The Curse of Low Task Diversity: On the Failure of Transfer Learning to Outperform MAML and Their Empirical Equivalence

Brando Miranda
University of Illinois Urbana-Champaign
Stanford University
brando9@stanford.edu

Patrick Yu
University of Illinois Urbana-Champaign
pzy@illinois.edu

Yu-Xiong Wang
University of Illinois Urbana-Champaign
yxw@illinois.edu

Sanmi Koyejo
University of Illinois Urbana-Champaign,
Stanford University
sanmi@stanford.edu

Abstract

Recently, it has been observed that a transfer learning solution might be all we need to solve many few-shot learning benchmarks – thus raising important questions about when and how meta-learning algorithms should be deployed. In this paper, we seek to clarify these questions by 1. proposing a novel metric – the *diversity coefficient* – to measure the diversity of tasks in a few-shot learning benchmark and 2. by comparing Model-Agnostic Meta-Learning (MAML) and transfer learning under fair conditions (same architecture, same optimizer, and all models trained to convergence). Using the diversity coefficient, we show that the popular Mini-ImageNet and CIFAR-FS few-shot learning benchmarks have low diversity. This novel insight contextualizes claims that transfer learning solutions are better than meta-learned solutions in the regime of low diversity under a fair comparison. Specifically, we empirically find that a low diversity coefficient correlates with a high similarity between transfer learning and MAML learned solutions in terms of accuracy at meta-test time and classification layer similarity (using feature based distance metrics like SVCCA, PWCCA, CKA, and OPD). To further support our claim, we find this meta-test accuracy holds even as the model size changes. Therefore, we conclude that in the low diversity regime, MAML and transfer learning have equivalent meta-test performance when both are compared fairly. We also hope our work inspires more thoughtful constructions and quantitative evaluations of meta-learning benchmarks in the future.

1 Introduction

The success of deep learning in computer vision [Krizhevsky et al., 2012, He et al., 2015], natural language processing [Devlin et al., 2018, Brown et al., 2020], game playing [Silver et al., 2016, Mnih et al., 2013, Ye et al., 2021], theorem proving [Rabe et al., Polu and Sutskever, 2020, Han et al.], code [Chen et al.] and more keeps motivating a growing body of applications of deep learning on an increasingly wide variety of domains. In particular, deep learning is now routinely applied to few-shot learning – a research challenge that assesses a model's ability to learn to adapt to new tasks, new distributions, or new environments. This has been the main research area where meta-learning algorithms have been applied – since such a strategy seems promising in a small data regime due to its

6th Workshop on Meta-Learning at NeurIPS 2022, New Orleans.

potential to *learn to learn* or *learn to adapt*. However, it was recently shown [Tian et al., 2020] that a transfer learning model with a fixed embedding can outperform many modern sophisticated meta-learning algorithms on numerous few-shot learning benchmarks [Chen et al., 2019, 2020, Dhillon et al., 2019, Huang and Tao, 2019]. This growing body of evidence raises the question if researchers are applying meta-learning with the right inductive biases [Mitchell, 1980, Shai Shalev-Shwartz, 2014] and designing appropriate benchmarks for meta-learning. Our evidence suggests this is not the case.

Our work is motivated by the inductive bias that when the diversity of tasks in a benchmark is low, then a meta-learning solution should provide no advantage compared to a minimally meta-learned algorithm – e.g. only fine-tuning the final layer. Therefore, in this work, we quantitatively show that when the task diversity – a novel measure of variability across tasks – is low, then MAML (Model-Agnostic Meta-Learning) [Finn et al., 2017] learned solutions have the same accuracy as a popular transfer learning algorithm – a supervised learned model with a fine-tuned final linear layer. We want to emphasize the importance of doing such an analysis fairly: with the same architecture, same optimizer, and all models trained to convergence. We hypothesize this was lacking in previous work [Chen et al., 2019, 2020, Dhillon et al., 2019, Huang and Tao, 2019]. This empirical equivalence remained true even as the model size changed – thus further suggesting this equivalence is more a property of the data than of the model. Therefore, we suggest taking a problem-centric approach to meta-learning and suggest applying Marr's level of analysis [Hamrick and Mohamed, 2020, Marr, 1982] to few-shot learning – to identify the family of problems suitable for meta-learning. We do this by analyzing the intrinsic diversity of the data.

Our contributions are summarized as follows:

1. *We propose a novel metric that quantifies the* ***intrinsic diversity*** *of the data of a few-shot learning benchmark*. We call it the *diversity coefficient*. It enables analysis of meta-learning algorithms through a *problem-centric framework*.
2. *We analyze the two most prominent few-shot learning benchmarks – MiniImagenet and Cifar-fs – and show that their diversity is low.*
3. *With this context, we clarify the previous results from [Tian et al., 2020] and show that MAML and transfer learning have statistically equivalent test performance under a fair comparison.* We define a fair comparison when the two methods are compared using the same architecture (backbone), same optimizer, and all models trained to convergence. This holds even as the model size increases.
4. *As an actionable conclusion, we provide a metric that can be used to analyze the intrinsic diversity of the data in a few-shot learning benchmarks and if it measures to a low value then the suggestion is to use simple transfer learning methods*.

We hope that this line of work inspires a problem-centric first approach to meta-learning – especially a **quantitative** approach to benchmark creation instead of focusing on making huge data sets.

2 Background

Model-Agnostic Meta-Learning (MAML): The MAML algorithm [Finn et al., 2017] attempts to meta-learn an initialization of parameters for a neural network so that it is primed for fast gradient descent adaptation. It consists of two main optimization loops: 1) an outer loop used to prime the parameters for fast adaptation, and 2) an inner loop that does the fast adaptation. During meta-testing, only the inner loop is used to adapt the representation learned by the outer loop.

Transfer Learning with Union Supervised Learning (USL): Previous work [Tian et al., 2020] shows that an initialization trained with supervised learning, on a union of all tasks, can outperform many sophisticated methods in meta-learning. In particular, their method consists of two stages: 1) first they use a union of all the labels in the few-shot learning benchmark during meta-training and train with standard supervised learning (SL), then 2) during the meta-testing, they use an inference method common in transfer learning: extract a fixed feature from the neural network and fully fine-tune the final classification layer (i.e., the head) with LBGFS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm).

Task2Vec Embeddings for Distance computation between Tasks: The diversity coefficient we propose is the expectation of the distance between tasks (details in section 3). Therefore, it is essential to define the distance between different pairs of tasks. We focus on the cosine distance between Task2Vec (vectorial) embeddings as in [Achille UCLA et al., 2019]. The vectorial representation of tasks provided by Task2Vec [Achille UCLA et al., 2019] is the vector of diagonal entries of the Fisher Information Matrix (FIM) given a fixed neural network as a feature extractor – also called a **probe network** – after fine-tuning the final classification layer to the task. The authors explain this is a good vectorial representation of tasks because 1. It approximately indicates the most informative weights for solving the current task (up to a second order approximation) 2. For rich probe networks like CNNs, the diagonal is more computationally tractable. The Task2Vec embedding of task τ is the diagonal of the following matrix:

$$\hat{F}_{D_\tau, f_w} = \hat{F}(D_\tau, f_w) = Diag\left(\mathbb{E}_{x,y\sim\hat{p}(x|\tau)p(y|x,f_w)}[\nabla_w \log p(y \mid x, f_w)\nabla_w p(y \mid x, f_w)^\top]\right) \quad (1)$$

where f_w is the neural network used as a feature extractor with architecture f and weights w, $\hat{p}(x \mid \tau)$ is the empirical distribution defined by the training data $D_\tau = \{(x_i, y_i)\}_{i=1}^n$ for task τ, and $p(y \mid x, f_w)$ is a deep neural network trained to approximate the (empirical) posterior $\hat{p}(y \mid x, \tau)$.

3 The Diversity Coefficient

The diversity coefficient aims to measure the intrinsic diversity (or variability) of tasks in a few-shot learning benchmark.

3.1 Ground Truth Distribution Based Diversity

We propose that the diversity coefficient should measure some aggregation of the distance of tasks from a few-shot learning benchmark. If the tasks are distributions then the measure should aggregate the distance between probability distributions. Therefore when the ground truth distributions are available, we propose the diversity coefficient to be the expected value of distances between the task probability distribution (see section 3.2, 2 for notation details):

$$div(B) = \mathbb{E}_{\tau_1\sim p(\tau|B),\tau_2\sim p(\tau|B):\tau_1\neq\tau_2}\left[d(p(x_1, y_1 \mid \tau_1), p(x_2, y_2 \mid \tau_2))\right] \quad (2)$$

3.2 Task Based Diversity Coefficient (with Task2Vec)

For real data the diversity coefficient is the expected distance between embeddings of different tasks **given a fixed probe network**. In this work, we choose the distance to be the cosine distance between vectorial representations of tasks according to Task2Vec as described in section 2. We define the **diversity coefficient** of a few-shot learning benchmark B as follows:

$$\hat{div}(B) = \mathbb{E}_{\tau_1\sim\hat{p}(\tau|B),\tau_2\sim\hat{p}(\tau|B):\tau_1\neq\tau_2}\mathbb{E}_{D_1\sim\hat{p}(x_1,y_1|\tau_1),D_2\sim\hat{p}(x_2,y_2|\tau_2)}\left[d(\hat{F}_{D_1,f_w}, \hat{F}_{D_2,f_w})\right] \quad (3)$$

where f_w is the neural networks used as a feature extractor with architecture f and weights w, $\hat{p}(x \mid \tau)$ is the empirical distribution defined by the training data $D_\tau = \{(x_i, y_i)\}_{i=1}^n$ for task τ, τ_1, τ_2 are tasks sampled from the empirical distribution of tasks $\hat{p}(\tau \mid B)$ for the current benchmark B (i.e. a batch of tasks with their data sets $\mathcal{D} = (\tau_i, D_{\tau_i})_{i=1}^N$), a task τ_i is the probability distribution $p(x, y \mid \tau)$ of the data, d is a distance metric (for us cosine), f_w is the neural networks used as a feature extractor with architecture f and weights w, and $\hat{p}(x \mid \tau)$ is the empirical distribution defined by the training data $D_\tau = \{(x_i, y_i)\}_{i=1}^n$ for task τ. We'd also like to recall to the reader that the definition of a task in this setting is of an n-way, k-shot few-shot learning task. Therefore, each task has n classes sampled with k examples used for the adaptation.

3.3 Justification for Task Based Diversity Coefficient with Task2Vec

Previous work showed Task2Vec is a solid task embedding method because: 1. it's able to predict task similarities that match human semantic relations between different visual tasks [Achille UCLA et al., 2019] 2. Task2Vec distances correlates positively with taxonomical distances [Achille UCLA et al., 2019] 3. importantly, Task2Vec is based on the FIM – which is a Riemannian metric on the space of probability distributions [Amari and Nagaoka, 2007]. This implies that distances based

on Task2Vec are good proxy to ground truth distances between task distribution. We confirm this in figure 11 where we should a strong correlation (with Pearson $r = 0.990$) of Task2Vec diversity compared to ground truth diversity (using the Hellinger distance).

We want to emphasize that one of our main contributions is the novel use of Task2Vec in non-trivial ways to analyze transfer and meta-learning algorithms.

4 Experiments

4.1 Experiments: Low Diversity Correlates with Equivalence of MAML and Transfer Learning when the diversity is low

We show that the diversity coefficient of the popular MiniImagenet and Cifar-fs benchmarks are low with good confidence intervals using four different probe networks in the left most plot of figure 1. We argue it's low because the diversity values are in the interval $[0.06, 0.117]$ – and the minimum and maximum values of the cosine distance are 0.0 and 1.0. The rightmost plot in figure 1 shows the failure of transfer learning (with USL) to outperform MAML. Finally, we study the role of model size. An alternative hypothesis to explain the equivalence of transfer learning (with USL) and MAML could be due to the capabilities of large neural networks to be better meta-learners in general – inspired by the impressive ability of large language models to be few-shot learners [Brown et al., 2020, Bommasani et al., 2021, Radford et al., 2021, Devlin et al., 2018]. In particular, we expect a meta-learned solution like MAML to help more for smaller-size models. We tested this and surprisingly the equivalence between MAML and USL seems to hold even as the model increased – strengthening our hypothesis that the low task diversity might be a bigger factor explaining our observations 2.

Probe Network	Diversity on MI	Diversity on Cifar-fs
Resnet18 (pt)	$0.117 \pm$ 2.098e-5	$0.100 \pm$ 2.18e-5
Resnet18 (rand)	$0.0955 \pm$ 1.29e-5	$0.103 \pm$ 1.05e-5
Resnet34 (pt)	$0.0999 \pm$ 1.95e-5	$0.0847 \pm$ 3.06e-5
Resnet34 (rand)	$0.0620 \pm$ 8.12e-6	$0.0643 \pm$ 9.64e-6

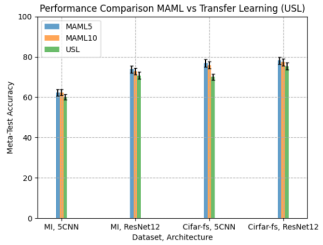

Figure 1: **MAML trained models and union supervised trained (USL) models have statistically equivalent meta-test accuracy for MiniImagenet and Cifar-fs when the diversity is low.** This holds for both the Resnet12 architecture used in [Tian et al., 2020] and the 5 layer CNN (indicated as "5CNN") in [Ravi and Larochelle, 2017]. We used Resnet18 and Resnet34 networks as probe networks – both pre-trained on ImageNet (indicated as "pt" on table) and randomly initialized (indicated as "rand" on table). We used 95% confidence intervals.

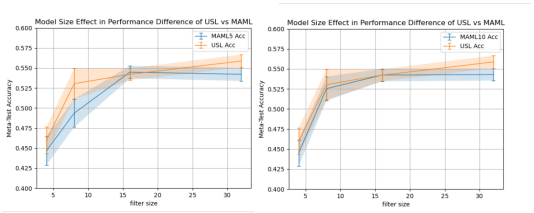

Figure 2: **The meta-test accuracy of MAML and transfer learning using USL is similar in a statistically significant way – regardless of the model size**. We used the MiniImagenet benchmark and the five-layer CNN used in [Finn et al., 2017, Ravi and Larochelle, 2017], and only increased the filter size using sizes 4, 8, 16, and 32. The MAML model was meta-trained using 5 inner steps. The legends MAML5 and MAML10 indicate the test time adaptation method.

Acknowledgments and Disclosure of Funding

We would like to acknowledge the work and the authors of Anatome, Torchmeta, Learn2Learn and Higher [Hataya, 2020, "Arnold et al., 2020, Deleu et al., 2019, Grefenstette et al., 2019] for making their code available and answering ours questions in their project's GitHub repository. We'd like to *especially* thank Sebastien Arnold from the great Learn2Learn meta-learning PyTorch library for great code discussions and for providing a demo of how to use his library for distributed training. We want to acknowledge the weights and biases (wandb) framework for powerful tracking of experiments [Biewald, 2020]. We especially acknowledge Pedro Cisneros-Velarde's insightful feedback on earlier versions of this manuscript, which revealed important subtleties that when clarified improved the paper's message and readability. We also acknowledge Shovik Guha for their feedback to improve earlier versions of this paper. This research was funded in part by a Sloan Fellowship, NSF 2046795, 1909577, 1934986, and NIFA award 2020-67021-32799.

References

Alessandro Achille, Glen Mbeng, and Stefano Soatto. The Dynamic Distance Between Learning Tasks: * From Kolmogorov Complexity to Transfer Learning via Quantum Physics and the Information Bottleneck of the Weights of Deep Networks. 2018. URL `https://arxiv.org/abs/1810.02440`.

Alessandro Achille, Glen Bigan Mbeng, and Stefano Soatto. Dynamics and Reachability of Learning Tasks. 2019.

Alessandro Achille, Giovanni Paolini, Glen Mbeng, and Stefano Soatto. The Information Complexity of Learning Tasks, their Structure and their Distance. 2020.

Alessandro Achille UCLA, Michael Lam AWS, Rahul Tewari AWS, Avinash Ravichandran AWS, Subhransu Maji UMass, Stefano Soatto UCLA, and Pietro Perona Caltech. TASK2VEC: Task Embedding for Meta-Learning Charless Fowlkes UCI and AWS. Technical report, 2019.

Shunichi Amari and Hiroshi Nagaoka. Methods of information geometry. 191, 4 2007. doi: 10.1090/MMONO/191. URL `http://www.ams.org/mmono/191`.

Sébastien M R "Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas" Zarkias. "learn2learn: A library for Meta-Learning research". aug 2020. URL "`http://arxiv.org/abs/2008.12284`".

Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, and William Wang. On the Opportunities and Risks of Foundation Models. aug 2021. URL `https://arxiv.org/abs/2108.07258v2`.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam Mccandlish, Alec Radford, Ilya Sutskever, and Dario Amodei Openai. Language Models are Few-Shot Learners. Technical report, 2020.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating Large Language Models Trained on Code. URL https://www.github.com/openai/human-eval.

Qi Chen, Changjian Shui, and Mario Marchand. Generalization Bounds For Meta-Learning: An Information-Theoretic Analysis. sep 2021. doi: 10.48550/arxiv.2109.14595. URL https://arxiv.org/abs/2109.14595v2.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A Closer Look at Few-shot Classification. *7th International Conference on Learning Representations, ICLR 2019*, 2019. URL http://arxiv.org/abs/1904.04232.

Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A New Meta-Baseline for Few-Shot Learning. Technical report, 2020. URL https://github.com/.

François Chollet. On the Measure of Intelligence. 2019. URL http://arxiv.org/abs/1911.01547.

Tristan Deleu, Tobias Würfl, Mandana Samiei, Joseph Paul Cohen, and Yoshua Bengio. Torchmeta: A Meta-Learning library for PyTorch, 2019. URL https://arxiv.org/abs/1909.06576. Available at: https://github.com/tristandeleu/pytorch-meta.

Giulia Denevi, Massimiliano Pontil, and Carlo Ciliberto. The Advantage of Conditional Meta-Learning for Biased Regularization and Fine-Tuning. *Advances in Neural Information Processing Systems*, 2020-December, aug 2020. ISSN 10495258. doi: 10.48550/arxiv.2008.10857. URL https://arxiv.org/abs/2008.10857v1.

Jacob Devlin, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, 1:4171–4186, oct 2018. URL https://arxiv.org/abs/1810.04805v2.

Guneet S. Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A Baseline for Few-Shot Image Classification. 2019. URL http://arxiv.org/abs/1909.02729.

Frances Ding, Jean-Stanislas Denain, and Jacob Steinhardt. Grounding Representation Similarity with Statistical Testing. 2021. URL https://github.com/js-d/sim_metric.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. 2017. URL http://arxiv.org/abs/1703.03400.

Katelyn Gao and Ozan Sener. Modeling and Optimization Trade-off in Meta-learning. *Advances in Neural Information Processing Systems*, 2020-December, oct 2020. ISSN 10495258. doi: 10.48550/arxiv.2010.12916. URL https://arxiv.org/abs/2010.12916v2.

Micah Goldblum, Steven Reich, Liam Fowl, Renkun Ni, Valeriia Cherepanova, and Tom Goldstein. Unraveling Meta-Learning: Understanding Feature Representations for Few-Shot Tasks. *37th International Conference on Machine Learning, ICML 2020*, PartF168147-5:3565–3574, feb 2020. doi: 10.48550/arxiv.2002.06753. URL https://arxiv.org/abs/2002.06753v3.

Edward Grefenstette, Brandon Amos, Denis Yarats, Phu Mon Htut, Artem Molchanov, Franziska Meier, Douwe Kiela, Kyunghyun Cho, and Soumith Chintala. Generalized inner loop meta-learning. *arXiv preprint arXiv:1910.01727*, 2019.

Yunhui Guo, Noel C. Codella, Leonid Karlinsky, James V. Codella, John R. Smith, Kate Saenko, Tajana Rosing, and Rogerio Feris. A Broader Study of Cross-Domain Few-Shot Learning. dec 2019. URL http://arxiv.org/abs/1912.07200.

Jessica Hamrick and Shakir Mohamed. LEVELS OF ANALYSIS FOR MACHINE LEARNING. 2020.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models.

Ryuichiro Hataya. anatome, a pytorch library to analyze internal representation of neural networks, 2020. URL https://github.com/moskomule/anatome.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016-December:770–778, dec 2015. ISSN 10636919. doi: 10.1109/CVPR.2016.90. URL https://arxiv.org/abs/1512.03385v1.

Shaoli Huang and Dacheng Tao. All you need is a good representation: A multi-level and classifier-centric representation for few-shot learning. 2019. URL http://arxiv.org/abs/1911.12476.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of Neural Network Representations Revisited. Technical report, may 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. 2012. URL http://code.google.com/p/cuda-convnet/.

Ramnath Kumar, Tristan Deleu, and Yoshua Bengio. The Effect of Diversity in Meta-Learning. jan 2022. doi: 10.48550/arxiv.2201.11775. URL https://arxiv.org/abs/2201.11775v1.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building Machines That Learn and Think Like People. *Behavioral and Brain Sciences*, 40, 2016. URL http://arxiv.org/abs/1604.00289.

Kaizhao Liang, Jacky Y Zhang, Boxin Wang, Zhuolin Yang, Oluwasanmi Koyejo, and Bo Li. Uncovering the Connections Between Adversarial Transferability and Knowledge Transferability. 2021.

David Marr. Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. *Phenomenology and the Cognitive Sciences*, 8(4):397, 1982. ISSN 15687759. URL http://mitpress.mit.edu/catalog/item/default.asp?ttype=2&tid=12242&ref=nf.

Brando Miranda. An empirical study of the properties of meta-learning - presentation. *Illinois Digital Environment for Access to Learning and Scholarship (IDEALS)*, dec 2020a. URL https://www.ideals.illinois.edu/handle/2142/109112.

Brando Miranda. An empirical study of the properties of meta-learning - presentation. *Illinois Digital Environment for Access to Learning and Scholarship (IDEALS)*, 2020b. URL https://www.ideals.illinois.edu/handle/2142/109112.

Brando Miranda, Yu-Xiong Wang, and Sanmi Koyejo. Does MAML Only Work via Feature Re-use? A Data Centric Perspective. dec 2021. URL https://arxiv.org/abs/2112.13137v1.

Tom M Mitchell. The Need for Biases in Learning Generalizations by The Need for Biases in Learning Generalizations. 1980.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. 2013.

Ari S. Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. Technical report, 2018.

Stanislas Polu and Ilya Sutskever. Generative Language Modeling for Automated Theorem Proving. sep 2020. URL http://arxiv.org/abs/2009.03393.

Markus N Rabe, Google Research, Dennis Lee, Kshitij Bansal, and Christian Szegedy. Mathematical Reasoning via Self-supervised Skip-tree Training. Technical report.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. feb 2021. URL https://arxiv.org/abs/2103.00020v1.

Aniruddh Raghu, Maithra Raghu, and Samy Bengio. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. Technical report, 2020. URL https://arxiv.org/abs/1909.09157.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. SVCCA: Singular Vector Canonical Correlation Analysis for Deep Learning Dynamics and Interpretability. Technical report, 2017.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. Technical report, 2017.

Elan Rosenfeld, Pradeep Ravikumar, and Andrej Risteski. An Online Learning Approach to Interpolation and Extrapolation in Domain Generalization. feb 2021. doi: 10.48550/arxiv.2102.13128. URL https://arxiv.org/abs/2102.13128v2.

Shai Ben-David Shai Shalev-Shwartz. Understanding Machine Learning: From Theory to Algorithms. *Cambridge University Press*, 2014.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature 2016 529:7587*, 529(7587):484–489, jan 2016. ISSN 1476-4687. doi: 10.1038/nature16961. URL https://www.nature.com/articles/nature16961.

Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B. Tenenbaum, and Phillip Isola. Rethinking Few-Shot Image Classification: a Good Embedding Is All You Need?, 2020.

Eleni Triantafillou, Tyler Zhu, Vincent Dumoulin, Pascal Lamblin, Utku Evci, Kelvin Xu, Ross Goroshin, Carles Gelada, Kevin Swersky, Pierre-Antoine Manzagol, and Hugo Larochelle. Meta-Dataset: A Dataset of Datasets for Learning to Learn from Few Examples. 2019. URL http://arxiv.org/abs/1903.03096.

Ruohan Wang, Massimiliano Pontil, and Carlo Ciliberto. The Role of Global Labels in Few-Shot Classification and How to Infer Them. aug 2021. doi: 10.48550/arxiv.2108.04055. URL https://arxiv.org/abs/2108.04055v2.

David H Wolpert and William G Macready. No Free Lunch Theorems for Optimization. *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, 1(1):67, 1997.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, Yang Gao, Tsinghua University, U C Berkeley, Shanghai Qi, and Zhi Institute. Mastering Atari Games with Limited Data. oct 2021. URL https://arxiv.org/abs/2111.00210v1.

A Synthetic Experiments showing closeness of MAML and Transfer Learning as Diversity Changes

In this section, we show the closeness of MAML and transfer learning (with USL) for synthetic experiments for low and high diversity regimes in Figure 3. In the low regime, the two methods are equivalent in a statistically significant way – which supports the main claims of our paper. As the diversity increases, however, the difference between USL and MAML increases (in favor of USL). This will be explored further in future work.

The tasks are the usual n-way, k-shot tasks, but the data comes from Gaussian classes and the meta-learners are tasked with classifying from which Gaussian class the data points came from in a few-shot learning manner. Benchmarks are created by sampling the parameters of individual classes from a Gaussian benchmark distribution, where class means move farther away from each other and the origin as the diversity of the benchmark increases. Therefore, the Gaussian benchmark with the highest diversity coefficient has Gaussians that are the furthest from each other and the origin. We computed the task diversity coefficients using the Task2Vec method as outlined in Section 3, using a random 3-layer fully connected probe network described in Section F.

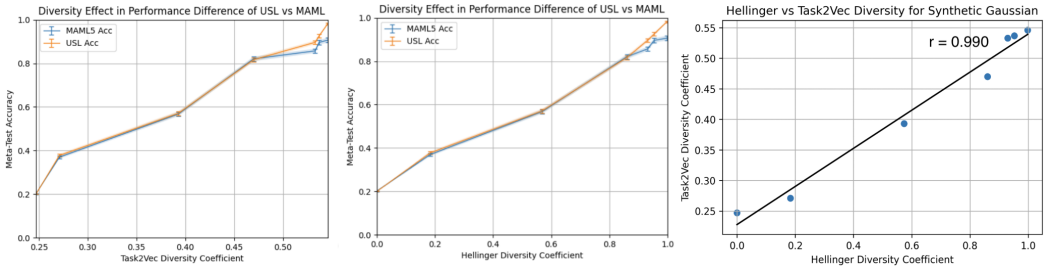

Figure 3: **Left and center plots show the meta-test accuracy of MAML and transfer learning using USL is similar in a statistically equivalent way in the low diversity regime in the 5-way, 10-shot Gaussian Benchmarks. The right most figure shows the strong correlation between Task2Vec diversity and the ground truth Hellinger Diversity.** Results used a (meta) batch-size of 500 tasks, 95% confidence intervals and the Pearson value was $r = 0.990$. MAML5 and MAML10 indicate adaptation at test time of the MAML5 trained model.

B Can we trust the computed diversity coefficient indeed reflects the diversity of a benchmark?

In this section, we argue that of all our low diversity computations are and valid because: 1. the task based diversity using Task2Vec reflects the true diversity as argued in section 3.3 2. our results are robust to the choice of the probe network because Task2Vec depends solely on the task, and ignores interactions with the model [Achille UCLA et al., 2019] – implying our results are valid regardless of the probe network 3. despite the evidence provided by the authors of Task2Vec [Achille UCLA et al., 2019], we nevertheless tested this dependence and used 4 probe networks to compute the task based diversity coefficient 4. the choice of using the expectation was empirically based on the bell shape of the histogram that shows the distribution of task distances (via Task2Vec) in figures 15, 16 17. They were Gaussian shaped, so the expectation of task distances was a sufficient statistic. 5. Although there are a large number of possible tasks based on combinatorial arguments (e.g. $C_5^{64} = 7624512$), 500 samples are enough to make strong statistical inferences about the population. If we assume the distribution of the data is Gaussian, then we expect to see a single mode with an approximate bell curve. If we plot the histogram of task pair distances of the 500 tasks and see this, then we can infer our Gaussian assumption is approximately correct. Given that the histograms in figure 15, 16 17 appear Gaussian, we can infer our assumption is approximately correct. This implies we can make strong statistical assumptions about the population – in particular, that we have a good estimate of the diversity coefficient using 500 samples. 6. The heat maps for MiniImagenet and Cifar-fs were homogeneous (uniform color) as shown in figures 14,13, 12.

C Discussion

In this work, we presented a problem-centric framework when comparing transfer learning methods with meta-learning algorithms – using USL and MAML as the representatives of transfer and meta-learning methods respectively. We showed that the diversity coefficient of the popular MiniImagenet and Cifar-fs benchmark is low and that under a fair comparison – MAML is very similar to transfer learning (with USL) at test time. This was also true even when changing the model size – removing the alternative hypothesis that the equivalence of MAML and transfer learning with USL held due to large models. Instead, this strengthens our hypothesis that the diversity of the data might be the driving factor. The equivalence of MAML and USL was also replicated in synthetic experiments. Therefore, we challenge the suggestions from previous work Tian et al. [2020] that only a good embedding can beat more effective than sophisticated meta-learning – especially in the low diversity regime, and instead suggest this observation might be due to lack of good principles to design meta-learning benchmarks. In addition, our synthetic experiments show a promising scenario where we can systematically differentiate meta-learning algorithms from transfer learning algorithms – which supports our actionable suggestion to use the diversity coefficient to effectively study meta-learning and transfer learning algorithms. In addition, this problematizes the observations that fo-MAML in meta-data set [Triantafillou et al., 2019] is better than transfer learning solutions – since our synthetic experiments show MAML is not better than USL in the high diversity regime. To further problematize, we want to point out that meta-learning methods are not better than transfer learning as observed by [Guo et al., 2019] – as observed in our synthetic experiments. Meaning that further research is needed in both data sets – especially from a problem-centric perspective with quantitative methods like the ones we suggest.

We also have theoretical results from a statistical decision perspective in the supplementary section Q that inspired this work and suggest that when the distance between tasks is zero – then the predictions of transfer learning, meta-learning, and even a fixed model with no adaptation are all equivalent (with respect to the l2 loss).

We hope this work inspires the community in meta-learning and machine learning to construct benchmarks from a problem-centric perspective – that go beyond only large scale data sets – and instead use quantitative metrics for the construction of such research challenges.

We'd like to emphasize that our synthetic experiments are promising because we can systematically differentiate meta-learning algorithms from transfer learning algorithms – which supports our actionable suggestion to: 1. use the diversity coefficient to effectively study meta-learning and transfer learning algorithms, and 2. to use the diversity coefficient to design better benchmarks. In addition, this problematizes the observations that fo-proto-MAML in meta-data set [Triantafillou et al., 2019] is better than transfer learning solutions – since our synthetic experiments show MAML is not better than USL in the high diversity regime. To further problematize, we want to point out that meta-learning methods are not better than transfer learning as observed by [Guo et al., 2019] – as observed in our synthetic experiments. We hypothesize however that the two scenarios in [Triantafillou et al., 2019] are different [Guo et al., 2019]. The first one focuses on the same meta-training and meta-testing conditions, while the latter focuses on a cross-domain. We hypothesize that the cross-domain scenario might benefit from a meta-learning which lower variance (e.g., a fixed embedding [Tian et al., 2020]) – which might explain why sophisticated meta-learning solutions might perform worse on the cross-domain setting as observed in [Guo et al., 2019]. Further research is needed in both benchmarks – especially from a problem centric perspective with quantitative methods like the ones we suggest.

In addition, we hypothesize that diversity might be a good proxy to predict the difference between meta-learning and transfer learning methods. More precisely, we conjecture that in a low diversity setting meta-learning methods are equivalent at meta-test time to transfer learning methods but their difference increases as the diversity of tasks in a benchmark increases. In the high diversity regime, we conjecture that the difference between meta-learning and transfer learning methods increases as the diversity increases. We are optimistic that meta-learning algorithms might outperform transfer learning methods, once we start comparing them in more thoughtfully designed benchmarks. It is possible that despite our efforts, meta-learning algorithms – as currently designed – are too sophisticated and in fact lead to meta-overfitting, as shown in previous work [Miranda et al., 2021].

We'd like to emphasize, that up until now, the meta-learning community has evaluated meta-learning algorithms in benchmarks that might not be the most appropriate. We conjecture high diversity benchmarks are more appropriate, since they might capture the meta-learning inductive prior: high diversity means that adaptation is required by construction. Thus, we conjecture that previous conclusions should be taken with a grain of salt until a more in depth study can be made in the high diversity regime – especially with benchmarks with real world data that have been analyzed extensively with metrics like the diversity coefficient that we propose. We conjecture that we can finally do meta-learning research effectively – given that a regime where meta-learning and transfer learning methods can be differentiated has been discovered, and previous low diversity benchmarks have been understood.

We also conjecture that meta-learning research is different from classical machine learning research. Historically, a seminal paper is the one where AlexNet was proposed [Krizhevsky et al., 2012]. In that time we had low performance on a fixed task e.g., Imagenet and couldn't even interpolate the data (i.e., reach zero train error). We conjecture that meta-learning is different because if we have a diversity so large where all possible tasks are incorporated, then we should reach the no-free lunch theorem regime [Wolpert and Macready, 1997] – where all algorithms should perform the same on average. Therefore, we hypothesize that a deliberate and quantitative efforts to design benchmarks is essential. A great example of such an attempt is the Abstraction and Reasoning Corpus (ARC) benchmark [Chollet, 2019] – which was made very thoughtfully with Artificial General Intelligence (AGI) in mind. We conjecture meta-learning is the most promising path in that direction, and hope this work inspires the design of benchmarks that lead to actionable and deliberate attempts to make progress to build such AGI technologies.

D Related Work

Our work proposes a problem-centric framework for the analysis of meta-learning algorithms inspired from previous puzzling results [Tian et al., 2020]. We propose the use of a pair-wise distance between tasks and analyze how this metric might correlate with meta-learning. The closest line of work for this is the long line of work by [Achille UCLA et al., 2019] where they suggest methods to analyze the complexity of a task, propose unsymmetrical distance metrics for data sets, reachability of tasks with SGD, ways to embed entire data sets and more [Achille UCLA et al., 2019, Achille et al., 2018, 2019, 2020]. We hypothesize this line of work to be very fruitful and hope that more people adopt tools like the ones they suggest and the tools we propose in this paper before researching or deploying meta-learning algorithms. We hope this helps meta-learning methods succeed in practice – since cognitive science suggests meta-learning is a powerful method humans use to learn [Lake et al., 2016]. In the future, we hope to compare [Achille UCLA et al., 2019]'s distance metrics between tasks with ours to provide a further unified understanding of meta-learning and transfer learning. A contrast between their work and ours is that we focus our analysis from a meta-learning perspective applied to few-shot learning – while their focus is understanding transfer learning methods between data sets.

Our analysis of the feature extractor layer is identical to the analysis by [Raghu et al., 2020]. They showed that MAML functions mainly via feature re-use than by rapid learning, i.e., that a model trained with MAML changes very little after the MAML adaptation. The main difference of their work with our is: 1) that we compare MAML trained models against union supervised learned models (USL) instead of only comparing MAML against adapted MAML, and 2) that we explicitly analyzed properties of the data sets. In addition, we use a large set of distance metrics for our analysis including: SVCCA, PWCCA, LINCKA and OPD as proposed by [Raghu et al., 2017, Morcos et al., 2018, Kornblith et al., 2019, Ding et al., 2021].

Our work is most influenced by previous work suggesting modern meta-learning requires rethinking [Tian et al., 2020]. The main difference of our work with theirs is that we analyzed the internal representation of the meta-learning algorithms and contextualize these with quantifiable metrics of the problem being solved. Unlike their work, we focused on a fair comparison between meta-learning methods by ensuring the same neural network backbone was used. Another difference is that they gained further accuracy gains by using distillation – a method we did not analyze and leave for future work.

Another related line of work is the predictability of adversarial transferability and transfer learning. They show this both theoretically and with extensive experiments [Liang et al., 2021]. The main

difference between their work and ours is that they focus their analysis mainly on transfer learning, while we concentrated on meta-learning for few-shot learning. In addition, we did not consider adversarial transferability – while that was a central piece of their analysis.

In terms of benchmarks, we'd like to start with the ARC benchmark [Chollet, 2019]. ARC was designed with AGI in mind – arguably the ultimate meta-learner. Its focus is primarily on visual reasoning using program synthesis techniques. We hypothesize that it's a very promising path, but our work inspires extension that go beyond program synthesis approaches. The meta-data set benchmark is an attempt to make the data set for few-shot learning at a larger scale and more diverse [Triantafillou et al., 2019]. The main difference of their work and ours is that we propose a quantitative metric to measure the intrinsic diversity in the data and go beyond data set size or number of classes. They also showed that a meta-learning algorithms – fo-Proto-MAML – is capable of beating transfer learning. However, they also showed transfer learning baselines are in fact quite difficult to beat. The IBM Cross-Domain few-shot learning benchmark [Guo et al., 2019] is a fascinating benchmark to evaluate meta-learning algorithms. Their central premise however is to transfer from a source domain to a different target domain – instead of our setting where tasks are created from the same meta-distribution. This is why their paper is considered, in addition to few-shot learning, a *cross-domain benchmark*. We believe this is an essential scenario to think about, but consider it different from our setting or the setting of meta-data set. We'd also like to emphasize that they do not employ a metric like our diversity coefficient that quantitatively assesses the diversity of their benchmarks. These two last benchmarks, although fascinating, are missing the essential quantitative analysis of the data itself we are trying to propose.

The work by [Chen et al., 2021] gives, to the best of our knowledge – the first non-vacuous generalization bounds for the (supervised) meta-learning setting. Their statements apply to a non-convex loss function and use stability theory at the task level. The bound depends on the mutual information on the input data vs the output data of the meta-learner. The results, although fascinating, are not built to separate classes of meta-learning – like our work attempts to do empirically.

The work by [Wang et al., 2021] proposes the idea of global labels as a way to indirectly optimize for the meta—learning objective for a fixed feature extractor. Global labels are equivalent to the concept we call USL in this paper. They show that pre-training (i.e. using USL/global labels) provides excellent meta-test results – including with their method (named MeLa) that can infer global labels given only local labels provided at in episodic meta-training. Their theoretical analysis depends on a fixed feature extractor, instead of considering the whole end-to-end meta-learner as a whole – meaning two different deep learning models cannot be used in their analysis. Therefore, their analysis fails to separate how different feature extractors might be trained, e.g. comparing USL vs MAML directly in an end-to-end fashion. In contrast, we instead tackle this question head on theoretically Q (with limited results) but instead show that the feature extractors indeed are different empirically G.

The work by [Denevi et al., 2020] proposes a theoretical treatment of meta-learning using meta-learners with closed-form equations derived from ridged regularization using fixed features. They formulate the conditional and unconditional formulation using side information for the task (e.g. the support set) and show the conditional method is superior. In relation to our work, they do not provide characterizations of the role of a neural network doing end-to-end meta-learning (in their empirical or theoretical analysis). In contrast, our findings make an explicit effort in understanding the role of the neural network in meta-learning in an end-to-end fashion through emperical analysis. Another contrast is that their results are highly theoretical, while ours focus on empirical results. In addition, their results are on synthetic experiments and do not explore their findings in the context of modern few-shot learning benchmarks like MiniImagenet or Cifar-fs.

The work by [Goldblum et al., 2020] provide strong evidence that adaptation at test time is best done when the meta-trained model matches the adaptation it was meta-trained with. This is shown because their classically pre-trained nets cannot perform better than the MetaOpt models with any fine-tuning method. However, their results cannot beat [Tian et al., 2020] and thus does not help separate the role of meta-training and union supervised learning (USL). Their Resnet12 results do provide further support to our hypothesis that large enough neural networks all perform the same, since 78.63 (Goldblum) vs 79.74 (RFS) have very close errors, in line with our findings. However, we hypothesize it is not due to the model size in accordance with our experiments 2.

The work by [Gao and Sener, 2020] provides theoretical bounds of when the expected risk of MAML and DRS (Domain Randomized Search) by bounding the gradient norm. DRS attempts to model

USL but fails to do so completely, because USL is capable of modeling adaptation because the final layer is capable of adaption. Thus, it does not address the capabilities of the feature extractor being able to learn all the information needed to meta-learn. Concisely, their analysis is not capable of separating performance of MAML and USL. Even if it hypothetically could, their analysis remains an upper bounds (with assumptions). This raising the question if their method truly explain the observations that transfer learning methods – like USL – beat meta-learning methods. In addition, they do not provide in depth empirical analysis with respect to any real few-shot learning benchmarks like MiniImagenet or Cifar-fs.

The work by [Kumar et al., 2022] provides an exploration of the effects of diversity in meta-learning. The main difference with our work is that they focus mostly on sampling strategies, and it's effect on diversity, while we focused on the intrinsic diversity in the benchmarks themselves.

The work by [Rosenfeld et al., 2021] provides a theoretical analysis on the difference between interpolation and extrapolation in transfer learning (and domain generalization). We believe this type of theory may be helpful as an inspiration to explore why in the high diversity regime there seems to be a difference between the performance of meta-learning and transfer learning methods.

A related line of work [Miranda, 2020b,a] first showed that there exist synthetic data sets that are capable of exhibiting higher degrees of adaptation as compared to the original work by [Raghu et al., 2020]. The difference is that they did not compare MAML models against transfer learning methods like we did here. Instead, they focused on comparing adapted MAML models vs. unadapted MAML models.

E Convergence of Learning Curves for Fair Comparison

E.1 Convergence of Learning Curves for MiniImagenet and Cifar-fs

In this section, we show that learning curves for the models used in figure 1 achieve convergence. Note that the learning curves for the models trained with MAML look noisier because the distributed training reduces the (meta) batch size for logging purposes. In addition, due to episodic meta-training, (meta) batch sizes have to be smaller compared to batch sizes used in USL.

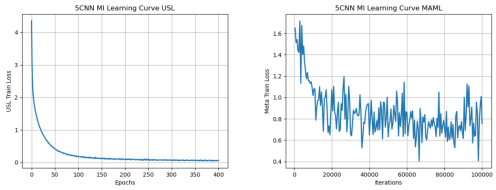

Figure 4: Plot showing convergence of 5CNN on MiniImagenet.

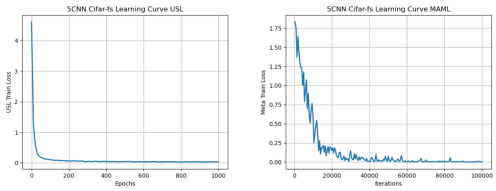

Figure 5: Plot showing convergence of 5CNN on Cifar-fs.

F Experimental Details

F.1 Experimental Details on MiniImagenet and Cifar-fs

We will explain the details for the four models we trained on figure 1.

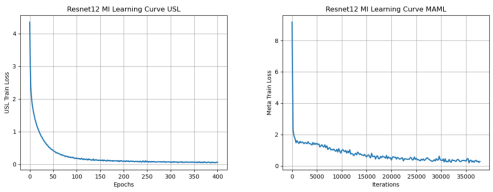

Figure 6: Plot showing convergence of Resnet12 on MiniImagenet.

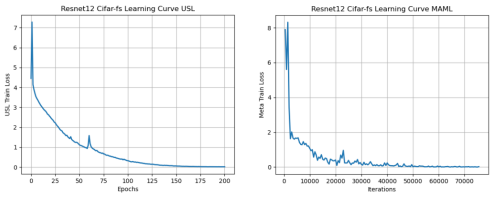

Figure 7: Plot showing convergence of Resnet12 on Cifar-fs.

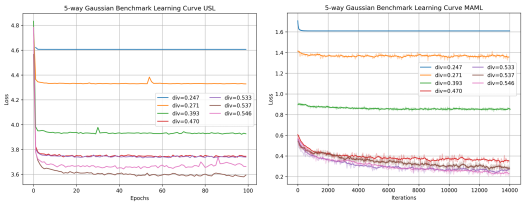

Figure 8: Plot showing convergence of a custom 3-layer fully connected network, for all synthetic Gaussian benchmarks tested. Each curve represents a different synthetic Gaussian benchmark tested on either MAML (left plot) or USL (right plot). The curves are then color-coded by the value of the Task2Vec-based task diversity coefficient of the Gaussian benchmark tested in that particular run.

Summary: We trained a five layer CNN (5CNN) and Resnet12 on both MiniImagenet and Cifar-fs to convergence. We used the Adam optimizer with learning rate 1e-3. We used the standard MiniImagenet and Cifar-fs data augmentations as provided in ["Arnold et al., 2020] matching [Tian et al., 2020].

Experimental Details for 5CNN on MiniImagenet: We used the five layer CNN from [Finn et al., 2017, Ravi and Larochelle, 2017]. We used 32 filters as used in previous work. We used the Adam optimizer with learning rate 1e-3 for both MAML and USL. We used no scheduler. We trained the USL model for 1000 epochs. We trained the MAML model for 100,000 episodic iterations (outer loop iterations). We used a batch size of 128 for USL and a (meta) batch size of 8 for MAML. For MAML we used an inner learning rate of 1e-1 and 5 inner learning steps. We did not use first order MAML. It took 3 hours 5 minutes 5 seconds to train USL to convergence with a single GPU. It took 1 day 6 hours 21 minutes 8 seconds to train MAML to convergence with 4 NVIDIA GeForce GTX TITAN X GPUs.

Experimental Details for Resnet12 for MiniImagenet: We used the Resnet12 provided by [Tian et al., 2020]. We used the Adam optimizer with learning rate 1e-3 for both MAML and USL. We used the same cosine scheduler as in [Tian et al., 2020] for USL and no cosine scheduler for MAML. We trained the USL model for 186 epochs. We trained the MAML model for 37,800 episodic iterations (outer loop iterations). We used a batch size of 512 for USL and a (meta) batch size of 4 for MAML. For MAML we used an inner learning rate of 1e-1 and 4 inner learning steps. We did not use first order MAML. It took 1 day 17 hours 2 minutes 41 seconds to train USL to convergence with a single dgx A100-SXM4-40GB GPU. The MAML model was trained with Torchmeta [Deleu et al., 2019] which didn't support multi gpu training when we ran this experiment, so we estimate it took 1-2 weeks to train on a single GPU. In addition, it was ran with an earlier version of our code, so we

unfortunately did not record the type of GPU but suspect it was either an A100, A40 or Quadro RTX 6000.

Experimental Details for 5CNN for Cifar-fs: We used the five layer CNN from [Finn et al., 2017, Ravi and Larochelle, 2017] provided by ["Arnold et al., 2020]. But we used 1024 filters instead of 32 (to speed up convergence). We used the Adam optimizer with learning rate 1e-3 for both MAML and USL. We used the same cosine scheduler as in [Tian et al., 2020] for MAML and no cosine scheduler for USL. We trained the USL model for 1000 epochs. We trained the MAML model for 100,000 episodic iterations (outer loop iterations). We used a batch size of 256 for USL and a (meta) batch size of 8 for MAML. For MAML we used an inner learning rate of 1e-1 and 5 inner learning steps. We did not use first order MAML. It took 10 hours 43 minutes 31 seconds to train USL to convergence with a single GPU dgx A100-SXM4-40GB. It took 2 days 9 hours 26 minutes 27 seconds to train MAML to convergence with 4 Quadro RTX 6000 GPUs.

Experimental Details for Resnet12 for Cifar-fs: We used the Resnet12 provided by [Tian et al., 2020]. We used the Adam optimizer with learning rate 1e-3 for both MAML and USL. We used the cosine scheduler used in [Tian et al., 2020] for both USL and MAML. We trained the USL model for 200 epochs. We trained the MAML model for 75,500 episodic iterations (outer loop iterations). We used a batch size of 1024 for USL and a (meta) batch size of 8 for MAML. For MAML we used an inner learning rate of 1e-1 and 5 inner learning steps. We did not use first order MAML. It took 45 minutes 54 seconds to train USL to convergence with a single GPU. It took 1 day 19 hours 29 minutes 31 seconds to train MAML to convergence with 4 dgx A100-SXM4-40GB GPUs.

Why the Adam optimizer? We hypothesize that the Adam optimizer is the most appropriate optimizer for a fair comparison for various reasons. First, the Adam optimizer is widely used – making our results most relevant and broadly applicable. Adam is generally a stable optimizer – especially for sophisticated meta-learning algorithms like MAML. It is not uncommon to have SGD result in exploding gradients or end up diverging – especially for MAML. Most importantly however, we hope to stay faithful whenever possible to how modern transformer models are trained – because they have been shown to be good meta-learners, e.g., GPT-3 is often cited as a zero-shot learner [Brown et al., 2020]. These type of models do use more complicated learning schemes besides only Adam (e.g., warm-ups, decay rates etc.) but we hypothesize using Adam is a good first step. We conjecture that the small benefits that SGD might provide are negligible compared to the stability that Adam provides, especially as the scale of the data sets starts to increase. Without Adam, we conjecture it would be hard to even perfectly fit the data for large scale data sets as it's usually done in Deep Learning. This was definitively true in our own experiments. Therefore, we decided to use Adam for our experiments, since it would be too hard to use SGD reliably at scale or with sophisticated meta-learning algorithms.

Details for left most plot on figure 1: Results on the left most plot used a (meta) batch-size of 500 tasks and 95% confidence intervals 1. All MAML models were trained with 5 inner steps during meta-training. MiniImagenet is abbreviated as "MI" in the figure. We used 20 shots (number of examples per class) since we can use the whole task data to compute the diversity coefficient (no splitting of support and query set required for the diversity coefficient). We used Resnet18 and Resnet34 networks as probe networks – both pre-trained on ImageNet (indicated as "pt" on table) and randomly initialized (indicated as "rand" on table). We observe that both type of networks and weights give similar diversity results.

Details for right most plot on figure 1: "MAML5" and "MAML10" in the bar plot indicates the adaptation method used at test time i.e. we used 5 inner steps and 10 inner steps at test time. MiniImagenet is abbreviated as "MI" in the figure. All confidence intervals were at 95%. We used 500 few-shot learning tasks. This results in $(500^2 - 500)/2 = 124,750$ pair-wise distances used to compute the diversity coefficient. We used 95% confidence intervals.

F.2 Experimental Details on N-way Gaussian Tasks

We used a custom 3-layer fully connected network derived from Learn2Learn's OmniglotFC model ["Arnold et al., 2020], with parameters $input_size = 1$, $output_size = 5$, and hidden layer sizes $sizes = [128, 128]$. We used the Adam optimizer with learning rate 1e-3 for both MAML and USL, but did not use a cosine scheduler for either USL or MAML. We trained the USL model for 100 epochs. We trained the MAML model for 14,000 episodic iterations (outer loop iterations). We used

a batch size of 100 for USL and a (meta) batch size of 100 for MAML. For MAML we used an inner learning rate of 1e-1 and 5 inner learning steps. We did not use first order MAML. It took 19 minutes 24 seconds to train USL to convergence with a single Titan X. It took 2 days and 13 hours 6 minutes 27 seconds to train MAML to convergence with a single Titan X.

F.2.1 Experimental details of empirical equivalence figure in main body

Experimental details for figure 11. MAML models were trained with 5 inner steps. MAML5 and MAML10 indicate the adaptation procedure at test time. Results used a (meta) batch-size of 500 tasks and 95% confidence intervals. As the diversity of the benchmark increases, the Gaussian tasks are sampled further away from the origin. Note, as the diversity increases, the difference between USL and MAML increases (in favor of USL).

F.3 Experimental Details on 5CNN test performance equivalence on MiniImagenet

For figure 1 we used Logistic Regression (LR) with LBFGS with the default value for the l2 regularization parameter given by Python's Sklearn. Note that an increase in inner steps from 5 to 10 with the MAML5 trained model does not provide an additional meta-test accuracy boost, consistent with previous work [Miranda, 2020a]. Note that the fact that the MAML5 representation matches the USL representation when both use the same adaptation method is not surprising – given that: 1) previous work has shown that the distance between the body of an adapted MAML model is minimal compared to the unadapted MAML (which we reproduce in 9 in the green line) and 2) the fact that a MAML5 adaptation is only 5 steps of MAML while LR fully converges the prediction layer. We want to highlight that only the MAML5 model achieved the maximum meta-test performance of 0.6 with the MAML5 adaptation – suggesting that the USL and MAML5 meta-learning algorithms might learn different representations. For USL to have a fair comparison during meta-test time when using the MAML adaptation, we provide the MAML final layer learned initialization parameters to the USL model (but any is fine due to convexity when using a fixed feature extractor). This is needed since during meta-training, USL is trained with a union of all the labels (64) – so it does not even have the right output size of 5 for few-shot prediction. Meta-testing was done in the standard 5-way, 5-shot regime.

G Feature Extractor Analysis of USL and MAML

G.1 MAML learns a different base model compared to Union Supervised Learned models – even in the presence of low task diversity

The first four layers of figure 10 shows how large the distance is of a MAML representation compared to a SL representation. In particular, it is much larger than the distance value in the range $[0, 0.1]$ from previous work that compared MAML vs. adapted MAML [Raghu et al., 2020]. We reproduced that and indeed MAML vs. adapted MAML has a small difference (smaller for us) – supporting our observations that a MAML vs. a USL learned representation show a statistically significant difference at the feature extractor layer, even when the diversity is low.

In this section, we discuss figure 9 that suggests that the feature extractor of MAML trained model is markedly different from a USL model – even when the diversity is low. Figure 9 shows that the initial difference is already much larger than previous work [Raghu et al., 2020] were, $0.0 < 0.1$ and statistically significant. In fact, we reproduce their results from [Raghu et al., 2020] in figure 9 with the green curve. In addition, their original distance remains much larger than our reproduction of the distance of previous work with the green line. This suggests that the adaptation makes even a smaller difference in MAML according to our experiments. This is true for all four metrics: SVCCA, PWCCCA, LINCKA and OPD.

H Classification Layer Analysis of USL and MAML show predictions are similar

The main contribution of this section is to show the similarity of the predictions of USL and MAML through figure 10. This is additional evidence to our claims – through a different metric than accuracy

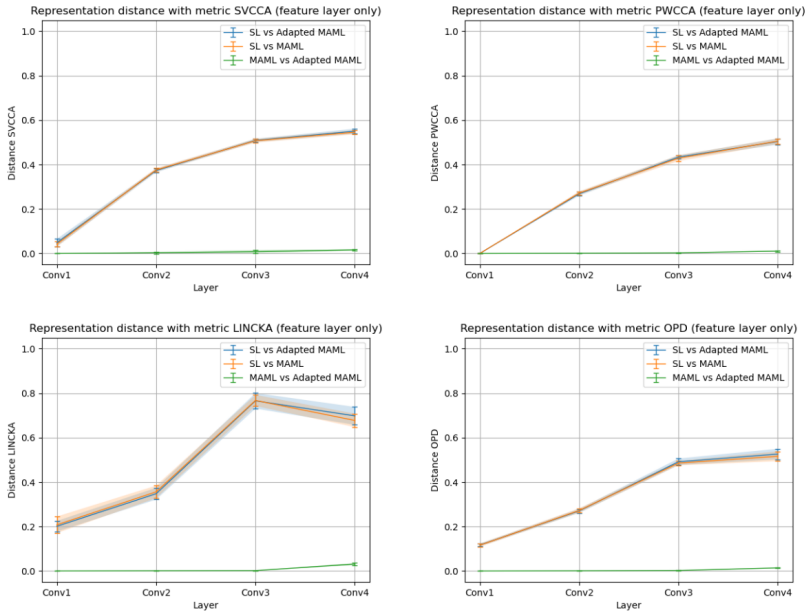

Figure 9: **Shows the significant difference between the feature extractor layers of a MAML trained model vs. a union supervised learned model – especially in contrast to the small change in the adapted MAML model (green line).** This figure suggests that although benchmark diversity is small, a meta-learned representation still learns through a different mechanism than a supervised learned representation. Note that the green line is our reproduction of previous work [Raghu et al., 2020] that showed that a MAML trained model does not change after using the MAML adaptation. They term this observation as "feature re-use".

difference – that USL and MAML are empirically equivalent. For the background on SVCCA, PWCCA, CKA, and, OPD see section O.

I Task2Vec embeddings are also related to the Complexity of a Task

In addition, Task2Vec exhibits impressive properties that are strongly suggestive that it also captures the amount of information about a task. Some of these properties are: 1. the ability to use Task2Vec to select a pre-trained features extractor close to the ground truth optimal expert – while costing substantially less [Achille UCLA et al., 2019]. 2. the correlation of the (nuclear) norm of the Task2Vec embedding with the difficulty of the task and test performance. 3. In addition, Task2Vec (through FIM) is related to the (Kolmogorov) complexity of a task [Achille et al., 2018].

J Background of Few-shot Learning Basics

The goal of few-shot learning is to learn to classify from a limited set of training samples. A few-shot benchmark is utilized to evaluate few-shot learning algorithms and typically contains many classes and a smaller number of samples per class. Typically, few-shot learning algorithms learn in episodes, where in each episode, a task consisting of a train (or support) set and a held-out validation (or query) set is sampled. In particular, a task is a n-way k-shot classification problem, means that the support and query sets each consist of n classes sampled from the benchmark, and each of the n classes are represented by k shots or examples. The learner uses the support set to adapt to the task, and the query set to evaluate the performance on the given task.

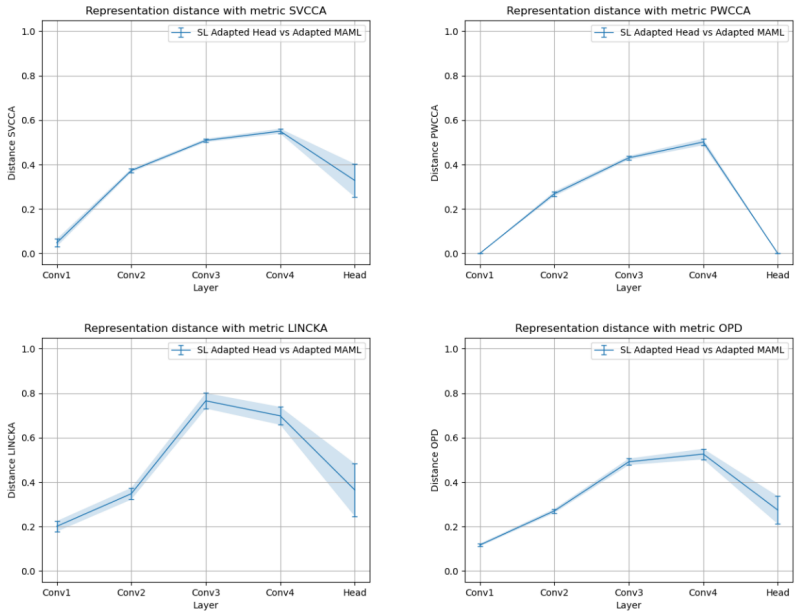

Figure 10: **The classification layer of transfer learning and a MAML5 model decrease in distance – implying similar predictions.** More precisely, an initialization trained with 5 inner steps (MAML5) has an increasingly similar head (classifier) after adaptation with MAML5 compared to the classifier layer of the Union Supervise-Learned (USL) model that has been adapted only at the final layer. In particular, the USL model has been adapted with Logistic Regression (LR) with LBFGS with the default value for the l2 regularization parameter given by Python's Sklearn (as in [Tian et al., 2020]). We showed this trend with four different distance metrics - SVCCA, PWCCA, LICKA, and OPD - as referenced in section 2. Observe that according to PWCCA, the distance between the predictions is zero. This is true because the distance of classification layer (indicated as "head" in the figure) is zero. The architecture used here is a five layer CNN as in [Finn et al., 2017, Ravi and Larochelle, 2017] with their same setup. The benchmark used for this analysis is MiniImagenet.

K Synthetic Gaussian Benchmark and N-way Gaussian Tasks

We create a series of synthetic few-shot benchmarks, where each Gaussian benchmark B is defined by four parameters $B = (\mu_m, \sigma_m, \mu_s, \sigma_s)$. To form the dataset of our benchmark, we first sample 100 meta-train, 100 meta-test, and 100 meta-validation classes, where class $1 \leq i \leq 300$ is a Gaussian parameterized by $(\mu_{class_i}, \sigma_{class_i})$ where

$$\mu_{class_i} \sim N(\mu_m, \sigma_m), \sigma_{class_i} \sim |N(\mu_s, \sigma_s)|$$

Then, for each class i, we sample 1000 data points $(x_{i,1}, i) \ldots (x_{i,1000}, i)$ where each datapoint (x, y) is composed of a input value $x \in \mathbb{R}$ and class label $1 \leq y \leq 300$. The input values $x_{i,1} \ldots x_{1,1000}$ are each sampled from class i's class distribution:

$$x_{i,1} \ldots x_{i,1000} \sim N(\mu_{class_i}, \sigma_{class_i})$$

Having defined our dataset underlying our benchmark, we may now sample individual tasks from our benchmark. Each task in our benchmark is 5-way, 10-shot - that is, each task is formed by first sampling 5 ways from the benchmark dataset, then sampling 10 shots from each of the 5 ways. The goal of each task is to correctly predict which of the 5 ways an input value $x \in \mathbb{R}$ falls into. We conducted experiments using 7 different benchmarks, with each benchmark defined by four parameters and its corresponding Hellinger distribution diversity coefficient and Task2Vec task diversity coefficient, as listed in Table 1:

Benchmark Parameters $(\mu_m, \sigma_m, \mu_s, \sigma_s)$	Hellinger-based Distribution Diversity	Task2Vec-based Task Diversity
(0, 0.01, 1, 0.01)	7.475e-05 $\pm$ 4.891e-07	0.247 $\pm$ 1.04e-3
(0, 1, 1, 0.01)	0.183 $\pm$ 1.24e-3	0.271 $\pm$ 1.15e-3
(0, 3, 1, 0.01)	0.574 $\pm$ 2.28e-3	0.393 $\pm$ 1.79e-3
(0, 10, 1, 0.01)	0.860 $\pm$ 1.75e-3	0.470 $\pm$ 2.35e-3
(0, 20, 1, 0.01)	0.929 $\pm$ 1.31e-3	0.533 $\pm$ 2.47e-3
(0, 30, 1, 0.01)	0.952 $\pm$ 1.10e-3	0.537 $\pm$ 2.57e-3
(0, 1000, 1, 0.01)	0.998 $\pm$ 2.07e-4	0.546 $\pm$ 2.74e-3

Table 1: **Benchmarks of increasing diversity are created by increasing σ_m, or, the standard deviation of the class mean. This table also shows that as the ground truth diversity increases, so does the Task2Vec diversity – implying the Task2Vec diversity is a good proxy to the ground truth distribution based diversity.** A larger σ_m increases the variance of the class means, making their respective class distributions farther apart on average and causing both the Hellinger-based distribution diversity and Task2Vec-based task diversity coefficients to increase. We varied σ_m from 0.01 to 1000 and fixed all remaining benchmark parameters to obtain 7 different Gaussian benchmarks. The corresponding Hellinger-based distribution diversity coefficients were obtained by numerically approximating the expected Hellinger distance between two classes sampled from the benchmark and computing the 95% confidence interval of the approximation. We also computed Task2Vec-based task diversity coefficients as an alternative measure to diversity using a random 3-layer fully connected probe network described in Section F. Figure 12 visualizes the Task2Vec task diversities among the synthetic benchmarks via a heatmap showing the relative pairwise distance between sampled tasks.

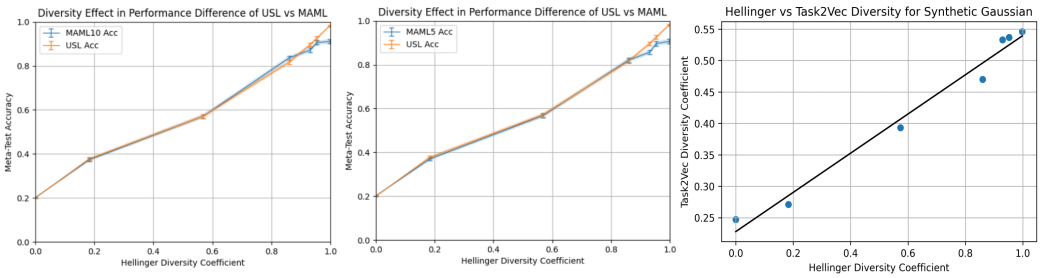

Figure 11: **Shows the strong relation between Hellinger distribution diversity and the Task2Vec task diversity coefficients, as both coefficients may be used interchangeably as a measure for the diversity of a given synthetic Gaussian benchmark** The first two plots show the relation between the Hellinger-based distribution diversity of a synthetic Gaussian benchmark and the benchmark's performance on the MAML5, MAML10, and USL methods. These first two plots are noticeably similar to Figure 3 (where Task2Vec-based task diversity was used as a measure of diversity instead of Hellinger-based distribution diversity), which indicates that our Hellinger-based distribution diversity metric also serves as a good proxy for task diversity. The rightmost plot shows a strong positive correlation between Hellinger-based distribution diversity and Task2Vec task diversity (Pearson $r = 0.990$).

L Distribution-based Diversity Metrics

In addition to the task diversity methods (such as Task2Vec) that we chose as a measure of diversity across our experiments in our main paper, we would like to introduce an additional class of diversity metrics that we call *distribution diversity*. Unlike task diversity, which quantifies diversity through the expected distance between any two distinct tasks sampled from the benchmark, distribution diversity quantifies diversity through the expected distance between any two distinct *distributions* that underlie the benchmark. In our synthetic Gaussian experiments, we define the distribution diversity of our Gaussian benchmark as the expected Hellinger distance between two distinct Gaussian class

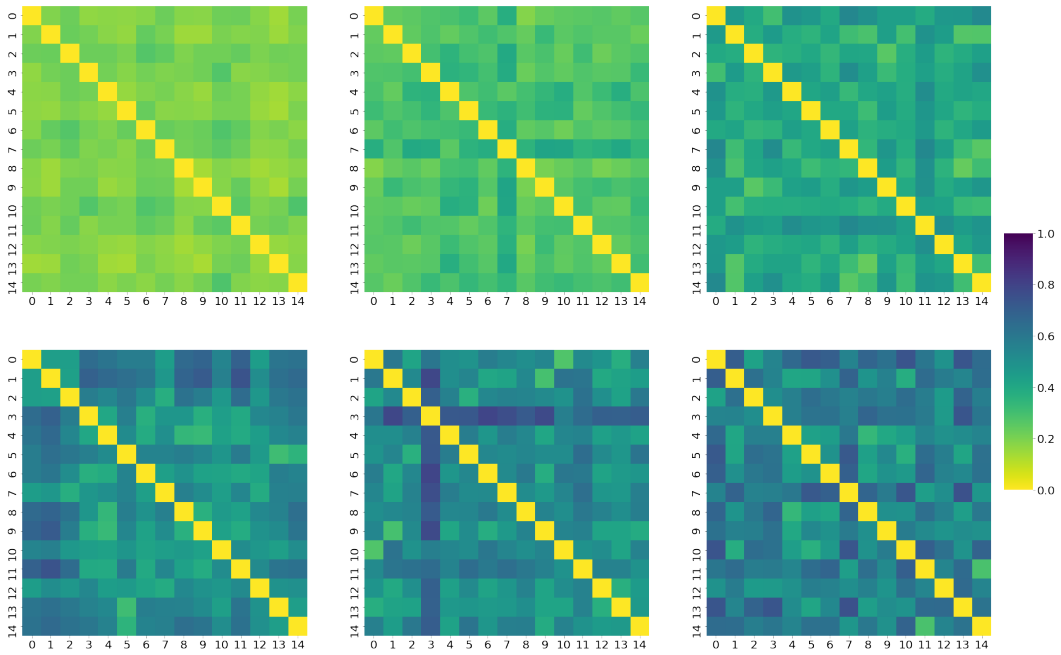

Figure 12: **Heatmaps show how benchmarks with larger Task2Vec-based task diversity coefficient show more heterogeneity between sampled tasks** Each heatmap below shows the pairwise distances between fifteen 5-way, 10-shot few-shot learning tasks sampled from the various synthetic Gaussian benchmarks described in Table 1. Note that as σ_m (the standard deviation of the class mean) increases, the distance between two tasks becomes larger on average and more varied, which can be seen as the heatmaps become more heterogeneous. This increase in expected distance among different tasks in turn increases the Task2Vec-based task diversity coefficient, which summarizes the average distance between any two tasks. From left to right, top to bottom, the benchmarks tested have parameters $\sigma_m = 0.01, 1, 3, 20, 30, 1000$ and Task2Vec-based task diversity coefficient parameters $div = 0.247, 0.271, 0.393, 0.533, 0.537, 0.546$.

distributions sampled from the benchmark - we describe the calculation of the distribution diversity of our synthetic Gaussian benchmark in more detail in Section M.

M Hellinger Diversity Coefficient and Hellinger Distance

An alternative metric to the Task2Vec-based task diversity metric is the Hellinger-based distribution diversity metric. The Hellinger-based distribution diversity of our Gaussian benchmark is obtained by computing the expected Hellinger distance between any two classes sampled from the benchmark. That is, for some benchmark parameterized by $B = (\mu_m, \sigma_m, \mu_s, \sigma_s)$, the diversity coefficient is given by

$$div(B) = \mathbb{E}_{\mu_1,\mu_2 \sim N(\mu_m,\sigma_m)} \mathbb{E}_{\sigma_1,\sigma_2 \sim |N(\mu_s,\sigma_s)|} [H^2(N(\mu_1,\sigma_1), N(\mu_2,\sigma_2))]$$

where H^2 denotes the squared Hellinger distance metric and $N(\mu_1,\sigma_1), N(\mu_2,\sigma_2)$ denote the distributions of the two classes sampled from the benchmark. The Hellinger-based distribution diversity metric provides an intuitive, model-agnostic characterization of the diversity of a benchmark - the larger the diversity, the less similar any two classes within the benchmark are, and the easier it is to distinguish between two classes. Conversely, the lower the diversity, the more similar any two classes within the benchmark are, and the harder it is to distinguish between two classes due to a larger overlap between the two classes' distributions.

Note that the closed-form equation for the Hellinger distance between the two class distributions $N(\mu_1, \sigma_1), N(\mu_2, \sigma_2)$ is given by

$$H^2(N(\mu_1, \sigma_1), N(\mu_2, \sigma_2)) = 1 - \sqrt{\frac{2\sigma_1\sigma_2}{\sigma_1^2 + \sigma_2^2}} e^{-\frac{1}{4}\frac{(\mu_1-\mu_2)^2}{\sigma_1^2+\sigma_2^2}}$$

However, there is no simple closed-form equation for computing the diversity $div(B)$ itself. As a result, we computed the diversity coefficient as a numerical approximation by repeatedly sampling two classes from the benchmark distribution and calculating the Hellinger distance between the two classes. These samples ultimately provide a 95% confidence interval that represents the expected Hellinger distance between two classes sampled from the benchmark.

We also compared our Hellinger-based distribution diversity coefficient with the Task2Vec-based task diversity coefficient for each of the synthetic Gaussian benchmarks tested in Table 1. We observe a strong positive correlation between the Hellinger-based distribution diversity and Task2Vec-based task diversity coefficients according to Figure 11, indicating that the Hellinger-based distribution diversity serves as an effective proxy for task diversity when the number of ways and shots of all tasks are fixed.

N Analysis of distribution of task distances in few-shot learning benchmarks

N.1 Heat Maps show Low Diversity and Homogeneity of tasks from MiniImagenet and Cifar-fs

In this section, we show the heat maps showing the distances between 5-way, 20-shot few-shot learning tasks from MiniImagenet and Cifar-fs in figure 13 and 14. We used 20-shots because we do not need to separate the data into support and query set to compute the diversity coefficient. We show that tasks sampled from these benchmarks create not only a low diversity coefficient on average, but also at the level of individual distances between pairs of tasks. In addition, the heat map's uniform coloring reveals that it is also justifiable to call the tasks from these benchmarks *homogeneous*. Low diversity is shown because the distances are between 0.07-0.12 given that max is 1.0 and minimum is 0.0.

N.2 Histograms of distances of tasks in the synthetic Gaussian Benchmark, MiniImagenet and Cifar-fs

In this section, we show the histograms of the cosine distances between pairs of tasks for the Gaussian Benchmark, MiniImagenet and Cifar-fs. The main purpose of this is to argue that a (relatively small) sample of the tasks is sufficient to estimate population statistics – like the expected distance between tasks i.e. diversity coefficient.

For ease of exposition for the argument, consider the case where we have 500 distance from a large population of size $\binom{64}{5} = 7,624,512$. The goal is to argue that 500 samples are enough to make strong statistical inferences about the population – even if it's as large as $7,624,512$. If we assume the distribution of the data is Gaussian, then we expect to see a single mode with an approximate bell curve. Therefore, if we plot the histogram of task pair distances of the 500 tasks and see this, then we can infer our Gaussian assumption is approximately correct. Given that the 15, 16, 17 appear Gaussian, we can infer our assumption is approximately correct. This implies we can make strong statistical assumptions about the population – in particular, that we have a good estimate of the diversity coefficient using 500 samples. Additionally, in histograms also discard the presence of outlier tasks.

O Background on Deep Neural Network Distance metrics

Distances for Deep Neural Network Feature Analysis: To compute the distance between neural networks, we use the distance versions of Singular Value Canonical Correlation Analysis (SVCCA) [Raghu et al., 2017], Projection Weighted Canonical Correlation (PWCCA) [Morcos et al., 2018], Linear Centered Kernel Analysis (LINCKA) [Kornblith et al., 2019] and Orthogonal Procrustes

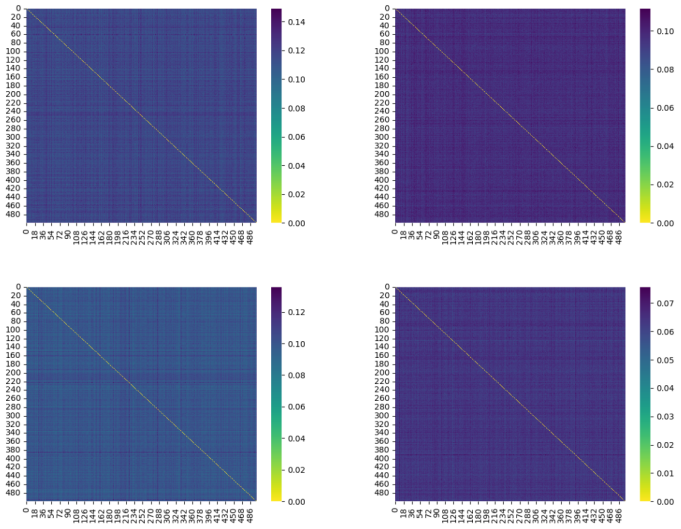

Figure 13: **Shows homogeneity and low diversity of 5-way, 20-shot tasks from MiniImagenet using the Task2Vec distance [Achille UCLA et al., 2019].** The top left heat map uses a Resnet18 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The top right heat map uses a Resnet18 with random weights to compute the Task2Vec distance between tasks. The bottom left heat map uses a Resnet34 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The bottom right heat map uses a Resnet34 with random weights on Imagenet to compute the Task2Vec distance between tasks. Homogeneity is shown because of the uniform color shown in the heat map. Low diversity is shown because the distance is between 0.07-0.12 given that max is 1.0 and minimum is 0.0. Note the diagonal is exactly zero because it is comparing the same tasks. The axis indices indicate the arbitrary name for the tasks. Indices between heat maps do not indicate the same task. We used the cosine distance between task Task2Vec embeddings.

Distance (OPD) [Ding et al., 2021]. These distances are in the interval $[0, 1]$ and are not necessarily a formal distance metric but are guaranteed to be zero when their inputs are equal and nonzero otherwise. This is true because SVCCA, PWCCA, LINCKA are based on similarity metrics and OPD is already a distance. Note that we use the formula $d(X, Y) = 1 - sim(X, Y)$ for our distance metrics where sim is one either SVCCA, PWCCA, LINCKA similarity metric and X, Y are matrices of activations (called layer matrices). The distance between two models is computed by choosing a layer and then comparing the features/activations after adaptation for that layer given a batch of tasks represented as a support and query set.

O.1 Neuron Vectors

The representation of a neuron d in layer l is the vector $z_d^{(l)}(X) \in \mathbb{R}^N$ of activations for a set of N examples, where $X \in \mathbb{R}^{N,D}$ is the data matrix with N examples.

O.2 Layer Matrix

A layer matrix L for layer l is a matrix of neuron vectors $z_d^{(l)}(X) \in \mathbb{R}^N$ with shape, $[N, D_i]$ i.e. $L \in \mathbb{R}^{N,D_i}$. In other words, the layer matrix L is the subspace of $\mathbb{R}^N$ spanned by its neuron vectors $z_d^{(l)}(X)$. In short, L is the layer matrix $[z_d^l; \ldots; z_{D_1}^l] \in \mathbb{R}^{N,D_i}$ with neuron vector z_d^l.

O.3 CCA

Canonical Correlation Analysis (CCA) is a well established statistical technique for comparing the (linear) correlation of two sets of random variables (or vectors of random variables). In the empirical

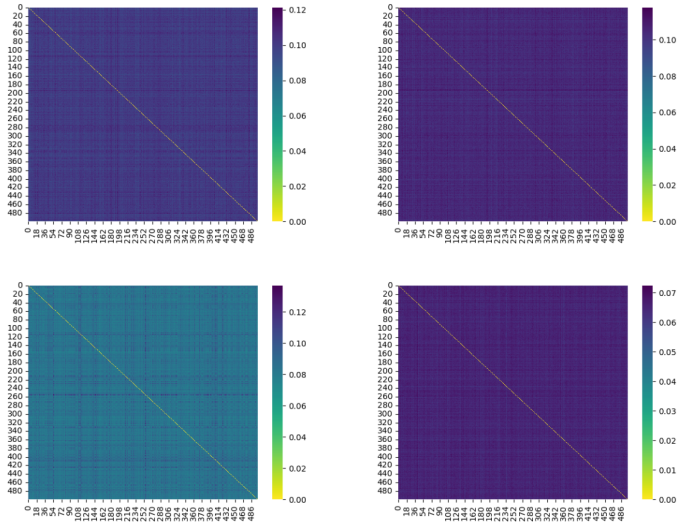

Figure 14: **Shows homogeneity and low diversity of 5-way, 20-shot tasks from Cifar-fs using the Task2Vec distance.** The top left heat map uses a Resnet18 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The top right heat map uses a Resnet18 with random weights to compute the Task2Vec distance between tasks. The bottom left heat map uses a Resnet34 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The bottom right heat map uses a Resnet34 with random weights on Imagenet to compute the Task2Vec distance between tasks. Homogeneity is shown because of the uniform color shown in the heat map. Low diversity is shown because the distance is between 0.07-0.12 given that max is 1.0 and minimum is 0.0. The axis indices indicate the arbitrary name for the tasks. Indices between heat maps do not indicate the same task. We used the cosine distance between task Task2Vec embeddings.

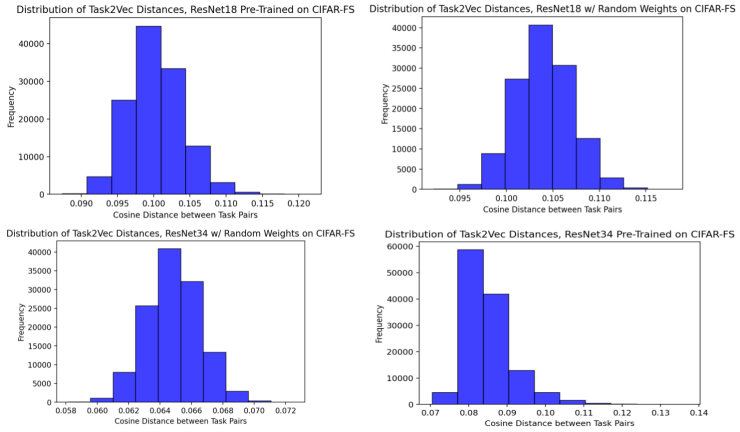

Figure 15: **Histogram of distances of 5-way, 20-shot tasks from Cifar-fs using the Task2Vec distance. This plot justifies the use of a subsample of the population to estimate the diversity coefficient because of its approximate Gaussian distribution.** For the full argument, see the main text, section N.2. The top left histogram uses a Resnet18 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The top right histogram uses a Resnet18 with random weights to compute the Task2Vec distance between tasks. The bottom left histogram uses a Resnet34 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The bottom right histogram uses a Resnet34 with random weights on Imagenet to compute the Task2Vec distance between tasks.

case, however, one computes the correlations between two sets of data sets (e.g. two matrices $X \in \mathbb{R}^{N,D_1}$ and $Y \in \mathbb{R}^{N,D_2}$ with N examples and D_1, D_2 features or layer matrices).

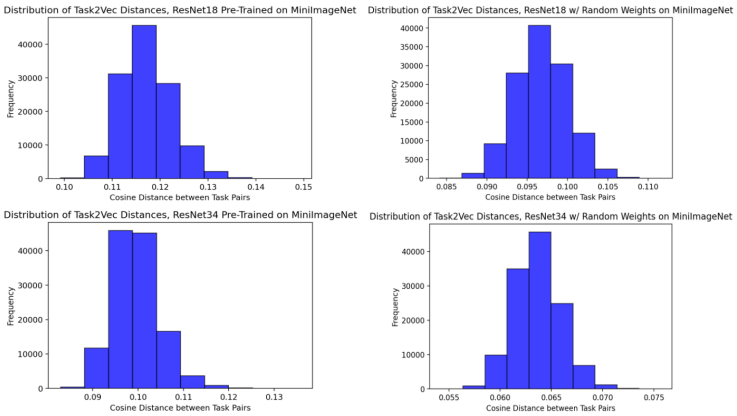

Figure 16: **Histogram of distances of 5-way, 20-shot tasks from MiniImagenet using the Task2Vec distance. This plot justifies the use of a subsample of the population to estimate the diversity coefficient because of its approximate Gaussian distribution.** For the full argument, see the main text, section N.2. The top left histogram uses a Resnet18 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The top right histogram uses a Resnet18 with random weights to compute the Task2Vec distance between tasks. The bottom left histogram uses a Resnet34 pre-trained on Imagenet to compute the Task2Vec distance between tasks. The bottom right histogram uses a Resnet34 with random weights on Imagenet to compute the Task2Vec distance between tasks.

True distribution based Canonical Correlation Analysis (CCA): What we call true distribution based CCA is the standard CCA measure using the true but known distribution of the data $p^*(x)$ and $p^*(y)$. In this case, CCA searches for a pair of linear combinations a^*, b^* of two set of random variables (or vectors of random variables) $\boldsymbol{x} = [X_1, \dots, X_{D_1}]$ and $\boldsymbol{y} = [Y_1, \dots, Y_{D_2}]$ that maximizes the Pearson correlation coefficient:

$$a^*, b^* = arg\max_{a,b} \frac{\mathbb{E}_{X,Y}[(a^\top \boldsymbol{x})((b^\top \boldsymbol{y}))]}{\sqrt{\mathbb{E}_X[(a^\top \boldsymbol{x})^2]}\sqrt{\mathbb{E}_Y[(a^\top \boldsymbol{y})^2]}} = arg\max_{w_1,w_2} \frac{a^\top \Sigma_{X,Y} b}{\sqrt{a^\top \Sigma_{X,X} a}\sqrt{b^\top \Sigma_{Y,Y} b}}$$

where $\Sigma_{X,Y}, \Sigma_{X,X}\Sigma_{Y,Y}$ are the (true) covariance and variance matrices respectively (e.g. $\Sigma_{X,Y}[i,j] = Cov[X_i, X_j] = [X_i Y_j]$ for centered random variables). All of these can be replaced by empirical data matrices in the obvious way.

O.4 SVCCA

At a high level, SVCCA is a similarity measure of two matrices that aims in removing redundant neurons (i.e. redundant features) with the truncated SVD by keeping 0.99 of the variance and then measure the overall similarity by averaging the top C CCA values.

SV: Given two matrices $L_1 \in \mathbb{R}^{N,D_1}, L2 \in \mathbb{R}^{N,D_2}$ (e.g. layer matrices) first reduce the effective dimensionality of the matrix via a low rank approximation $L_1' \in \mathbb{R}^{N,D_1'}, L2' \in \mathbb{R}^{N,D_2'}$ by choosing the top k singular values that keeps 0.99 of the variance. In particular, for each layer matrix, L_i keep the top D_i' singular values (and vectors) such that $\sum_{j=1}^{D_i'} |\sigma_j| \geq 0.99 \sum_{j=1}^{rank(L_i)} |\sigma_j|$.

SVCCA: SVCCA is a statistical technique for the measuring the (linear) similarity of two sets of data sets $L_1 \in \mathbb{R}^{N,D_1}, L2 \in \mathbb{R}^{N,D_2}$ (e.g. data matrices, layer matrices) by first reducing the effective dimensionality of the matrix via a low rank approximation $L_1' \in \mathbb{R}^{N,D_1'}, L_2' \in \mathbb{R}^{N,D_2'}$ (e.g. by choosing the top k singular values that keeps 0.99 of the variance) and then applying the standard empirical CCA to the resulting matrices. This is repeated $C = \min(D_1', D_2')$ times and the overall similarity of the two matrices is computed as the average CCA: $svcca = sim(L_1', L_2') = \frac{1}{C}\sum_{c=1}^{C} \rho_c$

Concretely:

1. Get the D_i' components that keep 0.99 of the variance (i.e. D_i' such that $\sum_{j=1}^{D_i'} |\sigma_j| \geq 0.99 \sum_{j=1}^{rank(L_i)} |\sigma_j|$).

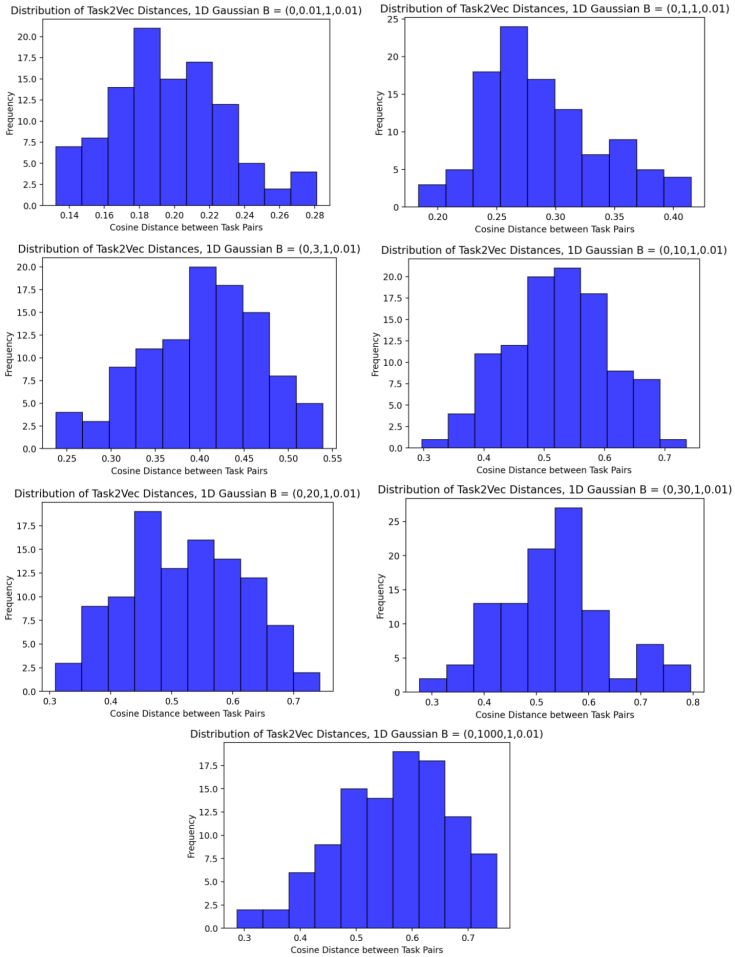

Figure 17: **Histogram of distances of 5-way, 20-shot tasks from the Synthetic Gaussian benchmark using the Task2Vec distance. This plot justifies the use of a subsample of the population to estimate the diversity coefficient because of its approximate Gaussian distribution.** For the full argument, see the main text, section N.2. The meta parameters generating tasks for each benchmark are denoted by $B = (0, x, 1, 0.01)$ where x is in the list $[0.01, 1, 3, 10, 20, 30, 1000]$ indicating the mean to generate the mean of the Gaussian tasks. For full details of the synthetic Gaussian benchmark, see section M.

2. Get the SVD: $U_1, \Sigma_1, V_1^\top = SVD(L_1)$ and $U_2, \Sigma_2, V_2^\top = SVD(L2)$
3. Then produce the SVD dimensionality reduction by $L_1' = L_1 V_1[1 : k_i] \in \mathbb{R}^{N,D_1}$ and $L_2' = L_2 V_2[1 : k] \in \mathbb{R}^{N,D_2}$ where $V_i[1 : D_i]$ gets the top D_i columns of a layer matrix i.
4. Get the CCA of the reduced layer matrix: $[\rho_c]_{c=1}^{C} = CCA(L_1', L_2')$ where, $C = \min(D_1', D_2')$
5. Finally return the mean CCA: $svcca = \frac{1}{C}\sum_{c=1}^{C} \rho_c$, where is the k-th CCA value of the reduced layer matrix.

O.5 PWCCA

At a high level, PWCCA was developed to increase the robustness (to noise) of SVCCA in the context of deep neural networks. In particular, Maithra et al. [Morcos et al., 2018] noticed that when the performance of the neural networks stabilized, so did the set of CCA vectors (or principle neuron vectors) related to the network stabilized on the data set in question. Thus, they suggest to give higher weighting to the canonical correlation ρ_c of these stable CCA vectors – in particular to the ones that

are similar to the final output layer matrix, e.g. L_1. Note this is simpler than trying to track the stability of these CCA vectors during training and then give those higher weighting.

PWCCA: Formally let L_1 be the layer matrix $[z_d^l; \ldots; z_{D_1}^l] \in \mathbb{R}^{N,D_i}$ with neuron vectors z_d^l for some layer l. Recall that the k-th left CCA vector for layer matrix L_1 is defined as follows, $\tilde{x}_c = L_1 a_c = L_1(\Sigma^{-\frac{1}{2}} u_c)$ where a_c is the cth CCA direction and u_c is the c-th left singular value from the matrix $M = \Sigma_{L_1}^{-\frac{1}{2}} \Sigma_{L_1,L_2} \Sigma_{L_2}^{-\frac{1}{2}} = U \Lambda V^\top$. Then, PWCCA can be computed as follows:

1. Calculate the CCA vectors $\tilde{x}_c = L_1 a_c = L_1(\Sigma_{L_1}^{-\frac{1}{2}} u_c)$ and explicitly orthonormalize with Gram-Schmidt for numerical stability.
2. Compute the weight $\tilde{\alpha}_c$ of how much the layer matrix L_1 is account for by each CCA vector $\tilde{x}_k$ with equation $\tilde{\alpha}_c(h_c, L_1) = \sum_{c=1}^{C} |\langle \tilde{x}_c, z_c^l \rangle_{\mathbb{R}^N}|$ where z_c^l is the c-th column of the layer matrix L_1
3. Normalize the weight indicating how much each CCA vector h_c accounts for L_1 and denote it with, $\alpha_c(\tilde{x}_k, L_1) = \frac{\tilde{\alpha}_c(\tilde{x}_k, L_1)}{\sum_{c=1}^{C} \alpha_c(\tilde{x}_k, L_1)}$
4. Finally return the mean CCA weighted by $\alpha_c(\tilde{x}_k, L_1)$: $pwcca = \sum_{c=1}^{C} \alpha_c(\tilde{x}_k, L_1)\rho_c$ where $C = \min(D_1', D_2')$.

The original authors could have used the right CCA vectors, i.e. $\tilde{y}_c = L_2 b_k = L_2(\Sigma^{-\frac{1}{2}} v_k)$ and in fact the details of their code suggest they choose the one that would have lead to fewer values removed by SVD. This choice seems to already be robust to noise, as shown in [Morcos et al., 2018]. Note that the CCA vectors $\tilde{x}_k, \tilde{y}_k$ are of size $\mathbb{R}^N$ and thus could be viewed as the principle neuron vectors that correlate two layers $L_1, L2$. With this view, PWCCA computes the mean CCA normalized by of the c principle neuron vectors are account for the output layer matrix the most.

O.6 CCA for CNNs

The input to CCA are two data matrices, but CNNs have intermediate representations that are 4D tensors. Therefore, some justification is needed in how to create the data matrices needed for computing CCA for CNNs. Note that it's the same reasoning for both SVCCA and PWCCA.

Each channel as the dimensionality of the data matrix: One option is to get the intermediate representation of size $[M, C, H, W]$ and get a layer matrix of size $[MHW, C]$. Thus, MHW is the effective number of data points and the channels (or number of filters) is the effective dimensionality of the (layer) matrix. In this view, each patch of an image processed by the CNN is effectively considered a data point. This view is very natural because it also considers each filter as its own "neuron" – which seams reasonable considering that each filter uniquely responds to each stimulus (e.g., data patch). This view results in HW images for every sample in the data set (or batch) of size M and C effective neurons.

Although the original authors suggest this metric as a good metric mainly for comparing two layers that are the same – we hypothesize it is also good for comparing different layers (as long as the effective number of data points match for the two layer matrices). The reason is that CCA tries to compute the maximum correlation of two data sets (or sets of random variables) and assumes no meaning in the ordering of the data points and assumes no process for generating each individual sample for the set of random variables, thus meaning that this metric (CCA) can be used for any two layers in a matrix. Overall, in this view, we are comparing the representation learned in each channel.

Each activation as the dimensionality of the data matrix: One option is to get the intermediate representation of size $[M, C, H, W]$ and get a layer matrix of size $[M, CHW]$. Thus, M is the effective number of data points (which matches the number of samples in the data set or batch) and therefore each activation value is the effective dimensionality of the (layer) matrix. In this view, each activation is viewed as a neuron of size M and we have CHW effective neurons for each activation. The authors suggest this metric for comparing different layers (potentially at different depths). However, because CCA assumes no correspondence between the data points nor the same dimensionality in the data matrix – we hypothesize this way to define the data matrix is as valid as the previous definition for comparisons between any models at any layer. One disadvantage however is that it will often result in data matrices that are very large due to CHW being very large – which

results in artificially high CCA similarity values. Potential ways to deal with it are noticing that there is no correspondence between the data matrices, so a cross comparison of every data point with every other data point in CCA is possible (resulting is $O(M^2)$ comparisons for the empirical covariance matrix). Alternatively one can pool in the spatial dimensions $[H, W]$ resulting in potential smaller layer matrices e.g. of shape $[M, C]$ with a pool over the entire spatial dimension. For these reasons and the fact that we hypothesize an image patch being its own image – we prefer to interpret the number of channels as the natural way to compare CNNs so that the layer matrices results of size $[MHW, C]$.

Subsampling of representations for channels as dimensionality: In this section, we review the subsampling we did when comparing the representations learned in each channel, i.e. the layer matrix has size $[MHW, C]$. The effective number of data points MHW will often be much larger than needed (e.g. for 16 data samples $M = 16$ and $H = W = 84$ results in $MHW = 112,896$), especially compared to the number of filters/channels (e.g. $C = 64$). Previous work [Raghu et al., 2017, Morcos et al., 2018] suggest using the number of effective data points to be from 5-10 times the size of the dimensionality in a layer matrix of size $[N', D']$ that means $N' = 10D'$. Based on our reproductions of that number, we choose $N' = 20D'$ which results in $NHW = 20C$

O.7 Centered Kernel Alignment (CKA)

At a high level, CKA is based on the insight that one can first measure the similarity between every pair of examples in each representation separately and then use the similarity structure to compute an overall similarity metric. In our case, we can treat the examples as the neuron vectors and compare all neuron vectors using some kernel function. Usually this will end in a kernel matrix of size M, M' where M and M' are the number of examples. In our case, they would be D, D' for the number of neurons of each layer matrix. Note, the layers matrices can correspond to neurons of different layers in a neural network.

Linear CKA: We use the linear kernel function as used in previous work [Kornblith et al., 2019, Ding et al., 2021]. Given two layer matrices $X_1 \in \mathbb{R}^{N,D_l}$ and $X_2 \in \mathbb{R}^{N,D_{l'}}$ for layers l, l', we compute the linear kernel $X_1^\top X_2$ to get the D_l by $D_{l'}$ kernel matrix indicating the (linear) similarity per neuron vector for the two layers. Then to obtain a single distance value we compute the Frobenius norm of the kernel matrix and subtract by one after normalization:

$$d_{linearCKA}(A, B) = 1 - \frac{\|A^\top B\|_F}{\|A^\top A\|_F \|B^\top B\|_F} \tag{4}$$

Note that depending on how the examples in matrices A, B are organized the cross-product could be computed with $AB^\top$ instead. Other kernel functions have been tested (e.g., the RBF kernel) for CKA but similar results are obtained, resulting in linear CKA being the most popular CKA method [Ding et al., 2021, Kornblith et al., 2019] to the best of our knowledge.

O.8 Orthogonal Procrustes Distance (OPD)

At a high level, the orthogonal Procrustes distance computes the distances between two matrices after using for the best orthogonal matrix that tries to match the two. Usually this is done after centering and dividing by the Frobenious the matrices, i.e. normalizing the matrices. In addition, previous work [Ding et al., 2021] finds that OPD is a better metric at detecting changes that matter functionally and robust against changes that do not matter.

OPD: Formally, the Orthogonal Procrustes Distance is the smallest distance between two matrices X and Y (with columns as the vectors in question) found by finding the orthogonal matrix Q which most closely maps A to B. Therefore, the OPD distance is the distance value from solving the orthogonal Procrustes problem:

$$d_{OPD}(X, Y) = \min_Q \|X - YQ\|_F \tag{5}$$

where $\|\cdot\|_F$ is the Frobenius norm. When matrices are normalized (centered and divided by their Frobenious norm) this is called the general Procrustes problem. However, the closed for equation we used is the following:

$$d'_{OPD}(X, Y) = \frac{1}{2}\left(\|X\|_F + \|Y\|_F - 2\|X^\top Y\|_*\right) \tag{6}$$

where $\|\cdot\|_*$ is the nuclear norm, i.e. the sum of singular values $\sum_i \sigma(A)_i = \|A\|_*$. The division by 2 is to guarantee that the OPD distance is between $[0, 1]$ instead of $[0, 2]$. We do the standard normalization of the matrices before computing the OPD distance – by centering and dividing by the Frobenious norm of the matrix. This is done because the orthogonal matrix in the orthogonal Procrustes problem does not allow for translation or rescaling of the matrices. Therefore, this normalization enforces invariance to this type of transformations – i.e. we don't want large OPD values due to rescaling or translation (and even if present, the orthogonal matrix wouldn't be able to reflect it).

Therefore, the final equation for OPD we use is:

$$d_{OPD}(X, Y) = 1 - \|X^\top Y\|_* \tag{7}$$

Why OPD? We use OPD due to the findings of [Ding et al., 2021]. They find that OPD is a more robust metric (compared to SVCCA, PWCCA, and CKA) because it is *sensitive* to changes that affect real functional behavior (so it detects changes to behavior that "matter") and it's *specific* against changes that do not. As a summary, some of the evidence that they provide for this is that OPD is able to detect when 0.75 of the principal components are removed, while CKA cannot detect removal of principal components until 0.97 are removed. CCA like metrics on the other hand are not specific – even random initialization noise overwhelms the distances it reports, while OPD is more robust to this random noise. For the last point, this means that even if we compare two different layers with CCA, the noise will dominate the distance reported instead of the difference caused by comparing different layer.

O.9 Correctly using Feature Based Distances

When comparing two layers of a neural network using two layer matrices, one needs to be careful with the number of data points (or batch size) being used. This is because metrics like CCA intrinsically are formulated as an optimization and if the number of examples is not larger than the number of dimensionality of the examples – then the similarity can be pathologically be perfect (e.g., the distance is zero when it's actually not zero). Therefore, we follow the suggestions by the original authors of SVCCA [Raghu et al., 2017] and always use at least 10 times more examples than there are features for our feature based comparisons. We call this value the *safety margin*. To illustrate this idea, we produce two random matrices and compare how the similarity (SVCCA) values varies as a function of the dimensionality of the data and the number of points. Since the two matrices are completely random, we know they should not be very similar and thus SVCCA should report a high similarity value (or low distance value). Therefore, we can see in figure 18 how as the dimensionality increases, the similarity value approaches a perfect similarity of 1.0. In figure 19 we can see how as the number of points increases, we approach a smaller similarity – closer to the true similarity for random matrices.

In general, given two matrices $X, Y \in \mathbb{R}^{M', D'}$ with the number of (effective) data points M' and (effective) dimensionality (number of features) D' – we want the number of points to be larger by a safety factor s. Formally, it must satisfy this inequality to avoid the pathological case for feature based distances:

$$D' \leq sM' \tag{8}$$

where we suggest to use $s \leq 10$ (as used in previous work [Raghu et al., 2017]). Note the effective number of data points used and dimensionality can be different depending on how one reshapes the CNN tensors to produce layer matrices as explained in section O.6. For example, if one uses the channel as the dimensionality (i.e., use the image patches as an effective data point) then one has to obey the following inequality:

$$C \leq sMHW \tag{9}$$

where M is the batch size, H, W is the height and width of the images, and C is the number of filters/channels for the current layer. This means that for a given architecture processing images of a given size that the only parameter we can change to make the above inequality true is the batch size M.

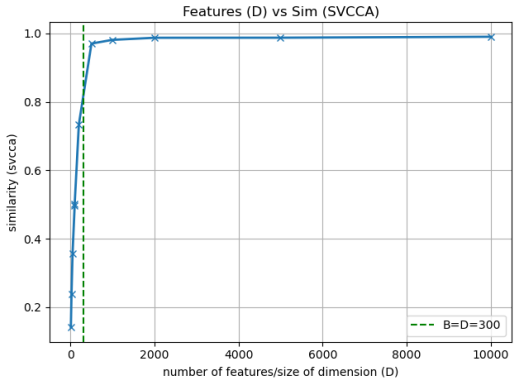

Figure 18: **Shows that as the dimensionality of a random data matrix increases – the SVCCA similarity approaches the pathological case by falsely reports the similarity is perfect.** The green line indicates when the number of examples and dimensionality are equal (and equal to 300). D denotes the dimensionality of the simulated data and B the size of the batch size/number of points.

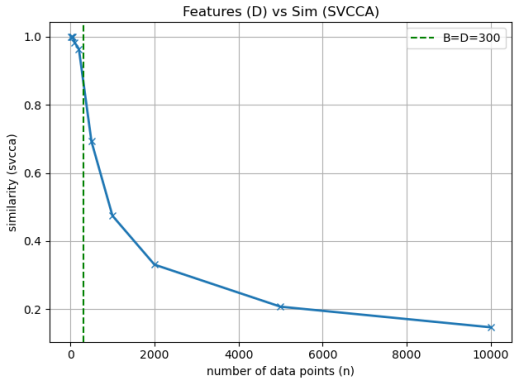

Figure 19: **Shows how to avoid the pathological case when using feature based similarities by increasing the number of data points (or batch size).** In particular, as the number of data points in two random data matrix increases – the true similarity approaches the true low similarity value. The green line indicates when the number of examples and dimensionality are equal (and equal to 300). D denotes the dimensionality of the simulated data and B the size of the batch size/number of points.

P Summary of Compute Required

We used an internal compute cluster with wide varied of GPUs. We used Titan X GPUs for most five layer CNN experiments. We used A40 and dgx-A100 GPUs for Resnet12 experiments, with 48 GB and 40 GB GPU memory respectively. We did notice that the Resnet12 architecture we used from previous work [Tian et al., 2020] required more memory than Resnet18 and Resnet34 used in Task2Vec [Achille UCLA et al., 2019]. By requiring more memory, we mean we did not have many memory out of bound issues with Resnet18/Resnet34, but did have memory issues with Resnet12. In addition, our episodic meta-learning training for MAML used Learn2Learn's ["Arnold et al., 2020] distributed training to speed up experiments. Experiments took 1-2 weeks with MAML in a single GPU to potentially 2-3 days with multiple GPUs (we used 2, 4 to 8 GPUs depending on availability). For synthetic experiments, we used Titan X GPUs with 16GB of GPU memory. Experiments took around 1-2 days on average with a single GPU. For more precision, check the experimental details section F.

Q A Statistical Decision view of the differences between Supervised Learning and Meta-learning

Recent work in meta-learning implies that feature-reuse might be all we need to solve modern few-shot learning benchmarks [Tian et al., 2020]. However, what it also reveals is our poor understanding of meta-learning algorithms. Therefore, in this section, we take the most foundational perspective to formulate and analyze meta-learning algorithms by analyzing them from an optimal statistical decision theory perspective?

We hope that this can help clarify the results from [Tian et al., 2020] and therefore help meta-learning researchers design better meta-learning benchmarks and meta-learning algorithms.

Q.1 Supervised Meta-Learning problem set-up

In this section, we introduce the notation for supervised meta-learning. Intuitively, we seek to find a function that minimizes the expected risk over tasks. To formalize it, we will use three formulations:

Monolithic meta-learner: for a monolithic decision rule g (or meta-learner), we want to find the optimal g by minimizing the *supervised meta-learning expected risk*:

$$R_{Mono}(g) = \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{x,y \sim p(x,y|\tau)} \left[l(g(x,\tau), y) \right] \tag{10}$$

where g is a single monolithic function, $p(\tau)$ is the true but unknown distribution of tasks, $p(x, y \mid \tau)$ is the true, but unknown distribution of data pair given a task τ and (x, y) is the data pair of input and target value sampled from a task.

Meta-learned meta-learner: for a meta-learned decision rule we usually have an adaptation rule A (e.g. SGD in MAML) and a function approximator h (e.g. a neural network) and minimize the follow over both:

$$R_{ML}(A, h) = \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{x,y \sim p(x,y|\tau)} \left[l(A(h,\tau)(x), y) \right] \tag{11}$$

$p(\tau)$ is the true but unknown distribution of tasks, $p(x, y \mid \tau)$ is the true, but unknown distribution of data pair given a task τ and (x, y) is the data pair of input and target value sampled from a task.

Fixed representation meta-learner without adaptation: one can also solve 10 using a single decision rule f that does not take the task τ as input as follows:

$$R_{SL}(f) = \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{x,y \sim p(x,y|\tau)} \left[l(f(x), y) \right] \tag{12}$$

where f is a function to be adapted (e.g. a neural network), $p(\tau)$ is the true but unknown distribution of tasks, $p(x, y \mid \tau)$ is the true, but unknown distribution of data pair given a task τ and (x, y) is the data pair of input and target value sampled from a task.

Fixed representation meta-learner with a final adaptation layer: one can also solve 10 using a single feature extractor g that does not take the task τ as input with a feature extractor g:

$$R_{SLA}(f, g) = \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{x,y \sim p(x,y|\tau)} \left[l((f(\tau) \circ g)(x), y) \right] \tag{13}$$

where g is the feature extractor from the raw inputs (e.g. a neural network), f the final layer adapted (e.g. a linear layer), $p(\tau)$ is the true but unknown distribution of tasks, $p(x, y \mid \tau)$ is the true, but unknown distribution of data pair given a task τ and (x, y) is the data pair of input and target value sampled from a task.

Remark Q.1. Note that in practice, the meta-learner does not usually take the full task τ as input, but instead a train and test set (often referred to as support set and query set) sampled from the task τ.

The goal of this work is to clarify the difference between 12 and 11 under the framework of statistical decision theory. Arguably the most important comparison between 11 and 13 is left for future work.

Q.2 Main Result: Difference between the Supervised Learned and Meta-learned decision rule

The proof sketch is as follows: we first show the optimal decision rules for both supervised learning and meta-learning when minimizing the expected meta-risk from equations 12 and 11 and then highlight that the main difference between them is that the meta-learned solution can act optimally if it identifies the task τ while the supervised learned solution has no capabilities of this since it learns an average based on tasks instead.

Theorem Q.2. *The minimizer to equation 11 is:*

$$A(h,\tau)(x) = \bar{y}^*_{y|x,\tau} = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right] \tag{14}$$

where $\bar{y}^*_{y|x,\tau} = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$ *and* l *is the squared loss* $l(\hat{y},y) = (\hat{y}-y)^2$.

Proof. The proof is the same as the standard decision rule textbook proof but instead of minimizing it point-wise w.r.t. x we minimize it point-wise w.r.t. (x,τ). In particular, we have:

$$R_{ML}(A,h) = \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{x,y\sim p(x,y|\tau)}\left[l(A(h,\tau)(x),y)\right]$$

$$\min_{A,h}\mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{x\sim p(x|\tau)}\mathbb{E}_{y\sim p(y|x,\tau)}\left[(A(h,\tau)(x)-y)^2\right]$$

without loss of generality (WLOG) and for clarity of exposition consider the special case for discrete variables:

$$\min_{A,h}\sum_{\tau}p(\tau)\sum_{x}p(x\mid\tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[(A(h,\tau)(x)-y)^2\right]$$

At this point we notice we can minimize the above point-wise w.r.t (x,τ) and ignore h. To do that, take the derivative of $R(A,h)$ with respect to $A(h,\tau)(x)$ because that $A(h,\tau)(x)\in\mathbb{R}$ and set it to zero:

$$\frac{d}{dA(h,\tau)(x)}\mathbb{E}_{y\sim p(y|x,\tau)}\left[(A(h,\tau)(x)-y)^2\right] = 0$$

$$\mathbb{E}_{y\sim p(y|x,\tau)}\left[(A(h,\tau)(x)-y)\right] = 0$$

$$\mathbb{E}_{y\sim p(y|x,\tau)}\left[(A(h,\tau)(x)\right] = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$$

$$A(h,\tau)(x) = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right] = \bar{y}^*_{y|x,\tau}$$

as desired. □

Corollary Q.3. *For a monolithic meta-learner defined in section Q.1 the solution to the meta supervised learning problem is the same as in equation 14 for the squared loss* $l(\hat{y},y) = (\hat{y}-y)^2$ *i.e.* $g(\tau,x) = \bar{y}^*_{y|x,\tau} = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$.

Proof. Proof is trivial, replace $A(h,\tau)(x)$ with $g(\tau,x)$ since h is not used. In this case, there is no difference with having an adaptation rule A equipped with another function h and a monolithic meta-learner g. □

Theorem Q.4. *The minimizer to equation 12:*

$$f(x) = \mathbb{E}_{\tau\sim p(\tau|x)}\left[\bar{y}^*_{y|x,\tau}\right] \tag{15}$$

where $\bar{y}^*_{y|x,\tau} = \mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$ *and* l *is the squared loss* $l(\hat{y},y) = (\hat{y}-y)^2$.

Proof. WLOG, consider the minimizer of equation 12 in the discrete case. In particular, we have:

$$R_{SL}(f) = \mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{x,y\sim p(x,y|\tau)}\left[l(f(x),y)\right]$$

$$\min_{f}\mathbb{E}_{\tau\sim p(\tau)}\mathbb{E}_{x\sim p(x|\tau)}\mathbb{E}_{y\sim p(y|x,\tau)}\left[(f(x)-y)^2\right]$$

$$\min_{f}\sum_{x}\mathbb{E}_{\tau\sim p(\tau)}p(x\mid\tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[(f(x)-y)^2\right]$$

Note we can minimize the above point-wise w.r.t. x only (and not also w.r.t. τ as we did in proof Q.2). Thus, we have want:

$$f(x) = \min_{f(x)\in\mathbb{R}}\mathbb{E}_{\tau\sim p(\tau)}p(x\mid\tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[(f(x)-y)^2\right]$$

at this point it is interesting to observe the disadvantage of supervised learning methods with fixed functions without dependence on the task is that they are forced to consider all task τ at once. We proceed to take derivatives as in proof Q.2 but with this objective:

$$\mathbb{E}_{\tau\sim p(\tau)}p(x\mid\tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[(f(x)-y)^2\right]$$

$$\frac{d}{df(x)}\mathbb{E}_{\tau\sim p(\tau)}p(x \mid \tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[(f(x)-y)^2\right]=0$$
$$\mathbb{E}_{\tau\sim p(\tau)}p(x \mid \tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[f(x)\right]=\mathbb{E}_{\tau\sim p(\tau)}p(x \mid \tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$$
$$f(x)\mathbb{E}_{\tau\sim p(\tau)}\left[p(x \mid \tau)\right]=\mathbb{E}_{\tau\sim p(\tau)}p(x \mid \tau)\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$$
$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\frac{p(x \mid \tau)}{\mathbb{E}_{\tau\sim p(\tau)}\left[p(x \mid \tau)\right]}\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right] \tag{16}$$

We proceed by noticing that $\mathbb{E}_{\tau\sim p(\tau)}\left[p(x \mid \tau)\right]=p(x)$, thus:

$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\frac{p(x \mid \tau)}{p(x)}\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right]$$
$$f(x)=\sum_{\tau}p(\tau)\frac{p(x \mid \tau)}{p(x)}\left[\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right]$$
$$f(x)=\sum_{\tau}\frac{p(\tau)}{p(x)}\frac{p(x,\tau)}{p(\tau)}\left[\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right]$$
$$f(x)=\sum_{\tau}\frac{p(x,\tau)}{p(x)}\left[\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right]$$
$$f(x)=\sum_{\tau}p(x \mid \tau)\left[\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right]$$
$$f(x)=\mathbb{E}_{\tau\sim p(x|\tau)}\left[\mathbb{E}_{y\sim p(y|x,\tau)}y\right]$$
$$f(x)=\mathbb{E}_{\tau\sim p(x|\tau)}\left[\bar{y}^*_{y|x,\tau}\right]$$

as required by the rightmost RHS of equation 15. □

Theorem Q.5. *The minimizer in equation 15 reduces to an expectation only over w.r.t. $p(\tau)$ of $\bar{y}^*_{y|x,\tau}$ under benchmarks that are balanced. Formally*

$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\bar{y}^*_{y|x,\tau}\right]=\mathbb{E}_{\tau\sim p(\tau)}\left[\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]\right] \tag{17}$$

where $\bar{y}^*_{y|x,\tau}=\mathbb{E}_{y\sim p(y|x,\tau)}\left[y\right]$ and under assumption A1: $p(x \mid \tau)$ is a constant, i.e. $p(x \mid \tau)=k_{XT}\in\mathbb{R},\forall x\in X,\forall\tau\in T$ and l is the squared loss $l(\hat{y},y)=(\hat{y}-y)^2$.

Proof. Recall equation 15:

$$f(x)=\mathbb{E}_{\tau\sim p(\tau|x)}\left[\bar{y}^*_{y|x,\tau}\right]$$

due to Bayes's rule we have $p(\tau \mid x)=\frac{p(\tau)p(x|\tau)}{p(x)}$ and equation 15 can be re-written as follows:

$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\frac{p(x \mid \tau)}{p(x)}\bar{y}^*_{y|x,\tau}\right]$$

under assumption A1 we have that $p(x \mid \tau)$ does not depend on as a function of x or τ. Thus, we have:

$$p(x)=\sum_{\tau}p(\tau)p(x \mid \tau)=p(x \mid \tau)\sum_{\tau}p(\tau)=p(x \mid \tau)$$

Thus we have:

$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\frac{p(x)}{p(x)}\bar{y}^*_{y|x,\tau}\right]$$
$$f(x)=\mathbb{E}_{\tau\sim p(\tau)}\left[\bar{y}^*_{y|x,\tau}\right]$$

as required. □

Remark Q.6. Note that assumption A1 holds for the common MiniImagenet few-shot learning data set, where $p(x \mid \tau)=\frac{1}{600}$.

Remark Q.7. In addition, because all classes are equally likely (e.g. $p(class) = \frac{1}{64}$ for the meta-train set) we have $p(\tau)$ is the same constant independent of the task τ. Proof in the appendix, lemma Q.8.

Theorem Q.8. *If the tasks are equally likely, then equation 17 becomes an average over conditional predictions over all tasks. Formally, if $p(\tau) = \frac{1}{T}$ then equation 17 becomes:*

$$f(x) = \frac{1}{T} \sum_{\tau} \bar{y}^*_{y|x,\tau} \tag{18}$$

under the squared loss $l(\hat{y}, y) = (\hat{y} - y)^2$.

Proof. Since $f(x) = \mathbb{E}_{\tau \sim p(\tau)} \left[\bar{y}^*_{y|x,\tau} \right]$ then, plugging $p(\tau) = \frac{1}{T}$ completes proof. □

Remark Q.9. It is interesting to note that without adaptation or dependence on the task τ being solved, the supervised learned meta-learner is suboptimal compared to the meta-learned solution. The proof is simple, and it follows because the meta-learned decision rule was chosen to minimize each term individually, but the supervised learned decision is not of that form. Proof in appendix Q.11. Unfortunately, note that this does not necessarily apply to previous work [Tian et al., 2020].

Remark Q.10. Note that remark Q.9 does not apply to work [Tian et al., 2020] because that work does depend on a task τ during meta-test time by adapting the final layer even if the representation is fixed.

Remark Q.11. The supervised learning decision rule is suboptimal compared to the meta-learned decision rule.

Q.3 The supervised Learning Solution is equivalent to the Meta-Learning solution when there is low task diversity

Sketch argument: The main idea is that because all tasks are very similar (task diversity is low) – it essentially means that τ is not truly an input to the adaptation rule or monolithic meta-learner). Equivalently, the problem is essentially a single task problem, so the task is implicitly an input to any method used. Therefore, since the task conditioning does not exist, then the optimization problem is the same for the meta-learned solution and when there is a fixed supervised learning feature extractor.

Theorem Q.12. *Assume $\tau_1 = \tau_2$ for any tasks in T and the data sets are balanced (i.e. same number of images x for each task). Then we have the meta-learned solution is the same as the supervised learning solution with shared embeddings: $f_{sl}(x) = A(f_{ml}, \tau)(x)$.*

Proof. Consider the optimization problem, for supervised learning:

$$\min_{A,h} \mathbb{E}_{\tau \sim p(\tau)} \mathbb{E}_{x \sim p(x|\tau)} \mathbb{E}_{y \sim p(y|x,\tau)} \left[(A(h, \tau)(x) - y)^2 \right]$$

If every pair of tasks is equal, it means their distributions are equal $p(x, y \mid \tau) = p(x, y)$ (meaning τ can be ignored). Thus, the solution to the supervised learning problem is: $f_{sl}(x) = \mathbb{E}_{\tau} \mathbb{E}_{p(x,y)}[y] = \mathbb{E}_{p(x,y)}[y] = y^*_{|x}$. Now for the meta-learning problem we have: $A(f_{ml}, \tau)(x) = y^*_{|x,\tau} = \mathbb{E}_{y \sim p(y|x,\tau)}[y]$ but due to every pair of tasks being equal means $p(x, y \mid \tau) = p(x, y)$ (i.e. all task share the same distributions) we have: $A(f_{ml}, \tau)(x) = \mathbb{E}_{y \sim p(y|x,\tau)}[y] = \mathbb{E}_{y \sim p(y|x}[y] = y^*_{|x}$ which is the same as the solution as in f_{sl}. Thus $f_{sl}(x) = A(f_{ml}, \tau)(x)$. □

Remark Q.13. Proofs were presented in the discrete case clearly, but it is trivial to expand them to the continuous case – e.g., using integrals instead of summations.