REPRODUCING META-LEARNING WITH DIFFERENTIABLE CLOSED-FORM SOLVERS

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work we report on the reproduction of the ICLR 2019 paper *"Meta-learning with differentiable closed-form solvers"* by Bertinetto et al. (2019). To this end, we build on existing work in this field to make sure results are comparable. Our attempt to reproduce part of the paper was successful. We achieve performance competitive with or superior to the paper on two benchmarks, for several settings. New baseline (algorithm) results, based on a new dataset presented in the paper, are also evaluated. However, multiple remarks and recommendations about reproducibility and comparability are given as well. Our contributions mainly consist of clarifying the paper, reproducing its most important results, and providing a first open-source implementation.

1 INTRODUCTION

The ability to adapt to new situations and learn quickly is a cornerstone of human intelligence. When given a previously unseen task, humans can use previous experience and learning abilities to do well on that new task in a matter of seconds and with a relatively small amount of data. On the other hand, artificial learning methods have shown to be very effective on specific tasks, often times surpassing human performance (Silver et al. (2016), Esteva et al. (2017)). However, relying on standard supervised-learning or reinforcement learning training paradigms, those artificial methods still require a lot of training data and training time to adapt to a new task.

An area of machine learning that is dealing with learning and adapting from a small amount of data is called *few-shot learning*. A *shot* corresponds to an example, e.g. a single image and its label. In few-shot learning the learning scope is expanded to a variety of tasks with a few shots each, compared to the classical setting of a single task with a lot of shots. A promising approach for few-shot learning is the field of *meta-learning*. Meta-learning, also known as *learning-to-learn*, is a paradigm that leverages cross-task information and training experience to perform well on a new unseen task.

The paper which is considered for reproduction (Bertinetto et al. (2019), further referenced as *"the paper"*) falls into the class of gradient-based meta-learning algorithms that try to learn a good model parameter intialization for rapid fine-tuning with few shots (Finn et al. (2017), Nichol & Schulman (2018)). In the paper, the authors present a new meta-learning method that combines a deep neural network feature extractor with differentiable learning algorithms that have closed-form solutions. This reduces the overall complexity of the gradient based meta-learning process, and at the same time advances the state-of-the-art in terms of accuracy across multiple few-shot benchmarks.

2 BACKGROUND IN META-LEARNING

The goal of few-shot meta-learning is to train a model that can quickly adapt to a new task using only a few datapoints and training iterations. In this work we will only consider classification tasks, but it should be noted that meta-learning is generally applicable to regression or reinforcement learning tasks as well (Finn et al. (2017)).

In order to provide a solid definition of meta-learning, we need to define its different components. We denote the set of tasks by $\mathbb{T}$. A task $\mathcal{T}_i \in \mathbb{T}$ corresponds to a classification problem, with a probability distribution of example inputs $\boldsymbol{x}$ and (class) labels y, $(\boldsymbol{x}, y) \sim \mathcal{T}_i$. For each task, we are given training samples $\mathcal{Z}_\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\} \sim \mathcal{T}$ with K shots per class and evaluation samples $\mathcal{Z}'_\mathcal{T} = \{(\boldsymbol{x}'_i, y'_i)\} \sim \mathcal{T}$ with Q shots (*queries*) per class, sampled independently from the same distribution $\mathcal{T}$. In meta-learning, we wish to reuse the learning experience used for tasks $\mathcal{T}_i,\ i \in [0, L]$ to learn a new task $\mathcal{T}_j$, where $j > L$, from only K examples, for every single one of the N classes in the task. Commonly, this is denoted as an *N-way K-shot* problem. To this end, in meta-learning two different kinds of learners can be at play: (1) a *base-learner* which works at the task level and aims at learning a single task (e.g. classifier with N classes) and (2) a *meta-learner* which aims to produce those model parameters that allow for the fastest average fine-tuning (using the *base-learner*) on unseen tasks.

In the paper a specific view of meta-learning is put forward. There, the meta-learning system consists of a generic feature extractor $\Phi(\boldsymbol{x})$, parametrized by ω, and a task-specific predictor $f_\mathcal{T}(X)$, parametrized by $w_\mathcal{T}$, that adapts separately to every task $\mathcal{T} \in \mathbb{T}$ based on the few shots available. In the case of a deep neural network architecture, this task-specific predictor $f_\mathcal{T}$ can be seen as the last layer(s) of the network and is specific to a task $\mathcal{T}$. The preceding layers Φ can be trained across tasks to provide the best feature extraction on which the task-specific predictor can finetune with maximum performance. This is visualized in Figures 1 and 2.

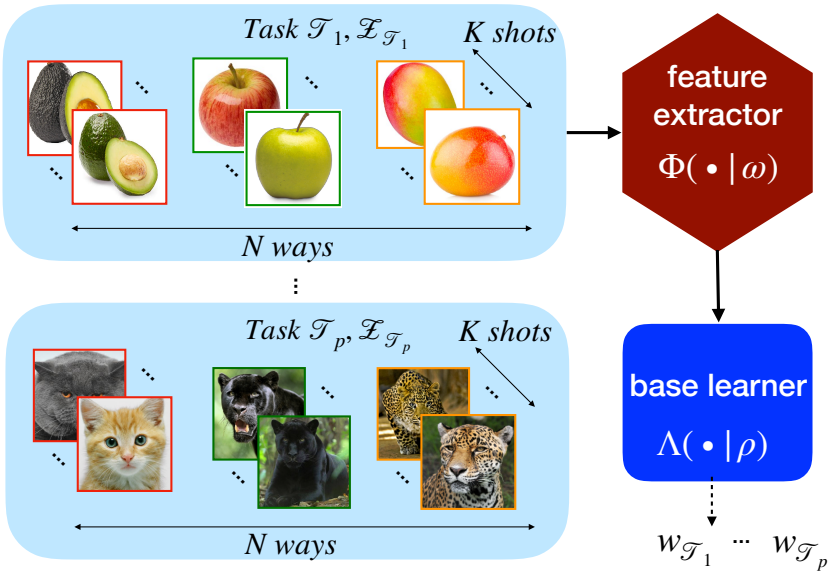

Figure 1: Base-learning of the task-specific parameters $w_{\mathcal{T}_i}$ over p tasks following steps 3 to 6 of Algorithm 1.

The base-learning phase in the paper assumes that the parameters ω of the feature extractor Φ are fixed and aims at computing the parameters $w_\mathcal{T}$ of $f_\mathcal{T}$ through a learning process Λ, which on its own is parametrized by ρ, to obtain $f_\mathcal{T}$. The meta-learning phase in the paper aims at learning a parametrization of Φ and Λ (respectively ω and ρ). In order to learn those meta-parameters, the algorithm minimizes the expected loss on test sets from unseen tasks in $\mathbb{T}$ with gradient descent.

Most of the recent meta-learning works are tested against image datasets and their feature extractor consists of a convolutional neural network (CNN). The variability between works mainly resides in the base learner $f_\mathcal{T}$ and its parameter obtaining training procedure Λ. Examples are an (unparametrized) k-nearest-neighbour algorithm (Vinyals et al. (2016)), a CNN with SGD (Mishra et al. (2017), and a nested SGD (Finn et al. (2017)). Systems in Vinyals et al. (2016) and Snell et al. (2017) are based on comparing new examples in a learned metric space and rely on matching. In particular, MATCHINGNET from Vinyals et al. (2016) uses neural networks augmented with

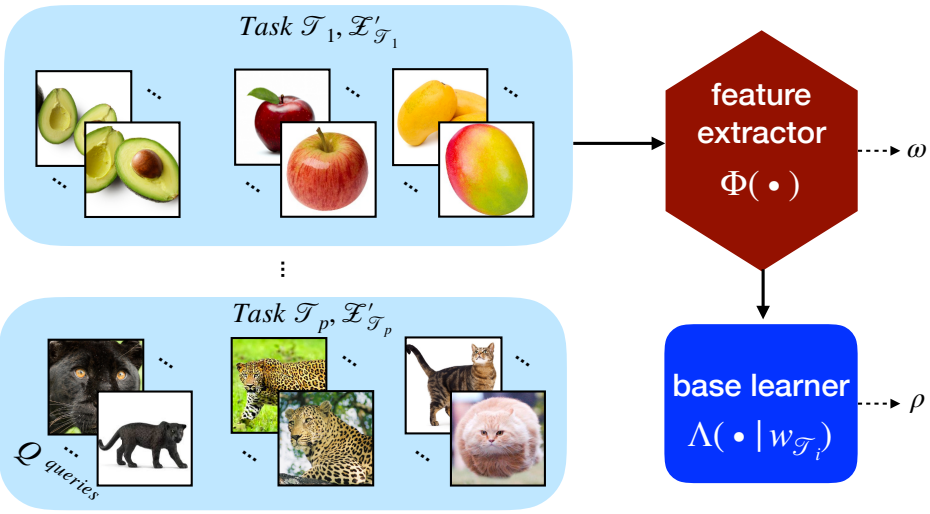

Figure 2: Meta-learning of the meta-parameters ω and $\boldsymbol{\rho}$ over the evaluation sets of each task $\mathcal{Z}'_{\mathcal{T}_i}$ using the previously learned $w_{\mathcal{T}_i}$ following steps 7 to 9 of Algorithm 1.

memory and recurrence with attention in a few-shot image recognition context. Mishra et al. (2017) builds on this attention technique by adding temporal convolutions which allow to use information from past tasks. Another use of matching-based methods is with a Graph Neural Network to learn the correspondence between the training and testing sets Garcia & Bruna (2017). A different approach is to consider the SGD update as a learnable function for meta-learning. In particular, sequential learning algorithms such as recurrent neural networks and LSTM-based methods allow to make use of long-term dependencies between the data and gradient updates as pointed out by Ravi & Larochelle (2017). Finally, Finn et al. (2017) introduced a technique called model agnostic meta-learning (MAML). In MAML, meta-learning is done by backpropagating through the fine-tuning gradient update itself of the differentiable model parameters.

3 Analysis of the Paper

In the paper (Bertinetto et al. (2019)), the authors present a new approach that relies on using fast and simple base learners such as ridge regression (R2D2) or regularized logistic regression (LRD2). In this reproducibility work we will focus on the R2D2 algorithm (*Ridge Regression Differentiable Discriminator*), because it is the only proposed algorithm with a truly closed-form solver for the base-learner. A detailed version of the R2D2 algorithm is given in Algorithm 1, which is elaborated upon in the following.

In R2D2, during base-learning with $\mathcal{Z}_{\mathcal{T}}$, the linear predictor $f_{\mathcal{T}}$ is adapted for each training task $\mathcal{T}$, using the learning algorithm Λ, and the meta-parameters ω (of Φ) and $\boldsymbol{\rho}$ (of Λ) remain fixed. It is only in the meta-training phase that meta-parameters ω and $\boldsymbol{\rho}$ are updated, using $\mathcal{Z}'_{\mathcal{T}}$. The linear predictor is seen as $f_{\mathcal{T}}(x) = xW$ with W a matrix of task-specific weights $w_{\mathcal{T}}$, and x the feature extracted version of $\boldsymbol{x}$, $x = \Phi(\boldsymbol{x})$. This approach leads to a ridge regression evaluation such that it learns the task weights $w_{\mathcal{T}}$:

$$\Lambda(X, Y) = \arg\min_{W} \|XW - Y\|^2 + \lambda \|W\|^2 \tag{1}$$

$$= (X^T X + \lambda I)^{-1} X^T Y \tag{2}$$

where X contains all feature extracted inputs from the training set of the considered task. A key insight in the paper is that the closed form solution of Equation 2 can be simplified using the *Woodbury matrix identity* yielding $W = \Lambda(X, Y) = X^T(XX^T + \lambda I)^{-1}Y$. This considerably reduces

Algorithm 1 Ridge Regression Differentiable Discriminator (R2D2)

Require: Distribution of tasks $\mathbb{T}$.
Require: Feature extractor Φ parameterized by ω.
Require: Finetuning predictor $f_{\mathcal{T}}$ with base-learning algorithm Λ and task-specific parameters $w_{\mathcal{T}}$, and meta-parameters $\boldsymbol{\rho} = (\alpha, \beta, \lambda)$

1: Initialize Φ, Λ, and $f_{\mathcal{T}}$ with pre-trained or random parameters ω_0 and $\boldsymbol{\rho_0}$
2: **while** not done **do**
3: Sample batch of tasks $\mathcal{T}_i \sim \mathbb{T}$
4: **for all** $\mathcal{T}_i$ **do**
5: Sample K datapoints for every class from $\mathcal{T}_i$ and put in them in the *training set* $\mathcal{Z}_{\mathcal{T}_i}$
6: Base-learn $f_{\mathcal{T}_i}$ using Λ:
$W_i = w_{\mathcal{T}_i} = \Lambda(\mathcal{Z}_{\mathcal{T}_i}) = X_i^T(X_i X_i^T + \lambda.I)^{-1} Y_i$
with $X_i = \Phi(\mathcal{Z}_{\mathcal{T}_i})$ and Y_i the one-hot labels from $\mathcal{Z}_{\mathcal{T}_i}$.
7: Sample datapoints for every class from $\mathcal{T}_i$ and put in them in the *evaluation set* $\mathcal{Z}'_{\mathcal{T}_i}$
8: **end for**
9: Update meta-parameters $\theta = (\omega, \boldsymbol{\rho})$ through gradient descent :

$$\theta \leftarrow \theta - \varepsilon . \sum_i \nabla_\theta \mathcal{L}(f_{\mathcal{T}_i}(\Phi(\mathcal{Z}'_{\mathcal{T}_i})), Y'_i)$$

with ε the learning rate, $\mathcal{L}$ the cross-entropy loss, and $f_{\mathcal{T}_i}(X'_i) = \alpha X'_i W_i + \beta$.
10: **end while**

the complexity of the matrix calculations in the special case of few-shot learning. Specifically, XX^T is of size $K \times K$, and thus will be, together with the regularization, relatively easily inverted. Normally regression is not fit for classification, but the authors noticed that it still had considerable performance. Therefore, in order to transform the regression outputs to work with the cross-entropy loss function, the meta-parameters $(\alpha, \beta) \in \mathbb{R}^2$ serve as a scale and bias respectively:

$$\hat{Y} = \alpha X' X^T (XX^T + \lambda I)^{-1} Y + \beta \tag{3}$$

4 REPRODUCIBILITY

As a first step in the reproducibility, we decided to look at the results on different datasets for a baseline. In this perspective, we decided to focus on the Model-Agnostic Meta-Learning (MAML) algorithm from Finn et al. (2017). We used the TensorFlow implementation of MAML provided by one of the original authors in Finn (2018) to reproduce the baseline results and new results on the CIFAR-FS dataset proposed by the paper (Bertinetto et al. (2019). A first consideration is that the feature extractors in MAML and R2D2 are very different. The former uses four convolutional blocks with an organization of [32, 32, 32, 32] filters whereas the latter's four blocks employ a [96, 192, 384, 512] scheme, as visualized in Figure 3. In other words, the feature extractor in R2D2 is more complex and is hence expected to yield better results. In order to provide a meaningful comparison we decided to implement both the simple and more complex feature extractors for R2D2 to evaluate the difference in performance.

Overall, we retrieved the implementation of MAML from Finn (2018), and extended it to run on the new CIFAR-FS dataset presented in the paper. Then we modified the pipeline to reproduce the R2D2 algorithm with the same feature extractor as MAML. We denote this version by R2D2*. Finally we modified the feature extractor to arrive at the architecture from the paper, and denote this as R2D2.

However, in order to reproduce the paper we had to make the following assumptions. First, we considered the aforementioned architecture and feature extractor. Those are based on our understanding of the report made by the authors. In particular for the feature extractor, we had to make some assumptions on the convolutional block options. We considered a 3x3 convolution block with a 'same' padding and a stride of 1. For the 2x2 maximum pooling, we opted for a stride of 2 and no padding. Regarding the ridge regression base-learner, we opted for a multinomial regression returning the class with the maximum value through one-hot encoding.

For comparison purposes, we decided to use the same number of classes during training and testing: 5-way uses five classes and 2-way uses two classes during training. Although the paper mentions to use a higher number of classes during training, this choice was motivated by the fact that a major contribution of Bertinetto et al. (2019) comes from the comparison with existing work. Hence it is of paramount importance to keep the results comparable. In particular in Finn et al. (2017), the results for the 5-way (1 and 5 shots) on both the miniImageNet and Omniglot datasets yield the same results as reported in Bertinetto et al. (2019), without any mention of increasing the number of classes in the multiclass classification training. We understand that, as mentioned in Bertinetto et al. (2019), increasing the amount of classes during training would yield a more generic feature extractor but since the focus of this work was on reproducibility, we decided to keep the strategy used in the baseline: 5-way means training with five classes as well.

Finally, a last assumption was made on the algorithm stopping criterion. In the paper, the stopping criterion is vaguely defined as "*the error on the meta-validation set does not decrease meaningfully for 20k episodes*". This is very vague to reproduce. Therefore we opted to meta-train for a fixed 20k iterations for all simulations involving closed-forms solvers for this reason, and also due to resource constrains.

Before going to the the reproducibility results and their analysis, it is worth mentioning that despite our best efforts, we could not reproduce part of the paper. Specifically, following the guidelines for the feature extractor presented in section 4.2 of the paper, we were not successful in reproducing the exact number of features at the output of the feature extractor. In the paper, the overall number of features at the output of the extractor add to 3584, 72576 and 8064 for Omniglot, miniImageNet and CIFAR-FS, respectively. However, by implementing the feature extractor described in the paper we get 3988, 51200 and 8192 respectively.

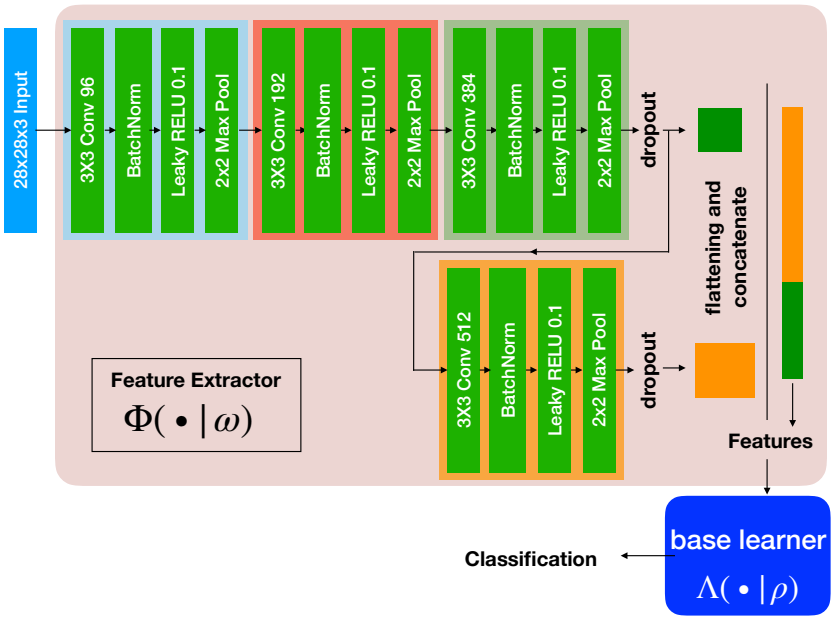

Figure 3: Overall architecture of the R2D2 system considering [96, 192, 384, 512] filters in the feature extractor with 4 convolutional blocks for the CIFAR-FS dataset.

Our implementations were made in Python 3.6.2 and TensorFlow 1.8.0 (Abadi et al. (2016)). The source code of all implementations is available[1] online[2]. Simulations were run on a server with 24 Xeon e5 2680s at 2.5 GHz, 252GB RAM and a Titan X GPU with 12 GB RAM.

[1]R2D2 and R2D2*: `https://github.com/ArnoutDevos/r2d2`
[2]MAML with cifar-fs: `https://github.com/ArnoutDevos/maml-cifar-fs`

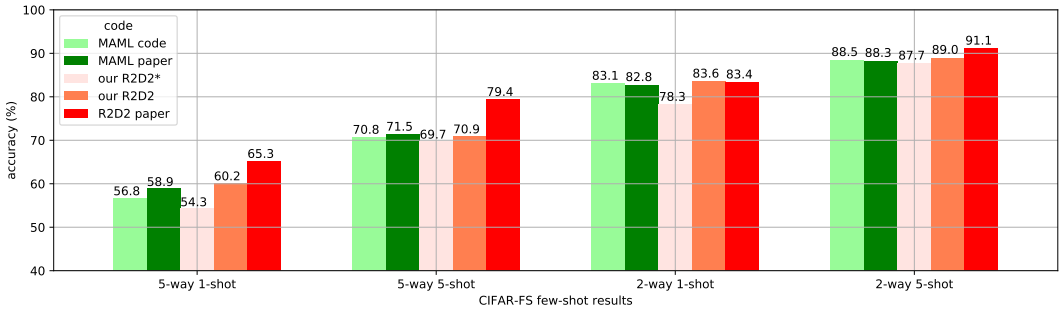

Figure 4: N-way K-shot classification accuracies on CIFAR-FS. Detailed results in Table 1.

Method	MAML paper Bertinetto et al. (2019)	MAML code ours	R2D2* ours	R2D2 ours	R2D2 paper Bertinetto et al. (2019)
5-way, 1-shot	$58.9 \pm 1.9\%$	$56.8 \pm 1.9\%$	$54.3 \pm 1.8\%$	$60.2 \pm 1.8\%$	$65.3 \pm 0.2\%$
5-way, 5-shot	$71.5 \pm 1.0\%$	$70.8 \pm 0.9\%$	$69.7 \pm 0.9\%$	$70.9 \pm 0.9\%$	$79.4 \pm 0.1\%$
2-way, 1-shot	$82.8 \pm 2.7\%$	$83.1 \pm 2.6\%$	$78.3 \pm 2.8\%$	$83.6 \pm 2.6\%$	$83.4 \pm 0.3\%$
2-way, 5-shot	$88.3 \pm 1.1\%$	$88.5 \pm 1.1\%$	$87.7 \pm 1.1\%$	$89.0 \pm 1.0\%$	$91.1 \pm 0.2\%$

Table 1: N-way K-shot classification accuracies on CIFAR-FS with 95% confidence intervals.

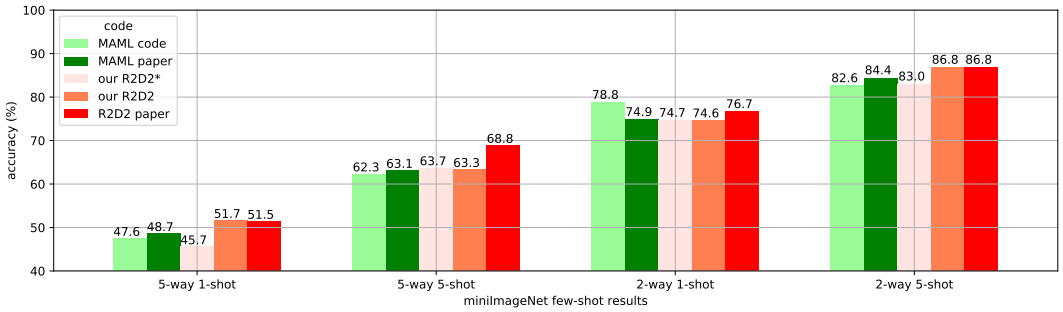

Figure 5: N-way K-shot classification accuracies on miniImageNet. Detailed results in Table 2.

Method	MAML Finn et al. (2017)	MAML Finn (2018)	R2D2* ours	R2D2 ours	R2D2 Bertinetto et al. (2019)
5-way, 1-shot	$48.7 \pm 1.8\%$	$47.6 \pm 1.9\%$	$45.7 \pm 1.8\%$	$51.7 \pm 1.8\%$	$51.5 \pm 0.2\%$
5-way, 5-shot	$63.1 \pm 0.9\%$	$62.3 \pm 0.9\%$	$63.7 \pm 1.3\%$	$63.3 \pm 0.9\%$	$68.8 \pm 0.2\%$
2-way, 1-shot	$74.9 \pm 3.0\%$	$78.8 \pm 2.8\%$	$74.7 \pm 2.9\%$	$74.6 \pm 2.9\%$	$76.7 \pm 0.3\%$
2-way, 5-shot	$84.4 \pm 1.2\%$	$82.6 \pm 1.2\%$	$83.0 \pm 1.2\%$	$84.6 \pm 1.2\%$	$86.8 \pm 0.2\%$

Table 2: N-way K-shot classification accuracies on miniImageNet with 95% confidence intervals

5 RESULTS AND CONTRIBUTIONS

Results of the different implemented architectures and algorithms for several datasets can be observed in Figures 4 and 5. More detailed results with 95% confidence intervals are shown in Tables 1 and 2. The first and last column correspond to the baselines in the original papers.

We can see that although our results differ slightly from the original paper Bertinetto et al. (2019), R2D2 (with its more complex network architecture) performs better than the MAML method for most simulations. Moreover, it comes with no surprise that, in most of the cases, with the more complex feature extractor we obtain better results for the same algorithm (R2D2 vs R2D2*). Overall, we can confirm that the R2D2 meta-learning method with its corresponding architecture yields better performance than basic MAML (with its different simpler architecture). We believe that the difference of our results with the reported values might be due to our assumptions or the stopping criterion in the training. Also, as expected, the complexity (N-ways) and the amount of data (K-shots) play a major role in the classification accuracy. For all the methods, accuracy drops when the number of ways increases and number of shots decreases. An outlier worth mentioning is our MAML simulation on miniImageNet with 2-way 1-shot that behaves better than what was reported in Finn et al. (2017).

To summarize, we have reproduced the most important results presented in Bertinetto et al. (2019). Although the reproduced results and paper results slightly differ, the general observations of the author remain valid. Their meta-learning with differentiable closed-form solvers yields state of the art results and improves over another state of the art method. However, as a remark, we would state that the assumptions we made could have been clarified in the original paper. We indeed believe that they could be the source of the discrepancy in our results. Due to resource constraints and since it is only used for few-shot binary classification we have not focused on reproducibility of the Logistic Regression based algorithm LRD2.

Overall, through this reproducibility project, we have made the following contributions:

- An algorithmic description of the R2D2 version of meta-learning with differentiable closed-form solvers (Algorithm 1).
- Evaluation of the MAML pipeline from Finn (2018) on two datasets: the existing miniImageNet and new CIFAR-FS for different few-shot multi-class settings.
- Implementation of R2D2* in TensorFlow on the pipeline following Algorithm 1 with the original MAML feature extractor.
- Implementation of R2D2 in TensorFlow on the pipeline following Algorithm 1 with the Figure 3 architecture as mimicked from in the original paper (Bertinetto et al. (2019)).
- Evaluation of the reproducibility of Bertinetto et al. (2019).

6 CONCLUSION

In this paper we presented a reproducibility analysis of the ICLR 2019 paper *"Meta-learning with differentiable closed-form solvers"* by Bertinetto et al. (2019). Some parameters and training methodologies, which would be required for full reproducibility , such as *stride* and *padding* of the convolutional filters, and a clear stopping criterion, are not mentioned in the paper nor in its appendix (Bertinetto et al. (2019)). However, making reasonable assumptions, we were able to reproduce the most important part of the paper and achieve similar results. Most importantly we were able to reproduce the increase in performance of the proposed method over some reproduced baseline results, which supports the conclusions in the original paper. However, the different neural network architectures should be taken into consideration when comparing results. Specifically we have shown that when using the exact same baseline architecture the improvement in performance of the proposed method is not clear.

REFERENCES

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-

scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Luca Bertinetto, Joao F. Henriques, Philip Torr, and Andrea Vedaldi. Meta-learning with differentiable closed-form solvers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxnZh0ct7.

Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.

Chelsea Finn. Code for "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks": cbfinn/maml, December 2018. URL https://github.com/cbfinn/maml. original-date: 2017-06-18T01:36:06Z.

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

Victor Garcia and Joan Bruna. Few-Shot Learning with Graph Neural Networks. *arXiv:1711.04043 [cs, stat]*, November 2017. URL http://arxiv.org/abs/1711.04043. arXiv: 1711.04043.

Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A Simple Neural Attentive Meta-Learner. *arXiv:1707.03141 [cs, stat]*, July 2017. URL http://arxiv.org/abs/1707.03141. arXiv: 1707.03141.

Alex Nichol and John Schulman. Reptile: a scalable metalearning algorithm. *arXiv preprint arXiv:1803.02999*, 2018.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. pp. 11, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical Networks for Few-shot Learning. *arXiv:1703.05175 [cs, stat]*, March 2017. URL http://arxiv.org/abs/1703.05175. arXiv: 1703.05175.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching Networks for One Shot Learning. *arXiv:1606.04080 [cs, stat]*, June 2016. URL http://arxiv.org/abs/1606.04080. arXiv: 1606.04080.