Accelerating Gradient-based Meta Learner

Varad Pimpalkhute
IIIT Nagpur
India
pimpalkhutevarad@gmail.com

Amey Pandit, Mayank Mishra, Rekha Singhal
TCS Research
India
amey.pandit@tcs.com,mishra.m@tcs.com,rekha.singhal@tcs.com

ABSTRACT

Meta Learning has been in focus in recent years due to the meta-learner model’s ability to adapt well and generalize to new tasks, thus, reducing both the time and data requirements for learning. However, a major drawback of meta learner is that, to reach to a state from where learning new tasks becomes feasible with less data, it requires a large number of iterations and a lot of time. We address this issue by proposing various acceleration techniques to speed up meta learning algorithms such as MAML (Model Agnostic Meta Learning). We present 3.73X acceleration on a well known RNN optimizer based meta learner proposed in literature [11]. We introduce a novel method of training tasks in clusters, which not only accelerates the meta learning process but also improves model accuracy performance.

KEYWORDS

Meta learning, RNN optimizer, AGI, Performance optimization

1 INTRODUCTION

Artificial general intelligence (AGI) is a hypothetical concept where machines are capable of learning and thinking like humans. Meta learning also known as learning to learn, brings us one step closer to achieve AGI by learning on a task in a manner similar to humans. To give an analogy, in Meta-learning paradigm, a distribution of related tasks (say racket sports: Badminton, Tennis etc.) are used to train a model (human) which can use this "experience" to learn any new task (Squash) with lesser amount of data (lesser practice of Squash than a person who doesn’t know any racket sport) and in lesser time.

The meta learning algorithms, like MAML [2] (and others [8, 10, 11]), are time and compute intensive. These algorithms require a lot of iterations over the available tasks (60000 [2]) to make a model converge and be ready to learn a new task quickly with good accuracy.

In this paper, we present our ongoing work involving several performance optimization techniques to reduce the training time of metalearning algorithms. We will discuss these techniques in the context of acceleration of a gradient based meta learner proposed in [11]. A gradient based meta learner learns how gradients change during gradient descent based optimizations of various learning tasks (learning task involves training a model). The goal is to enable rapid model parameter updates (faster than optimizers like SGD, ADAM, RMSProp) after each learning iteration 1.

The gradient based meta learner which is our focus in this paper employs a Hierarchical RNN 2 architecture to ensembles the loss landscapes of various small tasks. Figure 1 shows how this meta learner learnt from a single task’s training process (the same is repeated for every task). There are two learning loops: a) Task loop - which updates the task parameters for every batch of data using hierarchical RNN as optimizer; and b) Meta loop - which updates the parameters of RNN optimizer itself using loss accumulated across all batches of task learning using a RMSProp optimizer. This process is repeated multiple times (N epochs) for each task and sequentially for every task available in a single meta iteration. Large number of such meta iterations are required to let Hierarchical RNN based meta optimizer converge.

Figure 2 shows the sequential nature of the meta training approach and the iterations involved using a flow-chart. We improved

---

1There are other approaches of meta learning too, namely, metric based and loss based meta learners [4].
2RNN: Recurrent Neural Network
on the model performance by introducing a better and wider meta-
optimization pipeline, which incorporates the optimizer parameters 
being learned in form of task clusters. We achieved a speedup of 
more than 2x with our proposed approach and overall speedup of 
3.73x by also incorporating several coding related optimizations. 
The rest of the paper is structured as follows: We briefly discuss the 
previous related work in the field of meta learning and accelera-
tions in Section 2. We revisit definition of tasks in Section 3. We 
propose the approaches for accelerating Hierarchical RNN in Sec-
tion 4. We verify and discuss the effectiveness of our experiments in 
Section 5. Finally, we discuss the ongoing work in Section 6 and 
conclude the paper in Section 7.

2 RELATED WORK
Gradient based meta-learners proposed in literature [3, 5, 6, 8] have 
worked on improving model accuracy. However, very little work 
has been done on improving the training time of the meta learner, 
where meta-learner especially MAML is a time hungry algorithm. 
Gradient descent on the meta-parameters (w) involves comput-
ing second-order derivatives or a simplified and cheaper first order 
implementation [2, 7, 9]. [9] proposes ANIL which makes use of 
feature reuse for few-shot learning. ANIL simplifies on MAML by 
using almost no inner loop; thus being more computationally effi-
cient than MAML, getting a speed up as high as 1.8x. [7] focuses 
on decreasing the computations and memory by not unrolling a 
computational graph or calculating any second derivatives. [1] pro-
pose a distributed framework to accelerate the learning process of 
MAML.

In this work, we also aim to accelerate the learning process by 
proposing various optimization techniques. Contrary to the above 
approaches (which involve changing the mathematical model or 
framework), our approach is more intuitive and less complex.

3 TASKS PREPARATION FOR META 
LEARNING
The Hierarchical RNN based meta learner work proposed in [11] 
which we intend to accelerate is designed to be an optimizer with 
rapid updates. Consequently, it becomes important that the opti-

mizer should work for any model, may it be CNN, Fully connected, 
RNN, logistic regression, etc. Thus, optimizer is trained to give 
rapid updates by training it on various instances of these models. 
The instance of a model is considered as task. The diversity of 
tasks helps in generalizing the model well to new tasks. Datasets 
to train the task (model) are generated by following the approach 
mentioned in paper.

4 PROPOSED APPROACHES FOR 
ACCELERATION
The approaches we tried for speeding up Hierarchical RNN based 
meta learner can be divided into two categories A) Algorithmic 
optimizations, and B) Program optimizations, as shown in Figure 3.

We focus on discussions of Algorithmic optimizations performed 
in this section, while limiting the discussion around program opti-
mizations tried in experiments section as they are mostly straight-
forward and simple to understand.

4.1 Parallelization inside Task Loop
The task-loop shown in Figures 1 and 2 involved training over 
multiple batches of data sequentially. Instead of training over data 
batches sequentially we train over multiple batches in parallel. We 
calculate each batch’s loss and parameter gradients and then calcu-
late average of individual parameter’s gradients. Finally, updating 
the task parameters using these averaged parameter gradients. The 
updated model is used in the next iteration. The degree of paral-
lelism is configurable and depends on the underlying hardware.

4.2 Considering tasks in clusters and training 
clusters parallely
Although, meta learning approaches employ similar tasks for train-
ing the meta learner to begin with, some tasks in the task set are 
more similar to each other than other tasks. For example, a softmax 
regression task will be more similar to other softmax regression 
tasks in terms of how gradients get updated in each optimisation 
step than task which involves training a fully connected neural 
network. The Hierarchical RNN based meta optimizer approach 
doesn’t differentiate between such tasks and while meta training 
the "sequence" of tasks is randomly chosen.

We grouped the tasks into clusters depending on their similarity 
with each other and then we performed meta training across these 
clusters parallely. The Hierarchical RNN optimizer’s parameters 
were updated by using the averaged parameter gradients across 
these clusters. This iteration of meta training parallely over the clus-
ters is repeated for a number of times (number of meta-iterations is 
user defined). To properly utilise the underlying hardware and en-
suring that no one branch of parallelization becomes bottleneck and 
takes too long we group clusters into cluster groups and schedule 
cluster groups as shown in Figure 4.

The approach is that all cluster-groups should take similar time 
for meta learning iteration. Figure 4 shows 5 task clusters contain-
ing similar tasks namely Softmax Regression, Quadratic problems, 
2D problems, Bowl problems, and Fully connected problems are 
scheduled as two cluster groups.

Figure 5 gives the complete picture of the algorithmic optimiza-
tion we performed in form of a flow chart. The original sequential 
pipeline of meta training mentioned in flowchart of Figure 2 has 
been broadened by introducing parallelizations in task loop (box

There are various ways to create clusters of tasks. Thus for simplicity, we consider 
creation of clusters as a black box which returns k clusters, each cluster having similar 
tasks.
5 RESULTS & DISCUSSION

In this section, we perform experiments to verify the effectiveness of proposed acceleration techniques. The hardware and software configuration employed for experiments is mentioned in Table 1. Before performing experiments we upgraded the baseline code available from prior work [11] to newer libraries.

We performed experiments in a resource constrained setup as can be seen in Table 1. We believe that a higher configuration setup will exhibit better acceleration and performance gains due to possibility of higher parallelism. All the experiments are performed while ensuring that the achieved test accuracy of 0.65 for test “task-set” (consisting of softmax regression tasks) on base model is matched using our proposed approach. We now present details of experiments performed for each of proposed acceleration technique.

### 5.1 Acceleration achieved due to parallelism introduced

Table 2 lists the reduction in overall execution time due to parallelizations introduced. We varied the number of task clusters used for meta training and always created 2 cluster groups to keep degree of cluster training parallelization fixed at 2. This was done due to hardware limitations of our setup. However, the improvement in terms of time saving is encouraging. The experiments were speeded by more than 2x (base time vs optimized time) while the accuracy achieved on test task set remained same.

| Num Clusters | Base Time (in sec) | Optimized Time (in sec) |
|--------------|--------------------|------------------------|
| 2            | 1450.15            | 826.53                 |
| 3            | 6313.8             | 2977.83                |
| 4            | 11361.3            | 4918.31                |
| 5            | 21932.75           | 12473.1                |

We also introduced program and code level changes mentioned in Figure 3. Next we discuss what changes were made and how they improved the performance.

### 5.2 Program and Code Optimizations

There are several straightforward optimization like upgrading the libraries and platforms used in the available code. For example, upgrading python itself resulted in 15% speedup. However, to our surprise upgrading Tensorflow Library didn’t result in any speedup.
We are experimenting with different ways and granularities in which the task clusters can be created. We believe grouping similar tasks together in clusters and training meta learner over them will result in faster convergence than without clustering. This is because the the gradient updates for tasks in same cluster will be more similar to each other.

We are also experimenting whether sequence in which tasks-cluster are used for meta-learning has impact on convergence time of meta learning. A simple analogy from racket sports is that a person good at Racket-ball will find it easier to learn Squash as compared to Tennis 4.

We also experimented replacing GRU cells used in Hierarchical RNN optimizer with LSTM in hope of achieving convergence with lesser number of meta iterations. Although, the convergence happened in lesser number of iterations than when GRU cells were used, the time per iteration increased considerably, thus, nullifying the gains. We shelved this idea for future.

7 CONCLUSIONS

We presented our on-going work involving acceleration of a Hierarchical RNN meta learner proposed in [11]. We changed the sequential meta learning pipeline discussed in paper to a parallel pipeline by introducing parallizations inside training of the tasks as well as across clusters of tasks. We also introduced grouping of tasks clusters for better scheduling.

Apart from introducing wider meta-learning pipeline we also introduced certain program and code level optimizations. Put together the proposed techniques resulted in a 3.73X speed-up of the meta-learning algorithm.

REFERENCES

[1] Jan Bollenhacer, Florian Soulier, Beate Rhein, and Laurenz Wiskott. 2020. Investigating Parallelization of MAML In Discovery Science, Anzala Appice, Grigoris Tsoumakas, Yannis Manolopoulos, and Stan Matwin (Eds.). Springer International Publishing, Cham, 294–306.

[2] Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. CoRR abs/1703.03400 (2017). arXiv:1703.03400 http://arxiv.org/abs/1703.03400

[3] L. Franceschi, P. Frasconi, Saverio Salzo, Riccardo Grazzi, and M. Pontil. 2018. Bilevel Programming for Hyperparameter Optimization and Meta-Learning. ArXiv abs/1806.04910 (2018).

[4] Timothy Hospedales, Antreas Antoniou, Paul Micarelli, and Amos Storkey. 2020. Meta-learning in neural networks: A survey. arXiv preprint arXiv:2004.05439 (2020).

[5] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. 2019. Meta-Learning With Differentiable Convex Optimization. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

[6] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. 2017. Meta-Learning with Temporal Convolutions. CoRR abs/1707.03141 (2017). arXiv:1707.03141 http://arxiv.org/abs/1707.03141

[7] Alex Nichol, Joshua Achiam, and John Schulman. 2018. On First-Order Meta-Learning Algorithms. CoRR abs/1803.02999 (2018). arXiv:1803.02999 http://arxiv.org/abs/1803.02999

[8] Aniruddha Raghu, Mathira Raghu, Samy Bengio, and Oriol Vinyals. 2020. Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML. In 8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020. OpenReview.net. https://openreview.net/forum?id=rkgMkCEtPB

[9] S. Ravi and H. Larochelle. 2017. Optimization as a Model for Few-Shot Learning. In International Conference on Learning Representations.

[10] Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical Networks for Few-shot Learning. CoRR abs/1703.05175 (2017). arXiv:1703.05175 http://arxiv.org/abs/1703.05175

[11] Olga Wichrowska, Niru Maheswaranathan, Matthew W. Hoffman, Sergio Gomez Colmenarejo, Misha Denil, Nando de Freitas, and Jascha Sohl-Dickstein. 2017. Learned Optimizers that Scale and Generalize. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017 (Proceedings of Machine Learning Research, Vol. 70). Doina Precup and Yee Whye Teh (Eds.). PMLR, 3751–3760. http://proceedings.mlr.press/v70/wichrowska17a.html