A APPENDIX

A.1 QUADRATIC SETTING

Meta Mirror Descent is deployed on a family of 2D quadratic optimisation problems from which we can sample a disjoint set of meta-training and meta-testing optimisation problems. We sample tasks of the form:

$$\min_{\theta} \theta^T Q \theta - b^T \theta$$

where Q and b are random variables. b follows a Gaussian distribution with mean vector $[1, 1]^T$ and identity covariance. To generate Q which controls the flatness of the quadratic function, we sample $Q_{0,0}$ and $Q_{1,1,}$ independently with different Gaussian distributions. For example on the top row in Figure 1: the mean of $Q_{0,0}$ and $Q_{1,1}$ are 0.3 and 14 while on the bottom row, the mean of $Q_{0,0}$ and $Q_{1,1}$ are 0.02 and 14 respectively. In addition, the initialisation of each optimisation problem is also a random variable.

A.2 DERIVE OF THE CLOSED FORM MIRROR LOOP

In our setting, the mirror loop is described as

$$\theta_{t+1} = \arg\min_{\theta} \langle \nabla_\theta \mathcal{L}(\theta_t), \theta \rangle + \frac{1}{2\eta} B_\phi(\theta || \theta_t) \tag{14}$$

or, equivalently,

$$\theta_{t+1} = \arg\min_{\theta} \eta \langle \nabla_\theta \mathcal{L}_{tr}(\theta_t), \theta \rangle + B_\phi(\theta || \theta_t). \tag{15}$$

Setting the gradient w.r.t. θ to zero, we have

$$\eta \nabla \mathcal{L}(\theta_t) + \nabla \phi(\theta_{t+1}) - \nabla \phi(\theta_t) = 0, \tag{16}$$

which when rearranged yields

$$\nabla \phi(\theta_{t+1}) = \nabla \phi(\theta_t) - \eta \nabla \mathcal{L}(\theta_t) \tag{17}$$

$$\theta_{t+1} = \nabla \phi^{-1}(\nabla \phi(\theta_t) - \eta \nabla \mathcal{L}(\theta_t)). \tag{18}$$

In our case

$$\phi(\theta) = \frac{1}{2} \theta^T M \theta, \tag{19}$$

where M is a block diagonal matrix. Therefore,

$$\nabla \phi(\theta) = M\theta \tag{20}$$

$$\nabla \phi^{-1}(\theta) = M^{-1}\theta. \tag{21}$$

As a result, we also have

$$\theta_{t+1} = \nabla \phi^{-1}(\nabla \phi(\theta_t) - \eta \nabla \mathcal{L}(\theta_t)) \tag{22}$$

$$= M^{-1}(M\theta_t - \eta \nabla \mathcal{L}(\theta_t)) \tag{23}$$

$$= \theta_t - \eta M^{-1} \nabla \mathcal{L}(\theta_t). \tag{24}$$

A.3 PROOF OF THEOREM 4.1

We will make use of a well-known bound on suprema of empirical processes due to Bartlett & Mendelson (2002).

Theorem A.1. *For all functions $f \in \mathcal{F}$ with $\|f\|_\infty \leq a$, and i.i.d. Z_i, the following holds with probability at least $1 - \delta$,*

$$\mathbb{E}[f(Z)] \leq \frac{1}{n} \sum_{i=1}^{n} f(Z_i) + 2\hat{R}_n(\mathcal{F}) + 3a\sqrt{\frac{ln(2/\delta)}{2n}}. \tag{25}$$

In this theorem, $\hat{R}_n(\mathcal{F})$ is the empirical Rademacher complexity of the class $\mathcal{F}$, defined as

$$\hat{R}_n(\mathcal{F}) = \mathbb{E}_\epsilon \left[\sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i f(Z_i) \right], \tag{26}$$

where ϵ_i is a Rademacher random variable, so $P(\epsilon_i = -1) = P(\epsilon_i = 1) = 0.5$.

Proof. It suffices to bound, with high confidence, the difference between the first term of the meta-objective, and the expected Bregman divergence between initializations and solutions on new tasks sampled from the same task distribution. We will obtain such a bound using Rademacher complexity, and the main result will follow from standard applications of Rademacher complexity-based generalisation bounds (Bartlett & Mendelson, 2002), along with the observation that $B_\phi(\theta_* || \theta_1) \leq \frac{Cr^2}{2}$. That is, we will treat the initial and final points of an optimisation trajectory for each task as a random variable, and we will bound to what extent the mean Bregman divergence between initial and trained parameters on some meta-train tasks can deviate from the expected Bregman divergence on unseen tasks. In particular, we analyse the following class:

$$\mathcal{F} = \{(\theta_*, \theta_1) \mapsto B_\phi(\theta_* || \theta_1) \,:\, \phi(\theta) = \frac{1}{2}\theta^T M \theta,\ \|M - I\|_F \leq C\}. \tag{27}$$

Denoting $\theta_*^{(i)} - \theta_1^{(i)}$ by $\tilde{\theta}_i$, we can bound the Rademacher complexity of this class from above by

$$\begin{aligned}
\hat{R}_n(\mathcal{F}) &= \mathbb{E}_\epsilon \left[\sup_{B_\phi \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \epsilon_i B_\phi(\theta_*^{(i)} || \theta_1^{(i)}) \right] && (28)\\
&= \frac{1}{2n} \mathbb{E}_\epsilon \left[\sup_{M} \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i^T M \tilde{\theta}_i \right] && (29)\\
&= \frac{1}{2n} \mathbb{E}_\epsilon \left[\sup_{M} \sum_{i=1}^{n} \epsilon_i \langle M, \tilde{\theta}_i \tilde{\theta}_i^T \rangle_F \right] && (30)\\
&= \frac{1}{2n} \mathbb{E}_\epsilon \left[\sup_{M} \langle M, \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \rangle_F \right] && (31)\\
&= \frac{1}{2n} \mathbb{E}_\epsilon \left[\sup_{M} \langle (M - I), \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \rangle_F \right] + \frac{1}{2n} \mathbb{E}_\epsilon \left[\langle I, \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \rangle_F \right], && (32)
\end{aligned}$$

where $\langle \cdot, \cdot \rangle_F$ is the Frobenius inner product. Note that the second term in the final equality is equal to zero. As such we can continue by further bounding the first term using the Cauchy-Schwarz inequality,

$$\hat{R}_n(\mathcal{F}) \leq \frac{C}{2n} \mathbb{E}_\epsilon \left[\left\| \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \right\|_F \right] \tag{33}$$

The remainder of the proof follows a well-known sequence of steps using when bounding the expected norm of a Rademacher sum (see, e.g., (Shalev-Shwartz & Ben-David, 2014)), which we include here for completeness. Jensen's inequality tells us that

$$\begin{aligned}
\mathbb{E}_\epsilon \left[\left\| \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \right\|_F \right] &\leq \sqrt{\mathbb{E}_\epsilon \left[\left\| \sum_{i=1}^{n} \epsilon_i \tilde{\theta}_i \tilde{\theta}_i^T \right\|_F^2 \right]} && (34)\\
&= \sqrt{\mathbb{E}_\epsilon \left[\sum_{i=1}^{n} \sum_{j=1}^{n} \epsilon_i \epsilon_j \langle \tilde{\theta}_i \tilde{\theta}_i^T, \tilde{\theta}_j \tilde{\theta}_j^T \rangle_F \right]}. && (35)
\end{aligned}$$

Noting that $\mathbb{E}[\epsilon_i \epsilon_j]$ is one if $i = j$ and zero otherwise, we obtain

$$\sqrt{\mathbb{E}_\sigma \left[\sum_{i=1}^{n} \sum_{j=1}^{n} \epsilon_i \epsilon_j \langle \tilde{\theta}_i \tilde{\theta}_i^T, \tilde{\theta}_j \tilde{\theta}_j^T \rangle_F \right]} \leq \sqrt{\sum_{i=1}^{n} \|\tilde{\theta}_i \tilde{\theta}_i^T\|_F^2} \tag{36}$$

$$\leq \sqrt{\sum_{i=1}^{n} (\|\tilde{\theta}_i\|_2 \|\tilde{\theta}_i\|_2)^2} \tag{37}$$

$$\leq \sqrt{nr^4}. \tag{38}$$

Substituting this back into our earlier derivation yields

$$\hat{R}_n(\mathcal{F}) \leq \frac{C\sqrt{nr^4}}{2n} \tag{39}$$

$$= \frac{Cr^2}{2\sqrt{n}}, \tag{40}$$

which concludes the proof. □

A.4 Meta Regulariser Computation

We provide the meta regualriser computation details in this section. The proposed meta regulariser is expressed as

$$||M - I||_F^2 = ||M||_F^2 + || - I||_F^2 + 2\langle M, -I \rangle_F \tag{41}$$

$$= ||M||_F^2 + ||I||_F^2 - 2\text{tr}(M) \tag{42}$$

$$= ||M||_F^2 + ||I||_F^2 - 2\text{tr}(A)\text{tr}(B) \tag{43}$$

$$= \sum_{i,j} \upsilon_i(A)\upsilon_j(B) + ||I||_F^2 - 2\text{tr}(A)\text{tr}(B), \tag{44}$$

where $\upsilon_i(\cdot)$ denotes the i-th singular value of a matrix. To get from Eq. 42 to Eq. 43 we use that $\text{tr}(A \otimes B) = \text{tr}(A)\text{tr}(B)$, and from Eq. 43 to Eq. 44 we use that $\|M\|_F^2 = \sum_i \upsilon_i^2(A \otimes B) = \sum_{i,j} \upsilon_i(A)\upsilon_j(B)$.

A.5 Computing λ

In this section, we provide the computation details for λ, as used in Equation 8:

$$\lambda = \min_\zeta \upsilon_{min}(M_\zeta) \tag{45}$$

$$= \min_\zeta \upsilon_{min}(A_\zeta \otimes B_\zeta) \tag{46}$$

$$= \min_\zeta \min_{i,j} (\upsilon_i(A_\zeta)\upsilon_j(B_\zeta)) \tag{47}$$

where $\upsilon_i(\cdot)$ denotes the i-th singular value of a matrix. From Eq. 46 to Eq. 47, we use that $\upsilon(A \otimes B) = \upsilon(A) \cdot \upsilon(B)^T$, where $\upsilon(\cdot)$ gives all the eigenvalues in the column vector form.

A.6 Gradient Computation for Diagonal Matrix

Diagonal matrix is a special case of block diagonal matrix, which we applied in the 2D quadratic setting in section 5.2. With this simple parameterisation, the exact hypergradient w.r.t. the target optimiser can be computed with Forward Mode Differential (FMD). The gradient of the second term in the proposed meta-objective in Eq.8 is easy to compute while the first term with respect to ϕ is expressed as:

$$\frac{\partial B_{\phi_M}(\theta_*, \theta_1)}{\partial M} \approx \frac{\partial B_{\phi_M}(\theta_T, \theta_1)}{\partial M} \tag{48}$$

$$= \underbrace{\frac{\partial B_{\phi_M}(\theta_T, \theta_1)}{\partial M}}_{\text{direct gradient}} + \underbrace{\frac{\partial B_{\phi_M}(\theta_T, \theta_1)}{\partial \theta_T} \frac{\partial \theta_T}{\partial M}}_{\text{indirect gradient}} \tag{49}$$

when T is large enough to satisfy that $\theta_* \approx \theta_T$. The computation of the direct gradient can be easily solved by the existing auto-differentiation library. The indirect gradient in Eq 49, usually termed hypergradient, is much more computationally chanllenging as it is expressed in the form:

$$\frac{\partial \theta_T}{\partial M} = \sum_{t=1}^{T} \left(\prod_{t'=t+1}^{T} A_{t'} \right) B_t \tag{50}$$

$$\text{s.t. } A_t = \frac{\partial \pi_{\phi_M}(\theta_{t-1})}{\partial \theta_{t-1}} = I - \eta M^{-1} \frac{\partial^2}{\partial \theta^2} \mathcal{L}_{tr}(\theta_{t-1}), \tag{51}$$

$$B_t = \frac{\partial \pi_{\phi_M}(\theta_{t-1})}{\partial M} = 2\eta \frac{\partial}{\partial \theta} \mathcal{L}_{tr}(\theta_{t-1}) M^{-2}. \tag{52}$$

Forward-Mode Differentiation (FMD) and Reverse-Mode Differentiation Franceschi et al. (2017) are two algorithms to compute Eq 50. RMD computes the gradient from the last to the initial step, requiring one to store the entire optimisation trajectory in memory. When the optimisation trajectory is long, this computation is very expensive. In comparison, FMD updates the hypergradient in parallel with in inner loop optimisation by:

$$\frac{\partial \theta_t}{\partial M} = \frac{\partial \pi_{\phi_M}(\theta_{t-1})}{\partial \theta_{t-1}} \frac{\partial \theta_{t-1}}{\partial M} + \frac{\partial \pi_{\phi_M}(\theta_{t-1})}{\partial M}, \tag{53}$$

where it only requires the information from step $t-1$.

A.7 Hyperparameter Tuning

Grid Search For tuning the hyperparameters on 3-layer MLPs model settings in section 5.2, we sweep over the learning rates $\{0.1, 0.05, 0.01, 0.005, 0.001\}$ and weight decay parameters of $\{0.001, 0.0001, 0.0005\}$ for the SGD, SGD-M and RMSprop. In terms of Adam, we do grid search over the learn rates $\{0.3, 0.2, 0.1, 0.01, 0.001\}$ and weight decay $\{0.001, 0.0001, 0.0005\}$. For L2O, MetaCur, ARUBA and ARUBA++, we tune the models on the meta-test setting with the learning rates $\{0.1, 0.01, 0.005, 0.001, 0.0005, 0.0001\}$ and weight decay $\{0.001, 0.0001, 0.0005\}$.

Bayesian Optimisation We implement our BayesOpt using (Balandat et al., 2020) for the hyperparameter tuning on the ResNet18 and CIFAR10 setting. The model the expected performance using a Gaussian process with RBF kernel, which maps the learning rate and weight decay to the estimated validation accuracy. This also provides uncertainty information to the Upper Confidence Bound (UCB) acquisition function for exploring/exploiting the hyperparameter space. For each model selection in the meta-test stage, we run the Bayesian optimisation for 25 iterations.

A.8 Training loss learning curve for DiverseDigits dataset

We give all the training loss learning curves on DiverseDigits in Fig 5. It can be noticed that the conclusion we drew that MetaMD is clearly faster than SGD and SGD-M in training convergence in Section 5.2 is further supported.

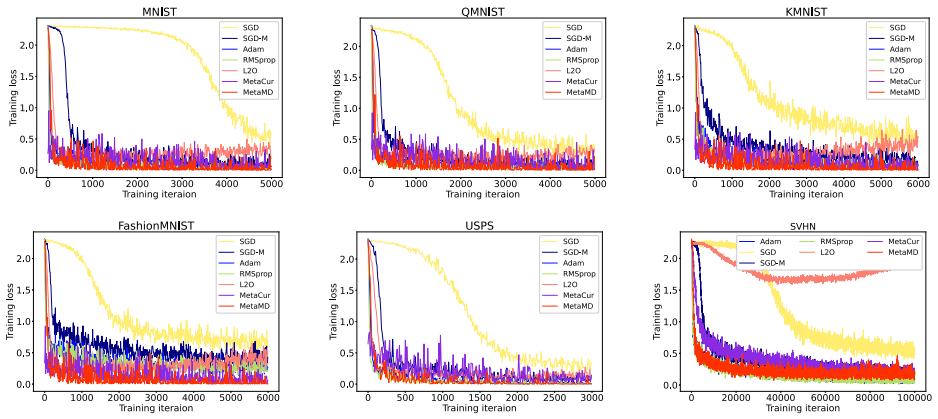

Figure 5: Convergence comparison of different optimisers on DiverseDigits.

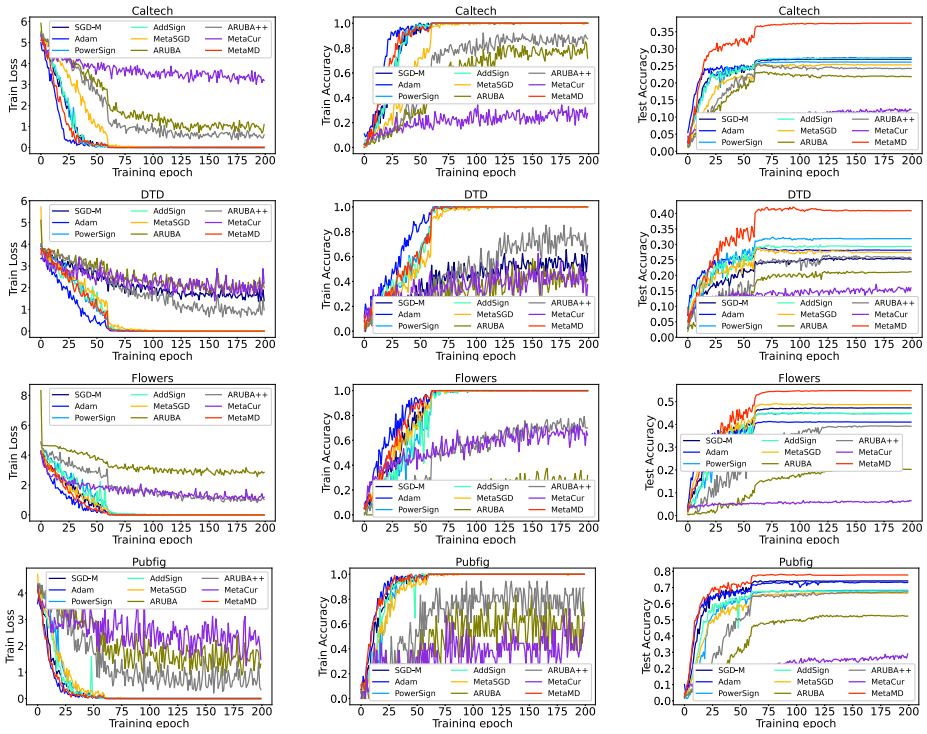

Figure 6: Learning curves on Caltech, DTD, Flowers and Pubfig. The left, middle and right collumn represent Training loss, Training accuracy and Test accuracy respectively

A.9 Experiment Details on High Resolution Image Datasets

We provide a summary of each dataset applied in our High Resolution tasks and give a clear split for the meta-train and meta-test datasets in Table 4. All the datasets except Caltech-256 has standard training, testing splits. As a result, we build our own split with 60 and 20 examples for each classes in training and testing stage respectively. Both MetaMD and competitors are tuned on learning rates,

Table 4: Statistics for the datasets used throughout the experiments. The Train, and Test columns contain the number of instances in each of the corresponding subsets.

	Dataset	Train	Test	Classes
Meta-Train	Aircraft (Maji et al., 2013)	6,667	3,333	100
	Butterfly (Chen et al., 2018)	10,270	15,009	200
	Pets (Parkhi et al., 2012)	4,000	3,390	37
Meta-Test	Caltech (Griffin et al., 2007)	12,800	5,120	256
	DTD (Cimpoi et al., 2014)	3,760	1,880	47
	Flowers (Nilsback & Zisserman, 2008)	2,040	6,149	102
	PubFig (Pinto et al., 2011)	12,178	1,660	83

$\{0.1, 0.01, 0.001, 0.0001\}$ and weight decays, $\{0.001, 0.0005, 0.0001, 0.00001, 0\}$. The learning curve for Caltech, DTD, Flowers and Pubfig are shown in Fig 6.

A.10 Loss Landscape Analysis

The generalisation ability is reflected by the flatness of the converged loss landscape (Foret et al., 2021). Motivated by this, we compare the converged loss landscapes achieved by MetaMD with those by SGD on the High Resolution Image setting. From Figure 7, it can be observed that the landscapes reached by MetaMD are much more flattened than SGD.

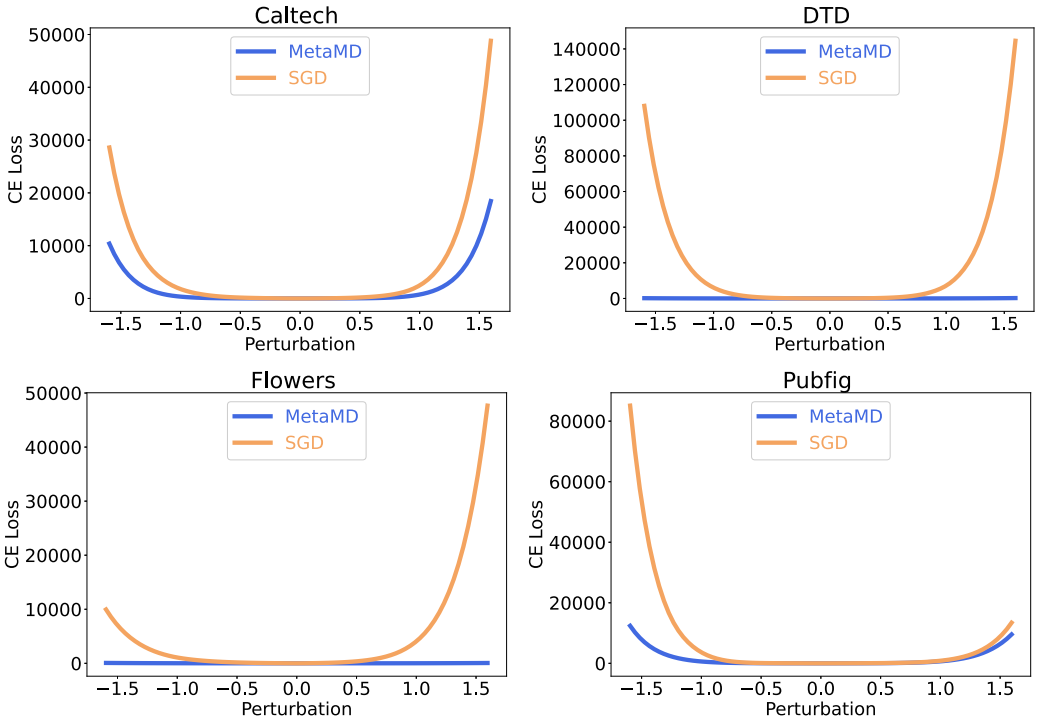

Figure 7: 1D Training Loss Landscape on Caltech, DTD, Flowers and Pubfig, where perturbations are performed in the direction of the eigenvector of the loss Hessian matrix corresponding to the largest eigenvalue.

A.11 Low rank representation Analysis

The recent research (Saunshi et al., 2021; Chen & Lee, 2021) claim that low rank representation generalizes well. Following the low-rank representation analysis proposed in (Chen & Lee, 2021), we compare the learned feature space of the model trained by SGD and that trained by MetaMD in the high resolution setting in Fig. 8. $-\sum_i \tilde{\lambda}_i \log \tilde{\lambda}_i$ denotes the low rank level, where the smaller value represents the lower rank and $\tilde{\lambda}_i = \lambda_i / \lambda_{max}$ represent the normalized singular values. One can observe that MetaMD has the lower rank feature spaces on all the meta-test tasks than SGD. The decay speed of the normalized singular values also indicates that MetaMD learns the lower rank feature space than SGD.

A.12 Hypergradient Computation

In this section, we introduce detailed inverse Hessian computation. The hypergradient of the meta-objective $\mathcal{E}$ w.r.t. to the optimiser parameters M is computed through

$$\frac{\partial \mathcal{E}}{\partial M} = \frac{\partial \mathcal{E}}{\partial \theta} \left(\frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial \theta} \right)^{-1} \frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial M} \bigg|_{\phi_M, \theta_*(M)} = \lim_{i \to \infty} \frac{\partial \mathcal{E}}{\partial \theta} \sum_{j=0}^{i} \left(I - \frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial \theta} \right)^j \frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial M} \bigg|_{\phi_M, \theta_*(M)},$$

where following (Lorraine et al., 2020) the inverse Hessian matrix is approximated by Neumann series:

$$\left(\frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial \theta} \right)^{-1} = \lim_{i \to \infty} \sum_{j=0}^{i} \left(I - \frac{\partial^2 B_{\phi_M}}{\partial \theta \, \partial \theta} \right)^j.$$

In practice, the approximation iteration j does not need to go to infinite and in our case, we set it as 20. We give the implementation details for computing $\frac{\partial \mathcal{E}}{\partial M}$ in Algorithm 2 by adapting the algorithm proposed in (Lorraine et al., 2020) to our problem and α is the hyperparameter controlling the iteration step length.

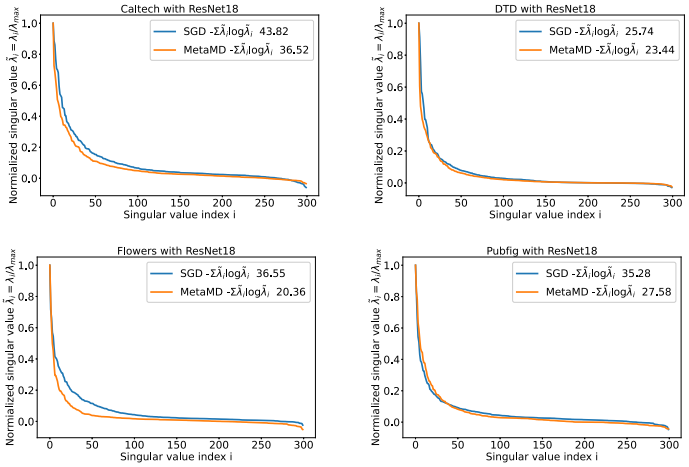

Figure 8: Low rank representation Analysis on Caltech, DTD, Flowers and Pubfig.

Algorithm 2 Computing the hypergradient of the meta-objective $\mathcal{E}$, with respect to the optimiser parameter M. The grad$(\cdot,\cdot,\cdot)$ function from PyTorch computes a Jacobian-vector product when called with a non-scalar first argument. Inspired by Lorraine et al. (2020), we use this to efficiently compute the Hessian required for approximating the Neumann series.

Input: $B_{\phi_M}, \mathcal{E}, \phi_M, \theta_i^*$
Output: $-p\frac{\partial^2 B_{\phi_M}}{\partial\theta\partial M}$
$v = p = \frac{\partial\mathcal{E}}{\partial\theta}|_{\phi_M,\theta_i^*}$
for all $j = 1, ..., J$ **do**
  $v- = \alpha \cdot \text{grad}(\frac{\partial B_{\phi_M}}{\partial\theta}, \theta, v)$
  $p \mathrel{+}= v$
end for

A.13 Convergence Rate comparison

We aim to learn optimisers with fast convergence speed. To evaluate the training speed of the learned MetaMD and to compare it with other methods, we measure the area under the curve (AUC) of the training loss (Rijn et al., 2015) where a small AUC value indicates an optimiser converges fast. Table 5 shows that the learned MetaMD has the fastest convergence speed in the CIFAR10 and ResNet18 setting while MetaMD ranks second in the High resolution setting in Table 6. To further illustrate the property of MetaMD, we compute the wall clock time of each optimisation iteration for each model in Table 7. Our model can still outperform other meta-learned precondition methods, such as MetaCur.

A.14 Fine tuning with Learned MetaMD

We studied the transferability of the learned MetaMD by shifting dataset between the meta-train and meta-test stage. A different kind of domain shift is to deploy the learned MetaMD for fine-tuning the base model, which is different to the meta-train stage where the base model is learned from scratch in each each inner loop. We reuse the High resolution setting in Section 5.2, MetaMD is trained on all the meta-train datasets with the Algorithm 1 but deployed to fine-tune the ResNet18 pre-trained on ImageNet on each individual meta-test dataset in the meta-test stage. Compared with the hand-craft optimisers including, SGD-M and Adam, with fair hyperparameters selection, MetaMD has competitive performance in terms of test accuracy in Table 8.

Table 5: Area under the learning curve of the training loss for CIFAR10 and Resnet18. Comparison with various optimisers.

Method	SGD-M	Adam	AdamW	KFAC	PowerSign	AddSign	Meta-SGD	MetaMD
CIFAR10	32.04	15.27	20.05	27.77	34.06	32.90	19.47	**11.82**

Table 6: Area under the learning curve of the training loss for high resolution task. Comparison with various optimisers.

Method	SGD-M	Adam	PowerSign	AddSign	Meta-SGD	ARUBA	ARUBA ++	MetaCur	MetaMD
Caltech	106.80	**69.77**	108.23	111.86	176.59	409.97	327.83	736.83	90.20
DTD	434.23	**94.35**	138.44	133.88	153.68	509.23	379.49	469.52	132.84
Flowers	111.32	**74.78**	145.12	142.23	112.06	672.46	386.41	332.81	94.25
Pubfig	56.80	**44.19**	82.33	68.77	81.08	416.87	274.47	512.78	47.88
Average Rank	4.00	1.00	5.00	4.25	5.00	8.50	7.00	8.25	2.00

A.15 NUMBER OF META-PARAMETERS

We compute the number of the meta-parameters for every meta-optimiser including Meta-SGD MetaCur and MetaMD in Table9. It can be noticed that MetaMD has a significantly small number of meta-parameters to learn compared with other meta-optimiser, which is one of the reasons that MetaMD suffers less from meta-overfitting.

Table 7: Wall clock time per iteration (mean and standard deviation) for High resolution task with ResNet18. Comparison with various optimisers.

Method	SGD-M	Adam	PowerSign	AddSign	Meta-SGD	ARUBA	ARUBA ++	MetaCur	MetaMD
Time	118.21 ± 2.37	121.16 ± 2.62	121.30 ± 2.38	122.45 ± 3.21	120.29 ± 3.32	118.43 ± 3.63	118.38 ± 3.62	132.63 ± 3.63	131.12 ± 3.62

Table 8: Test Accuracy (%) and AUC on high resolution datasets by fine-tuning ImageNet pre-trained ResNet18. Comparison with various optimisers.

Datasets	Caltech	DTD	Flowers	Pubfig	Avg Rank
SGD-M (Fine-tuning)	77.81 ± 0.73	67.13 ± 0.58	91.56 ± 0.54	89.52 ± 0.39	2.5
Adam (Fine-tuning)	72.44 ± 0.19	68.46 ± 0.77	92.03 ± 0.81	90.12 ± 0.28	2.5
MetaMD (Fine-tuning)	77.93 ± 0.44	68.66 ± 0.93	91.67 ± 0.75	89.94 ± 0.41	1.5
SGD-M (AUC)	154.98	78.58	22.12	59.06	2.5
Adam (AUC)	70.64	42.39	26.86	59.62	2.0
MetaMD (AUC)	150.40	77.23	21.63	54.29	1.5

Table 9: Number of Meta-parameters for various Meta-optimisers

Model	MetaSGD	MetaCur	MetaMD
	11176512	2974717	37664