# Supplementary material: MAHA

**Anonymous Author(s)**
Affiliation
Address
`email`

## A Explanation on Set Transformer

Set Transformer [14] is proven to be a flexible function approximator that considers a high-order interaction between set elements. It can be decomposed into the following 4 attention modules:

$$\begin{aligned}
&\text{MAB}(A, B) = \text{LN}(H + \text{rFF}(H)) \in \mathbb{R}^{n\times d}\\
&\text{SAB}(A) = \text{MAB}(A, A) \in \mathbb{R}^{n\times d}\\
&\text{ISAB}_m(A) = \text{MAB}(A, \text{MAB}(I, A)) \in \mathbb{R}^{n\times d}\\
&\text{PMA}_k(A) = \text{MAB}(S, \text{rFF}(A)) \in \mathbb{R}^{k\times d}
\end{aligned}$$

where $H = \text{LN}(A + \text{Multihead}(A, B, B))$ is a basic building block for every module. Here, $A, B \in \mathbb{R}^{n\times d}$ are random sets, and $I \in \mathbb{R}^{m\times d}, S \in \mathbb{R}^{k\times d}$ are additional learnable parameters. Note that the randomly initialized inducing points $I$ in ISAB have a lower cardinality than $A$.

A multi-head attention block (MAB) and set attention block (SAB) are the two main key components, which reinforce the multi-head-attention and self-attention with a layer normalization and a skip connection. The induced set attention block (ISAB), with $m$ inducing points, is further devised as a substitute for the SAB in terms of computational efficiency and generalization. The output size is fixed to $k$ by another complex pooling module: pooling by multi-head attention (PMA). Throughout the experiments, 2 ISAB, 1 PMA, 2 SAB are composited in order. $m$ is set to 32 in regression, 256 in classification, and $k$ is set to 1 imitating the mean-pooling operation over *shot* in NP.

## B Distilling an obtainable knowledge from $T$ to $C$

Meta-learning has shown to be vulnerable to overfitting due to task ambiguity [19, 33, 16, 36, 32]. According to [20], there are too many local optima in meta-learning that can lead to bad test results when it comes to a limited number of both *way* and *shot*. Here, we mainly deal with the few-shot nature of meta-learning such that additional regularization terms are devised to distill an obtainable knowledge from $T$ to $C$.

Since the output distribution accounts for a significant portion of the variability of the neural processes [12], we minimize KL divergence between the following output distributions:

$$o(T_y|C) := \mathbb{E}_{q(r|C)}\left[p(T_y|T_x, r, z)\right], \quad o(T_y|T) := \mathbb{E}_{q(r|T)}\left[p(T_y|T_x, r, z)\right]$$

Notice that the deterministic representations are conditioned on the different sets, one from the context set $C$, another from the target set $T$, and we assume that the stochastic representations are given in advance. For a general-purpose, we derive an upper bound as follows since the KL divergence can be computed in a closed-form only in a limited family of probability distributions:

$$\begin{aligned}
KL\left(o(T_y|C)\|o(T_y|T)\right) &= -\int o(T_y|C)\log o(T_y|T)\, dT_y - \mathcal{H}\left(o(T_y|C)\right)\\
&\approx -\log o(\hat{T}_y|T) - \mathcal{H}\left(o(T_y|C)\right) \quad \text{s.t.} \quad \hat{T}_y \sim o(T_y|C)\\
&\leq -\mathbb{E}_{q(r|T)}\left[\log p(\hat{T}_y|T_x, r, z)\right] - \mathcal{H}\left(o(T_y|C)\right)
\end{aligned}$$

where $\mathcal{H}(\cdot)$ indicates entropy. The approximation is conducted using a Monte Carlo sample, and the inequality is from Jensen's inequality on the concave $\log(\cdot)$ function. The first term in the last line is conceptually similar to cross entropy, which leads the model prediction to refer to the pseudo-label which we detach from the computational graph to avoid cycle following [15]. The second term helps the model to avoid overconfidence and degeneracy as discussed in [25, 8, 10]. As a result, the loss function can be rewritten as follow by augmenting the regularization term :

$$\mathcal{L}_{KD} = -\mathbb{E}_{q(r|C)q(z|T)}\left[\log p(T_y|T_x, r, z)\right] + \beta_3 \cdot KL\left(q(z|T))\|q(z|C)\right)$$
$$- \beta_3 \cdot \left(\mathbb{E}_{q(r|T)q(z|T)}\left[\log p(\hat{T}_y|T_x, r, z)\right] + \mathbb{E}_{q(z|T)}\left[\mathcal{H}\left(o(T_y|C)\right)\right]\right) \quad (1)$$

Note that it can bridge to studies on knowledge distillation, specifically a self-distillation [34, 9, 30, 13], where a network is trained not only with the true output $T_y$, but also with the soft output $\hat{T}_y$ that is estimated by the network itself.

# C Implementation details

## C.1 Dataset

**Gaussian Process** A batch of size 16, context set of variable size ranged from 5 to 10, and target set of size 30 are considered. For the squared exponential kernel $k(x, x') = \sigma^2 \exp\left(-0.5(x - x')^2/l^2\right)$, the hyperparameters are chosen to be $l = 0.5$ and $\sigma = 1$. Inputs are uniformly sampled from [-2.0, 2.0] and outputs are computed based on the Cholesky decomposition of the kernel with the noise parameter $\sigma_n = 0.02$ [21].

**Sine**&**Polynomial** A batch of size 25, context set of size 5 or 10, and target set of size 15 or 20 are considered. Input domain is fixed to [-5.0, 5.0], and a task is defined among the four functions whose coefficients are uniformly sampled from the intervals summarized in Table 1.

Table 1: Coefficient settings

| | Sine | Line | Quad | Cubic |
|---|---|---|---|---|
| A | [0.1, 5.0] | [-3.0, 3.0] | [-0.2, 0.2] | [-0.1, 0.1] |
| B | [0.8, 1.2] | [-3.0, 3.0] | [-2.0, 2.0] | [-0.2, 0.2] |
| C | [0.0, $2\pi$] | - | [-3.0, 3.0] | [-2.0, 2.0] |
| D | - | - | - | [-3.0, 3.0] |

**Mini-ImageNet, Tiered-ImageNet** A batch of size 12 is considered where each batch instance is generated by sampling five random classes from the meta set with randomly assigned labels from $\{0, 1, 2, 3, 4\}$. Then, for each of the chosen classes, 1 or 5 images are selected as the context set, and 15 other images are additionally selected to construct the target set.

**Multi-dataset** Task generation process and size of the context/target set are equal to the setting in mini-ImageNet and tiered-ImageNet. However, unlike mini-ImageNet or tiered-ImageNet, images are not pre-processed in advance by the deep residual network. Instead, all images in the meta-train set, meta-valid set, and meta-test set are resized to 84×84×3, and Conv-blocks are utilized to extract the feature from the images. Due to extensive memory usage during the feature extraction, a small batch of size 4 is considered where each batch instance is generated among the four fine-grained image classification datasets.

Table 2: Summary of classification datasets

| Dataset | mini-ImageNet | tiered-ImageNet | Bird | Texture | Aircraft | Fungi |
|---|---|---|---|---|---|---|
| Source | [23] | [23] | [28] | [7] | [18] | [1] |
| Split setting | [26] | [22] | [31] | [31] | [31] | [31] |
| Fineness | Coarse | Coarse | Fine | Fine | Fine | Fine |

## C.2 Architecture design

We show the detailed architectures used for the feature extractor in Table 3. Here, Conv(d, k, s, n, p) is a convolutional block with d output channels, k kernel size, s stride size, n normalization, p pooling method. LRN and BN indicate a local response normalization and a batch normalization, respectively, and MAX is a max-pooling with a kernel size 3 and stride 2. By default, two linear layers are commonly exploited, which come after the convolutional layers if the model input is a 3D image. For the convolutional layers, we follow the exact setting of [31] depending on whether the model is for task clustering or prediction. For the dropout rate $p$, please refer to Section C.3.

Table 3: Feature extractor g($\cdot$) architecture

| Gaussian Process<br>Sine&Polynomial | mini-ImageNet<br>tiered-ImageNet | multi-dataset | |
|---|---|---|---|
| | | 2 Conv | 4 Conv |
| Linear$(1, 128)$ | Dropout($p$) | Conv(32, 5, 1, LRN, MAX) | Conv(32, 3, 1, BN, MAX) |
| ReLU | Linear$(640, 128)$ | Conv(32, 5, 1, LRN, MAX) | Conv(32, 3, 1, BN, MAX) |
| Linear$(128, 128)$ | ReLU | Linear$(32 \times 21 \times 21, 384)$ | Conv(32, 3, 1, BN, MAX) |
| | Linear$(128, 128)$ | ReLU | Conv(32, 3, 1, BN, MAX) |
| | | Linear$(384, 128)$ | Dropout($p$) |
| | | | Linear$(32 \times 5 \times 5, 128)$ |
| | | | ReLU |
| | | | Linear$(128, 128)$ |

In Table 4 and 5, the encoder-decoder pipeline of MAHA is summarized. Note that the encoders for $r$ and $z$ are almost the same except for the output size, which is doubled in $z$ due to reparameterization. Also, note that the size of the inputs is different between regression and classification. This is because $g(X)$ and $Y$ are concatenated in regression while *shot*s of $g(X)$ is first divided along *way* by $Y$ and then separately feed-forwarded in classification. Lastly, in regression, the encoder outputs, $r$ and (reparameterized) $z$, are reshaped from from $[batch, 1, 256]$ into $[batch, 2, 128]$ before feed-forwarded into the decoder. By default, all networks use the Adam optimizer with a constant learning rate and an l2 regularization of weight 1e-4. Please refer to the attached code for actual implementation.

Table 4: Regression architecture

| Encoder | | Decoder |
|---|---|---|
| Enc$_r(\cdot)$ | Enc$_z(\cdot)$ | rFF$(\cdot)$ |
| ISAB$_{32}(128+1, 128)$ | ISAB$_{32}(128+1, 128)$ | Linear$(128, 128)$ |
| ISAB$_{32}(128, 128)$ | ISAB$_{32}(128, 128)$ | ReLU |
| PMA$_1(128, 128)$ | PMA$_1(128, 128)$ | Linear$(128, 128)$ |
| SAB$(128, 128)$ | SAB$(128, 128)$ | |
| SAB$(128, 256)$ | SAB$(128, 512)$ | |

Table 5: Classification architecture

| Encoder | | Decoder |
|---|---|---|
| Enc$_r(\cdot)$ | Enc$_z(\cdot)$ | rFF$(\cdot)$ |
| ISAB$_{256}(128, 128)$ | ISAB$_{256}(128, 128)$ | Linear$(128, 128)$ |
| ISAB$_{256}(128, 128)$ | ISAB$_{256}(128, 128)$ | ReLU |
| PMA$_1(128, 128)$ | PMA$_1(128, 128)$ | Linear$(128, 128)$ |
| SAB$(128, 128)$ | SAB$(128, 128)$ | |
| SAB$(128, 128)$ | SAB$(128, 256)$ | |

### C.3 Hyperparameter

Hyperparameters are optimized with the validation loss of the model trained on the train meta-set. Among the many hyperparameter optimization processes [3, 4, 17, 29], we use the random search whose outcomes are summarized in Table 6 and 9. For the heterogeneous datasets, an agglomerative clustering is applied to the t-SNE embeddings of $\mu_{\bar{z}}$ from the stochastic path where we use the default setting of Scipy [27], an open-source scientific tools for Python. In Table 7 to 8 and Table 10 to 11, the clustering results for randomly generated 2500 (in regression) or 4000 (in classification) data points are presented by cross-tabulation between the cluster index and the true dataset label. For the reason why only two clusters are considered in Sine&Polynomial, please refer to Section D.

Table 6: Hyperparameters for regression

| | Gaussian Process | Sine&Polynomial 5-*shot* | | Sine&Polynomial 10-*shot* | |
|---|---|---|---|---|---|
| epoch (pre) | - | 1e+6 | | 1e+6 | |
| lr (pre) | - | 1e-4 | | 1e-4 | |
| $\beta_1$ | - | 1 | | 1 | |
| Cluster index | | 1st | 2nd | 1st | 2nd |
| epoch | 1e+6 | 1e+6 | 1e+6 | 1e+6 | 1e+6 |
| lr | 1e-4 | 1e-4 | 1e-4 | 1e-4 | 1e-4 |
| $\beta_2$ | 1 | 1 | 1 | 1 | 1 |
| $\beta_3$ | - | 0.120 | 0.941 | 0.156 | 0.224 |

Table 7: 5-*shot* Sine&Polynomial

| Cluster index | Sine | Line | Quad | Cubic |
|---|---|---|---|---|
| 1st | 584 | 9 | 10 | 7 |
| 2nd | 11 | 589 | 614 | 676 |

Table 8: 10-*shot* Sine&Polynomial

| Cluster index | Sine | Line | Quad | Cubic |
|---|---|---|---|---|
| 1st | 587 | 0 | 0 | 0 |
| 2nd | 1 | 598 | 658 | 656 |

Table 9: Hyperparameters for classification

| | mini-ImgeNet 1-*shot* | mini-ImgeNet 5-*shot* | tiered-ImgeNet 1-*shot* | tiered-ImgeNet 5-*shot* | multi-dataset 1-*shot* | | | | multi-dataset 5-*shot* | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| epoch (pre) | - | - | - | - | 1.4e+4 | | | | 1.4e+4 | | | |
| lr (pre) | - | - | - | - | 4.1e-5 | | | | 4.3e-5 | | | |
| $\beta_1$ | - | - | - | - | 0.017 | | | | 0.016 | | | |
| Cluster index | | | | | 1st | 2nd | 3rd | 4th | 1st | 2nd | 3rd | 4th |
| epoch | 5e+4 | 6e+4 | 6e+4 | 7e+4 | 2e+4 | 1.5e+4 | 5e+3 | 2e+4 | 5e+3 | 1e+4 | 1e+4 | 1e+4 |
| lr | 8.7e-5 | 9.2e-5 | 8.8e-5 | 9.9e-5 | 4.4e-5 | 3.0e-5 | 7.8e-5 | 2.5e-5 | 8.1e-5 | 7.1e-5 | 2.9e-5 | 7.9e-5 |
| $p$ | 0.3 | 0.3 | 0.47 | 0.41 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| $\beta_2$ | 0.098 | 0.004 | 0.091 | 0.001 | 0.063 | 0.099 | 0.064 | 0.020 | 0.032 | 0.050 | 0.032 | 0.075 |
| $\beta_3$ | - | - | - | - | 0.085 | 0.068 | 0.063 | 0.051 | 0.018 | 0.085 | 0.028 | 0.053 |

Table 10: 1-*shot* multi-dataset

| Cluster index | Bird | Texture | Aircraft | Fungi |
|---|---|---|---|---|
| 1st | 9 | 0 | 950 | 0 |
| 2nd | 930 | 17 | 45 | 27 |
| 3rd | 11 | 12 | 0 | 921 |
| 4th | 6 | 1023 | 0 | 49 |

Table 11: 5-*shot* multi-dataset

| Cluster index | Bird | Texture | Aircraft | Fungi |
|---|---|---|---|---|
| 1st | 1030 | 0 | 0 | 0 |
| 2nd | 0 | 0 | 0 | 979 |
| 3rd | 0 | 1006 | 0 | 3 |
| 4th | 0 | 0 | 982 | 0 |

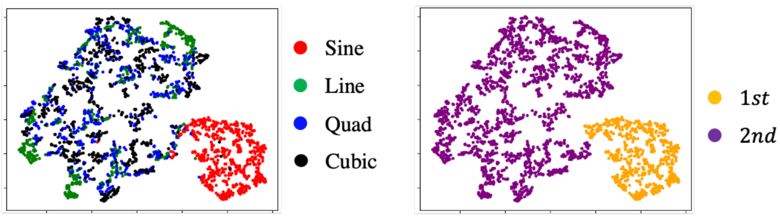
### C.4 Computing Resource

We run experiments on NVIDIA GeForce RTX 2080 Ti, which takes two days for regression and three days for classification. Compared to the basic neural processes, MAHA is about two times slower in terms of the convergence speed, mainly due to the flexible encoder, Set Transformer. However, the compatibility and necessity are empirically verified by the outstanding performance of both homogeneous and heterogeneous datasets. Many follow-up studies are emerging these days to speed up the training of Transformers when applying the attention-based modules [11, 6], which would make our work more valid.

# D Additional experimental results

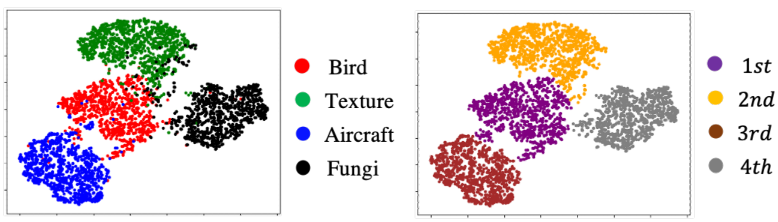
## D.1 Clustering result

In Table 1, note that Quad is perfectly covered by Cubic, and Quad and Cubic mostly cover Line. Hence, rather than four separate clusters, only two are shown in Figure 1, each of which implies either sine or polynomial. On the other hand, in Figure 2, four distinct fine-grained image classification datasets are clearly discriminated by separate clusters.




(a) True label (b) Clustered label

Figure 1: t-SNE visualization of the task representation from Sine&Polynomial




(a) True label (b) Clustered label

Figure 2: t-SNE visualization of the task representation from multi-dataset

## D.2 POOL resolves KL collapse

Although a small subset $C$ of $T$ is expected to reproduce the stochastic representation through the KL divergence in the loss function, the representation inferred by $T$ is restricted to be underutilized, which is the KL collapse [5, 2, 24, 35]. By dimension-wise pooling operations, we intended to prevent $z$ from being redundant by allowing the information flow to go through the stochastic path whenever heterogeneous tasks occur in *batch*. In Figure 3, the training curve of the KL divergence is visualized for many different encoder-decoder pipelines: NP, NP+LD, NP+FE, FELD. It can be observed that the KL collapse no more occurs when accompanied by the pooling operations, which implies that the posterior does not simply converge to the prior.

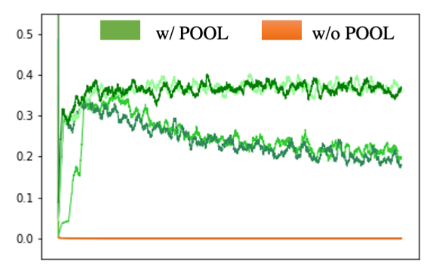



Figure 3: Moving average of KL divergence along epochs

## D.3 POOL needs AE for interpretability

Although the pooling operations are applied to disentangle the task representation from the stochastic path, we observe in the paper that the auto-encoding structure is additionally required to achieve interpretability and high purity values. It is mainly due to the restricted flexibility of $\bar{r}$, which encourages $\bar{z}$ to imply not only the heterogeneity but also the local features that are initially in charge of the deterministic path. Here, the auto-encoding structure allows $r$ to be inferred by the (large) target set $T$, not the (small) context set $C$, which is advantageous to obtain a more flexible set representation. Therefore, the restricted flexibility of $\bar{r}$ can be resolved so that $\bar{z}$ can provide well-clustered and interpretable task representation.

# References


[1] 2018 fgcvx fungi classification challenge. 2018.

[2] Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*, 2016.

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2), 2012.

[4] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[5] Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

[6] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[7] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Vedaldi a.: Describing textures in the wild. In *In Computer Vision and Pattern Recognition (CVPR*. Citeseer, 2014.

[8] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[9] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1013–1021, 2019.

[10] Xu Ji, João F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9865–9874, 2019.

[11] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[12] Tuan Anh Le, Hyunjik Kim, Marta Garnelo, Dan Rosenbaum, Jonathan Schwarz, and Yee Whye Teh. Empirical evaluation of neural process objectives. In *NeurIPS workshop on Bayesian Deep Learning*, 2018.

[13] Hankook Lee, Sung Ju Hwang, and Jinwoo Shin. Self-supervised label augmentation via input transformations. In *International Conference on Machine Learning*, pages 5714–5724. PMLR, 2020.

[14] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pages 3744–3753. PMLR, 2019.

[15] Juho Lee, Yoonho Lee, Jungtaek Kim, Eunho Yang, Sung Ju Hwang, and Yee Whye Teh. Bootstrapping neural processes. *arXiv preprint arXiv:2008.02956*, 2020.

[16] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10657–10665, 2019.

[17] Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[18] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[19] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017.

[20] Eunbyung Park and Junier B Oliva. Meta-curvature. *arXiv preprint arXiv:1902.03356*, 2019.

[21] Carl Edward Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[22] Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*, 2018.

[23] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[24] Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[25] Yee Whye Teh, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *arXiv preprint arXiv:1707.04175*, 2017.

[26] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *arXiv preprint arXiv:1606.04080*, 2016.

[27] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

[28] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011.

[29] James T Wilson, Frank Hutter, and Marc Peter Deisenroth. Maximizing acquisition functions for bayesian optimization. *arXiv preprint arXiv:1805.10196*, 2018.

[30] Chenglin Yang, Lingxi Xie, Chi Su, and Alan L Yuille. Snapshot distillation: Teacher-student optimization in one generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2859–2868, 2019.

[31] Huaxiu Yao, Ying Wei, Junzhou Huang, and Zhenhui Li. Hierarchically structured meta-learning. *arXiv preprint arXiv:1905.05301*, 2019.

[32] Mingzhang Yin, George Tucker, Mingyuan Zhou, Sergey Levine, and Chelsea Finn. Meta-learning without memorization. *arXiv preprint arXiv:1912.03820*, 2019.

[33] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 7343–7353, 2018.

[34] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019.

[35] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models. *arXiv preprint arXiv:1702.08658*, 2017.

[36] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702. PMLR, 2019.