STRUCTURED PREDICTION FOR CONDITIONAL META-LEARNING USING DEEP MODELS: REPRODUCTION AND ANALYSIS

Muhammad Uzair Abid
mua1n20@soton.ac.uk

James Evans
je4g16@soton.ac.uk

Marco Edoardo Palma
mep2g17@soton.ac.uk

ABSTRACT

Optimisation-based meta-learning aims to derive a shared initialisation of parameters from a distribution of tasks to be used for a newly drawn task. Unlike standard model-agnostic approaches that make no assumption about the affinity of the training tasks with the target task, such as model-agnostic meta-learning (MAML) [Finn et al. (2017)], *Task Adaptive Structured Meta Learning* (TASML) [Wang et al. (2020)] proposes a non parametric approach to analytically weigh the contribution of each training task to aid fine-tuning on the target task. This report not only investigates the reproducibility of the performance claims made by [Wang et al. (2020)] on benchmark datasets, but also comments on the performance of a number of deep models implemented for TASML and MAML.

1 INTRODUCTION

Meta-learning is a family of learning strategies that tackle problems having small datasets by exploiting inductive biases offered by the training data to derive an initialisation state for fine-tuning on the target dataset. The *n-way k-shot* problem is one application of meta-learning. Here, the problem statement is a collection of tasks: disjoint datasets of n classes each with k training samples. Parameters learnt from the training tasks are then used for fine-tuning a target task. The accuracy of the model is then measured on its ability to classify the unseen samples in the target task [Vinyals et al. (2017)].

This report delivers a view on the performance of TASML meta-learning strategy; a conditional variant of the MAML strategy that gives structured outputs. MAML and TASML utilise a generic bi-level [Wang et al. (2020)] optimisation approach which allows for performing task-specific optimization in the inner algorithm and uses the gathered experience to learn meta-parameters in the outer algorithm. Testing out [Wang et al. (2020)] claims on the arbitrary choice of the inner algorithm we implement deep models[1] and record their performance on benchmark datasets. Finally, we provide a review of TASML's implementation using the proposed least-squares model, and report on the reproducibility of the original accuracy figures.

2 TASML IMPLEMENTATION REVIEW

Traditional approaches to meta-learning such as MAML aim to learn a shared initialisation $\boldsymbol{\theta} \in \Theta$, a space of meta-parameters, across the set of all training tasks $S = \{D_i^{tr}, D_i^{val}\}_{i=1}^{N}$ for a given target task. In a n-way k-shot problem, D_i^{tr} and D_i^{val} refer to the support and query set respectively where training is done using the k-examples per class in the training set and prediction loss is minimized on the query set. This initialisation is then used to run one or several parameter update steps on the support set D^{tr} of the target task D to predict classes of the query set D^{val}.

In this section we deliver a summary of the theoretical and implementation details of the provided TASML implementation[2] formalised in [Wang et al. (2020)]. The process is broken into three steps. N-way k-shot tasks are first constructed using common few-shot learning benchmarks, two of which

[1] https://github.com/uzborg950/COMP6248-CW-MetaLearning-SP

[2] https://github.com/RuohanW/Tasml

are the *mini*ImageNet and *tiered*ImageNet datasets. This is done using pre-trainined embeddings generated from [Rusu et al. (2019)]. For each task, the embeddings are used to construct a kernel mean embedding $\bar{\phi} : D \rightarrow \mathbb{R}^p$ which is an average of the query and support set of the task D. The kernel mean embedding is then later used to find the maximum mean discrepancy,

$$\boldsymbol{k}(D, D') = exp(\frac{-||\bar{\phi}(D) - \bar{\phi}(D')||^2}{\sigma^2}) \tag{1}$$

where $\sigma^2 > 0$ is a bandwidth parameter set as 1 in the implementation.

The generated database of training tasks is then optionally used for unconditional meta-learning to warmstart the model for a better parameter initialisation. The architecture uses a least-squares solution as the inner algorithm that finds parameters in closed form by minimizing the mean square error loss. Note that in our implementation we use deep models of varying width and depth to minimize the cross-entropy loss in the inner algorithm. As suggested in [Wang et al. (2020)] the least-squares solution is only provided to give theoretical guarantees and can be replaced with another learning algorithm. The outer algorithm then uses the Adam optimizer [Kingma & Ba (2017)] to minimize the cumulative loss on the query sets of the training tasks. It can be stated that the TASML algorithm is the same as MAML if all the weights $\alpha_i(D) = 1$ for each training task D_i and there isn't an additional regularising term.

Agreeing with the algorithm formalised in [Wang et al. (2020)], the model aims at finding parameters $\tau(D) \in \Theta$ by minimising the loss function in [Ciliberto et al. (2019)] and adds an additional optimisation term $\mathcal{L}(Alg(\theta, D), D)$ to encourage the model to learn signals from the training set in a way that is compatible with the target task.

$$\tau(D) = \underset{\boldsymbol{\theta} \in \Theta}{argmin} \sum_{i=1}^{N} \alpha_i(D) \mathcal{L}(Alg(\theta, D_i^{tr}), D_i^{val}) + \mathcal{L}(Alg(\theta, D), D) \tag{2}$$

$$\alpha_i(D) = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1} \upsilon(D) \in \mathbb{R}^N \tag{3}$$

The similarity of each training task D_i with the target task is found using a weighting function $\alpha : D \rightarrow \mathbb{R}$. $\boldsymbol{K} \in \mathbb{R}^{N \times N}$ is a reproducing kernel matrix where each entry $\boldsymbol{K}_{i,j} = \boldsymbol{k}(D_i^{tr}, D_j^{tr})$ is the maximum mean discrepancy [Gretton et al. (2012)] between the two tasks (See Equation (1)). Similarly the distance between the training task D_i^{tr} and target task D is found using the evaluation vector $\upsilon(D)_i = \boldsymbol{k}(D_i^{tr}, D) \in \mathbb{R}^N$ which is the maximum mean discrepancy between the two tasks. A higher $\alpha_i(D)$ weight determines the impact of a training task on finding structured outputs conditioned on the target task. TASML proposes improvements to the loss function in [Ciliberto et al. (2019)] to speed up convergence.

1. **Warm-start by MAML**: Run unconditional MAML for a fixed number of steps using all the training tasks in order to initialize parameters of the inner algorithm of the model. This enables the model to converge early but is an optional step.
2. **Top-M Filtering**: Filters M training tasks for each target task that have the highest $\alpha(D)$ weights. The figure M is normally kept at 1% of the total training tasks. In this way, only the most relevant tasks contribute to training leading to performance gains.
3. **Task Adaptation**: Adds another optimisation term $\mathcal{L}(Alg(\theta, D), D)$ to exploit past training signals that were relevant to the target task thus acting as a regularising term.

3 EXPERIMENTS

For a fair comparison we implemented MAML and TASML using deep models in the inner algorithm to test out the claims of improvement by [Wang et al. (2020)]. This includes the implementation of finding weights $\alpha_i(D)$ and performing top-m filtering. A baseline model (BASE) was used as a vanilla trained and fine-tuned neural network, with the aim of comparing the accuracy produced by the meta-learning strategies. Testing was carried out on four different deep neural networks of increasing depth and width limited by the computing power at the team's disposal. *Model 0* was

reproduced from [Finn et al. (2017)], using the same testing scenarios reported. It consists of four ReLu activated dense layers of sizes 256, 128, 64 and 64 with batch normalisation. *Model 1* replicates the former but drops the batch-norm layers. *Model 2* is a wider version of *Model 1* with deeper layers consisting of sizes 512, 256, 128, 64 and 64. Finally, *Model 3* is a wider instance of *Model2* with layer sizes 1024, 512, 246, 128 and 64 with batch normalisation at every layer. Finally, the performance of MAML and TASML was recorded for 10 target tasks for both the *5-way 1-shot* and *5-way 5-shot* settings. As reported in [Wang et al. (2020)], embeddings of the *tiered*ImageNet and *mini*ImageNet benchmark datasets were used for training. Due to hardware limitations, the team was unable to execute few-shot problems with 30000 training samples. The team limited the training of MAML using 500 training samples (the maximum tractable) and thus provided a comparison of TASML and BASE at this amount. Furthermore, tests with 10000 training samples were carried out by feeding MAML and BASE with a top-m filtered dataset, in order to highlight the effects of top-m filtering alone. To test original claims, we first reproduce results using the provided TASML implementation.

Table 1: Classification accuracy of original and reproduced TASML model for 30000 training samples.

Accuracy (%)				
Method	*mini* 1-shot	*mini* 5-shot	*tiered* 1-shot	*tiered* 5-shot
TASML(Original)	62.04 ± 0.52	78.22 ± 0.47	66.42 ± 0.37	82.62 ± 0.31
TASML(Reproduced)	59.20 ± 11.10	78.34 ± 6.34	66.48 ± 10.62	82.33 ± 7.43

Table 2: Classification accuracy of implemented models for 500 training samples.

Accuracy (%) - Model 0				
Method(TopMFiltering)	*mini* 1-shot	*mini* 5-shot	*tiered* 1-shot	*tiered* 5-shot
BASE(False)	21.88 ± 1.91	22.79 ± 1.04	22.19 ± 2.81	27.19 ± 2.50
BASE(True)	37.88 ± 5.61	58.27 ± 9.24	52.94 ± 5.14	66.76 ± 4.20
MAML(False)	34.80 ± 6.33	58.93 ± 6.59	34.71 ± 6.35	65.38 ± 8.58
MAML(True)	37.24 ± 6.39	61.18 ± 9.48	53.17 ± 8.31	69.05 ± 5.78
TASML(True)	37.34 ± 7.91	53.57 ± 10.39	44.59 ± 12.62	65.71 ± 11.56

Table 3: Classification accuracy of implemented models for 10000 training samples.

Accuracy (%) - Model 0				
Method(TopMFiltering)	*mini* 1-shot	*mini* 5-shot	*tiered* 1-shot	*tiered* 5-shot
BASE(True)	27.11 ± 5.62	32.59 ± 4.20	32.95 ± 5.13	42.02 ± 8.28
MAML(True)	44.26 ± 8.85	58.05 ± 8.06	47.83 ± 9.04	66.61 ± 6.64
TASML(True)	38.78 ± 9.15	50.39 ± 11.37	45.07 ± 9.85	65.47 ± 10.36
	Accuracy (%) - Model 1		Accuracy (%) - Model 2	
Method(TopMFiltering)	*mini* 1-shot	*tiered* 1-shot	*mini* 1-shot	*tiered* 1-shot
BASE(True)	19.13 ± 2.14	25.39 ± 3.63	21.24 ± 2.72	25.92 ± 3.59
MAML(True)	29.98 ± 11.20	24.82 ± 9.78	25.14 ± 7.17	22.64 ± 5.55
TASML(True)	33.91 ± 8.58	39.20 ± 8.07	28.27 ± 5.11	33.22 ± 11.91
Accuracy (%) - Model 3				
Method(TopMFiltering)	*mini* 1-shot	*mini* 5-shot	*tiered* 1-shot	*tiered* 5-shot
BASE(True)	22.97 ± 2.71	32.86 ± 6.17	29.39 ± 4.92	46.50 ± 5.71
MAML(True)	41.50 ± 8.93	57.92 ± 8.59	47.07 ± 7.02	68.18 ± 5.69
TASML(True)	40.74 ± 10.35	57.11 ± 8.01	46.62 ± 9.50	63.88 ± 14.34

3.1 Experiment Results

As the deepest model (*Model 3*) took 10 hours for 1 run of 10 test tasks consisting of BASE, MAML and TASML, we decided to average test task performance over 1 run only as opposed to 50 in [Wang et al. (2020)]. As a consequence, large standard deviations were reported across the board including the tests done using the provided TASML implementation. Despite the changes in experimental setup, a similar mean accuracy to the original TASML results was reported in our reproduced results.

For all the methods and models, a higher accuracy was observed with *tiered*ImageNet which further agrees with the results reported in [Wang et al. (2020)].

Top-M filtering reduces the number of training tasks to the top *M* with highest affinity to the target task. We were able to pinpoint the performance of top-m filtering alone by using it alongside BASE and MAML; an alteration not tested previously in existing works. We observe in the batch-norm models *Model 0* and *Model 3*, filtering causes MAML and TASML to perform similarly. One argument against this observation is that TASML weighs each top-*M* training tasks while MAML does not so TASML should perform better. In reality, the $\alpha_i(D)$ weights are generally small with little variance between them within the top m set. Thus these have a negligible effect on the following meta-learning process leading to TASML to behave like MAML. The performance of top-M filtering is further validated by the *Model 0* - 500 samples setup where $M = 5$. Despite a small number of training samples, BASE, MAML and TASML perform better when trained on this reduced set. In fact, BASE gains a boost of 30.45% in average accuracy with the property enabled across all the few-shot learning experiments.

We can observe that in the non-batch norm models *Model 1* and *Model 2* TASML performs better than the rest but has the lowest accuracy overall as compared to the other models. We hypothesize that this could be because the regularising effect of batch-norm leads to better generalization performance and thus provides boosts in accuracy for *Model 0* and *Model 3*. We stress that the significant differences in performance to the original implementation is likely due to the use of deep models in the inner algorithm with a smaller size of training samples and lesser computational time availability. The least squares closed form solution in the original implementation is likely to perform better and our implementation provides a strict comparison of the use of the two different approaches.

4 Conclusions

In this work we aimed at reproducing the performance boost of meta-learning with the novel TASML architecture. After validating the implementation of such strategy from the authors, we replicated the testing scenarios reported in the literature, and were able to successfully collect comparable accuracies as the ones reported. Custom implementation of both the MAML and TASML strategies were developed and tested on a number of deep models with the *n-way k-shot* format on *mini*ImageNet and *tiered*ImageNet embeddings. We were able to detect some benefits with the use of the approach subject of this enquiry. This was the case despite the team's hardware limitations constraining the maximum number training tasks, which handicapped the performance of TASML as this made its ability to weight tasks affinity with the target's vain in some scenarios.

References

Carlo Ciliberto, Francis Bach, and Alessandro Rudi. Localized structured prediction, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(null):723–773, March 2012. ISSN 1532-4435.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

Andrei A. Rusu, Dushyant Rao, Jakub Sygnowski, Oriol Vinyals, Razvan Pascanu, Simon Osindero, and Raia Hadsell. Meta-learning with latent embedding optimization, 2019.

Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning, 2017.

Ruohan Wang, Yiannis Demiris, and Carlo Ciliberto. Structured prediction for conditional meta-learning, 2020.