Appendix

Organization The appendix is organized as follows: In Section A, we provide proofs of the theorem and propositions in Section 4. In Section B, we show additional experimental results — Meta-interpolation with first order MAML, the effect of the number of meta-training and meta-validation tasks, ablation study for location of interpolation, and the effect of the number of tasks for interpolation. In Section C, we provide detailed descriptions of the experimental setup used in the main paper together with the exact data splits for meta-training, meta-validation and meta-testing for all the datasets used. Finally, we specify the exact architecture of the Prototypical Networks used for all the experiments and further describe the Set Transformer in detail in Section C.3. All hyperparameters are specified in Section C.4.

A Proofs

A.1 Proof of Theorem 1

Proof. Define $\phi \coloneqq \phi^l_{\theta_l} = f^l_{\theta_l} \circ \cdots \circ f^1_{\theta_1}$, and $g \coloneqq f^L_{\theta_L} \circ \cdots \circ f^{l+1}_{\theta_{l+1}}$. Define the dimensionality as $\phi(\mathbf{x}^s_{t,i}) \in \mathbb{R}^d$, and $g(\phi(\mathbf{x}^s_{t,i})) \in \mathbb{R}^D$. From the definition, since we use $\hat{f}_{\theta,\lambda}$ in both training and testing time, we have

$$\mathcal{L}_{\text{singleton}}\left(\lambda, \theta; \mathcal{T}_t\right) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \mathbf{c}_{y^q_{t,i}}))}{\sum_{k'} \exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \mathbf{c}_{k'}))}$$

where

$$\mathbf{c}_k = \frac{1}{N_{t,k}} \sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i}=k\}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i})\})$$

$$N_{t,k} = \sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i}=k\}}$$

In the analysis of the loss functions, without the loss of generality, we can set the permutation σ_t on $\{1, \ldots, K\}$ to be the identity since the every combination can be realized by one permutation σ instead of the two permutations $\sigma_t, \sigma_{t'}$. Therefore, using the definition of the $\hat{\mathcal{T}}_{t,t'}$, we can write the corresponding loss by

$$\mathcal{L}_{\text{mix}}(\lambda, \theta, \hat{\mathcal{T}}_{t,t'}) = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \hat{\mathbf{c}}_{y^q_{t,i}}))}{\sum_{k'} \exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \hat{\mathbf{c}}_{k'}))}$$

where

$$\hat{\mathbf{c}}_k \coloneqq \frac{1}{|S_k|} \sum_{(\{\mathbf{x}^s_{t,i}, \mathbf{x}^s_{t',j}\}, k) \in S_k} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i}), \phi(\mathbf{x}^s_{t',j})\})$$

$$S_k \coloneqq \{(\{\mathbf{x}^s_{t,i}, \mathbf{x}^s_{t',j}\}, k) \mid (\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t, y^s_{t,i} = k, (\mathbf{x}^s_{t',j}, y^s_{t',j}) \in \mathcal{D}^s_{t'}, y^s_{t',j} = \sigma(k)\}$$

This can be rewritten as:

$$\hat{\mathbf{c}}_k = \frac{1}{N_{t,t',k}} \sum_{(\mathbf{x}^s_{t',j}, y^s_{t',j}) \in \mathcal{D}^s_{t'}} \mathbb{1}_{\{y^s_{t',i}=\sigma(k)\}} \sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i}=k\}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i}), \phi(\mathbf{x}^s_{t',j})\})$$

$$\begin{aligned} N_{t,t',k} &= \sum_{(\mathbf{x}^s_{t',j}, y^s_{t',j}) \in \mathcal{D}^s_{t'}} \mathbb{1}_{\{y^s_{t',i}=\sigma(k)\}} \sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i}=k\}} \\ &= \left(\sum_{(\mathbf{x}^s_{t',j}, y^s_{t',j}) \in \mathcal{D}^s_{t'}} \mathbb{1}_{\{y^s_{t',i}=\sigma(k)\}} \right) \left(\sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i}=k\}} \right) \\ &= N_{t',k} N_{t,k} \end{aligned}$$

Thus,

$$\hat{\mathbf{c}}_k = \frac{1}{N_{t',k}} \sum_{(\mathbf{x}^s_{t',j}, y^s_{t',j}) \in \mathcal{D}^s_{t'}} \mathbb{1}_{\{y^s_{t',j} = \sigma(k)\}} \left(\frac{1}{N_{t,k}} \sum_{(\mathbf{x}^s_{t,i}, y^s_{t,i}) \in \mathcal{D}^s_t} \mathbb{1}_{\{y^s_{t,i} = k\}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i}), \phi(\mathbf{x}^s_{t',j})\}) \right).$$

Define the set

$$I_{t,k} \coloneqq \{i : y^s_{t,i} = k\}.$$

Then,

$$\hat{\mathbf{c}}_k = \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i}), \phi(\mathbf{x}^s_{t',j})\}).$$

Summarizing the computation so far, we have that

$$\mathcal{L}_{\text{singleton}}(\lambda, \theta; \mathcal{T}_t) = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \mathbf{c}_{y^q_{t,i}}))}{\sum_{k'} \exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \mathbf{c}_{k'}))}$$

$$\mathbf{c}_k = \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i})\})$$

and

$$\mathcal{L}_{\text{mix}}(\lambda, \theta, \hat{\mathcal{T}}_{t,t'}) = -\frac{1}{n} \sum_{i=1}^{n} \log \frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \hat{\mathbf{c}}_{y^q_{t,i}}))}{\sum_{k'} \exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}), \hat{\mathbf{c}}_{k'}))}$$

$$\hat{\mathbf{c}}_k = \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} (g \circ \varphi_\lambda)(\{\phi(\mathbf{x}^s_{t,i}), \phi(\mathbf{x}^s_{t',j})\})$$

with

$$I_{t,k} = \{i : y^s_{t,i} = k\}.$$

We now analyze the relationship of $\varphi_\lambda(\{h, h'\})$ and $\varphi_\lambda(\{h\})$. From this definition, we first compute $\varphi_\lambda(\{h, h'\})$ as follows. By writing $\sigma_1 = \text{softmax}(\sqrt{d^{-1}} Q_1^{\{h,h'\}} (K_1^{\{h,h'\}})^\top)$ and $\sigma_2 = \text{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h,h'\}})^\top)$,

$$\begin{aligned}
\varphi_\lambda(\{h, h'\}) &= A(Q_2, K_2^{\{h,h'\}}, V_2^{\{h,h'\}}) \\
&= \text{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h,h'\}})^\top)(H_2^{\{h,h'\}} W_2^V + \mathbf{1}_2 b_2^V) \\
&= \text{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h,h'\}})^\top)(A(Q_1^{\{h,h'\}}, K_1^{\{h,h'\}}, V_1^{\{h,h'\}}) W_2^V + \mathbf{1}_2 b_2^V) \\
&= \sigma_2(\sigma_1(H^{\{h,h'\}} W_1^V + \mathbf{1}_2 b_1^V) W_2^V + \mathbf{1}_2 b_2^V) \\
&= \sigma_2 \sigma_1 H^{\{h,h'\}} W_1^V W_2^V + \sigma_2 \sigma_1 \mathbf{1}_2 b_1^V W_2^V + \sigma_2 \mathbf{1}_2 b_2^V.
\end{aligned}$$

Here, by writing $p_1 = p_1^{(t,t',i,j)}$, $p_1' = \tilde{p}_1^{(t,t',i,j)}$, and $p_2 = p_2^{(t,t',i,j)}$, we can rewrite that

$$\sigma_1 = \text{softmax}(\sqrt{d^{-1}} Q_1^{\{h,h'\}} (K_1^{\{h,h'\}})^\top) = \begin{bmatrix} p_1 & 1-p_1 \\ p_1' & 1-p_1' \end{bmatrix} \in \mathbb{R}^{2\times 2}$$

$$\sigma_2 = \text{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h,h'\}})^\top) = [p_2 \quad 1-p_2] \in \mathbb{R}^{1\times 2}$$

Thus, by letting $\bar{p} = p_2 p_1 + (1-p_2) p_1' \in [0,1]$,

$$\sigma_2 \sigma_1 = [p_2 \quad 1-p_2] \begin{bmatrix} p_1 & 1-p_1 \\ p_1' & 1-p_1' \end{bmatrix} = [p_2 p_1 + (1-p_2) p_1' \quad p_2(1-p_1) + (1-p_2)(1-p_1')] = [\bar{p} \quad 1-\bar{p}].$$

Using these,

$$\sigma_2 \sigma_1 H^{\{h,h'\}} = [\bar{p} \quad 1-\bar{p}] \begin{bmatrix} h^\top \\ (h')^\top \end{bmatrix} = \bar{p} h^\top + (1-\bar{p})(h')^\top$$

$$\sigma_2 \sigma_1 \mathbf{1}_2 = [\bar{p} \quad 1-\bar{p}] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1,$$

and

$$\sigma_2 \mathbf{1}_2 = [p_2 \quad 1-p_2] \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1.$$

Thus, by letting $W = (W_1^V W_2^V)^\top \in \mathbb{R}^{d \times d}$ and $b = (b_1^V W_2^V + b_2^V)^\top \in \mathbb{R}^d$, and by defining $\alpha = 1 - \bar{p}$, we have that

$$\varphi_\lambda(\{h, h'\}) = W\left(h + \alpha(h' - h)\right) + b.$$

For $\varphi_\lambda(\{h\})$, similarly, by writing $\sigma_1 = \mathrm{softmax}(\sqrt{d^{-1}} Q_1^{\{h\}} (K_1^{\{h\}})^\top)$ and $\sigma_2 = \mathrm{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h\}})^\top)$,

$$\begin{aligned} \varphi_\lambda(\{h\}) &= A(Q_2, K_2^{\{h\}}, V_2^{\{h\}}) \\ &= \sigma_2 \sigma_1 h^\top W_1^V W_2^V + \sigma_2 \sigma_1 b_1^V W_2^V + \sigma_2 b_2^V. \end{aligned}$$

Here, since $\sigma_1 = \mathrm{softmax}(\sqrt{d^{-1}} Q_1^{\{h\}} (K_1^{\{h\}})^\top) \in \mathbb{R}$ and $\sigma_2 = \mathrm{softmax}(\sqrt{d^{-1}} Q_2 (K_2^{\{h\}})^\top) \in \mathbb{R}$, we have $\sigma_1 = \sigma_2 = 1$ and thus,

$$\varphi_\lambda(\{h\}) = Wh + b.$$

Therefore, we have that

$$\begin{aligned} &\varphi_\lambda(\{h, h'\}) = W\left(h + \alpha(h' - h)\right) + b, \\ &\varphi_\lambda(\{h\}) = Wh + b. \end{aligned}$$

where W and b does *not* depend on the (h, h'). Using these and by defining $h_{t,i} = \phi(\mathbf{x}_{t,i}^s)$, we have that

$$\mathbf{c}_k = \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} g(h_{t,i}^\top W + b) \in \mathbb{R}^D$$

$$\hat{\mathbf{c}}_k \coloneqq \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} g\left(W\left[h_{t,i} + \alpha_{ij}^{(t,t')}(h_{t',j} - h_{t,i})\right] + b\right) \in \mathbb{R}^D.$$

With these preparation, we are now ready to prove the regularization form. Fix t, t' and write $\alpha_{ij} = \alpha_{ij}^{(t,t')}$. Define a vector α such that $\alpha = (\alpha_{ij})_{i,j}$. Then,

$$\hat{c}_k(\alpha) \coloneqq \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} g\left(W\left[h_{t,i} + \alpha_{ij}(h_{t',j} - h_{t,i})\right] + b\right)$$

Then, from the results of the calculations above, we have that

$$\begin{aligned} \hat{c}_k(\alpha) &= \hat{\mathbf{c}}_k, \\ \hat{c}_k(0) &= \mathbf{c}_k. \end{aligned}$$

Using the assumptions that $\partial^r g(z) = 0$ for all $r \geq 2$, for any J, the J-th approximation of $\hat{c}_k(\alpha)$ is given by

$$\hat{c}_k(\alpha)_J = \hat{c}_k(0) + \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} \alpha_{ij} \partial g\left(W h_{t,i} + b\right) W (h_{t',j} - h_{t,i})$$

Let $\bar{\gamma} \geq 1$ to be set later and define

$$\Delta_k \coloneqq \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} \frac{\alpha_{ij}}{\bar{\gamma}} \partial g\left(W h_{t,i} + b\right) W (h_{t',j} - h_{t,i}).$$

Then,

$$\hat{c}_k(\alpha)_J = \hat{c}_k(0) + \bar{\gamma} \Delta_k.$$

Define $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_K)$ and $\Delta = (\Delta_1, \ldots, \Delta_K)$. For any given t, define

$$L_t(\mathbf{c}) \coloneqq -\frac{1}{n} \sum_{i=1}^n \log \frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}_{t,i}^q), \mathbf{c}_{y_{t,i}^q}))}{\sum_{k'} \exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}_{t,i}^q), \mathbf{c}_{k'}))}.$$

Then, we have that

$$L_t(\mathbf{c}+\bar{\gamma}\Delta) = -\frac{1}{n}\sum_{i=1}^{n}\log\frac{\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}),(\mathbf{c}+\bar{\gamma}\Delta)_{y^q_{t,i}}))}{\sum_{k'}\exp(-d(\hat{f}_{\theta,\lambda}(\mathbf{x}^q_{t,i}),(\mathbf{c}+\bar{\gamma}\Delta)_{k'}))} = \mathcal{L}_{\text{mix}}(\lambda,\theta,\hat{\mathcal{T}}_{t,t'}),$$

$$L_t(\mathbf{c}) = \mathcal{L}_{\text{singleton}}\left(\lambda,\theta;\mathcal{T}_t\right).$$

The J-th approximation of $\mathcal{L}_{\text{mix}}(\lambda,\theta,\hat{\mathcal{T}}_{t,t'})$ is given by

$$\begin{aligned}\mathcal{L}_{\text{mix}}(\lambda,\theta,\hat{\mathcal{T}}_{t,t'})_J = L_t(\mathbf{c}+\bar{\gamma}\Delta)_J &= L_t(\mathbf{c}) + \sum_{j=1}^{J}\frac{\bar{\gamma}^j}{j!}\varphi^{(j)}(0)\\ &= \mathcal{L}_{\text{singleton}}\left(\lambda,\theta;\mathcal{T}_t\right) + \sum_{j=1}^{J}\frac{\bar{\gamma}^j}{j!}\varphi^{(j)}(0),\end{aligned}$$

where $\varphi^{(j)}$ is the j-th order derivative of $\varphi:\gamma\mapsto L_t(\mathbf{c}+\gamma\Delta)$. Here, for any $j\in\mathbb{N}^+$, by defining $b'=\mathbf{c}+\gamma\Delta\in\mathbb{R}^{KD}$,

$$\varphi^{(j)}(\gamma) = \sum_{i_1=1}^{KD}\sum_{i_2=1}^{KD}\cdots\sum_{i_j=1}^{KD}\frac{\partial^j L_t(b')}{\partial b'_{i_1}\partial b'_{i_2}\cdots\partial b'_{i_j}}\Delta_{i_1}\Delta_{i_2}\cdots\Delta_{i_j}.$$

Then, by using the vectorization of the tensor $\text{vec}[\partial^j L_t(b')]\in\mathbb{R}^{(KD)^j}$, we can rewrite this equation as

$$\varphi^{(j)}(\gamma) = \text{vec}[\partial^j L_t(\mathbf{c}+\gamma\Delta)]^\top\Delta^{\otimes j}, \tag{9}$$

where $\Delta^{\otimes j}=\Delta\otimes\Delta\otimes\cdots\otimes\Delta\in\mathbb{R}^{(KD)^j}$. By combining these with $\bar{\gamma}=1$,

$$\mathcal{L}_{\text{mix}}(\lambda,\theta,\hat{\mathcal{T}}_{t,t'})_J = \mathcal{L}_{\text{singleton}}\left(\lambda,\theta;\mathcal{T}_t\right) + \sum_{j=1}^{J}\frac{1}{j!}\text{vec}[\partial^j L_t(\mathbf{c})]^\top\Delta^{\otimes j},$$

where

$$\Delta_k = \frac{1}{|I_{t',\sigma(k)}|}\sum_{j\in I_{t',\sigma(k)}}\frac{1}{|I_{t,k}|}\sum_{i\in I_{t,k}}\alpha_{ij}^{(t,t')}\partial g\left(Wh_{t,i}+b\right)W(h_{t',j}-h_{t,i}).$$

□

A.2 Proof of Proposition 1

Proof. We now apply our general regularization form theorem to this special case from the previous paper [50] on the ProtoNet loss. In this special case, we have that $\phi(\mathbf{x})=\mathbf{x}$, $W=I, b=0, h_{t,i}=\mathbf{x}^s_{t,i}$, $g(\mathbf{x})=\mathbf{x}^\top\theta$, and

$$L_t(\mathbf{c}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{1+\exp(\langle\mathbf{x}^q_{t,i},\theta\rangle-\mathbf{c}_1/2-\mathbf{c}_2/2)}.$$

Here, for $\mathbf{c}=(\mathbf{c}_1,\mathbf{c}_2)$ with

$$\mathbf{c}_1 = \frac{1}{N_{t,1}}\sum_{(\mathbf{x}^s_{t,i},y^s_{t,i})\in\mathcal{D}^s_t}\mathbb{1}_{\{y^s_{t,i}=1\}}\varphi_\lambda(\{\mathbf{x}^s_{t,i}\})^\top\,\theta = \theta^\top\mathbf{c}'_1 \tag{10}$$

and

$$\mathbf{c}_2 = \frac{1}{N_{t,2}}\sum_{(\mathbf{x}^s_{t,i},y^s_{t,i})\in\mathcal{D}^s_t}\mathbb{1}_{\{y^s_{t,i}=2\}}\varphi_\lambda(\{\mathbf{x}^s_{t,i}\})^\top\,\theta = \theta^\top\mathbf{c}'_2, \tag{11}$$

we recover that

$$L_t(\mathbf{c}) = \mathcal{L}\left(\lambda,\theta;\mathcal{T}_t\right).$$

Thus, to instantiate our general theorem to this special case, we only need to compute the $\partial^j L_t(\mathbf{c})$ and $\partial g(h_{t,i}^\top W + b)$ up to the second order approximation.

$$\partial g(h_{t,i}^\top W + b) = \partial g(\mathbf{x}_{t,i}^s) = \theta$$

$$\partial^j g(h_{t,i}^\top W + b) = \partial^j g(\mathbf{x}_{t,i}^s) = 0 \qquad \forall j \geq 2$$

Define $z_{t,i} \coloneqq \langle \mathbf{x}_{t,i}^q - (\mathbf{c}_1' + \mathbf{c}_2')/2, \theta \rangle$. Since $\mathbf{c}_1 = \theta^\top \mathbf{c}_1'$ and $\mathbf{c}_2 = \theta^\top \mathbf{c}_2'$ by Equation 10 and 11,

$$\begin{aligned}
\langle \mathbf{x}_{t,i}^q - (\mathbf{c}_1' + \mathbf{c}_2')/2, \theta \rangle &= \langle \mathbf{x}_{t,i}^q, \theta \rangle - (\langle \mathbf{c}_1'/2, \theta \rangle + \langle \mathbf{c}_2'/2, \theta \rangle) \\
&= \langle \mathbf{x}_{t,i}^q, \theta \rangle - \theta^\top \mathbf{c}_1'/2 - \theta^\top \mathbf{c}_2'/2 \\
&= \langle \mathbf{x}_{t,i}^q, \theta \rangle - \mathbf{c}_1/2 - \mathbf{c}_2/2
\end{aligned}$$

Thus, we have $L_t(\mathbf{c}) = \frac{1}{n}\sum_{i=1}^n \frac{1}{1+\exp(z_{t,i})}$. Then,

$$\frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1} = -\frac{1}{n}\sum_{i=1}^n [1+\exp(z_{t,i})]^{-2}\exp(z_{t,i})\frac{\partial z_{t,i}}{\partial \mathbf{c}_1} = \frac{1}{n}\sum_{i=1}^n \frac{1}{2}[1+\exp(z_{t,i})]^{-2}\exp(z_{t,i})$$

Similarly,

$$\frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2} = -\frac{1}{n}\sum_{i=1}^n [1+\exp(z_{t,i})]^{-2}\exp(z_{t,i})\frac{\partial z_{t,i}}{\partial \mathbf{c}_2} = \frac{1}{n}\sum_{i=1}^n \frac{1}{2}[1+\exp(z_{t,i})]^{-2}\exp(z_{t,i})$$

For the second order, by defining the logistic function $\psi(z_{t,i}) = \frac{\exp(z_{t,i})}{1+\exp(z_{t,i})}$,

$$\begin{aligned}
\frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_1} &= \frac{1}{n}\sum_{i=1}^n \frac{-2}{2}[1+\exp(z_{t,i})]^{-3}\exp(z_{t,i})\frac{\partial \exp(z_{t,i})}{\partial \mathbf{c}_1} + \frac{1}{2}[1+\exp(z_{t,i})]^{-2}\frac{\partial \exp(z_{t,i})}{\partial \mathbf{c}_1} \\
&= \frac{1}{n}\sum_{i=1}^n \frac{1}{2}[1+\exp(z_{t,i})]^{-3}\exp(z_{t,i})^2 - \frac{1}{4}[1+\exp(z_{t,i})]^{-2}\exp(z_{t,i}) \\
&= \frac{1}{n}\sum_{i=1}^n \frac{1}{2}[1+\exp(z_{t,i})]^{-2}\exp(z_{t,i})(\psi(z_{t,i}) - 0.5) \\
&= \frac{1}{n}\sum_{i=1}^n \frac{1}{2}\frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1+\exp(z_{t,i})}
\end{aligned}$$

Similarly,

$$\frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2 \partial \mathbf{c}_2} = \frac{1}{n}\sum_{i=1}^n \frac{1}{2}\frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1+\exp(z_{t,i})}$$

$$\frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_2} = \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2 \partial \mathbf{c}_1} = \frac{1}{n}\sum_{i=1}^n \frac{1}{2}\frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1+\exp(z_{t,i})}$$

Therefore, the second approximation of $\mathbb{E}_{t',\sigma}[\mathcal{L}_{\text{mix}}(\lambda, \theta, \hat{\mathcal{T}}_{t,t'})]$ is

$$\begin{aligned}
&\mathcal{L}_{\text{singleton}}(\lambda, \theta; \mathcal{T}_t) + \mathbb{E}_{t',\sigma}\left[\sum_{j=1}^2 \frac{1}{j!}\operatorname{vec}[\partial^j L_t(\mathbf{c})]^\top \Delta^{\otimes j}\right] \\
&= \mathcal{L}_{\text{singleton}}(\lambda, \theta; \mathcal{T}_t) + \mathbb{E}_{t',\sigma}\left[\operatorname{vec}[\partial L_t(\mathbf{c})]^\top \Delta + \frac{1}{2}\operatorname{vec}[\partial^2 L_t(\mathbf{c})]^\top \Delta^{\otimes 2}\right]
\end{aligned}$$

where $\Delta = [\Delta_1^\top, \Delta_2^\top]^\top$ with

$$\begin{aligned}
\Delta_k &= \frac{1}{|I_{t',\sigma(k)}|}\sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|}\sum_{i \in I_{t,k}} \alpha_{ij}^{(t,t')}(h_{t',j} - h_{t,i})^\top \theta \\
&= \theta^\top \delta_{t,t',\sigma,k}
\end{aligned}$$

where

$$\delta_{t,t',\sigma,k} \coloneqq \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} \alpha_{ij}^{(t,t')} (h_{t',j} - h_{t,i}).$$

For the first order term,

$$\begin{aligned}
\mathrm{vec}[\partial L_t(\mathbf{c})]^\top \Delta &= \frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} [1 + \exp(z_{t,i})]^{-2} \exp(z_{t,i}) (\theta^\top \delta_{t,t',\sigma,k} + \theta^\top \delta_{t,t',\sigma,k}) \\
&= \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} [1 + \exp(z_{t,i})]^{-2} \right) \exp(z_{t,i}) \theta^\top \delta_{t,t',\sigma}
\end{aligned}$$

where

$$\delta_{t,t',\sigma} = \sum_{k=1}^{2} \frac{1}{|I_{t',\sigma(k)}|} \sum_{j \in I_{t',\sigma(k)}} \frac{1}{|I_{t,k}|} \sum_{i \in I_{t,k}} \alpha_{ij}^{(t,t')} (h_{t',j} - h_{t,i}).$$

For the second order term,

$$\begin{aligned}
\mathrm{vec}[\partial^2 L_t(\mathbf{c})]^\top \Delta^{\otimes 2} &= [\Delta^\top \otimes \Delta^\top] \ \mathrm{vec}[\partial^2 L_t(\mathbf{c})] \\
&= \Delta^\top \partial^2 L_t(\mathbf{c}) \Delta \\
&= \begin{bmatrix} \Delta_1^\top & \Delta_2^\top \end{bmatrix} \begin{bmatrix} \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_1} & \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_2} \\ \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2 \partial \mathbf{c}_1} & \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2 \partial \mathbf{c}_2} \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \end{bmatrix} \\
&= \Delta_1^2 \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_1} + \Delta_2^2 \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_2 \partial \mathbf{c}_2} + 2\Delta_1 \Delta_2 \frac{\partial L_t(\mathbf{c})}{\partial \mathbf{c}_1 \partial \mathbf{c}_2} \\
&= \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1 + \exp(z_{t,i})} \right) \left(\Delta_1^2 + \Delta_2^2 + 2\Delta_1 \Delta_2 \right) \\
&= \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1 + \exp(z_{t,i})} \right) (\Delta_1 + \Delta_2)^2 \\
&= \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{2} \frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1 + \exp(z_{t,i})} \right) \left(\theta^\top \delta_{t,t',\sigma} \right)^2
\end{aligned}$$

Since $\mathbb{E}_{t',\sigma}[\delta_{t,t',\sigma}] = 0$ with the α being balanced, the second approximation of $\mathbb{E}_{t',\sigma}[\mathcal{L}_{\text{mix}}(\lambda, \theta, \hat{\mathcal{T}}_{t,t'})]$ becomes:

$$\begin{aligned}
&\mathcal{L}_{\text{singleton}}(\lambda, \theta; \mathcal{T}_t) + \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{4} \frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1 + \exp(z_{t,i})} \right) \mathbb{E}_{t',\sigma} \left[\left(\theta^\top \delta_{t,t',\sigma} \right)^2 \right]. \\
&= \mathcal{L}_{\text{singleton}}(\lambda, \theta; \mathcal{T}_t) + \left(\frac{1}{n} \sum_{i=1}^{n} \frac{1}{4} \frac{\psi(z_{t,i})(\psi(z_{t,i}) - 0.5)}{1 + \exp(z_{t,i})} \right) \theta^\top \mathbb{E}_{t',\sigma} \left[\delta_{t,t',\sigma} \delta_{t,t',\sigma}^\top \right] \theta.
\end{aligned}$$

□

A.3 Proof of Proposition 2

Proof. Let $\xi_1, \ldots, \xi_n$ be independent uniform random variables taking values in $\{-1, 1\}$; i.e., Rademacher variables. We first bound the empirical Rademacher complexity part as follows:

$$\begin{aligned}
\mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \hat{\mathcal{R}}_n(\mathcal{F}_R) &= \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \mathbb{E}_\xi \sup_{f \in \mathcal{F}_R} \frac{1}{n} \sum_{i=1}^{n} \xi_i f(\mathbf{x}_i) \\
&= \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \mathbb{E}_\xi \sup_{\theta : \|\theta\|_\Sigma^2 \leq R} \frac{1}{n} \sum_{i=1}^{n} \xi_i \theta^\top \mathbf{x}_i \\
&= \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \mathbb{E}_\xi \sup_{\theta : \|\theta\|_\Sigma^2 \leq R} \mathbb{E}_{\mathbf{x}'} \frac{1}{n} \sum_{i=1}^{n} \xi_i \theta^\top (\mathbf{x}_i - \mathbf{x}')
\end{aligned}$$

$$
\begin{aligned}
&\leq \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{1}{n} \mathbb{E}_{\xi} \sup_{\theta:\theta^\top \Sigma \theta \leq R} \|\Sigma^{1/2}\theta\|_2 \mathbb{E}_{\mathbf{x}'} \left\| \sum_{i=1}^{n} \xi_i \Sigma^{\dagger/2}(\mathbf{x}_i - \mathbf{x}') \right\|_2 \\
&\leq \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{\sqrt{R}}{n} \mathbb{E}_{\mathbf{x}'} \mathbb{E}_{\xi} \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{n} \xi_i \xi_j (\Sigma^{\dagger/2}(\mathbf{x}_i - \mathbf{x}'))^\top (\Sigma^{\dagger/2}(\mathbf{x}_j - \mathbf{x}'))} \\
&\leq \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{\sqrt{R}}{n} \sqrt{\mathbb{E}_{\mathbf{x}'} \mathbb{E}_{\xi} \sum_{i=1}^{n}\sum_{j=1}^{n} \xi_i \xi_j (\Sigma^{\dagger/2}(\mathbf{x}_i - \mathbf{x}'))^\top (\Sigma^{\dagger/2}(\mathbf{x}_j - \mathbf{x}'))} \\
&= \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{\sqrt{R}}{n} \sqrt{\mathbb{E}_{\mathbf{x}'} \sum_{i=1}^{n} (\Sigma^{\dagger/2}(\mathbf{x}_i - \mathbf{x}'))^\top (\Sigma^{\dagger/2}(\mathbf{x}_i - \mathbf{x}'))} \\
&= \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{\sqrt{R}}{n} \sqrt{\mathbb{E}_{\mathbf{x}'} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{x}')^\top \Sigma^{\dagger} (\mathbf{x}_i - \mathbf{x}')}
\end{aligned}
$$

where $\Sigma^\dagger$ denotes the Moore–Penrose inverse of Σ. By taking expectation and using this bound on the empirical Rademacher complexity, we now bound the Rademacher complexity as follows:

$$
\begin{aligned}
\mathcal{R}_n(\mathcal{F}_R) = \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \hat{\mathcal{R}}_n(\mathcal{F}_R) &\leq \mathbb{E}_{\mathbf{x}_1,\ldots,\mathbf{x}_n} \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \mathbb{E}_{\mathbf{x}'} (\mathbf{x}_i - \mathbf{x}')^\top \Sigma^{\dagger} (\mathbf{x}_i - \mathbf{x}')} \\
&\leq \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \mathbb{E}_{\mathbf{x}_i,\mathbf{x}'} (\mathbf{x}_i - \mathbf{x}')^\top \Sigma^{\dagger} (\mathbf{x}_i - \mathbf{x}')} \\
&= \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \sum_{k,l} (\Sigma^{\dagger})_{kl} \mathbb{E}_{\mathbf{x}_i,\mathbf{x}'} (\mathbf{x}_i - \mathbf{x}')_k (\mathbf{x}_i - \mathbf{x}')_l} \\
&= \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \sum_{k,l} (\Sigma^{\dagger})_{kl} (\Sigma)_{kl}} \\
&= \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \mathrm{tr}(\Sigma \Sigma^{\dagger})} \\
&= \frac{\sqrt{R}}{n} \sqrt{\sum_{i=1}^{n} \mathrm{rank}(\Sigma)} \\
&= \frac{\sqrt{R}\sqrt{\mathrm{rank}(\Sigma)}}{\sqrt{n}}
\end{aligned}
$$

□

B Additional Experiments

B.1 First-order MAML

As stated in the introduction, we focus solely on metric based meta-learning due to their efficiency and better empirical performance over the gradient based methods on the few-task meta-learning problem. Moreover it is challenging to combine our method with Model-Agnostic Meta-Learning (MAML) [10] since it yields a tri-level optimization problem which requires differentiating through second order derivatives. Furthermore, tri-level optimization is still known to be a challenging problem [4, 7] and currently an active line of research. Thus, instead of using the original MAML, we perform additional experiments on the ESC-50 and Metabolism dataset with first-order MAML (FOMAML)

Table 7: First-order MAML (FOMAML) on Metabolism ESC-50 dataset.

	Metabolism	**ESC50**
Method	5shot	5shot
FOMAML	65.18 ± 2.20	72.14 ± 0.73
FOMAML + MLTI	63.94 ± 2.87	71.52 ± 0.55
FOMAML + Meta-Interpolation	66.79 ± 2.35	76.68 ± 1.02
ProtoNet + Meta-Interpolation	$\mathbf{72.92 \pm 1.74}$	$\mathbf{79.22 \pm 0.96}$

Table 8: Acc. on ESC-50 as varying # of **meta-training** tasks.

Model	**5 Tasks**	**10 Tasks**	**15 Tasks**	**20 Tasks**
ProtoNet	51.41 ± 3.93	60.63 ± 3.61	65.49 ± 2.05	69.05 ± 1.49
MLTI	58.98 ± 3.54	61.60 ± 2.04	66.29 ± 2.41	70.62 ± 1.96
Ours	$\mathbf{72.74 \pm 0.84}$	$\mathbf{74.78 \pm 1.43}$	$\mathbf{77.47 \pm 1.33}$	$\mathbf{79.22 \pm 0.96}$

Table 9: Acc. on ESC-50 as varying # of **meta-validation** tasks.

Model	**5 Tasks**	**10 Tasks**	**15 Tasks**
ProtoNet	69.48 ± 1.03	69.20 ± 1.17	69.05 ± 1.49
MLTI	68.09 ± 2.07	69.40 ± 2.02	70.62 ± 1.96
Ours	$\mathbf{77.68 \pm 1.38}$	$\mathbf{77.13 \pm 1.23}$	$\mathbf{79.22 \pm 0.96}$

which approximates the Hessian with a zero matrix. As shown in Table 7, the experimental results show that Meta-Interpolation with first-order MAML outperforms MLTI, which again confirms the general effectiveness of our set-based task augmentation scheme. However, it largely underperforms our original Meta-Interpolation framework with metric-based meta-learning.

B.2 Effect of the number of meta-training and validation tasks

We analyze the effect of the number of meta-training tasks on ESC-50 dataset. As shown in Table 8, Meta-Interpolation consistently outperforms the baselines by large margins, regardless of the number of the tasks. Furthermore, we report the test accuracy as a function of the number of meta-validation tasks in Table 9. Although the generalization performance of Meta-Interpolation slightly decreases as we reduce the number of meta-validation tasks, it still outperforms the relevant baselines by a large margin.

B.3 Location of interpolation

Contrary to Manifold Mixup, we fix the layer of interpolation as shown in Table 17. Otherwise, we cannot use the same architecture of Set Transformer to interpolate the output of different layers since the hidden dimension of each layer is different. Moreover, we report the test accuracy by changing the layer of interpolation. Interpolating hidden representation of support sets from layer 2, which is the model used in the main experiments on ESC50 dataset, achieves the best performance.

Table 10: Accuracy for different location of interpolation on ESC-50.

Layer	**Accuracy**
Input Layer	66.83 ± 1.31
Layer 1	74.04 ± 2.05
Layer 2	$\mathbf{79.22 \pm 0.96}$
Layer 3	77.62 ± 1.46

B.4 Effect of Cardinality for Interpolation

Since the set function φ_λ can handle sets of arbitrary cardinality $n \in \mathbb{N}$, we plot the test accuracy on the ESC-50 dataset with varying set sizes from one to five. As shown in Figure 4, for sets of cardinality one, where we do not perform any interpolation, we observe significant degradation in performance. This suggests that the interpolation of instances is crucial for better generalization. On the other hand, for set sizes from two to five, the gain is marginal with increasing cardinality. Furthermore, increasing the set size introduces extra computational overhead. Thus, we set the cardinality to two for task interpolation in all the experiments.

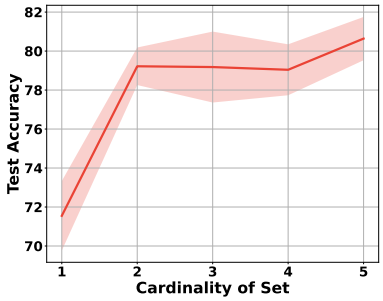

Figure 4: Acc. as a function of set size.

C Experimental Setup

C.1 Dataset Description

Metabolism We use the following split for the metabolism dataset:

$$\texttt{Meta-Train} = \{\texttt{CYP1A2\_Veith}, \texttt{CYP3A4\_Veith}, \texttt{CYP2C9\_Substrate\_CarbonMangels}\}$$
$$\texttt{Meta-Validation} = \{\texttt{CYP2D6\_Veith}, \texttt{CYP2D6\_Substrate\_CarbonMangels}\}$$
$$\texttt{Meta-Test} = \{\texttt{CYP2C19\_Veith}, \texttt{CYP2C9\_Veith}, \texttt{CYP3A4\_Substrate\_CarbonMangels}\}$$

Following Yao et al. [50], we balance each subdataset by selecting 1000 samples from each subdataset. Each data sample is processed by extracting a 1024-bit fingerprint feature from the SMILES representation of each chemical compound using the RDKit [15] library.

Tox21 We use the following split for the metabolism dataset[2]:

$$\texttt{Meta-Train} = \{\texttt{NR-AR}, \texttt{NR-AR-LBD}, \texttt{NR-AhR}, \texttt{NR-Aromatase}, \texttt{NR-ER}, \texttt{NR-ER-LBD}\}$$
$$\texttt{Meta-Validation} = \{\texttt{NR-PPAR-gamma}, \texttt{SR-ARE}\}$$
$$\texttt{Meta-Test} = \{\texttt{SR-ATAD5}, \texttt{SR-HSE}, \texttt{SR-MMP}, \texttt{SR-p53}\}$$

NCI We download the dataset from the github repository [3] and use the following splits for meta-training, meta-validation, and meta-testing:

$$\texttt{Meta-Train} = \{41, 47, 83, 109\}$$
$$\texttt{Meta-Validation} = \{81, 145\}$$
$$\texttt{Meta-Test} = \{1, 33, 123\}$$

GLUE-SciTail We use Hugging Face Datasets library [25] to download MNLI, QNLI, SNLI, RTE, and SciTail datasets and tokenize it ELECTRA tokenizer with setting maximum length to 128. We list meta-train, meta-validation, and meta-test split as follows:

$$\texttt{Meta-Train} = \{\texttt{MNLI}, \texttt{QNLI}\}$$
$$\texttt{Meta-Validation} = \{\texttt{SNLI}, \texttt{RTE}\}$$
$$\texttt{Meta-Test} = \{\texttt{SciTail}\}$$

ESC-50 We download ESC-50 dataset [34] from the github repository[4] and use the meta-training, meta-validation, and meta-test split as follows:

Meta-train set:

dog, rooster, pig, cow, frog, cat, hen, insects, sheep, crow,
rain, sea_waves, crackling_fire, crickets, chirping_birds,
water_drops, wind, pouring_water, toilet_flush, thunderstorm

Meta-validation set:

crying_baby, sneezing, clapping, breathing, coughing, footsteps,
laughing, brushing_teeth, snoring, drinking_sipping, door_wood_knock,
mouse_click, keyboard_typing, door_wood_creaks, can_opening

Meta-test set:

washing_machine, vacuum_cleaner, clock_alarm, clock_tick,
glass_breaking, helicopter, chainsaw, siren,
car_horn, engine, train, church_bells, airplane, fireworks, hand_saw

[2] https://tdcommons.ai/single_pred_tasks/tox/#tox21
[3] https://github.com/GRAND-Lab/graph_datasets/tree/master/Graph_Repository
[4] https://github.com/karolpiczak/ESC-50

RMNIST Following Yao et al. [50], we create RMNIST dataset by changing the size, color, and angle of the images from the original MNIST dataset. To be specific, we merge training and test split of MNIST and randomly select 5,600 images for each class and split into 56 subdatasets where each class has 100 examples. We only choose 16/6/10 subdatasets for meta-train, meta-validation, and meta-test split, respectively and do not use the rest of them. Each subdatset with the corresponding composition of image transformations is considered a distinct task. Following are the specific splits.

Meta-train set:

$$(\texttt{red}, \texttt{full}, 90^\circ), (\texttt{indigo}, \texttt{full}, 0^\circ), (\texttt{blue}, \texttt{full}, 270^\circ), (\texttt{orange}, \texttt{half}, 270^\circ),$$
$$(\texttt{green}, \texttt{full}, 90^\circ), (\texttt{green}, \texttt{full}, 270^\circ), (\texttt{orange}, \texttt{full}, 180^\circ), (\texttt{red}, \texttt{full}, 180^\circ),$$
$$(\texttt{green}, \texttt{full}, 0^\circ), (\texttt{orange}, \texttt{full}, 0^\circ), (\texttt{violet}, \texttt{full}, 270^\circ), (\texttt{orange}, \texttt{half}, 90^\circ),$$
$$(\texttt{violet}, \texttt{half}, 180^\circ), (\texttt{orange}, \texttt{full}, 90^\circ), (\texttt{violet}, \texttt{full}, 180^\circ), (\texttt{blue}, \texttt{full}, 90^\circ)$$

Meta-validation set:

$$(\texttt{indigo}, \texttt{half}, 270^\circ), (\texttt{blue}, \texttt{full}, 0^\circ), (\texttt{yellow}, \texttt{half}, 180^\circ),$$
$$(\texttt{yellow}, \texttt{half}, 0^\circ), (\texttt{yellow}, \texttt{half}, 90^\circ), (\texttt{violet}, \texttt{half}, 0^\circ)$$

Meta-test set:

$$(\texttt{yellow}, \texttt{full}, 270^\circ), (\texttt{red}, \texttt{full}, 0^\circ), (\texttt{blue}, \texttt{half}, 270^\circ), (\texttt{blue}, \texttt{half}, 0^\circ), (\texttt{blue}, \texttt{half}, 180^\circ),$$
$$(\texttt{red}, \texttt{half}, 270^\circ), (\texttt{violet}, \texttt{full}, 90^\circ), (\texttt{blue}, \texttt{half}, 90^\circ), (\texttt{green}, \texttt{half}, 270^\circ), (\texttt{red}, \texttt{half}, 90^\circ)$$

Mini-ImageNet-S Following Yao et al. [50], we reduce the number of meta-training tasks by choosing subset of the original meta-training classes. We specify the classes used for meta-training tasks as follows:

$$\texttt{n03017168}, \texttt{n07697537}, \texttt{n02108915}, \texttt{n02113712}, \texttt{n02120079}, \texttt{n04509417},$$
$$\texttt{n02089867}, \texttt{n03888605}, \texttt{n04258138}, \texttt{n03347037}, \texttt{n02606052}, \texttt{n06794110}$$

For meta-validation and meta-testing classes, we use the same classes as the original Mini-ImageNet.

CIFAR-100-FS Similar to Mini-ImageNet-S, we choose 12/16/20 classes for meta-training, meta-validation, meta-test classes. Followings are the specific classes for each split:

$$\texttt{Meta-Train} = \{0, \ldots, 11\}$$
$$\texttt{Meta-Validation} = \{64, \ldots, 79\}$$
$$\texttt{Meta-Test} = \{80, \ldots, 99\}$$

C.2 Prototypical Network Architecture

We summarize neural network architectures for ProtoNet on each datasets in Table 11, 12, 13, 14, 15, and 16. For Glue-SciTail, we use pretrained ELECTRA-small [8] which we download from Hugging Face [48].

Table 11: Conv4 architecture for RMNIST.

Output Size	Layers
$3 \times 28 \times 28$	Input Image
$32 \times 14 \times 14$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 7 \times 7$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 3 \times 3$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 1 \times 1$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
32	Flatten

Table 12: Conv4 architecture for Mini-ImageNet-S.

Output Size	Layers
$3 \times 84 \times 84$	Input Image
$32 \times 42 \times 42$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 21 \times 21$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 10 \times 10$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 5 \times 5$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
800	Flatten

Table 13: Conv4 architecture for CIFAR-100-FS.

Output Size	Layers
$3 \times 32 \times 32$	Input Image
$32 \times 16 \times 16$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 8 \times 8$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 4 \times 4$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
$32 \times 2 \times 2$	conv2d(3×3, stride 1, padding 1), BatchNorm2D, ReLU, Maxpool(2×2, stride 2)
128	Flatten

Table 14: Fully connected networks for Metabolism, Tox21, and NCI.

Output Size	Layers
1024	Input SMILE
500	Linear(1024, 500, bias=True), BatchNorm1D, LeakyReLU
500	Linear(500, 500, bias=True), BatchNorm1D, LeakyReLU
500	Linear(500, 500, bias=True)

Table 15: Fully connected networks for ESC-50.

Output Size	Layers
650	Input VGGish feature
500	Linear(650, 500, bias=True), BatchNorm1D, LeakyReLU
500	Linear(500, 500, bias=True), BatchNorm1D, LeakyReLU
500	Linear(500, 500, bias=True)

Table 16: ELECTRA-small for GLUE-SciTail.

Output Size	Layers
128	Input sentence
128×256	ElectraModel("google/electra-small-discriminator")
256	[CLS] embedding
256	Linear(256, 256, bias=True), ReLU
256	Linear(256, 256, bias=True), ReLU
256	Linear(256, 256, bias=True)

C.3 Set Transformer

We describe Set Transformer [23], φ_λ, in more detail. Let $X \in \mathbb{R}^{n \times d}$ be a stack of n d-dimensional row vectors. Let $W^Q_{1,j}, W^K_{1,j}, W^V_{1,j} \in \mathbb{R}^{d \times d_k}$ be weight matrices for self-attention and let $b^Q_{1,j}, b^K_{1,j}, b^V_{1,j} \in \mathbb{R}^{d_k}$ be bias vectors for $j = 1, \ldots, 4$. For encoding an input X, we compute

self-attention as follows:

$$\begin{aligned}
Q_1^{(j)} &= XW_{1,j}^Q + \mathbf{1}(b_{1,j}^Q)^\top \in \mathbb{R}^{n\times d_k} \\
K_1^{(j)} &= XW_{1,j}^K + \mathbf{1}(b_{1,j}^K)^\top \in \mathbb{R}^{n\times d_k} \\
V_1^{(j)} &= XW_{1,j}^V + \mathbf{1}(b_{1,j}^V)^\top \in \mathbb{R}^{n\times d_k} \\
A_1^{(j)}(X) &= \text{LayerNorm}\left(Q_1^{(j)} + \text{softmax}\left(Q_1^{(j)}(K_1^{(j)})^\top/\sqrt{d_k}\right)V_1^{(j)}\right) \in \mathbb{R}^{n\times d_k} \\
O_1(X) &= \text{Concat}(A_1^{(1)}(X),\dots,A_1^{(4)}(X)) \in \mathbb{R}^{n\times d_h}
\end{aligned} \tag{12}$$

where $\mathbf{1} = (1,\dots,1)^\top \in \mathbb{R}^n$ is a vector of ones, $d_h = 4d_k$, and softmax is applied for each row. After self-attention, we add a skip connection with layer normalization [1] as follows:

$$g_1(X) \coloneqq \text{LayerNorm}\left((O_1(X)) + \text{ReLU}(W_1O_1(X) + \mathbf{1}b_1^\top)\right) \tag{13}$$

where $W_1 \in \mathbb{R}^{d_h\times d_h}, b_1 \in \mathbb{R}^{d_h}$. Similarly, we compute another self-attention on top of $g_1(X)$ with $W_{2,j}^Q, W_{2,j}^K, W_{2,j}^V \in \mathbb{R}^{d_k\times d_k}$ weight matrices for self-attention and let $b_{2,j}^Q, b_{2,j}^K, b_{2,j}^V \in \mathbb{R}^{d_k}$ be bias vectors, for $j = 1\dots,4$.

$$\begin{aligned}
Q_2^{(j)} &= g_1(X)W_{2,j}^Q + \mathbf{1}(b_{2,j}^Q)^\top \in \mathbb{R}^{n\times d_k} \\
K_2^{(j)} &= g_1(X)W_{2,j}^K + \mathbf{1}(b_{2,j}^K)^\top \in \mathbb{R}^{n\times d_k} \\
V_2^{(j)} &= g_1(X)W_{2,j}^V + \mathbf{1}(b_{2,j}^V)^\top \in \mathbb{R}^{n\times d_k} \\
A_2^{(j)}(g_1(X)) &= \text{LayerNorm}\left(Q_2^{(j)} + \text{softmax}\left(Q_2^{(j)}(K_2^{(j)})^\top/\sqrt{d_k}\right)V_2^{(j)}\right) \in \mathbb{R}^{n\times d_k} \\
O_2(g_1(X)) &= \text{Concat}(A_2^{(1)}(g_1(X)),\dots,A_2^{(4)}(g_1(X)) \in \mathbb{R}^{n\times d_h}
\end{aligned} \tag{14}$$

After self-attention, we also add a skip connection after the second self-attention with dropout [41].

$$(g_2\circ g_1)(X) \coloneqq \text{Dropout}(H_2(X)) \tag{15}$$

$$H_2(X) = \text{LayerNorm}(\left(O_2(g_1(X)) + \text{ReLU}(O_2(g_1(X))W_2 + \mathbf{1}b_2^\top)\right)) \tag{16}$$

where $W_2 \in \mathbb{R}^{d_h\times d_h}, b_2 \in \mathbb{R}^{d_h}$.

After encoding X with two-layers of self-attention, we perform pooling with attention. Let $S \in \mathbb{R}^{1\times d_h}$ be learnable parameters and $W_{3,j}^Q, W_{3,j}^K, W_{3,j}^V \in \mathbb{R}^{d_h\times d_k}$ be weight matrices for the pooling-attention and let $b_{3,j}^Q, b_{3,j}^K, b_{3,j}^V \in \mathbb{R}^{d_k}$ be bias vectors, for $j = 1,\dots,4$. We pool $(g_2\circ g_1)(X)$ as follows:

$$\begin{aligned}
Q_3^{(j)} &= SW_{3,j}^Q + \mathbf{1}(b_{3,j}^Q)^\top \in \mathbb{R}^{1\times d_k} \\
K_3^{(j)} &= (g_2\circ g_1)(X)W_{3,j}^Q + \mathbf{1}(b_{3,j}^K)^\top \in \mathbb{R}^{n\times d_k} \\
V_3^{(j)} &= (g_2\circ g_1)(X)W_{3,j}^K + \mathbf{1}(b_{3,j}^V)^\top \in \mathbb{R}^{n\times d_k} \\
A_3^{(j)}((g_2\circ g_1)(X)) &= \text{LayerNorm}\left(Q_3^{(j)} + \text{softmax}\left(Q_3^{(j)}(K_3^{(j)})^\top/\sqrt{d_k}\right)V_3^{(j)}\right) \in \mathbb{R}^{1\times d_k} \\
O_3((g_2\circ g_1)(X)) &= \text{Concat}\left(A_3^{(1)}((g_2\circ g_1)(X)),\dots,A_3^{(4)}((g_2\circ g_1)(X))\right) \in \mathbb{R}^{1\times d_h}
\end{aligned} \tag{17}$$

After pooling, we add another skip connection with dropout as follows:

$$\begin{aligned}
(g_3\circ g_2\circ g_1)(X) &\coloneqq \text{Dropout}\left(H_3(X)\right) \\
H_3(X) &= \text{LayerNorm}\left(O_3\left((g_2\circ g_1)(X)\right) + \text{ReLU}(O_3((g_2\circ g_1)(X))W_3 + \mathbf{1}b_3^\top)\right)
\end{aligned} \tag{18}$$

where $W_3 \in \mathbb{R}^{d_h\times d_h}, b_3 \in \mathbb{R}^{d_h}$. Finally, we perform affine-transformation after the pooling as follows:

$$(g_4\circ g_3\circ g_2\circ g_1)(X) \coloneqq \left((g_3\circ g_2\circ g_1)(X)W_4 + \mathbf{1}b_4^\top\right)^\top \tag{19}$$

where $W_4 \in \mathbb{R}^{d_h\times d}, b_4 \in \mathbb{R}^d$.

To summarize, Set Transformer is the set function $\varphi_\lambda \coloneqq g_4\circ g_3\circ g_2\circ g_1 : \mathbb{R}^{n\times d} \to \mathbb{R}^d$ that can handle a set with arbitrary cardinality $n \in \mathbb{N}$.

C.4 Hyperparameters

In Table 17 and 18, we summarize all the hyperparameters for each datasets, where MI stands for Mini-ImageNet-S. For CIFAR-100-FS, we use the same hyperparameters for both 1-hot and 5-shot.

Table 17: Hyperparameters for non-image domains.

Hyperparameters	Metabolism	Tox21	NCI	GLUE-SciTail	ESC-50
learning rate α	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-3}$
optimizer	Adam [21]	Adam	Adam	AdamW [28]	Adam
scheduler	none	none	none	linear	none
batch size	4	4	4	1	4
query size for meta-training	10	10	10	10	5
maximum training iterations	10,000	10,000	10,000	50,000	10,000
number of episodes for meta-test	3,000	3,000	3,000	3,000	3,000
hyper learning rate η	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
hyper optimizer	Adam	Adam	Adam	Adam	Adam
hyper scheduler	linear	linear	linear	linear	linear
update period S	100	100	100	1,000	100
input size for set function d	500	500	500	256	500
hidden size for set function d_h	1,024	1,024	500	1,024	1,024
layer for interpolation l	1	1	1	2	2
iterations for Neumann series q	5	5	5	10	5
distance metric $d(\cdot, \cdot)$	Euclidean	Euclidean	Euclidean	Euclidean	Euclidean

Table 18: Hyperparameters for image domains.

Hyperparameters	RMNIST	MI (1-shot)	MI (5-shot)	CIFAR-100-FS (1-shot, 5-shot)
learning rate α	$1 \cdot 10^{-1}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$	$1 \cdot 10^{-3}$
optimizer	SGD	Adam	Adam	Adam
scheduler	none	none	none	none
batch size	4	4	4	4
query size for meta-training	1	15	15	15
maximum training iterations	10,000	50,000	50,000	50,000
number of episodes for meta-test	3,000	3,000	3,000	3,000
hyper learning rate η	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
hyper optimizer	Adam	Adam	Adam	Adam
hyper scheduler	linear	linear	linear	linear
update period S	1000	1,000	1,000	1,000
input size for set function d	1,568	500	500	256
hidden size for set function d_h	1,568	14,112	56,448	8,192
layer for interpolation l	2	2	1	1
iterations for Neumann series q	5	5	5	5
distance metric $d(\cdot, \cdot)$	Euclidean	Euclidean	Euclidean	Euclidean