# A APPENDIX

## A.1 EXPERIMENTAL DETAILS

For all of the experiments and all the baselines, we fix the architecture of the meta-learner $f_\theta$ to be a multi-layer perceptron with 3 hidden layers of 64 hidden units together with ReLU activations. We use a meta batch size of 10 tasks and train all the models with Adam Kingma & Ba (2014) optimizer with a learning rate of $10^{-3}$. We use the inner learning rate $\alpha = 0.1$ for the adaptation step and MSE as the adaptation loss. All experiments were run for 5 different seeds to compute mean and standard deviations. For LLAMA we use $\eta = 10^{-6}$, for PLATIPUS we scale the KL loss by 0.1, for BMAML we use 10 particles and use MSE rather than the chaser loss for a fair comparison. Other experiment-specific details include:

- **Sine and Polynomial** regression: models are trained for $10^5$ outer loss steps, and the dimension of the context used is equal to the task's dimensionality.
- **Mass-Spring**: models are trained for $10^6$ outer loss steps, the dimension of the context used is 6, the ODE dynamics are defined as:

$$m\frac{d^2y}{dt^2} + b\frac{dy}{dt} + ky = 1 \tag{14}$$

  where $k, m, b$ are sampled uniformly in $[0.5, 2.0]$.
- **Double Pendulum**: models are trained for $10^6$ outer loss steps, the dimension of the context used is 4. The dynamics are described through the canonical Hamiltonian equations as:

$$\dot{\alpha}_1 = \frac{p_1 l_2 - p_2 l_1 \cos(\alpha_1 - \alpha_2)}{l_1^2 l_2 [m_1 + m_2 \sin^2(\alpha_1 - \alpha_2)]} \tag{15}$$

$$\dot{\alpha}_2 = \frac{p_2(m_1 + m_2) l_1 - p_1 m_2 l_2 \cos(\alpha_1 - \alpha_2)}{m_2 l_1 l_2^2 [m_1 + m_2 \sin^2(\alpha_1 - \alpha_2)]} \tag{16}$$

$$\dot{p}_1 = -(m_1 + m_2) g l_1 \sin\alpha_1 - A_1 + A_2 \tag{17}$$

$$\dot{p}_2 = -m_2 g l_2 \sin\alpha_2 + A_1 - A_2 \tag{18}$$

  with

$$A_1 = \frac{p_1 p_2 \sin(\alpha_1 - \alpha_2)}{l_1 l_2 [m_1 + m_2 \sin^2(\alpha_1 - \alpha_2)]} \tag{19}$$

$$A_2 = \frac{p_1^2 m_2 l_2^2 - 2 p_1 p_2 m_2 l_1 l_2 \cos(\alpha_1 - \alpha_2) + p_2^2 (m_1 + m_2) l_1^2}{2 l_1^2 l_2^2 [m_1 + m_2 \sin^2(\alpha_1 - \alpha_2)]^2} \tag{20}$$

  for angles $\alpha_1, \alpha_2$ between the pendulums and generalized momenta $p_1, p_2$. From this, we find the coordinate $(x_2, y_2)$ of tip of the pendulum as

$$x_2 = l_1 \sin(\alpha_1) + l_2 \sin(\alpha_2) \tag{21}$$

$$y_2 = -l_1 \cos(\alpha_1) - l_2 \cos(\alpha_2). \tag{22}$$

  We sample parameters as $m_1, m_2 \sim \mathcal{U}[0.5, 1.5]^2$ and $g \sim \mathcal{U}[5, 15]$. The pendulum lengths $l_1, l_2$ are both constants with a value of 2.0.
- **Multi Mass-Spring**: models are trained for $10^6$ outer loss steps, the dimension of the context used equals 8. The dynamics can be simulated by iterating through each of the particles and summing the forces applied by the other particles:

$$v_i = v_i + \Delta t \frac{F_i}{m} \tag{23}$$

$$x_i = x_i + \Delta t v_i \tag{24}$$

  where $v_i, x_i$ denotes the velocity and position of particle $i$ respectively and $m$ denotes the mass. $F_i$ denotes the total spring force acted upon particle $i$:

$$F_i = \sum_{j=1}^{n} F_{ij} \tag{25}$$

$$F_{ij} = K_{ij}(1 - \|x_i - x_j\|) \frac{x_i - x_j}{\|x_i - x_j\|}. \tag{26}$$

**Algorithm 1** LAVA Pseudo-Code

```
Require: $p(\mathcal{T})$ distribution of tasks
Require: $\alpha, \eta, \epsilon$ hyperparameters.
Ensure: Output results

Randomly initialize $\theta_0$
while not done do
    Sample batch of tasks $\mathcal{T} \sim p(\mathcal{T})$
    for all $\tau \in \mathcal{T}$ do
        Sample $D^S_\tau, D^Q_\tau \sim \tau$
        for all $(x_i, y_i) \in D^S_\tau$ do
            Evaluate $\hat{\theta}_i = \theta_0 - \alpha \nabla_\theta \mathcal{L}(\theta_0, x_i, y_i)$
            Evaluate $H_i = \frac{d^2}{d\theta^2} \mathcal{L}(\hat{\theta}_i, x_i, y_i)$
            Evaluate $\tilde{H}_i = \frac{1}{1+\epsilon} (H_i + \epsilon I)$
        end for
        Evaluate $\tilde{H} = \sum_i \tilde{H}_i$
        Evaluate $\hat{\theta}_\tau = \tilde{H}^{-1} \sum_i \tilde{H}_i \hat{\theta}_i$
    end for
    Update $\theta_0 = \theta_0 - \eta \nabla_{\theta_0} \sum_{\tau \in \mathcal{T}} \mathcal{L}(\hat{\theta}_\tau, D^Q_\tau)$ using each $D^Q_\tau$
end while
```

where $F_{ij}$ denotes the force acted upoin particle $i$ by particle $j$. We sample spring constant $K_{ij} = K_{ji}$ between particle $i, j$ uniformly in $[0.5, 2.0]$. We let mass $m = 1$ and use a total of 4 particles.

- **Omnipush**: models are trained for $2 \times 10^5$ outer loss steps, and the dimension of the context used is 8.

### A.2 GRADIENT-BASED META-LEARNING IS AN UNBIASED ESTIMATOR

Here, we show that GBML is an unbiased estimator. Define the loss for one task as:

$$\mathcal{L}(\theta, \tau) = \underset{x \sim p(x|\tau)}{\mathbb{E}} [\mathcal{L}(\theta, x)] \tag{27}$$

Then the gradient w.r.t $\theta$ is an unbiased estimator:

$$\underset{x \sim p(x|\tau)}{\mathbb{E}} [\nabla_\theta \mathcal{L}(\theta, x)] = \nabla_\theta \underset{x \sim p(x|\tau)}{\mathbb{E}} [\mathcal{L}(\theta, x)] = \nabla_\theta \mathcal{L}(\theta, \tau) \tag{28}$$

Moreover, we measure empirically the bias of GBML and LAVA estimators. As a comparison, we include a fully learned network implemented as a HyperNetwork Ha et al. (2016) that takes as input the entire support dataset and outputs the adapted parameters directly. Both the adaptation and the aggregation are learned end-to-end together with the downstream task.

We train these three models until convergence on the sine regression dataset. Then, we measure their performance on each task corrupted by Gaussian noise with a standard deviation of 3 on the support labels. The experiment is designed to test how the performance changes when increasing the support size. Figure 4 shows the difference in the loss between adaptation with and without noise for the three models and for different support sizes. Thus, we are effectively testing the ability of these estimators to recover the performances of the noiseless adaptation. Ideally, an unbiased estimator converges to the correct posterior with enough samples as long as the noise has zero mean. As can be seen in the figure, GBML methods are much more robust to these kinds of perturbations, while learned networks are not unbiased.

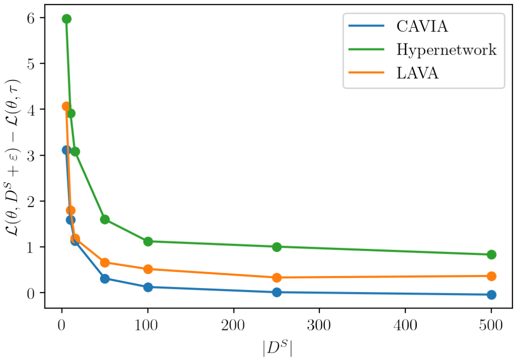

Figure 4: Scaled MSE when adding noise to the support labels for the sine experiment with increasing support size.

### A.3 VARIANCE REDUCTION

Below we give an account of the proof of Proposition 1. Consider the variance reduction problem defined in Equation 12

$$\min_{W} \quad \text{Var}\left(\sum_{i=1}^{n} W_i Z_i\right) \quad \textit{subject to} \quad \sum_{i=1}^{n} W_i = I \tag{29}$$

We have that

$$\text{Var}\left(\sum_{i=1}^{n} W_i Z_i\right) = \sum_{i=1}^{n} W_i \Sigma_i W_i^T \tag{30}$$

By introducing a Lagrange multiplier $\lambda$, we reach the following optimization problem:

$$\min_{W} \quad F(Z, W) \tag{31}$$

$$F(Z, W) = \sum_{i=1}^{n} W_i \Sigma_i W_i^T + \lambda \left(\sum_{i=1}^{n} W_i - I\right) \tag{32}$$

By taking the derivative w.r.t $W_i$ and using the fact that $\Sigma_i$ is symmetric we find that

$$\frac{dF}{dW_i} = 2W_i\Sigma_i - \lambda \tag{33}$$

Setting this equal to $0$, we get

$$W_i = \frac{\lambda}{2}\Sigma_i^{-1} \tag{34}$$

From the condition defined in 29, we have

$$\frac{\lambda}{2}\sum_{i=1}^{n}\Sigma_i^{-1} = I \tag{35}$$

$$\lambda = 2\left(\sum_{i=1}^{n}\Sigma_i^{-1}\right)^{-1} \tag{36}$$

$$\tag{37}$$

Plugging this into Equation 34, it follows that

$$W_i = \left(\sum_{i=1}^{n}\Sigma_i^{-1}\right)^{-1}\Sigma_i^{-1} \tag{38}$$

Given these weights, $W_i$, the distribution of $\sum_{i=1}^{n} W_i Z_i$ follows a normal distribution which is equivalent to Equation 10. $\square$

| Models | $\lvert D^S \rvert = 5$ | $\lvert D^S \rvert = 10$ | $\lvert D^S \rvert = 20$ |
|---|---|---|---|
| MAML | $0.834 \pm 0.129$ | $0.518 \pm 0.071$ | $0.316 \pm 0.042$ |
| meta-SGD | $0.63 \pm 0.08$ | $0.35 \pm 0.05$ | $0.20 \pm 0.02$ |
| ANIL | $0.72 \pm 0.09$ | $0.41 \pm 0.05$ | $0.23 \pm 0.03$ |
| PLATIPUS | $0.921 \pm 0.165$ | $0.386 \pm 0.064$ | $0.134 \pm 0.024$ |
| CAVIA | $0.651 \pm 0.105$ | $0.372 \pm 0.059$ | $0.199 \pm 0.029$ |
| LLAMA | $0.647 \pm 0.112$ | $0.371 \pm 0.058$ | $0.199 \pm 0.028$ |
| BMAML | $0.599 \pm 0.120$ | $0.349 \pm 0.075$ | $0.190 \pm 0.035$ |
| VFML | $0.64 \pm 0.08$ | $0.38 \pm 0.05$ | $0.21 \pm 0.02$ |
| MetaMix | $0.76 \pm 0.10$ | $0.67 \pm 0.09$ | $0.58 \pm 0.09$ |
| **LAVA** | $\mathbf{0.047 \pm 0.020}$ | $\mathbf{0.016 \pm 0.003}$ | $\mathbf{0.010 \pm 0.002}$ |

Table 3: Test MSE on sine regression with different support sizes.

# B ADDITIONAL EXPERIMENTS

## B.1 ADDITIONAL SINE RESULTS

Here we provide additional results for the sine regression experiment. Using the same experimental settings described in Section 6.1 and Appendix A.1, we present MSE and standard deviations for 5 seeds for LAVA and baselines in Table 3. Additional qualitative results are shown in Figure 5

## B.2 ADDITIONAL COMPUTATIONAL TIME EXPERIMENTS

We present additional results on computational times for the polynomial experiment in Figure 6 for different number of degrees and size of support.

## B.3 MINI-IMAGENET

We further experiment with classification on the Mini-Imagenet dataset Vinyals et al. (2016). We use the training-set split as used in Ravi & Larochelle (2017) which leaves 64 classes for training, 16 for validation and 20 for test. We experiment with 5-way classification in either a 1-shot or 5-shot setting. We train the models for 1000 epochs and perform model selection by choosing the one with the best performance on the validation set. We present results on the test set in Table 4. In the 1-shot setting, we achieve results comparable to Meta-SGD and ANIL, outperforming Bayesian alternatives such as BMAML. In the 5-shot classification, we find that our model outperforms BMAML while being within the error bars of standard GBML methods such as CAVIA, MAML and Meta-SGD.

Standard classification benchmarks such as Mini-Imagenet test the capability of the model to incorporate high-dimensional data in the form of images. Some of the methods, such as the best-performing method BOIL, are optimized towards image data and attempt to efficiently learn a well-structured representation space of images, such that the adaptation reduces to modifying decision boundaries. In particular, few-shot image classification problems in this form are inherently discrete problems that do not suffer as extensively from the task-overlap assumption as outlined in 2.

## B.4 A NOTE ON COMPUTATIONAL COMPLEXITY

A limitation of the described method lies in an increased time complexity. Computing the Hessian on each single support point can, in fact, severely affect the training time of methods that already require complex second-order calculations like in GBML. As a first consideration, we point out that the Hessian is computed on the contextual parameters only, leading to a more efficient computation rather than the full model parameters. When the network is expressive enough, this should lead to no difference in performance over the full adaptation framework.

Nevertheless, the Hessian computation can result in a sensible increase in computational time and LAVA is, in fact, more expensive than the standard GBML model. However, this computational complexity increase is paired with stronger performances. LAVA provides a more effective adaptation

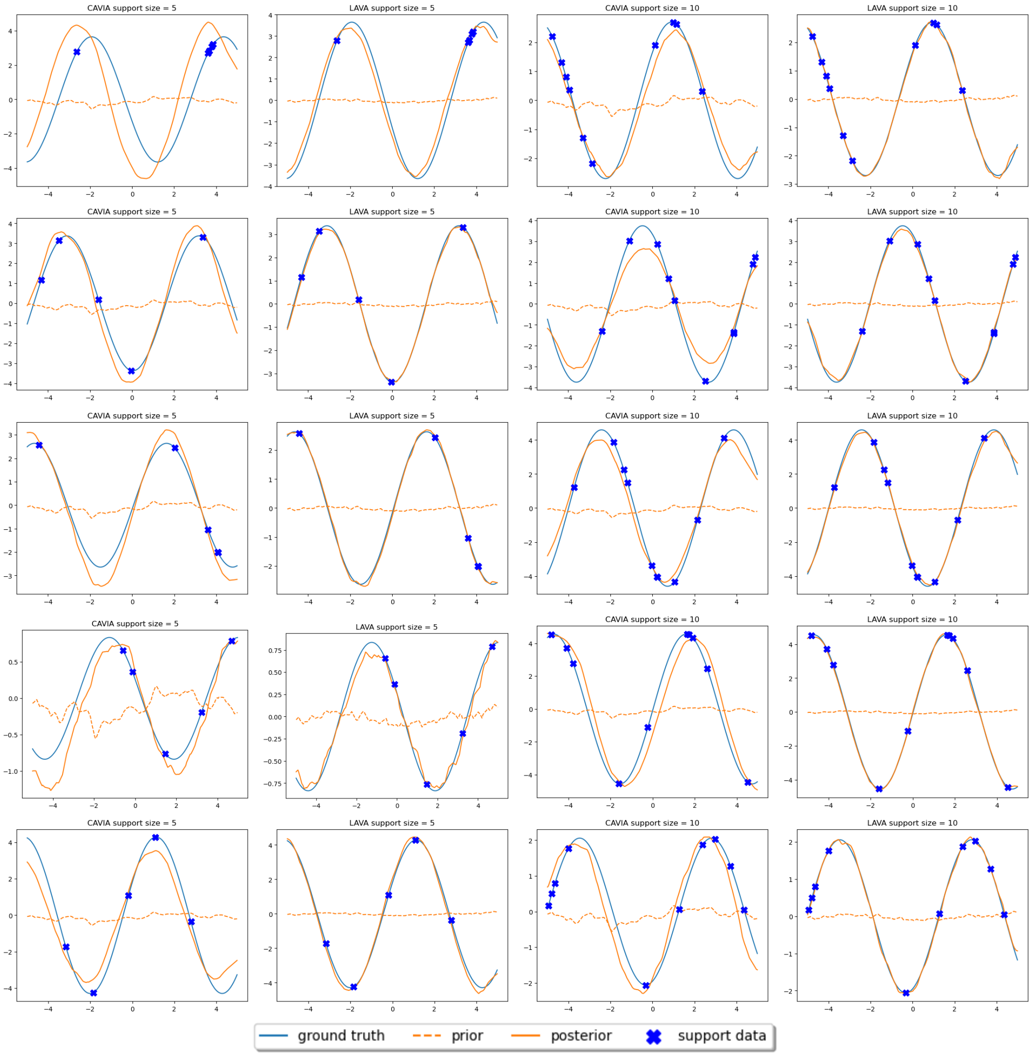

Figure 5: Qualitative results for the sine experiment for CAVIA and LAVA with 5 and 10 support size

| Model | 5-ways 1-shot | 5-ways 5-shot |
|---|---|---|
| ANIL | $45.94 \pm 0.94$ | $62.86 \pm 0.26$ |
| BMAML | $42.43 \pm 7.44$ | $59.76 \pm 0.43$ |
| BOIL | $50.12 \pm 0.33$ | $64.72 \pm 0.40$ |
| CAVIA | $47.84 \pm 0.41$ | $63.09 \pm 0.51$ |
| LLAMA | $40.19 \pm 0.85$ | $56.50 \pm 0.15$ |
| MAML | $\mathbf{48.60 \pm 0.80}$ | $\mathbf{63.19 \pm 1.57}$ |
| Meta-SGD | $46.18 \pm 0.45$ | $62.82 \pm 0.36$ |
| PLATIPUS | $34.71 \pm 0.68$ | $42.84 \pm 0.99$ |
| LAVA | $46.69 \pm 1.45$ | $61.51 \pm 0.97$ |

Table 4: Results Mini-Imagenet with support sizes 1 and 5

technique as one of its main features is the efficient use of the limited information given by the support. In this regard, LAVA provides a better trade-off between performances and computational complexity. To better analyze this trade-off we compared LAVA's performances against CAVIA in the Polynomial regression experiment by varying the number of inner loop adaptation steps. Figure 7

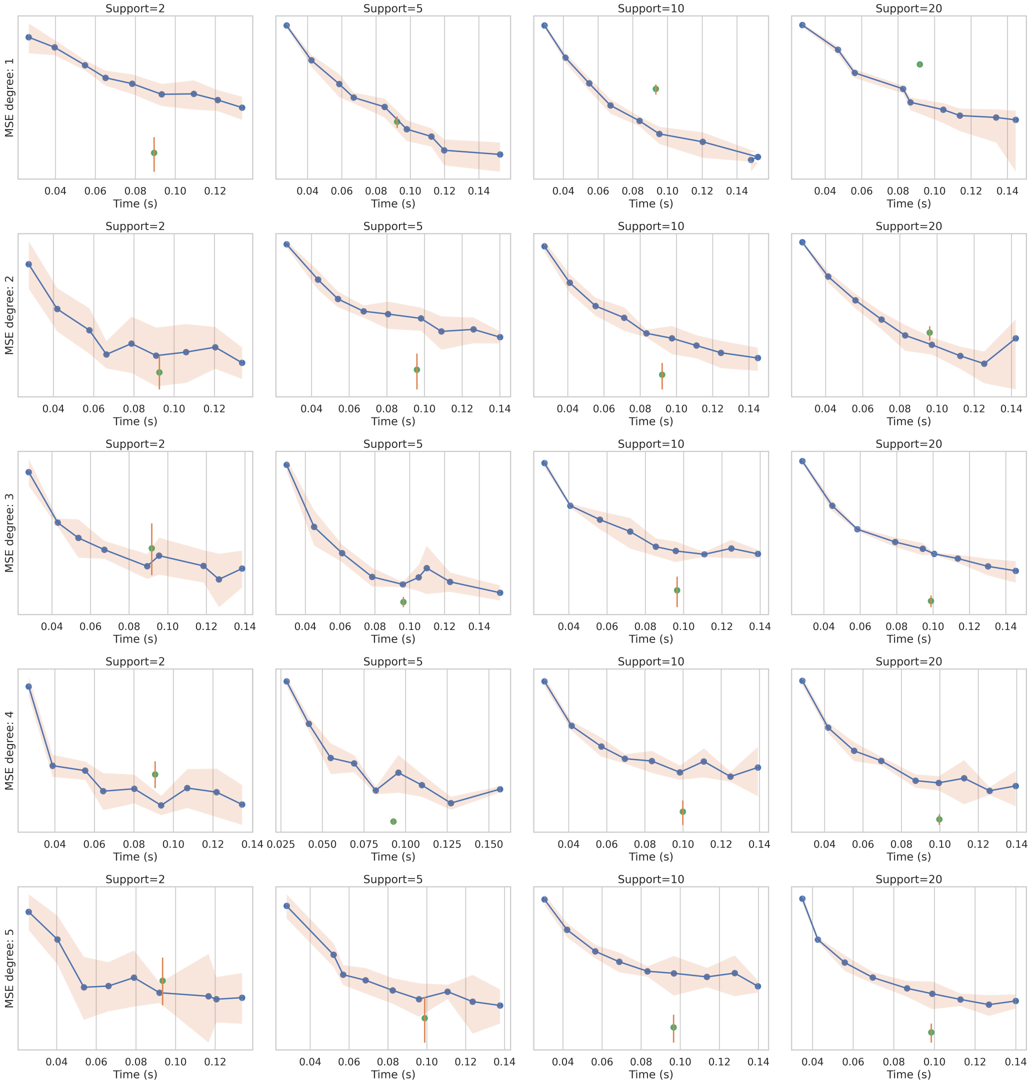

Figure 6: Computational times for the polynomial experiment of various degrees.

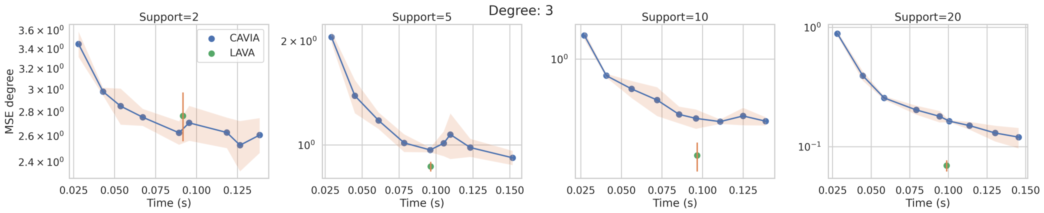

Figure 7: Computational times for third-degree polynomials with different sizes of support. The blue line represents results for CAVIA over a different number of adaptation steps. In comparison, LAVA achieves a computational time comparable to 6 inner-steps of CAVIA, while showcasing superior performance.

shows how LAVA has a computation complexity comparable to CAVIA with between $5$ and $6$ inner loop gradient steps. However, the performances of LAVA are consistently lower. A more extensive analysis is given in the Appendix in Figure 6.

There exists an inherent trade-off between computational complexity and model performance. Recent work exemplifies how deep learning modules benefit from more computational complexity, an example being the shift to Transformers from CNNs in computer vision tasks. LAVA represents another step in this direction as one of its main features is the efficient use of the limited information given by the support size at the expense of complex computations.