Meta-Learning via Classifier(-free) Diffusion Guidance

Elvis Nava* *enava@ethz.ch ETH AI Center & INI & Soft Robotics Lab, ETH Zurich*

Seijin Kobayashi* *seijink@ethz.ch Dept. of Computer Science, ETH Zurich*

Yifei Yin *yifyin@ethz.ch Dept. of Computer Science, ETH Zurich*

Robert K. Katzschmann *rkk@ethz.ch Soft Robotics Lab, D-MAVT, ETH Zurich*

Benjamin F. Grewe *bgrewe@ethz.ch Institute of Neuroinformatics, University of Zurich & ETH Zurich*

Reviewed on OpenReview: *https://https://openreview.net/forum?id=1irVjE7A3w*

Abstract

We introduce meta-learning algorithms that perform zero-shot weight-space adaptation of neural network models to unseen tasks. Our methods repurpose the popular generative image synthesis techniques of natural language guidance and diffusion models to generate neural network weights adapted for tasks. 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 multi-task and meta-learning methods in a series of zero-shot learning experiments on our Meta-VQA dataset.

1 Introduction

State-of-the-art machine learning algorithms often lack the ability to 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. Indeed, humans can solve novel tasks when prompted by a variety of different interaction modalities such as visual task observations or natural language prompts. In this work, we present new meta-learning techniques that allow models to perform a similar kind of multi-modal task inference and adaptation in the weight space of neural network models. In particular, we present two different approaches (**HyperCLIP guidance** and **HyperLDM**) that utilize natural language task descriptors for zero-shot task adaptation.

The development of deep learning models that "Learn to learn" is the focus of the field of meta-learning. Meta-learning can be defined as a bi-level optimization problem, a trend stemming from the success of Model-Agnostic Meta-Learning (Finn et al., 2017, MAML): an outer loop meta-model is trained with the goal of improving the performance of a base model when fine-tuned on a variety of related tasks. MAML was specifically introduced as a gradient-based method to find a network initialization with high few-shot performance over an entire set of tasks. Recent progress in large-scale transformer networks is however challenging this explicit meta-learning framework grounded in optimization over model weights. Large models trained on huge, rich, and diverse data sets have been shown to possess surprisingly good few-shot

*Equal Contribution

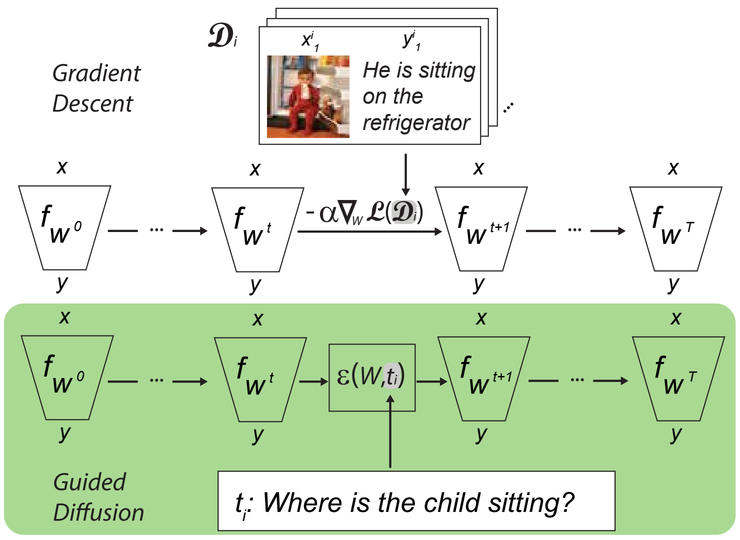

Figure 1: Given a task $\mathcal{T}_i$ and a network initialization W^0, with few-shot task data $\mathcal{D}_i$ one can use traditional gradient descent to perform task adaptation and obtain fine-tuned weights W^T. This adaptation requires that at every step t the gradient $\nabla_W\mathcal{L}(\mathcal{D}_i)$ is computed. Our methods (green) instead do not require few-shot data $\mathcal{D}_i$, but use natural language descriptors t_i to generate a surrogate adaptation towards W^T in a zero-shot manner. In the Figure, we summarize this adaptation step as $\epsilon(W, t_i)$, which depends only on t_i and not on D_i, which may be unavailable.

learning capabilities through in-context learning (Brown et al., 2020). Moreover, large-scale pre-training and fine-tuning often outperforms explicit meta-learning procedures (Mandi et al., 2022). Brown et al. (2020) dispense with the bi-level optimization formulation and use the word “Meta-Learning” to generally describe problem settings with an inner-loop/outer-loop structure, and use the words “zero-shot”, “one-shot”, or “few-shot” depending on how many demonstrations are provided in-context at inference time.[1]

These developments in transformer networks prompted us to develop alternative methods for meta-learning in weight-space which natively benefit from rich and multi-modal data like in-context learning. Inspired by recent advances in conditional image generation (Ramesh et al., 2022; Rombach et al., 2022), we recast meta-learning as a multi-modal generative modeling problem such that, given a task, few-shot data and natural language descriptions are considered equivalent conditioning modalities for adaptation (Figure 1). What we show is 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 & Nichol, 2021; Ho & Salimans, 2021; Nichol et al., 2022) can be repurposed for the meta-learning setting to generate adapted neural network weights. Using multi-step adaptation instead of traditionally conditioning the model on the natural language task information allows our models to achieve higher performance on each task by breaking down computations into multiple steps.

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; Schürholt et al., 2022) to map from its latent space to the weight space of a base model (Figure 2.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 2.B and 2.C). Our methods can thus benefit from large scale data through the pre-training phase, even when natural language descriptions are not available for all tasks.

We summarise our contributions as follows:

1) We introduce **HyperCLIP**, a contrastive learning method equivalent to Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021), producing CLIP embeddings of fine-tuned neural network weights. Using HyperCLIP as a guidance model then allows us to find task-adapted networks in the latent space of a hypernetwork model (Figure 2.B).

2) We introduce Hypernetwork Latent Diffusion Models (**HyperLDM**) as a costlier but more powerful alternative to pure HyperCLIP guidance to find task-adapted networks within the latent space of a hypernetwork model (Figure 2.C). We show how combining this approach with classifier-free guidance (Ho & Salimans, 2021) improves the performance of generated base networks.

[1]See the footnote in page 4 of Brown et al. (2020).

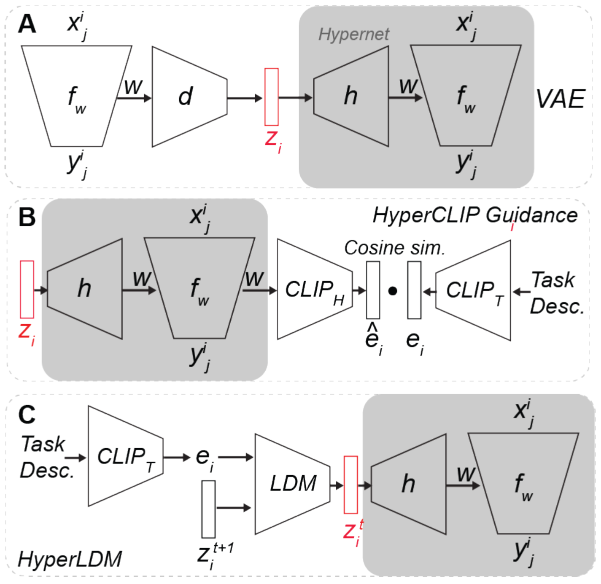

Figure 2: Schematic of the three main components of our proposed meta-learning approach. **A.** A task-unconditional variational autoencoder (VAE) models the latent space of adapted weights W for the network f trained on data (x_j^i, y_j^i). Its generator hypernetwork h producing the weights (highlighted in gray) can be re-used in the task-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.

3) We demonstrate the usefulness of our methods on **Meta-VQA**, our modification of the *VQA v2.0* dataset (Goyal et al., 2017) built to reflect the multi-task setting with natural language task descriptors. We show how our guidance methods outperform traditional multi-task and meta-learning techniques for zero-shot learning on this dataset.

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 such that it generally performs well on new unseen tasks drawn from $p(\mathcal{T})$.

2.1 Background on Model-Agnostic Meta-Learning

In Zintgraf et al. (2019)'s version of MAML, a model g is composed of context parameters ϕ that are adapted to 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, we sample training data D_i from each task $\mathcal{T}_i$ and split it 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:

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

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:

$$\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, \theta, \phi) = \phi - \eta \frac{1}{|D_i^{\mathrm{s}}|} \sum_{(x',y')\in D_i^{\mathrm{s}}} \nabla_\phi \mathcal{L}_{\mathcal{T}_i}(g(x', \theta, \phi), y') \tag{2}$$

with some learning rate η. The objective from Eq. 1 is itself solved with gradient descent. This is done 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 training process of the entire inner loop, averaged over batches of tasks.

2.2 Natural Language Task Embeddings

At test time, MAML-based meta-learning requires few-shot data D_i^{s} from test tasks to adapt its unconditioned network parameters through gradient descent. In contrast, in this work, to perform zero-shot task adaptation, we utilize an additional high-level context embedding e_i for each task $\mathcal{T}_i$. In practice, such embeddings can come from a natural language description t_i of the task, which can be encoded into a task embedding using pre-trained language models.

A simple baseline for incorporating task embeddings into a model during training is by augmenting the input of the network, concatenating such input with the task embedding during the forward pass, or using custom conditioning layers such as FiLM (Perez et al., 2017). We instead consider the use of hypernetworks (Ha et al., 2016; Zhao et al., 2020), a network that generates the weights of another network given a conditioning input.

Hypernetworks introduce multiplicative interactions between neural network model weights, similar to how the attention mechanism in transformer models allows for the efficient mixing and propagation of information across self-attention layers. In fact, transformers can be viewed as a composition of small hypernetworks. The key difference between adopting transformers for in-context learning and our approach is that we deliberately fix the architecture of our small base model f, and use a hypernetwork to sample such models. Effectively, we decouple the multiplicative hypernetwork mechanism from the downstream network specialized for a task.

2.3 Meta-Learning and Hypernetworks

Given a neural network $f(\cdot, W)$ parametrized by a weight vector W, we reparametrize the model by introducing a hypernetwork h. The hypernetwork h is parametrized by θ, and generates weights $h(z, \theta) = W$ from an embedding z. The overall model is then defined as $f(\cdot, h(z, \theta))$. In the multi-task setting, one can use a "task-conditioned" hypernetwork, in which the input embedding z directly depends on the task $\mathcal{T}_i$ (*e.g.* $z = e_i$). In this work, we will also consider "unconditional" hypernetworks, trained as generative models (see Section 3), with input embeddings z that don't depend on the task, but may for example be normally distributed.

Before introducing our new zero-shot techniques, we construct a hypernetwork-based baseline by rewriting the MAML objective (Eq. 1) with respect to the hypernetwork weight θ as

$$\min_{\theta} \mathop{\mathbb{E}}_{\mathcal{T}_i} \left[\frac{1}{|D_i^{\mathrm{q}}|} \sum_{(x,y) \in D_i^{\mathrm{q}}} \mathcal{L}_{\mathcal{T}_i} \left(f(x, h(\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, \theta, z), \theta)), y \right) \right]. \tag{3}$$

Forcing $\mathcal{A}_{\mathcal{T}_i}(D_i^{\mathrm{s}}, \theta, z) = e_i$, we recover the simple multi-task objective of a task-conditioned hypernetwork optimizing for zero-shot performance, taking e_i directly as input. When $\mathcal{A}_{\mathcal{T}_i}$ is instead the gradient descent algorithm on z, the objective corresponds to a variant of MAML, optimizing the few-shot performance of h when only adapting the embedding in the inner loop, initialized at z. For more details related to the baselines, see Appendix A.6.

3 Hypernetworks as Generative Models

A rich literature exists on hypernetworks interpreted as generative models of base network weights (see Section 7). Our work builds upon this interpretation to adapt multi-modal generative modeling techniques to the meta-learning domain.

In generative modeling, we aim to learn the distribution $p(x)$ over a high dimensional data domain $\mathcal{X}$, such as images, given samples from the distribution. To do so, we resort to techniques such as variational inference, adversarial training, or diffusion models. Our meta-learning setting can analogously be framed as modeling a distribution of diverse high-dimensional base network weights W. In the Bayesian setting, this distribution is made explicit as we seek to model the posterior $p(W|D)$ given data D. However, the framework is still useful even when no explicit posterior distribution is assumed, as it is the case for deep ensembles. In the present work, we indeed avoid explicit Bayesian inference: for each training task $\mathcal{T}_i$, we fine-tune the base model $f(x, W) = y$ on it, and use the resulting W_i as a sample to train a generative model of network weights.

The fundamental building block of our unconditional generative model is the hypernetwork $h(z, \theta) = W$ that we can train in two ways: **1. HVAE:** We define a Hypernetwork VAE (Figure 2.A), which, given samples of fine-tuned base network weights W_i, learns a low-dimensional normally distributed latent representation z_i. The encoder $d(W_i, \omega) = (\mu_{z_i}, \Sigma_{z_i})$ with parameters ω maps base network weights to means and variances used to sample a latent vector z_i, while the decoder (or generator) is a classic hypernetwork $h(z_i, \theta) = W_i$ which reconstructs the network weights from the latent vector (See Appendix A.7.1). This VAE setup is analogous to that proposed in recent work on *hyper-representations* (Schürholt et al., 2022). **2. HNet:** Using MAML, we learn both an *initialization 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. 3 (see Section 2.3).

Up to this point, we trained an unconditional hypernetwork generative model of neural network weights, comprising the *unconditional pre-training* phase of our meta-learning approach. This gives us a powerful generator $h(z, \theta) = W$, which maps from its latent space to the weight space of our base network. In the next Section, we investigate how to then perform task-conditional *guidance* within this latent space, finding adapted latent embeddings z_i for our test tasks in a zero-shot manner.

4 HyperCLIP: Training a CLIP Encoder for the "Model-Parameters Modality"

The first of our two meta-learning *guidance* techniques, **HyperCLIP Guidance**, consists of these steps:

1. We train an unconditional generative hypernetwork $h(z, \theta) = W$ over a family of training tasks (as in Section 3).
2. We train a **HyperCLIP** encoder, mapping fine-tuned neural network weights W_i for tasks $\mathcal{T}_i$ to multi-modal CLIP embeddings $e_i^{(H)}$.
3. Given a new task $\mathcal{T}_i$, we use **HyperCLIP guidance** to guide the exploration of the hypernetwork latent space, in order to find base model weights W_i with high zero-shot performance for the task.

To define **HyperCLIP guidance**, we borrow from the field of multi-modal contrastive learning. More specifically, we build on top of Contrastive Language-Image Pre-training (CLIP) (Radford et al., 2021), a popular method for joint learning of language and image embeddings with applications to zero-shot and few-shot classification.

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. Specifically, the formulation maximizes the cosine similarity $e_i^{(I)\top} e_j^{(T)} / \|e_i^{(I)}\| \|e_j^{(T)}\|$ for pair-wise matches $(i = j)$ and minimizes the cosine similarity for non-matches $(i \neq j)$. Beyond the original language-image setting, the CLIP approach can be easily adapted to include additional modalities, aligning the representation of more than two encoders at a time. Existing works such as AudioCLIP (Guzhov et al., 2022) demonstrate the possibility of training an encoder for an additional modality such as audio on the side of the pre-trained frozen CLIP language-image encoders.

4.1 Contrastive Learning on Neural Network Weights

In our work, we consider multi-modal representations of meta-learning tasks $\mathcal{T}_i$. A descriptor of a task may come from the language modality (t_i), but potentially also from image, video, or audio modalities. When we fine-tune a base machine learning model $f(x, W_i) = y$ for task $\mathcal{T}_i$, we then also consider the fine-tuned base model weights W_i as being part of an alternative *model-parameters modality* that describes task $\mathcal{T}_i$. Fine-tuned network weights from the *model-parameters modality* can then be paired in contrastive learning with the other multi-modal descriptions of $\mathcal{T}_i$. We thus define our new **HyperCLIP** encoder $CLIP_H(W_i) = e_i^{(H)}$, taking fine-tuned neural network weights W_i as input, and outputting a CLIP embedding $e_i^{(H)}$ optimized for high cosine similarity with the CLIP embedding for the textual (or image, audio, etc.) descriptor of the task. Figure 3 and Algorithm 1 illustrate the approach.

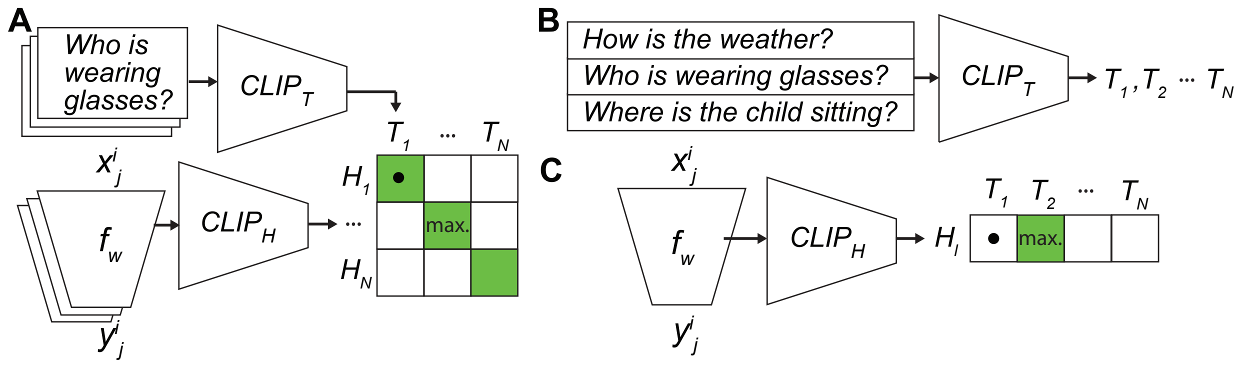

Figure 3: **A.** Our HyperCLIP encoder CLIP$_H$ is contrastively trained to map neural network weights W to the latent space of a pre-trained language encoder CLIP$_T$, which we use to embed the natural language questions associated to the tasks. **B.** To perform task inference given an already fine-tuned network one can encode all candidate task questions using the language CLIP encoder. **C.** We encode the fine-tuned network weights with HyperCLIP, and finally infer the correct task with a softmax operation over cosine similarities between HyperCLIP and language CLIP embeddings.

Algorithm 1 HyperCLIP Training

```
sample a batch of tasks $\mathcal{T}_{i=1,\dots,N}$ with loss functions $\mathcal{L}_{\mathcal{T}_i}$, training data $D_i^{\text{train}}$ 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 $i = 1,\dots,N$ do
    $T[i] = \text{CLIP}_T(t_i) \;/\; \|\text{CLIP}_T(t_i)\|$
    Fine-tune $W_i$: $\min_W \sum_{(x',y')\in D_i^{\text{train}}} \mathcal{L}_{\mathcal{T}_i}(f(x',W),y')$
    $H[i] = \text{CLIP}_H(W_i) \;/\; \|\text{CLIP}_H(W_i)\|$
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 $\nabla$loss
```

4.2 Classifier-Guided Meta-Learning

On their own, CLIP encoders are not capable of data generation. Recent popular image synthesis techniques, however, use CLIP encoders or other classifiers to *guide* generation from pre-trained unconditional generative models. *Classifier guidance* or *CLIP guidance* (Gal et al., 2021; Patashnik et al., 2021) use gradients with respect to a classifier or CLIP encoder to traverse a generative model's latent space.

In this work, we introduce **HyperCLIP guidance**, the first algorithm for classifier guidance in the meta-learning setting (Figure 2.B). Given a previously unseen validation task $\mathcal{T}_i$ and an unconditional generative hypernetwork model $h(z,\theta) = W$, we 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 descent over the guidance loss

$$\mathcal{L}_{\text{g}}(z) = -\frac{\text{CLIP}_H\left(h(z,\theta)\right)^\top \text{CLIP}_T(t_i)}{\|\text{CLIP}_H\left(h(z,\theta)\right)\|\|\text{CLIP}_T(t_i)\|} + \lambda\|z - z^0\|, \tag{4}$$

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.

5 HyperLDM: Task-conditional Diffusion of Hypernetwork Latents

Due to rapid innovation in the image synthesis community, simple CLIP guidance has been largely overcome in favor of applying classifier guidance and classifier-free guidance during the sampling process of a Diffusion Model (Dhariwal & Nichol, 2021; Ho & Salimans, 2021; Kim et al., 2022; Crowson, 2022; Nichol et al., 2022; Rombach et al., 2022). To investigate whether these advances also apply to our meta-learning setting, we introduce **HyperLDM**, a diffusion-based technique as an alternative to the previously introduced HyperCLIP guidance.

In summary, our **HyperLDM** technique for network parameter generation consists of the following steps:

1. We train an unconditional generative hypernetwork $h(z, \theta) = W$ over a family of training tasks (as in Section 3).
2. We train a conditional **HyperLDM** model, able to sample latent vectors $\hat{z}_i$ for high-performing base model neural networks, conditioned on the task embedding e_i for a task $\mathcal{T}_i$.
3. To further improve the generative quality of our HyperLDM model, we use **Classifier-free Guidance** during conditional sampling.

5.1 (Latent) 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 the data distribution are slowly corrupted with additive Gaussian noise over T steps with a variance schedule $\beta_1, \dots, \beta_T$, resulting in the Markov Chain

$$q(x^t|x^{t-1}) = \mathcal{N}(x^t; \sqrt{1-\beta_t}x^{t-1}, \beta_t \mathbf{I}) \tag{5}$$

$$q(x^{1:T}|x^0) = \prod_{t=1}^{T} q(x^t|x^{t-1}). \tag{6}$$

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 $p(x)$

$$\mathcal{L}_{\text{DM}}(\psi) = \mathbb{E}_{x, \epsilon \sim \mathcal{N}(0,1), t} \left[\|\epsilon - \epsilon_\psi(x^t, t)\|_2^2 \right]. \tag{7}$$

Sampling from the reverse process can then be done with

$$x^{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x^t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\psi(x^t, t) \right) + \sigma_t \xi, \tag{8}$$

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 an analog to Langevin Dynamics, a connection explicitly made in works exploring the diffusion technique from the "score matching" perspective (Song & Ermon, 2019; Song et al., 2020).

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 $h(z, \theta)$, 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 then propose a Hypernetwork Latent Diffusion Model (**HyperLDM**), which learns 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 fits 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

$$\mathcal{L}_{\text{LDM}}(\psi) = \mathbb{E}_{\mathcal{T}_i, d(W_i), \epsilon \sim \mathcal{N}(0,1), t} \left[\|\epsilon - \epsilon_\psi(z^t, t, e_i)\|_2^2 \right]. \tag{9}$$

5.2 Classifier-Free Guidance for Meta-Learning

To improve the quality of sampled networks, the classifier guidance technique presented in Section 4.2 can be also combined together with 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 & Salimans, 2021; Nichol et al., 2022) allows us to leverage a conditional diffusion model to obtain the same effect 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, and inputting zeros instead. We can then sample at each diffusion iteration with

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = (1-\gamma)\,\epsilon_\psi(z^t, t, 0) + \gamma\epsilon_\psi(z^t, t, e_i). \tag{10}$$

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.1.

6 Experimental Setup and Results

In this section, we demonstrate the competitiveness of our two approaches in zero-shot image classification experiments against a series of traditional meta-learning techniques. Throughout our experiments, we fix the choice of base network model f to a CLIP-Adapter model (see Appendix A.2), only varying the meta-learning techniques employed to obtain adapted base model weights. The CLIP-Adapter base model makes use of pre-trained CLIP encoders to obtain high base performance on image classification with textual labels while maintaining a relatively small trainable parameter count. It should not be confused with the usage of CLIP encoders to produce task embeddings, or to train HyperCLIP, all of which happens at the meta-level. Our base model f effectively performs classification in the same way that CLIP (Radford et al., 2021) does, by using natural language labels as opposed to a one-hot vector for classes, then encoding both image and answers with CLIP encoders, and scoring each answer using a dot product between its embedding and the image embedding.

6.1 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 & 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 $(x_j^i, y_j^i) \in D_i$. We refer to Appendix A.5 for a more in-depth discussion, and to Figure 4 for an illustrative example of a task from the dataset.

6.2 Zero-Shot Task Adaptation Experiments

In Table 1 we show how our methods compare to a series of baselines when tested on the Meta-VQA dataset in the zero-shot setting. For each training task $\mathcal{T}_i$, the algorithms are given access to the full 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 are given, and the algorithms are tasked with predicting the correct labels of images in the query set D_i^{q}.

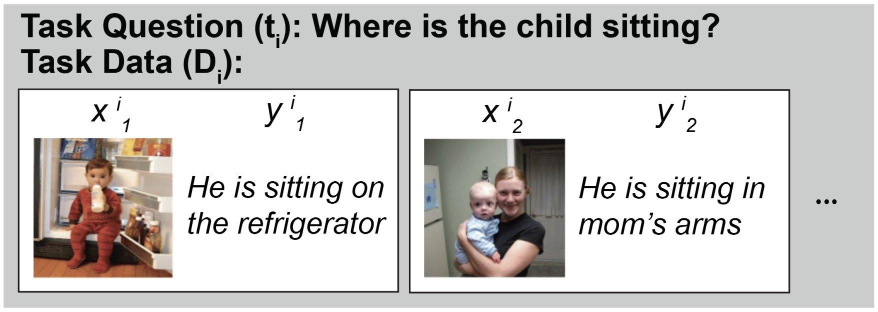
Task Question (t_i): Where is the child sitting?
Task Data (D_i):
x^i_1
y^i_1
He is sitting on the refrigerator
x^i_2
y^i_2
He is sitting in mom's arms
...

Figure 4: Example classification task from Meta-VQA, adapted from VQA v2 (Goyal et al., 2017). A task $\mathcal{T}_i$ is associated to a single question t_i and multiple image-answer tuples (x^i_j, y^i_j).

Table 1: Zero-shot accuracy (mean ± s.d.) averaged over Meta-VQA test tasks. Results should be interpreted as relative to a performance ceiling of 60.24% obtainable when task data is available (few-shot learning), and with our fixed choice of base model (see Appendix A.8). The columns separate the setting in which only half of task descriptors/questions are given (50% Q.), and that in which all of the task descriptors are given (100% Q.). (* ours)

Method	**Zero-shot (50% Q.)**	**Zero-Shot (100% Q.)**
CLIP as Base Model	44.99	
Uncond. Multitask	53.75 (± 0.36)	
Uncond. MNet-MAML	53.04 (± 0.69)	
Uncond. MNet-FOMAML	53.04 (± 0.42)	
Uncond. HNet-MAML	53.37 (± 0.29)	
Cond. Multitask	51.68 (± 0.33)	54.12 (± 0.80)
Cond. Multitask FiLM	51.60 (± 0.56)	53.84 (± 0.61)
Cond. HNet-MAML	51.54 (± 0.63)	53.02 (± 0.20)
* HNet + HyperCLIP guidance	53.51 (± 0.22)	53.98 (± 0.54)
* HVAE + HyperCLIP guidance	53.82 (± 0.07)	53.91 (± 0.08)
* HNet + HyperLDM $\gamma = 1$	53.66 (± 0.25)	54.06 (± 0.21)
* HNet + HyperLDM $\gamma = 1.5$	54.08 (± 0.11)	54.30 (± 0.27)
* HVAE + HyperLDM $\gamma = 1$	54.72 (± 0.23)	55.03 (± 0.32)
* HVAE + HyperLDM $\gamma = 1.5$	**54.84** (± 0.24)	**55.10** (± 0.08)

In addition, we also simulate a setting in which we possess a larger "unconditional" pre-training dataset. Our two-phased approach, which separates generative model pre-training and guidance, benefits from unconditional data: tasks without language descriptors can still be used to learn the unconditional HNet/HVAE model. To test this, we conduct additional runs in which we train our model while only keeping a fraction of task descriptors from the Meta-VQA dataset.

Using the original CLIP for zero-shot image classification (**CLIP as Base Model**) provides a 44.99% *floor* for performance on Meta-VQA. All other techniques will use CLIP-Adapter as the base model, as previously mentioned. We also obtained an approximate 60.24% performance *ceiling* from the best method in the few-shot setting, in which models have also access to a data support set D^s_i for every test task (see Appendix A.8). Our zero-shot techniques cannot surpass this ceiling while keeping the choice of base model fixed. The zero-shot scores should then be judged within a range between 44.99% and 60.24% accuracy.

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 Section 2.3) and a conditional one (**Cond. HNet-MAML**). We also benchmark against standard multitask learning (**Uncond. Multitask**, **Cond. Multitask**) on the base model without hypernetworks, and conditional multitask learning with the classic FiLM layer (Perez

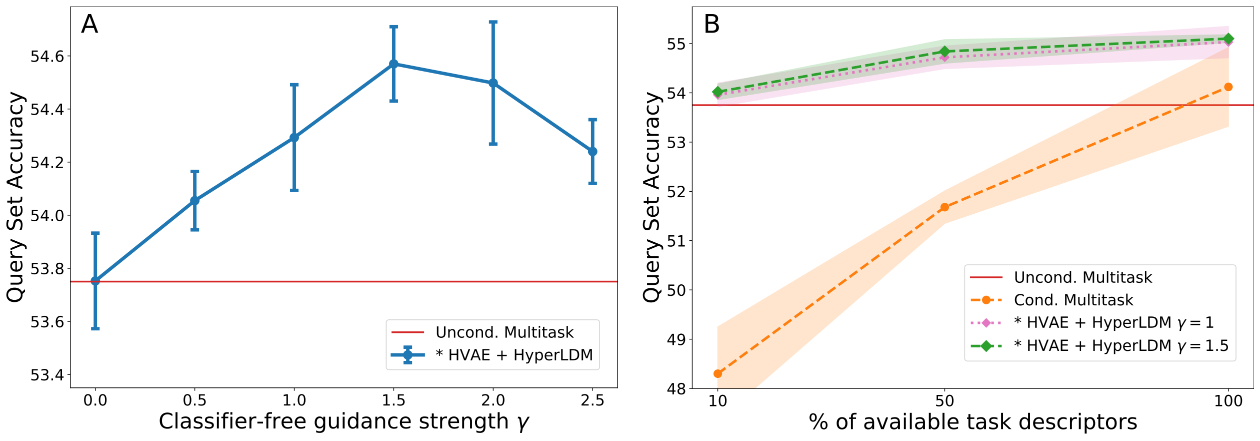

Figure 5: **A.** Zero-shot performance of HyperLDM (mean ± s.d.) over different classifier-free guidance parameters γ. For $\gamma = 0$ we sample from an unconditional latent diffusion model. For $\gamma = 1$ we sample with classic conditioning. For $\gamma > 1$, we are in the classifier-free guidance regime. **B.** Zero-shot performance of HyperLDM (mean ± s.d.) against baselines in the setting where only a fraction of natural language task labels are given.

et al., 2017) (**Cond. Multitask FiLM**). Note 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.2 and A.6 for more details on each algorithm.

We then test out two approaches, **HyperCLIP guidance** and **HyperLDM**, when trained on top of either a hypernetwork or a VAE generator (Section 3, see also Appendix A.2 and A.7 for more detail). **HyperCLIP guidance** allows for faster sampling than **HyperLDM** but is generally less performant. Still, **HyperCLIP guidance** performs on par with or slighly improves upon all other zero-shot baselines except for **Cond. Multitask**. The best-performing model for the zero-shot setting is **HVAE + HyperLDM**, and specifically for classifier-free guidance with $\gamma = 1.5$. As illustrated in Figure 5.A, to further show the effectiveness of the classifier-free guidance technique, we run a different experiment sweeping over several candidate γ parameters to find that the optimum occurs for $\gamma > 1$. As shown in Figure 5.B, when training our model while only keeping 50% or 10% of task descriptors, traditional **Cond. Multitask** learning heavily overfits, while **HyperLDM** is almost not affected due to its two-phased training regime based on an unconditional VAE. The gap between the multitask baseline and our HyperLDM technique is particularly striking in this setting.

7 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 (Henning et al., 2021). Another vein of work consists in using hypernetworks to distill ensembles of diverse networks (Wang et al., 2018; Ratzlaff & Fuxin, 2020; von Oswald et al., 2021a). Recent work also explored the properties of hypernetworks as autoencoder generative models of network weights (Schürholt et al., 2022).

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; 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 & Choi, 2018), instead of directly adapting in weight space. Recent works bypass the formal bi-level formulation of meta-learning (Brown et al., 2020) by, *e.g.*, using transformers to directly map the few-shot examples to the weights of the target network (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 & 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 & 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 & Nichol, 2021; Gal et al., 2021; Patashnik et al., 2021) and classifier-free guidance (Ho & 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).

Zero-shot Learning There exists a large literature on zero-shot learning, including both established benchmarks and well-known methods (Han et al., 2021; Su et al., 2022; Gupta et al., 2021). While these zero-shot learning works consider the zero-shot performance on unseen class labels within a single classification task, our setting considers that of the zero-shot performance where test tasks themselves are unseen, thus raising the zero-shot problem to the task level.

8 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.

Acknowledgments

We are grateful for funding received by the ETH AI Center, Swiss National Science Foundation (B.F.G. CRSII5-173721 and 315230 189251), ETH project funding (B.F.G. ETH-20 19-01), the Human Frontiers Science Program (RGY0072/2019), and Credit Suisse Asset Management.

Code Release

Our code is available at `https://github.com/elvisnava/hyperclip`.

Author Contributions

Elvis Nava * Original idea, implementation, and experiments (guidance techniques and diffusion), paper writing (overall manuscript structure, generative modeling and guidance sections, experimental results section).

Seijin Kobayashi * Implementation and experiments (MAML and hypernetworks, HyperCLIP guidance), paper writing (meta-learning background and related work).

Yifei Yin Creation of the Meta-VQA dataset, initial experiments on the CLIP-Adapter base model.

Robert K. Katzschmann Conceptualising of the project idea with a focus on down-stream learning applications in robotics, project co-lead. Participating in project meetings, giving feedback and conceptually guiding the approaches, Revising the manuscript and figures.

Benjamin F. Grewe Conceptualisation of the project with a focus on using language task descriptors, senior project lead, participating in project meetings, providing conceptual input for developing the two approaches, giving feedback on the manuscript, developing intuitive schematics for Figures 2 and 3.

References

Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. September 2018. URL `https://openreview.net/forum?id=HyxnZh0ct7`.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. October 2018. URL `https://openreview.net/forum?id=B1xsqj09Fm&noteId=HklmZ1xqhm&utm_campaign=Dynamically%20Typed&utm_medium=email&utm_source=Revue%20newsletter`.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html`.

Katherine Crowson. v-diffusion, May 2022. URL `https://github.com/crowsonkb/v-diffusion-pytorch`. original-date: 2021-12-16T17:04:35Z.

Prafulla Dhariwal and Alexander Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794. Curran Associates, Inc., 2021. URL `https://papers.nips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html`.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1126–1135. PMLR, July 2017. URL `https://proceedings.mlr.press/v70/finn17a.html`. ISSN: 2640-3498.

Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, Francesco Visin, Hujun Yin, and Raia Hadsell. Meta-Learning with Warped Gradient Descent, February 2020. URL `http://arxiv.org/abs/1909.00025`. arXiv:1909.00025 [cs, stat].

Rinon Gal, Or Patashnik, Haggai Maron, Gal Chechik, and Daniel Cohen-Or. StyleGAN-NADA: CLIP-Guided Domain Adaptation of Image Generators. Technical Report arXiv:2108.00946, arXiv, December 2021. URL `http://arxiv.org/abs/2108.00946`. arXiv:2108.00946 [cs] type: article.

Peng Gao, Shijie Geng, Renrui Zhang, Teli Ma, Rongyao Fang, Yongfeng Zhang, Hongsheng Li, and Yu Qiao. CLIP-Adapter: Better Vision-Language Models with Feature Adapters. *arXiv:2110.04544 [cs]*, October 2021. URL `http://arxiv.org/abs/2110.04544`. arXiv: 2110.04544.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2014. ISSN 0001-0782. doi: 10.1145/3422622. URL `https://doi.org/10.1145/3422622`.

Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. pp. 6904–6913, 2017. URL `https://openaccess.thecvf.com/content_cvpr_2017/html/Goyal_Making_the_v_CVPR_2017_paper.html`.

Akshita Gupta, Sanath Narayan, Salman Khan, Fahad Shahbaz Khan, Ling Shao, and Joost van de Weijer. Generative Multi-Label Zero-Shot Learning, January 2021. URL http://arxiv.org/abs/2101.11606. arXiv:2101.11606 [cs] version: 2.

Andrey Guzhov, Federico Raue, Jörn Hees, and Andreas Dengel. Audioclip: Extending Clip to Image, Text and Audio. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 976–980, May 2022. doi: 10.1109/ICASSP43922.2022.9747631. ISSN: 2379-190X.

David Ha, Andrew Dai, and Quoc V. Le. HyperNetworks. Technical Report arXiv:1609.09106, arXiv, December 2016. URL http://arxiv.org/abs/1609.09106. arXiv:1609.09106 [cs] type: article.

Zongyan Han, Zhenyong Fu, Shuo Chen, and Jian Yang. Contrastive Embedding for Generalized Zero-Shot Learning. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2371–2381, Nashville, TN, USA, June 2021. IEEE. ISBN 978-1-66544-509-2. doi: 10.1109/CVPR46437.2021.00240. URL https://ieeexplore.ieee.org/document/9578587/.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification, February 2015. URL http://arxiv.org/abs/1502.01852. arXiv:1502.01852 [cs].

Christian Henning, Johannes von Oswald, Joao Sacramento, Simone Carlo Surace, Jean-Pascal Pfister, and Benjamin F. Grewe. Approximating the Predictive Distribution via Adversarially-Trained Hypernetworks. In *Bayesian Deep Learning Workshop, NeurIPS 2018*, Montréal, Canada, December 2018. Yarin. doi: 10.5167/uzh-168578. URL http://bayesiandeeplearning.org/2018/papers/121.pdf.

Christian Henning, Maria R. Cervera, Francesco D'Angelo, Johannes von Oswald, Regina Traber, Benjamin Ehret, Seijin Kobayashi, João Sacramento, and Benjamin F. Grewe. Posterior Meta-Replay for Continual Learning. *arXiv:2103.01133 [cs]*, June 2021. URL http://arxiv.org/abs/2103.01133. arXiv: 2103.01133.

Yusuke Hirota, Yuta Nakashima, and Noa Garcia. Gender and Racial Bias in Visual Question Answering Datasets. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pp. 1280–1292, June 2022. doi: 10.1145/3531146.3533184. URL http://arxiv.org/abs/2205.08148. arXiv:2205.08148 [cs].

Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. November 2021. URL https://openreview.net/forum?id=qw8AKxfYbI&utm_campaign=The%20Batch&utm_source=hs_email&utm_medium=email&_hsenc=p2ANqtz-8x1u8iiVdztrPz7MsKz--4T7G3-b8L3RsGWCtkvf1hnN-nqvoAD_zpR8XSKjCoNR3kavee.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4c5bcfec8584af0d967f1ab10179ca4b-Abstract.html.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. pp. 7132–7141, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Hu_Squeeze-and-Excitation_Networks_CVPR_2018_paper.html.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural Tangent Kernel: Convergence and Generalization in Neural Networks, February 2020. URL http://arxiv.org/abs/1806.07572. arXiv:1806.07572 [cs, math, stat].

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. Technical Report arXiv:2205.09991, arXiv, May 2022. URL http://arxiv.org/abs/2205.09991. arXiv:2205.09991 [cs] type: article.

Siddhant M. Jayakumar, Wojciech M. Czarnecki, Jacob Menick, Jonathan Schwarz, Jack Rae, Simon Osindero, Yee Whye Teh, Tim Harley, and Razvan Pascanu. Multiplicative Interactions and Where to Find Them. September 2019. URL https://openreview.net/forum?id=rylnK6VtDH.

Heeseung Kim, Sungwon Kim, and Sungroh Yoon. Guided-TTS: A Diffusion Model for Text-to-Speech via Classifier Guidance. In *Proceedings of the 39th International Conference on Machine Learning*, pp. 11119–11133. PMLR, June 2022. URL `https://proceedings.mlr.press/v162/kim22d.html`. ISSN: 2640-3498.

Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017. URL `http://arxiv.org/abs/1412.6980`. arXiv:1412.6980 [cs].

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. URL `http://arxiv.org/abs/1312.6114`. arXiv: 1312.6114.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. Bayesian Hypernetworks, April 2018. URL `http://arxiv.org/abs/1710.04759`. arXiv:1710.04759 [cs, stat].

Yoonho Lee and Seungjin Choi. Gradient-Based Meta-Learning with Learned Layerwise Metric and Subspace, June 2018. URL `http://arxiv.org/abs/1801.05558`. arXiv:1801.05558 [cs, stat].

Zhao Mandi, Pieter Abbeel, and Stephen James. On the Effectiveness of Fine-tuning Versus Meta-reinforcement Learning. Technical Report arXiv:2206.03271, arXiv, June 2022. URL `http://arxiv.org/abs/2206.03271`. arXiv:2206.03271 [cs] type: article.

Alex Nichol, Joshua Achiam, and John Schulman. On First-Order Meta-Learning Algorithms, October 2018. URL `http://arxiv.org/abs/1803.02999`. arXiv:1803.02999 [cs].

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. GLIDE: Towards Photorealistic Image Generation and Editing with Text-Guided Diffusion Models. Technical Report arXiv:2112.10741, arXiv, March 2022. URL `http://arxiv.org/abs/2112.10741`. arXiv:2112.10741 [cs] type: article.

Or Patashnik, Zongze Wu, Eli Shechtman, Daniel Cohen-Or, and Dani Lischinski. StyleCLIP: Text-Driven Manipulation of StyleGAN Imagery. pp. 2085–2094, 2021. URL `https://openaccess.thecvf.com/content/ICCV2021/html/Patashnik_StyleCLIP_Text-Driven_Manipulation_of_StyleGAN_Imagery_ICCV_2021_paper.html`.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual Reasoning with a General Conditioning Layer, December 2017. URL `http://arxiv.org/abs/1709.07871`. arXiv:1709.07871 [cs, stat].

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020 [cs]*, February 2021. URL `http://arxiv.org/abs/2103.00020`. arXiv: 2103.00020.

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. Technical Report arXiv:2204.06125, arXiv, April 2022. URL `http://arxiv.org/abs/2204.06125`. arXiv:2204.06125 [cs] type: article.

Neale Ratzlaff and Li Fuxin. HyperGAN: A Generative Model for Diverse, Performant Neural Networks. Technical Report arXiv:1901.11058, arXiv, July 2020. URL `http://arxiv.org/abs/1901.11058`. arXiv:1901.11058 [cs, stat] type: article.

Sachin Ravi and Hugo Larochelle. Optimization as a Model for Few-Shot Learning. October 2016. URL `https://openreview.net/forum?id=rJY0-Kcll`.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-Resolution Image Synthesis With Latent Diffusion Models. pp. 10684–10695, 2022. URL `https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html`.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi (eds.), *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, Lecture Notes in Computer Science, pp. 234–241, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24574-4. doi: 10.1007/978-3-319-24574-4_28.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-Learning with Latent Embedding Optimization. September 2018. URL `https://openreview.net/forum?id=BJgklhAcK7&source=post_page---------------------------`.

Konstantin Schürholt, Boris Knyazev, Xavier Giró-i Nieto, and Damian Borth. Hyper-Representations as Generative Models: Sampling Unseen Neural Network Weights, September 2022. URL `http://arxiv.org/abs/2209.14733`. arXiv:2209.14733 [cs].

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning*, pp. 2256–2265. PMLR, June 2015. URL `https://proceedings.mlr.press/v37/sohl-dickstein15.html`. ISSN: 1938-7228.

Yang Song and Stefano Ermon. Generative Modeling by Estimating Gradients of the Data Distribution. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL `https://proceedings.neurips.cc/paper/2019/hash/3001ef257407d5a371a96dcd947c7d93-Abstract.html`.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. September 2020. URL `https://openreview.net/forum?id=PxTIG12RRHS&utm_campaign=NLP%20News&utm_medium=email&utm_source=Revue%20newsletter`.

Hongzu Su, Jingjing Li, Zhi Chen, Lei Zhu, and Ke Lu. Distinguishing Unseen from Seen for Generalized Zero-shot Learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7875–7884, New Orleans, LA, USA, June 2022. IEEE. ISBN 978-1-66546-946-3. doi: 10.1109/CVPR52688.2022.00773. URL `https://ieeexplore.ieee.org/document/9879149/`.

Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations (ICLR 2020)*, May 2020. doi: 10.48550/arXiv.1906.00695. URL `http://arxiv.org/abs/1906.00695`. arXiv:1906.00695 [cs, stat].

Johannes von Oswald, Seijin Kobayashi, Alexander Meulemans, Christian Henning, Benjamin F. Grewe, and João Sacramento. Neural networks with late-phase weights. *arXiv:2007.12927 [cs, stat]*, April 2021a. URL `http://arxiv.org/abs/2007.12927`. arXiv: 2007.12927.

Johannes von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *arXiv:2110.14402 [cs]*, October 2021b. URL `http://arxiv.org/abs/2110.14402`. arXiv: 2110.14402.

Kuan-Chieh Wang, Paul Vicol, James Lucas, Li Gu, Roger Grosse, and Richard Zemel. Adversarial Distillation of Bayesian Neural Network Posteriors. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 5190–5199. PMLR, July 2018. URL `https://proceedings.mlr.press/v80/wang18i.html`. ISSN: 2640-3498.

Dominic Zhao, Seijin Kobayashi, João Sacramento, and Johannes von Oswald. Meta-Learning via Hypernetworks. In *4th Workshop on Meta-Learning at NeurIPS 2020 (MetaLearn 2020)*, Virtual Conference, December 2020. IEEE. doi: 10.5167/uzh-200298. URL `https://www.zora.uzh.ch/id/eprint/200298/`.

Andrey Zhmoginov, Mark Sandler, and Max Vladymyrov. HyperTransformer: Model Generation for Supervised and Semi-Supervised Few-Shot Learning, July 2022. URL `http://arxiv.org/abs/2201.04182`. arXiv:2201.04182 [cs].

Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast Context Adaptation via Meta-Learning, June 2019. URL `http://arxiv.org/abs/1810.03642`. arXiv:1810.03642 [cs, stat].

A Appendix

A.1 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

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = \epsilon_\psi(z^t, t, e_i) - \eta\sigma_t \nabla_{z^t} \log p_\psi(e_i|z^t) \approx -\sigma_t \nabla_{z^t} \left[\log p(z^t|e_i) + \eta \log p_\psi(e_i|z^t)\right]. \tag{11}$$

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

$$-\sigma_t \nabla_{z^t} \left[\log p(z^t) + \gamma \log p(e_i|z^t)\right] \quad \text{with} \quad \gamma = 1 + \eta \tag{12}$$

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 5.2.

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. 12 we obtain

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

$$-\sigma_t \nabla_{z^t} \left[(1-\gamma) \log p(z^t) + \gamma \log p(z^t|e_i)\right], \tag{14}$$

which can be implemented with our conditional network as

$$\tilde{\epsilon}_\psi(z^t, t, e_i) = (1-\gamma)\, \epsilon_\psi(z^t, t, 0) + \gamma \epsilon_\psi(z^t, t, e_i). \tag{15}$$

A.2 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 (Radford et al., 2021) with added learned fully-connected layers refining the output embedding. Specifically, we use the *ViT-L/14@336px* CLIP encoder type with an embedding size of 768, and for the adapter we use an MLP with one hidden layer of 256 units, which are followed by a rectified linear activation, and a new linear output layer of size 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 6.2, when benchmarking the base model alone in the zero-shot setting (CLIP as Base Model), we drop the Adapter and use pre-trained zero-shot CLIP.

Hypernetwork (h) For the hypernetworks used in our baseline as well as the generative model, we use an 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 The variational autoencoder in our unconditioned generative model uses as decoder 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, and 8192.

A.3 Notes and Limitations

While optimal parameter counts for the task-conditional techniques vary between our techniques, as well as between our techniques and the baselines, all of the investigated approaches ultimately produce adapted weights for the same base network f, with the same architecture. The average performance of this base network with this fixed architecture when adapted and deployed on each individual test task is what allows us to fairly compare all meta-learning and multi-task algorithms. We acknowledge that, as a limitation of our work, the comparisons hold up when comparing relatively small fixed base networks f, and our approach might not be scalable to compete with massively pre-trained large-scale multi-task models. In any case, we believe that the weight space generation of compact models can be useful in a variety of contexts, such as when the adapted base model needs to be deployed in embedded systems and other domains with limited compute resources. The scaling behavior of our techniques is still an open problem, which can be of interest for future research.

A.4 Fairness and Bias

While our overall proposed methods belong to the realm of general-purpose techniques, their specific application may inadvertently raise issues related to gender and racial bias. As our MetaVQA dataset is a simple modification of VQA v2 (Goyal et al., 2017), it straightforwardly inherits gender and racial bias problems that have been found to exist within this dataset (Hirota et al., 2022). For example, answer distributions have been found to be different for the same question when the subject is a man or a woman. Such imbalances will necessarily exactly transfer to our MetaVQA dataset. Moreover, question-answer pairs have been found that express gender stereotypes. In any case, these problems are dataset-specific and can be ameliorated by future work on the source material from which MetaVQA is constructed, or by taking gender bias issues into account when constructing a new dataset from scratch.

A.5 The Meta-VQA Dataset

The original VQA problem is about choosing a suitable natural language answer y_k when prompted with both a natural language question t_k and an image x_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 $(t_k, x_k, y_k) \in D$, we can group the data by unique questions t_i (which will serve as task descriptor), each of which can then be associated with supervised image classification tuples $(x^i_j, y^i_j) \in D_i$. To make sure the designed tasks are meaningful, we filter out question-answer pairs with questions in a choice format, *e.g.,* "A or B?" or "yes/no" answers. From the remaining questions, we keep the ones that 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. There are on average 9.13 answer choices per question/task. The average size of the support set is 57.85 examples, while the average size of the query set is 25.9 examples.

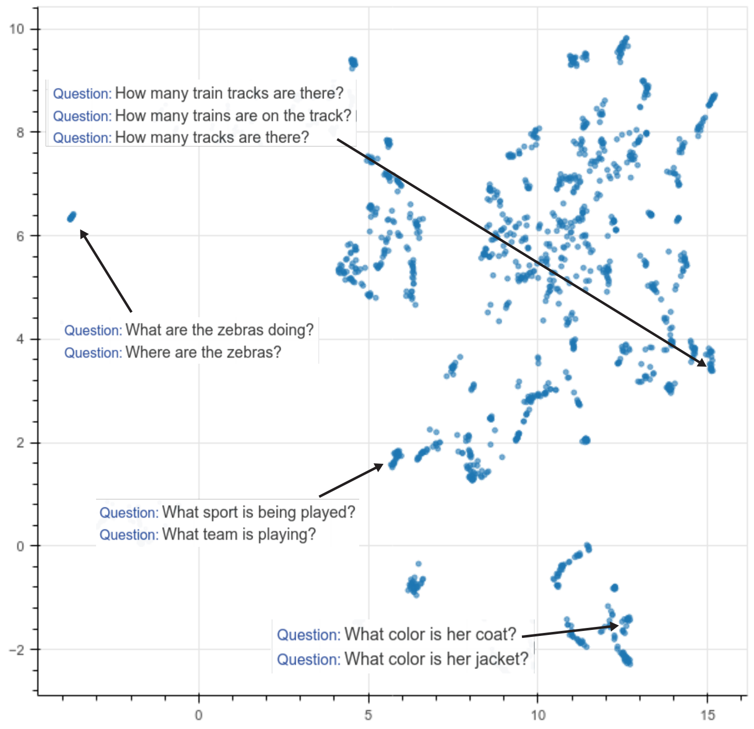

Figure 6: UMAP projection for CLIP embeddings of MetaVQA questions (using the *ViT-L/14@336px* CLIP encoder). Clusters of similar questions exist in the dataset, similarly to how they exist in VQA v2.

A.6 Baseline methods

We detail an overview of the baseline methods we benchmark in table 2, together with algorithm tables detailing each baseline method.

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 D_i from a sampled task $\mathcal{T}_i$ is randomly split into a support set D_i^{s} and a query set D_i^{q} during training. The support set is then used to perform the adaptation, while the query set is used to compute the loss on which the meta-parameters are updated (see Section 2.1). For baselines with no inner-loop adaptation $\mathcal{A}_{\mathcal{T}_i}$, all the data D_i from a sampled task $\mathcal{T}_i$ is used in training. *Unconditional* methods do not have access to the task embedding e_i, while *conditional* methods do. When the percentage of available task descriptors is reduced, conditional baselines are trained only on the tasks for which the descriptors are available, as they require such descriptors during training, unlike our two-phased techniques.

Evaluation: For each held-out test task $\mathcal{T}_i$ from the Meta-VQA dataset, we perform a zero-shot model evaluation on the fixed predefined query set D_i^{q} for the task. Zero-shot performance is evaluated before applying any adaptation procedure $\mathcal{A}_{\mathcal{T}_i}$.

Algorithm 2 Unconditional Multitask Training

```
Define the base network $f$ with parameters $W$.
for epoch $= 1, \dots, N$ do
    Sample a training batch of image-answer pairs $(x_k, y_k)$ from a mix of random training tasks $\mathcal{T}_i$.
    Update $W$ with gradient descent computed with respect to the classification loss over the sampled batch.
end for
```

Algorithm 3 Unconditional MNet-MAML Training

```
Define the base network $f$ with parameters $W$.
for meta-epoch $= 1, \dots, N$ do
    Sample a training task $\mathcal{T}_i$ and data $D_i$.
    Randomly split $D_i$ into support set $D_i^s$ and query set $D_i^q$.
    Run inner-loop adaptation $\mathcal{A}_{\mathcal{T}_i}$ using the support set $D_i^s$, fine-tuning $W$ into task-adapted $W_i = \mathcal{A}_{\mathcal{T}_i}(W)$.
    Use MAML gradient update to adapt $W$ given the inner-loop adaptation.
end for
```

Algorithm 4 Unconditional HNet-MAML Training

```
Define the base network $f$ with parameters $W$.
Define a hypernetwork $h$ with meta-parameters $\theta$, mapping a latent vector $z^0$ to base network weights $W$.
for meta-epoch $= 1, \dots, N$ do
    Sample a training task $\mathcal{T}_i$ and data $D_i$.
    Randomly split $D_i$ into support set $D_i^s$ and query set $D_i^q$.
    Run inner-loop adaptation $\mathcal{A}_{\mathcal{T}_i}$ using the support set $D_i^s$, fine-tuning $z^0$ into task-adapted $z_i = \mathcal{A}_{\mathcal{T}_i}(z^0)$.
    Use MAML gradient update to adapt $z^0$ and $\theta$ given the inner-loop adaptation.
end for
```

Algorithm 5 Conditional Multitask Training

```
Define the base network $f$ with parameters $W$.
Define a hypernetwork $h$ with meta-parameters $\theta$, mapping the clip embedding $e_i$ of the language task descriptor to base network weights $W_i$.
for epoch $= 1, \dots, N$ do
    Sample a training batch of task clip embedding, image and answer triples $(e_k, x_k, y_k)$ from a mix of random training tasks $\mathcal{T}_i$.
    Update $\theta$ with gradient descent computed with respect to the classification loss over the sampled batch.
end for
```

Algorithm 6 Conditional Multitask FiLM Training

```
Define the base network $f$ with parameters $W$.
Define a FiLM layer, mapping the clip embedding $e_i$ of the language task descriptor to modulation signals for the hidden activation layer of $f$.
for meta-epoch $= 1, \dots, N$ do
    Sample a training batch of task clip embedding, image and answer triples $(e_k, x_k, y_k)$ from a mix of random training tasks $\mathcal{T}_i$.
    Update $\theta$ with gradient descent computed with respect to the classification loss over the sampled batch.
end for
```

Algorithm 7 Conditional HNet-MAML Training

```
Define the base network $f$ with parameters $W$.
Define a hypernetwork $h$ with meta-parameters $\theta$, mapping the clip embedding $e_i$ of the language task descriptor to base network weights $W_i$.
for meta-epoch $= 1, \dots, N$ do
    Sample a training task $\mathcal{T}_i$, data $D_i$ and the clip embedding $e_i$ of the task descriptor.
    Randomly split $D_i$ into support set $D_i^s$ and query set $D_i^q$.
    Run inner-loop adaptation $\mathcal{A}_{\mathcal{T}_i}$ using the support set $D_i^s$, fine-tuning $e_i$ into task-adapted $\tilde{e_i} = \mathcal{A}_{\mathcal{T}_i}(e_i)$.
    Use MAML gradient update to adapt $\theta$ given the inner-loop adaptation.
end for
```

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 mini-batches 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**
Uncond. Multitask	$f(\cdot, W)$	W
Uncond. MNet-(FO)MAML	$f(\cdot, \mathcal{A}_{\mathcal{T}_i}(W^0))$	W^0
Uncond. HNet-MAML	$f(\cdot, h(\mathcal{A}_{\mathcal{T}_i}(z^0), \theta))$	θ, z^0
Cond. Multitask	$f(\cdot, h(e_i, \theta))$	θ
Cond. Multitask FiLM	$f(\cdot, e_i, W)$	W
Cond. HNet-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 & Ba, 2017) optimizer, with a 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 $\mathcal{A}$ has a range of possible steps, the number of steps is sampled uniformly from the range for every adaptation.

Method	`epochs`	`lr`	$\mathcal{A}$-`lr`	$\mathcal{A}$-`steps`
Uncond. MNet-Multitask	300	0.0001	-	-
Uncond. MNet-(FO)MAML	500	0.00003	0.01	0-10
Uncond. HNet-MAML	100	0.00003	0.1	0-10
Cond. Multitask	60	0.0001	-	-
Cond. Multitask FiLM	300	0.0001	-	-
Cond. HNet-MAML	200	0.00001	0.1	0-10

A.7 Guidance Models

A.7.1 Generative hypernetwork

To enable our guidance methods, we need to first train a generative hypernetwork h as in Section 3, either in the form of an Unconditional Hypernetwork, or of a Hypernetwork VAE:

- For **HNet + HyperCLIP guidance** and **HNet + HyperLDM**, we meta-learned an unconditional 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 a VAE on samples of fine-tuned network weights W_i using the base network architecture specified in Appendix A.2. We detail the procedure in Algorithm 8 and, as training samples W_i, we use adaptations over the base network (initialized from a learned **Uncond. MNet-MAML** initialization), using 50-step adaptation $\mathcal{A}_{\mathcal{T}_i}$ with learning rate 0.01 on randomly split support sets. We trained the VAE for 2000 epochs where each epoch is a single pass through all the tasks, with the Adam (Kingma & 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.

Algorithm 8 HVAE Training

```
Define the base network $f$ with parameters $W$.
Define an encoder $z = d(W, \omega)$ with parameters $\omega$ and a hypernetwork decoder $W = h(z, \theta)$ with parameters $\theta$.
Obtain a previously learned base network initialization $W^0$ according to **Uncond. MNet-MAML** (Algorithm 7).
for epoch $= 1, \ldots, N$ do
    Create an empty batch $B = \{\}$.
    for $b = 1, \ldots, M$ do
        Sample a training task $\mathcal{T}_i$ and data $D_i$.
        Randomly split $D_i$ into support set $D_i^s$ and query set $D_i^q$.
        Run inner-loop adaptation $\mathcal{A}_{\mathcal{T}_i}$ using the support set $D_i^s$, fine-tuning $W_i = \mathcal{A}_{\mathcal{T}_i}(W^0)$.
        Add the fine-tuned weights to the batch: $B = B \cup \{W_i\}$.
    end for
    Train the HVAE encoder and decoder using the VAE loss to reconstruct the weight batch $B$.
end for
```

A.7.2 HyperCLIP

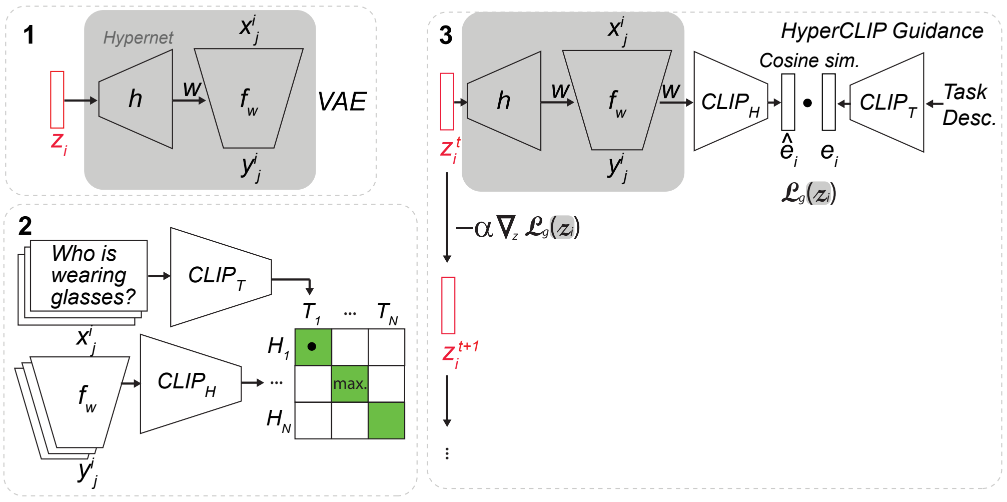

Figure 7: A comprehensive diagram of the steps involved for training and evaluating the HyperCLIP guidance technique. **(1)** Training an unconditional HVAE over the training tasks. **(2)** Training the HyperCLIP model according to the contrastive procedure described in Algorithm 1. **(3)** Exploring the latent space of the HVAE with gradient descent over the HyperCLIP guidance loss, as described in Section 4.2.

Training To train the HyperCLIP model, we need samples of fine-tuned network weights W_i. We use adaptations from **Uncond. HNet-MAML**, using 50-step adaptation $\mathcal{A}_{\mathcal{T}_i}$ with a learning rate 0.1, on randomly split support sets. We trained our HyperCLIP model for 600 epochs with the Adam (Kingma & 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, to perform guidance within either the HNet or HVAE latent spaces.

Evaluation For each held-out test task $\mathcal{T}_i$ from the Meta-VQA dataset, we perform a zero-shot model evaluation on the fixed predefined query set D_i^{q} for the task. Zero-shot performance is evaluated on the output of the generative hypernetwork h after applying latent space guidance.

Algorithm 9 HNet + HyperCLIP Training

```
Learn an unconditional hypernetwork $h(z^0, \theta)$ with the **Uncond. HNet-MAML** procedure from Algorithm 7.
Learn HyperCLIP network $\text{CLIP}_H(W)$ using the HyperCLIP training procedure from Algorithm 1. For sampling fine-tuned $W_i$, fine-tune the base network on training tasks.
```

Algorithm 10 HVAE + HyperCLIP Training

```
Learn an unconditional hypernetwork $h(z, \theta)$, as the decoder of a HVAE (Algorithm 8).
Learn HyperCLIP network $\text{CLIP}_H(W)$ using the HyperCLIP training procedure from Algorithm 1. For sampling fine-tuned $W_i$, fine-tune the base network on training tasks.
```

Algorithm 11 HyperCLIP guidance (Inference time)

```
Define a learned unconditional hypernetwork $h(z, \theta)$, as either a HNet $h(z^0, \theta)$ (Algorithm 7) or the decoder of a HVAE (Algorithm 8).
Define a learned HyperCLIP network $\text{CLIP}_H(W)$.
Define an unseen task $\mathcal{T}_i$ with natural language task descriptor $t_i$.
Randomly sample $z \sim \mathcal{N}(0, I)$ if using the decoder of a HVAE, or set $z = z^0$ where $z^0$ is the meta-learned embedding initialization of the Hnet.
Optimize $z$ with gradient descent over $\mathcal{L}_{\text{guidance}}(z)$ (Eq. 4), obtaining *guided* $z_i$.
Obtain *guided* base weights $W_i = h(z_i, \theta)$.
Use adapted base network $f$ with weights $W_i$ to classify examples from the unseen task $\mathcal{T}_i$.
```

A.7.3 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 randomly split support sets. We parametrize the diffusion process with a linear noise schedule, β starting at 0.0001 and ending at 0.06, and 350 diffusion timesteps. For all our experiments, we train the HyperLDM for 1000 epochs with the Adam optimizer, 0.00025 learning rate, and 128 epochs.

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.

Algorithm 12 HNet + HyperLDM Training

```
Learn an unconditional hypernetwork $h(z^0, \theta)$ with the **Uncond. HNet-MAML** procedure from Algorithm 7.
Learn the HyperLDM network $\epsilon_\psi(z^t, t, e_i)$ using the HyperLDM training procedure, optimizing reconstruction of $z_i^0$ with loss from Eq. 9. For sampling fine-tuned $z_i$, fine-tune the base network on training tasks, then encode the weights using the HNet.
```

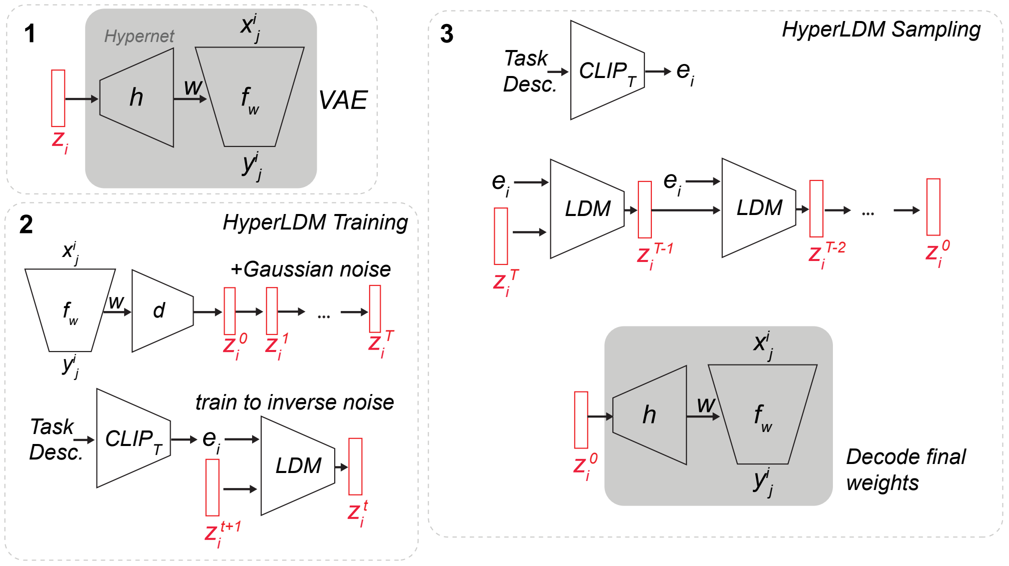

Figure 8: A comprehensive diagram of the steps involved for training and evaluating the HyperLDM model. **(1)** Training an unconditional HVAE over the training tasks. **(2)** Training the HyperLDM model as a conditional reversed diffusion process as described in Section 5.1. **(3)** Sampling from the conditional diffusion model.

Algorithm 13 HVAE + HyperLDM Training

```
Learn an unconditional hypernetwork $h(z, \theta)$, as the decoder of a HVAE (Algorithm 8).
Learn the HyperLDM network $\epsilon_\psi(z^t, t, e_i)$ using the HyperLDM training procedure, optimizing reconstruction of $z_i^0$ with loss from Eq. 9. For sampling fine-tuned $z_i$, fine-tune the base network on training tasks, then encode the weights using the HVAE.
```

Algorithm 14 HyperLDM Inference

```
Define a learned unconditional hypernetwork $h(z, \theta)$, as either a HNet $h(z^0, \theta)$ (Algorithm 7) or the decoder of a HVAE (Algorithm 8).
Define a learned HyperLDM network $\epsilon_\psi(z^t, t, e_i)$.
Define an unseen task $\mathcal{T}_i$ with natural language task descriptor $t_i$, with clip embedding $e_i$.
Randomly sample $z \sim \mathcal{N}(0, I)$.
Iteratively modify $z$ with diffusion sampling using the learned $\epsilon_\psi$ network, obtaining *guided* $z_i$.
Obtain *guided* base weights $W_i = h(z_i, \theta)$.
Use adapted base network $f$ with weights $W_i$ to classify examples from the unseen task $\mathcal{T}_i$.
```

Table 4: Few-Shot learning accuracy averaged over Meta-VQA test tasks. (* ours)

Method	**Few-Shot**
CLIP as Base Model	54.93 (± 0.11)
Uncond. Multitask	55.53 (± 0.40)
Uncond. MNet-MAML	**60.24** (± 0.84)
Uncond. MNet-FOMAML	60.03 (± 0.48)
Uncond. HNet-MAML	58.70 (± 0.10)
Cond. Multitask	59.46 (± 0.31)
Cond. HNet-MAML	59.48 (± 0.03)

A.8 Few-Shot Learning

For completeness, we include in Table 4 the results for few-shot learning on the test split of Meta-VQA. Our technique, unlike classic MAML, does not optimize specifically for the few-shot learning setting. Instead, the few-shot learning results are meant to contextualize performance gains in the zero-shot setting: zero-shot performance gains should be interpreted as relative to the few-shot accuracy ceiling of 60.24%, the maximum attained with our fixed choice of the base model.

For few-shot learning at test time, all adaptation is performed on the support set of the test tasks. For MAML baselines, we keep the same adaptation-time learning rate as during training, and we always adapt for 50 steps. For each multitask baseline, we use the same adaptation scheme (steps, learning rate, adapting parameters) as their MAML counterpart.