A. Derivations

A.1. Proof of Lemma 1

Proof. Under the assumptions that

$$\ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) := -\log p_\theta(\boldsymbol{y}|t, \boldsymbol{x});$$

and the prior $p(t|\boldsymbol{x})$ is a uniform distribution over t, the β-VAE objective can be written as

$$\begin{aligned}
&\mathcal{J}_{\beta\text{-VAE}}(\theta, q_\phi; \boldsymbol{x}, \boldsymbol{y}) := \\
&\quad \mathbb{E}_{q_\phi} \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x}) - \beta \mathrm{KL}(q_\phi(t)||p(t|\boldsymbol{x})) \\
&= -\mathbb{E}_{q_\phi} \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) - \beta \mathbb{E}_{q_\phi(t)} \log \frac{q_\phi(t)}{p(t|\boldsymbol{x})} \\
&= -\mathbb{E}_{q_\phi} \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) - \beta \mathbb{E}_{q_\phi(t)} \log q_\phi(t) \\
&\quad + \beta \mathbb{E}_{q_\phi(t)} \log p(t|\boldsymbol{x}) \\
&= -\left(\mathbb{E}_{q_\phi} \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) - \beta H(q_\phi)\right) + \beta \mathbb{E}_{q_\phi(t)} \log \frac{1}{T} \\
&= -\mathcal{L}(\theta, q_\phi; \boldsymbol{x}, \boldsymbol{y}) - \beta \log T.
\end{aligned}$$

Since the second term $-\beta \log T$ is a constant, maximizing $\mathcal{J}_{\beta\text{-VAE}}(\theta, q_\phi; \boldsymbol{x}, \boldsymbol{y})$ is equivalent to minimizing $\mathcal{L}(\theta, q_\phi; \boldsymbol{x}, \boldsymbol{y})$. □

A.2. Equivalence of reverse KL and maximum-entropy RL

The variational distribution q_ϕ actually depends on the input instance $\boldsymbol{x}$. For notation simplicity, we only write $q_\phi(t)$ instead of $q_\phi(t|\boldsymbol{x})$.

$$\min_\phi \mathrm{KL}(q_\phi(t)||q^*_\theta(t|\boldsymbol{y}, \boldsymbol{x})) \tag{A.1}$$

$$= \min_\phi -\sum_{t=1}^{T} q_\phi(t) \log q^*_\theta(t|\boldsymbol{y}, \boldsymbol{x})) - H(q_\phi) \tag{A.2}$$

$$= \min_\phi -\sum_{t=1}^{T} q_\phi(t) \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x})^\beta) - H(q_\phi) \tag{A.3}$$

$$+ \sum_{t=1}^{T} q_\phi(t) \log \sum_{\tau=1}^{T} p_\theta(\boldsymbol{y}|\tau, \boldsymbol{x})^\beta \tag{A.4}$$

$$= \min_\phi -\sum_{t=1}^{T} q_\phi(t) \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x})^\beta - H(q_\phi) \tag{A.5}$$

$$+ \sum_{t=1}^{T} q_\phi(t) C(\boldsymbol{x}, \boldsymbol{y}) \tag{A.6}$$

$$= \min_\phi -\sum_{t=1}^{T} q_\phi(t) \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x})^\beta - H(q_\phi) \tag{A.7}$$

$$+ C(\boldsymbol{x}, \boldsymbol{y}) \tag{A.8}$$

$$= \min_\phi -\sum_{t=1}^{T} q_\phi(t) \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x})^\beta - H(q_\phi) \tag{A.9}$$

$$= \max_\phi \sum_{t=1}^{T} q_\phi(t) \log p_\theta(\boldsymbol{y}|t, \boldsymbol{x})^\beta + H(q_\phi) \tag{A.10}$$

$$= \max_\phi \sum_{t=1}^{T} q_\phi(t) \beta \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) + H(q_\phi) \tag{A.11}$$

$$= \max_{\phi} \mathbb{E}_{t\sim q_\phi} \left[-\beta \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) - \log q_\phi(t)\right] \tag{A.12}$$

Define the action as $a_t \sim \pi_t = \pi_\phi(\boldsymbol{x}, \boldsymbol{x}_t)$, the reward function as

$$r(\boldsymbol{x}_t, a_t; \boldsymbol{y}) := \begin{cases} -\beta \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) & \text{if } a_t = 1 \text{ (i.e. stop)}, \\ 0 & \text{if } a_t = 0 \text{ (i.e. continue)}, \end{cases}$$

and the transition probability as

$$P(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, a_t) = \begin{cases} 1 & \text{if } \boldsymbol{x}_{t+1} = \mathcal{F}_\theta(\boldsymbol{x}_t) \text{ and } a_t = 0, \\ 0 & \text{else.} \end{cases}$$

Then the above optimization can be written as

$$\max_{\phi} \mathbb{E}_{t\sim q_\phi} \left[-\beta \ell(\boldsymbol{y}, \boldsymbol{x}_t; \theta) - \log q_\phi(t)\right] \tag{A.13}$$

$$= \max_{\phi} \mathbb{E}_{\pi_\phi} \sum_{t=1}^{T} r(\boldsymbol{x}_t, a_t; y) - \log \pi_t(a_t|\boldsymbol{x}, \boldsymbol{x}_t) \tag{A.14}$$

$$= \max_{\phi} \mathbb{E}_{\pi_\phi} \sum_{t=1}^{T} \left[r(\boldsymbol{x}_t, a_t; y) + H(\pi_t)\right]. \tag{A.15}$$

B. Experiment Details

B.1. Learning To Learn: Sparse Recovery

Synthetic data. We follow Chen et al. (2018) to choose $m = 250$, $n = 500$, sample the entries of A i.i.d. from the standard Gaussian distribution, i.e., $A_{ij} \sim \mathcal{N}(0, \frac{1}{m})$, and then normalize its columns to have the unit ℓ_2 norm. To generate $\boldsymbol{y}^*$, we decide each of its entry to be non-zero following the Bernoulli distribution with $p_b = 0.1$. The values of the non-zero entries are sampled from the standard Gaussian distribution. The noise $\boldsymbol{\epsilon}$ is Gaussian white noise. The signal-to-noise ratio (SNR) for each sample is uniformly sampled from 20, 30 and 40. For the testing phase, a test set of 3000 samples are generated, where there are 1000 samples for each noise level. This test set is fixed for all experiments in our simulations.

Evaluation metric. The performance is evaluated by NMSE (in dB), which is defined as $10\log_{10}(\frac{\sum_{i=1}^{N}\|\hat{\boldsymbol{x}}^i - \boldsymbol{x}^{*,i}\|_2^2}{\sum_{i=1}^{N}\|\boldsymbol{x}^{*,i}\|_2^2})$ where $\hat{\boldsymbol{x}}^i$ is the estimator returned by an algorithm or deep model.

B.2. Task-imbalanced Meta Learning

B.2.1. Details of setup

Hyperparameters We train MAML with batch size 16 on Omniglot imbalanced and batch size 2 on MiniImagenet imbalanced datasets. In both scenario we train with 60000 of mini-batch updates for the outer-loop of MAML. We report the results with 5 inner SGD steps for Omniglot imbalanced and 10 inner SGD steps for MiniImagenet imbalanced with other best hyperparameters suggested in (Finn et al., 2017), respectively. For MAML-stop we run 10 inner SGD steps for both datasets, with the inner learning rate to be 0.1 and 0.05 for Omniglot and MiniImagenet, respectively. The outer learning rate for MAML-stop is $1e^{-4}$ as we use batch size 1 for training.

When generating each meta-training dataset, we randomly select the number of observations within k_1 to k_2 for k_1-k_2-shot learning. The number of observations in test set is always kept the same within each round of experiment.

B.2.2. Memory efficient implementation

As our MAML-stop allows the automated decision of optimal stopping, it is preferable that the maximum number of SGD updates per each task is set to a larger number to fully utilize the capacity of the approach. This brings the challenge during training, as the loss on each meta-test set during training is required for *each single* inner update step. That is to say, if we allow maximumly 10 steps of inner SGD update, then the memory cost for running CNN prediction on meta-test set is 10x larger than vanilla MAML. Thus a straightforward implementation will not give us a feasible training mechanism.

To make the training of MAML-stop feasible on a single GPU, we utilize the following techniques:

- We use stochastic EM for learning the predictive model, as well as the stopping policy. Specifically, we sample $t \sim q_\theta^*(\cdot|\boldsymbol{y}, \boldsymbol{x})$ in each round of training, and only maximize $p_\theta(\boldsymbol{y}|t, \boldsymbol{x})$ in this round.
- As the auto differentiation in PyTorch is unable to distinguish between 'no gradient' and 'zero gradient', it causes extra storage for the unnecessary gradient computation. To overcome this, we first calculate $q_\theta^*(t|\boldsymbol{y}, \boldsymbol{x})$ for each t without any gradient storage (which corresponds to `no_grad()` in PyTorch), then recompute $p_\theta(\boldsymbol{y}|t, \boldsymbol{x})$ for the sampled t.

With the above techniques, we can train MAML-stop almost as (memory) efficient as MAML.

B.2.3. STANDARD META-LEARNING TASKS

For completeness, we also include the MAML-stop in the standard setting of few-shot learning. We mainly compared with the vanilla MAML for the sake of ablation study.

Hyperparameters The hyperparameter setup mainly follows the vanilla MAML paper. For both MAML and MAML-stop, we use the same batch size, number of training epochs and the learning rate. For Omniglot 20-way experiments and MiniImagenet 5-way experiments, we tune the number of unrolling steps in $\{5, 6, \ldots, 10\}$, β in $\{0, 0.1, 0.01, 0.001\}$ and the learning rate of inner update in $\{0.1, 0.05\}$. We simply use grid search with a random held-out set with 600 tasks to select the best model configuration.

B.3. Image Denoising

B.3.1. IMPLEMENTATION DETAILS

When training the denoising models, the raw images were cropped and augmented into 403K $50 * 50$ patchs. The training batch size was 256. We used Adam optimizer with the initial learning rate as $1e-4$. We first trained the deep learning model with the unweighted loss for 50 epochs. Then, we further train the model with the weighted loss for another 50 epoches. After hyper-parameter searching, we set the exploration coefficient β as 0.1. When training the policy network, we used the Adam optimizer with the learning rate as $1e-4$. We reused the above hyper-parameters during joint training.

B.3.2. VISUALIZATION

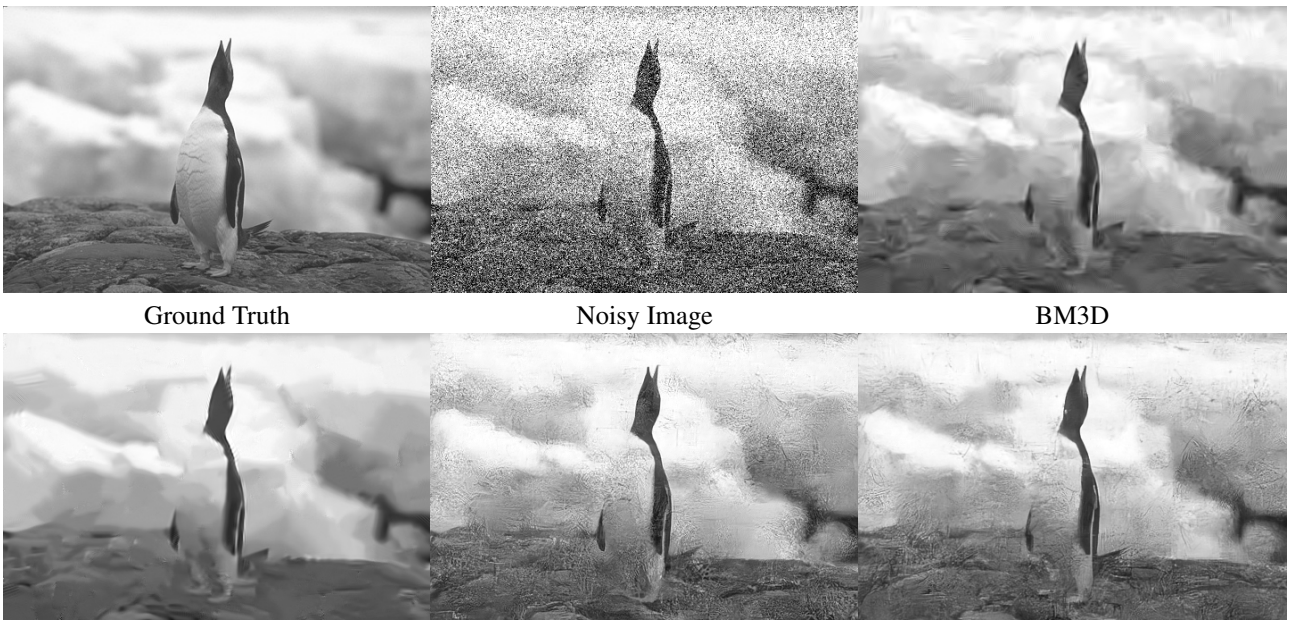

Figure B.1. Denoising results of an image with noise level 65.