 2022) and robotic planning (Janner et al., 2022).

A.2 Model-Agnostic Meta-Learning

We present here a slightly altered formulation of MAML (Finn et al., 2017) introduced in (Zintgraf et al., 2019), whereby the parameters of the model g are partitioned into two parts: context parameters ϕ that are adapted on individual tasks, and shared parameters θ that are meta-trained and shared across tasks. MAML and its variants focus on the few shot setting, which aims to learn an initialization for these parameters such that the model $g(\cdot, \theta, \phi)$ generalizes well on new tasks after fine-tuning ϕ on a few data points from that task. To train such a model, the data from each task $\mathcal{T}_i$ is split during training into a support set D_i^{s} and a query set D_i^{q}. The MAML objective aims to optimize the validation score evaluated on the query set when fine-tuning ϕ on the support set, *e.g.*, consider the following optimization problem:

$$\min_{\theta,\phi} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\frac{1}{|D_i^{\mathrm{q}}|} \sum_{(x,y)\in D_i^{\mathrm{q}}} \mathcal{L}_{\mathcal{T}_i} \left(g(x, \theta, \mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, \theta, \phi)), y \right) \right], \tag{4}$$

where $\mathcal{A}_{\mathcal{T}_i}$ is some differentiable algorithm, typically implementing a variant of few-step gradient descent on the loss computed on the support set, *e.g.*, in the case of one-step gradient descent:

$$\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, \theta, \phi) = \phi - \eta \frac{1}{|D_i^{\mathrm{s}}|} \sum_{(x',y')\in D_i^{\mathrm{s}}} \nabla_\phi \mathcal{L}_{\mathcal{T}_i}(g(x', \theta, \phi), y') \tag{5}$$

with some learning rate η. The objective from Eq. 4 is itself solved with gradient descent, by iteratively optimizing the parameters ϕ in the inner loop on the support set of a sampled task, and updating θ and the initialization of ϕ with their gradient with respect to the entire inner loop training process, averaged over batches of tasks. Note that the original formulation of MAML considers $\theta = \emptyset$.

If we consider hypernetworks, We can rewrite the MAML objective with respect to the hypernetwork weight θ as

$$\min_{\theta} \mathbb{E}_{\mathcal{T}_i \sim p(\mathcal{T})} \left[\frac{1}{|D_i^{\mathrm{q}}|} \sum_{(x,y)\in D_i^{\mathrm{q}}} \mathcal{L}_{\mathcal{T}_i} \left(f(x, h(\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, e_i, \theta), \theta))), y \right) \right], \tag{6}$$

when $\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, e_i, \theta) = e_i$, we recover the classic multi-task objective of a hypernetwork optimizing for zero-shot performance. When $\mathcal{A}_{\mathcal{T}_i}$ is instead the gradient descent algorithm on e_i, the objective aligns with the few-shot performance of h when adapting the embedding initialized at e_i.

A.3 Hypernetworks as Generative Models of Network Weights

We hereby define our two alternative choices for hypernetworks as unconditional generative models of base neural network weights: **1)** We define a Hypernetwork VAE as in Figure 1.A, which, given

samples of fine-tuned base network weights W_i, learns a low-dimensional normally distributed latent representation z. The encoder $d(W, \omega) = (\mu_z, \Sigma_z)$ maps base network weights to means and variances used to sample a latent vector z, while the decoder (or generator) is a classic hypernetwork $h(z, \theta) = W$ which reconstructs the network weights from the latent vector. **2)** Using MAML, we learn both an embedding z and hypernetwork weights θ such that, when fine-tuning only the embedding z on each task $\mathcal{T}_i$, we obtain high-performing base networks with weights $W_i = h(z_i, \theta)$. Concretely, we optimize θ and the initialization of z following the objective in Eq. 4 where z takes the role of the task-specific parameter ϕ.

A.4 The HyperCLIP Training Algorithm

Here we report the detailed learning procedure for training the HyperCLIP encoder.

Algorithm 1 HyperCLIP Training

sample a batch of tasks $\mathcal{T}_{i=1,\ldots,N}$ with loss functions $\mathcal{L}_{\mathcal{T}_i}$, training data D_i^{train} and text t_i
define two N-sized arrays of d-dimensional embeddings $T \in \mathbb{R}^{N \times d}$ and $H \in \mathbb{R}^{N \times d}$
for $i = 1, \ldots, N$ **do**
  $T[i] = \text{CLIP}_T(t_i)$ / $\|\text{CLIP}_T(t_i)\|$
  Fine-tune W_i with objective: $\min_W \sum_{(x',y') \in D_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f(x', W), y')$
  $H[i] = \text{CLIP}_H(W_i)$ / $\|\text{CLIP}_H(W_i)\|$
end for
loss $= \left(\mathcal{L}_{\text{cross-entropy}}(TH^\top) + \mathcal{L}_{\text{cross-entropy}}(HT^\top)\right)$ / 2
Update weights of $\text{CLIP}_H(.)$ using ∇loss

A.5 Diffusion Models

Denoising Diffusion Probabilistic Models (Sohl-Dickstein et al., 2015; Ho et al., 2020, DDPM) are a powerful class of generative models designed to learn a data distribution $p(x)$. They do so by learning the inverse of a *forward diffusion process* in which samples x^0 of our data distribution are slowly corrupted with additive Gaussian noise over T steps with a variance schedule $\beta_1, \ldots, \beta_T$, resulting in the Markov Chain

$$q(x^t|x^{t-1}) = \mathcal{N}(x^t; \sqrt{1-\beta_t}x^{t-1}, \beta_t \mathbf{I}) \qquad q(x^{1:T}|x^0) = \prod_{t=1}^{T} q(x^t|x^{t-1}). \tag{7}$$

A property of such a process is that we can directly sample each intermediate step from x^0 as $x^t = \sqrt{\bar{\alpha}_t}x^0 + (\sqrt{1-\bar{\alpha}_t})\epsilon$ given $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_t$. Then, to learn the reverse process $p_\psi(x^{t-1}|x^t)$, we parametrize the timestep-dependent noise function $\epsilon_\psi(x^t, t)$ with a neural network and learn it by optimizing a simplified version of the variational lower bound on $p(x)$

$$\mathcal{L}_{\text{DM}}(\psi) = \mathbb{E}_{x, \epsilon \sim \mathcal{N}(0,1), t}\left[\|\epsilon - \epsilon_\psi(x^t, t)\|_2^2\right]. \tag{8}$$

Sampling from the reverse process can then be done with

$$x^{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(x^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(x^t, t)\right) + \sigma_t \xi, \tag{9}$$

with $\xi \sim \mathcal{N}(0, \mathbf{I})$ and σ_t chosen between β_t and $\beta_t/\sqrt{1-\bar{\alpha}_t}$. Sampling from the learned diffusion model can be seen as analogue to Langevin Dynamics, a connection explicitly made in works exploring the diffusion technique from the "score matching" perspective (Song and Ermon, 2019; Song et al., 2020).

A.6 Classifier-Free Guidance

We hereby provide a rationale for the use of classifier guidance and classifier-free guidance during diffusion model sampling. As per the "score matching" interpretation of diffusion models, we

assume that our trained noise network approximates the score function of the true conditional latent distribution $p(z|e_i)$ as $\epsilon_\psi(z^t, t, e_i) \approx -\sigma_t \nabla_{z^t} \log p(z^t|e_i)$. For classifier guidance, we can perturb our diffusion sampling by adding the gradient of the log likelihood of our CLIP encoder $p_\psi(e_i|z_t)$ to the diffusion score as follows

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = \epsilon_\psi(z^t, t, e_i) - \eta\sigma_t \nabla_{z^t} \log p_\psi(e_i|z^t) \approx -\sigma_t \nabla_{z^t} \left[\log p(z^t|e_i) + \eta \log p_\psi(e_i|z^t)\right]. \tag{10}$$

We can rewrite this as classifier guidance on the unconditional score $\nabla_{z^t} \log p(z^t)$ with

$$-\sigma_t \nabla_{z^t} \left[\log p(z^t) + \gamma \log p(e_i|z^t)\right] \quad \text{with} \quad \gamma = 1 + \eta \tag{11}$$

using Bayes' rule, as $\log p(z^t|e_i) = \log p(e_i|z^t) + \log p(z^t) - \log p(e_i)$, and thus $\nabla_{z^t} \log p(z^t|e_i) = \nabla_{z^t} \log p(e_i|z^t) + \nabla_{z^t} \log p(z^t)$.

For classifier-free guidance, we aim to perform the above sampling without access to a classifier, as long we possess a conditional diffusion model $\epsilon_\psi(z^t, t, e_i)$ that doubles as an unconditional model $\epsilon_\psi(z^t, t, 0)$, as illustrated in Section 4.

Using Bayes' rule again, we can see that $\nabla_{z^t} \log p(e_i|z^t) = \nabla_{z^t} \log p(z^t|e_i) - \nabla_{z^t} \log p(z^t)$. If we substitute this into Eq. 11 we obtain

$$-\sigma_t \nabla_{z^t} \left[\log p(z^t) + \gamma \left(\log p(z^t|e_i) - \log p(z^t)\right)\right], \tag{12}$$

$$-\sigma_t \nabla_{z^t} \left[(1-\gamma) \log p(z^t) + \gamma \log p(z^t|e_i)\right], \tag{13}$$

which can be implemented with our conditional network as

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = (1-\gamma)\, \epsilon_\psi(z^t, t, 0) + \gamma \epsilon_\psi(z^t, t, e_i). \tag{14}$$

A.7 Network architectures

Base Network (f) Our choice for a base model is a CLIP-Adapter (Gao et al., 2021), which consists of a frozen CLIP image encoder with added learned fully-connected layers refining the output embedding. Specifically, we use the *ViT-L/14@336px* CLIP encoder type with embedding size of 768. The advantages of this model choice lie in its combination of high base performance (due to pre-trained knowledge contained in the CLIP component) and relatively small parameter count, enabling agile medium-small scale experiments. This base CLIP-Adapter network purely works as a base model and is not to be confused with HyperCLIP, which is employed at the meta-level. In Section 5, when benchmarking the base model in the zero-shot setting, we drop the Adapter and use pre-trained zero-shot CLIP (Radford et al., 2021).

Hypernetwork (h) For the hypernetworks used in our baseline as well as as the generative model, we use a MLP with one hidden layer of 256 units, which are followed by a rectified linear activation. For the unconditioned hypernetwork, the embedding to the hypernetwork is a vector of dimension 64, while for the conditioned counterpart, the task embedding is used. In order to ensure that the generated weights are properly normalized at initialization, we use the Kaiming initialization (He et al., 2015) for the hypernetwork weights, initialize the embedding as a sample from a multivariate standard gaussian distribution (for unconditioned models), and use the NTK parametrization (Jacot et al., 2020) for the target network.

Variational Autoencoder For the variational autoencoder used as our unconditioned generative model, we use an MLP of 2 hidden layers of size 512 and 256, each followed by the rectified linear non-linearity. We chose 32 as the latent code dimension. We use the same architecture for the decoder, except for the dimensionality of the 2 hidden layers being swapped. We use the Kaiming initialization (He et al., 2015) to initialize the weight of both the encoder and decoder.

HyperCLIP We parametrize our HyperCLIP model as a fully-connected MLP with a single hidden layer of dimension 256, taking as input the flattened weight of the base network and outputting the corresponding CLIP encoding. We chose the tangent hyperbolic function as the activation function in the hidden layer.

HyperLDM While the original LDM makes use of a time-conditional UNet (Ronneberger et al., 2015) to parametrize the noise network, we are unfortunately unable to make use of spatial information and convolutions due to the non-spatial nature of our latent space. We parametrize our HyperLDM as a fully-connected network with residual connections and squeeze-and-excitation layers (Hu et al., 2018). The time index t is embedded into a vector with a 150-dimensional sinusoidal positional embedding, and is concatenated together with the task-conditional embedding e_i at the input layer and at intermediate activations. Hidden layer dimensions are 8192, 16384, 8192.

A.8 The Meta-VQA Dataset

To evaluate the performance of our methods, we utilize a dataset that reflects the setting of meta-learning with multi-modal task descriptors. Existing meta-learning benchmarks such as MiniImagenet (Ravi and Larochelle, 2016) or CIFAR-FS (Bertinetto et al., 2018) are unsuitable, as they are built for the traditional few-shot learning setting, in which the task $\mathcal{T}_i$ is not associated with task descriptors but is meant to be inferred through exposure to the support set D_i^{s}. We thus introduce our own **Meta-VQA** dataset, a modification of the VQA v2.0 dataset (Goyal et al., 2017) for Visual-Question-Answering. The dataset is composed of training and test tasks $\mathcal{T}_i$, each associated with a natural language question t_i and a mini image classification dataset $(x_j^i, y_j^i) \in D_i$.

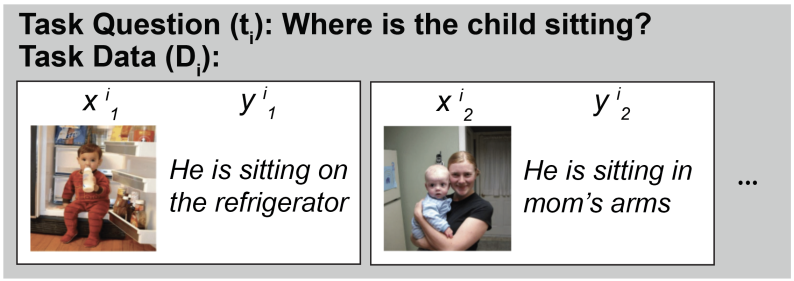

Figure 4: Example classification task from Meta-VQA, adapted from VQA v2 (Goyal et al., 2017). A single question t_i is associated to multiple image-answer tuples (x_j^i, y_j^i).

The original VQA problem is about choosing a suitable natural language answer a_k when prompted with both a natural language question q_k and an image I_k. Our observation is that the VQA problem can then easily be reformulated as a meta-learning image classification problem with natural language task descriptions: given question-image-answer triples $(q_k, I_k, a_k) \in D$, we can group the data by unique questions q_i (which will serve as task descriptor t_i), each of which can then be associated with supervised image classification tuples $(I_j^i, a_j^i) \in D_i$. To make sure the designed tasks are meaningful, we filter out question-answer pairs with questions in choosing form, *e.g.,* "A or B?" or yes/no answers. From the remaining questions we keep the ones which appear at least 20 times throughout the dataset, such that each task contains enough samples. In the end our Meta-VQA dataset is composed of 1234 unique tasks (questions), split into 870 training tasks and 373 test tasks, for a total of 104112 image-answer pairs.

A.9 Experimental Results

We point to Table 1 for a recount of zero-shot and few-shot classification accuracy obtained on Meta-VQA test tasks with both the baselines and our techniques.

For each training task $\mathcal{T}_i$, the algorithms are given access to the full image/answer support and query sets D_i^{s}, D_i^{q}, together with the question (task descriptor) t_i. At test time, in the zero-shot setting, only the task descriptors t_i for each test task $\mathcal{T}_i$ are given, and the algorithms are tasked with predicting the correct labels of images in the query set D_i^{q}. In the few-shot setting, beyond the task descriptors t_i, the support sets D_i^{s} of test tasks with accompanying answers (labels) are also given.

Table 1: Zero-Shot and Few-Shot learning accuracy averaged over Meta-VQA test tasks. (* ours)

Method	Zero-Shot	Few-Shot
Base CLIP(-Adapter)	44.99	54.93 ($\pm$ 0.11)
Uncond. Multitask	53.75 ($\pm$ 0.36)	55.53 ($\pm$ 0.40)
Uncond. MNet-MAML	53.04 ($\pm$ 0.69)	**60.24** ($\pm$ 0.84)
Uncond. MNet-FOMAML	53.04 ($\pm$ 0.42)	60.03 ($\pm$ 0.48)
Uncond. HNet-MAML	53.37 ($\pm$ 0.29)	58.70 ($\pm$ 0.10)
Cond. Multitask	54.12 ($\pm$ 0.80)	59.46 ($\pm$ 0.31)
Cond. HNet-MAML	53.02 ($\pm$ 0.20)	59.48 ($\pm$ 0.03)
* HNet + HyperCLIP Guidance	53.98 ($\pm$ 0.54)	58.82 ($\pm$ 0.27)
* HVAE + HyperCLIP Guidance	53.62 ($\pm$ 0.37)	58.75 ($\pm$ 0.29)
* HNet + HyperLDM $\gamma = 1$	54.06 ($\pm$ 0.21)	58.70 ($\pm$ 0.11)
* HNet + HyperLDM $\gamma = 1.5$	54.30 ($\pm$ 0.27)	58.60 ($\pm$ 0.09)
* HVAE + HyperLDM $\gamma = 1$	54.29 ($\pm$ 0.19)	58.97 ($\pm$ 0.09)
* HVAE + HyperLDM $\gamma = 1.5$	**54.57** ($\pm$ 0.14)	58.89 ($\pm$ 0.07)

A.10 Baseline methods

Classic zero-shot **CLIP** and its **CLIP-Adapter** extension for few-shot learning (our base network) provide a floor for performance on Meta-VQA. We then benchmark several unconditional and conditional methods, with only conditional methods having access to language task descriptors. We apply MAML and its first-order variant FOMAML (Nichol et al., 2018) directly to the base network (**MNet-MAML**, **MNet-FOMAML**), and to both an unconditional hypernetwork (**Uncond. HNet-MAML**, as in Appendix A.3) and a conditional one (**Cond. HNet-MAML**). We also benchmark against standard multitask learning (**Uncond. Multitask**, **Cond. Multitask**). It is apparent that the multitask approach, at least in this setting, leads to better zero-shot models than MAML, which instead optimizes for few-shot performance. We refer to Appendix A.7 for more details on each model.

We detail an overview of the baseline methods we benchmark in table 2.

Training: The number of epochs each model is trained on, the learning rate `lr` of the optimization, as well as the learning rate and number of steps of the adaptation algorithm used for each method can be found in table 3. For all methods using an adaptation $\mathcal{A}_{\mathcal{T}_i}$, the dataset from the task is randomly split into a support set and a query set during training, every time a task is sampled. The support set is then used to perform the adaptation (see Section A.2), while the query set is used to compute the loss on which the meta-parameters are updated. When no adaptation is used, all the data is used for this update. *Unconditional* methods do not have access to the task embedding e_i, while *conditioned* methods do. When the percentage of available task descriptor is reduced, conditioned methods are trained only on the tasks which descriptor is available.

Evaluation: Evaluation is performed on a fixed query set on the predefined query set of the held-out test tasks of the Meta-VQA dataset. Zero-shot performance is evaluated before applying the adaptation procedure $\mathcal{A}_{\mathcal{T}_i}$. For the few shot performance, all adaptation is performed on the support set of the test tasks. For MAML baselines, we keep the same adaptaiton-time learning rate as during trainng, while we always adapt for 50 steps. For each multitask baselines, we use the same adaptation scheme (steps, learning rate, adapting parameters) as their MAML counterpart.

Table 2: Overview of the different methods trained on MetaVQA. The **parameters** are optimized via the task loss evaluated on the output of the **function**, averaged over minibatches of tasks. The adaptation $\mathcal{A}_{\mathcal{T}_i}$ implements a few step gradient descent algorithm applied on the argument parameter, w.r.t the task loss evaluated on the support set.

Method	Function	Parameters
Unconditional MNet Multitask	$f(\cdot, W)$	W
Unconditional MNet (FO)MAML	$f(\cdot, \mathcal{A}_{\mathcal{T}_i}(W^0))$	W^0
Unconditional Hypernetwork MAML	$f(\cdot, h(\mathcal{A}_{\mathcal{T}_i}(z^0), \theta))$	θ, z^0
Conditional Multitask	$f(\cdot, h(e_i, \theta))$	θ
Conditional Hypernetwork MAML	$f(\cdot, h(\mathcal{A}_{\mathcal{T}_i}(e_i), \theta))$	θ

Table 3: Hyperparameters used for the baseline methods. All methods are trained with the Adam (Kingma and Ba, 2017) optimizer, with meta-batch size of 32 tasks. We use gradient norm clipping for all optimization, with the maximum norm set to 10. Note that when the adaptation algorithm has a range of possible steps, the number of step is sampled uniformly from the range for every adaptation.

Method	`epochs`	`lr`	$\mathcal{A}$-`lr`	$\mathcal{A}$-`steps`
Unconditional MNet Multitask	300	0.0001	-	-
Unconditional MNet (FO)MAML	500	0.00003	0.01	0-10
Unconditional Hypernetwork MAML	100	0.00003	0.1	0-10
Conditional Multitask	60	0.0001	-	-
Conditional Hypernetwork MAML	200	0.00001	0.1	0-10

A.11 Guidance Models Hyperparameters

A.11.1 Generative hypernetwork

To enable our guidance methods, we need to first train a generative hypernetwork h as in Section **??**, either in the form of an Unconditional Hypernetwork, or of a Hypernetwork VAE:

- For **HNet + HyperCLIP guidance** and **HNet + HyperLDM**, we meta-learnt an unconditioned hypernetwork with the exact same hyperparameters as the baseline **Uncond. HNet-MAML**, and used it as the generative hypernetwork.
- For **HVAE + HyperCLIP guidance** and **HVAE + HyperLDM**, we trained an unconditioned VAE on samples of fine tuned network weights W_i using the architecture specified in A.7. In order to be able to quickly sample new adapted weights, and to reduce the complexity of the manifold from which such weights are sampled, we use adaptations from our unconditional MAML baselines as W_i. Specifically, we used adaptations from **Uncond. HNet-MAML**, using 50-step adaptation $\mathcal{A}_{\mathcal{T}_i}$ with learning rate 0.1, on support set stochastically sampled for every adaptation phase. We trained the VAE on 2000 epochs where each epoch is a single pass through all the tasks, with the Adam (Kingma and Ba, 2017) optimizer and 0.0001 learning rate and batch size 32. We used gradient norm clipping independently for both the encoder and decoder, with the maximum norm capped at 1000.

A.11.2 HyperCLIP

Training In order to train the HyperCLIP model, we need samples of fine tuned network weights W_i. Similarly to HVAE, we used adaptations from **Uncond. HNet-MAML**, using 50-step adaptation $\mathcal{A}_{\mathcal{T}_i}$ with learning rate 0.1, on a support set stochastically sampled at every adaptation phase, as this would allow us to use the same HyperCLIP model for doing guidance on both HNet and HVAE. We trained our HyperCLIP model for 600 epochs, with the Adam (Kingma and Ba, 2017) optimizer, 0.0003 learning rate and batch size 64, for all our experiments.

Guidance We use 10 steps guidance with $\lambda = 0.01$ and learning rate 0.1, for both when performed on HNet and HVAE.

Evaluation Evaluation is performed on a fixed query set on the predefined query set of the held-out test tasks of the Meta-VQA dataset. Zero-shot performance is evaluated on the output of the generative hypernetwork h after applying latent space guidance. For the few shot performance, all adaptation is performed on the support set of the test tasks, on the latent space initialized at the output of the guidance procedure. Similarly to our baselines, we use 50-steps gradient descent adaptation with learning rate 0.1.

A.11.3 HyperLDM

Training Similarly to HyperCLIP, to train HyperLDM we need samples of fine tuned network weights W_i, for which we use adaptations from **Uncond. HNet-MAML**, using 50-step adaptation $\mathcal{A}_{\mathcal{T}_i}$ with learning rate 0.1, on a support set stochastically sampled at every adaptation phase. We parametrize the diffusion process with a linear noise schedule, β starting at 0.0001 and ending at 0.06, and 350 diffusion timesteps. We train the HyperLDM for 1000 epochs with the Adam optimizer, 0.00025 learning rate and 128 epochs, for all our experiments.

Evaluation Evaluation is performed as for HyperCLIP guidance, except for the fact that adaptation is performed natively through sampling from the learned reversed diffusion process, with parameters derived from the chosen β schedule. The guidance parameter $\gamma > 0$ can be tuned during inference to accentuate the effect of classifier-free guidance.