Neural Computing and Applications (2022) 34:13529–13537
https://doi.org/10.1007/s00521-022-07160-1

ORIGINAL ARTICLE

One-step model agnostic meta-learning using two-phase switching optimization strategy

Saad Mahmud[1] · King Hann Lim[1]

Received: 28 July 2021 / Accepted: 2 March 2022 / Published online: 30 March 2022

Abstract
Conventional training mechanisms often encounter limited classification performance due to the need of large training samples. To counter such an issue, the field of meta-learning has shown great potential in fine tuning and generalizing to new tasks using mini dataset. As a variant derived from the concept of Model Agnostic Meta-Learning (MAML), an one-step MAML incorporated with the two-phase switching optimization strategy is proposed in this paper to improve performance using less iterations. One-step MAML uses two loops to conduct the training, known as the inner and the outer loop. During the inner loop, gradient update is performed only once per task. At the outer loop, gradient is updated based on losses accumulated by the evaluation set during each inner loop. Several experiments using the BERT-Tiny model are conducted to analyze and compare the performance of the one-step MAML with five benchmark datasets. The performance of evaluation shows that the best loss and accuracy can be achieved using one-step MAML that is coupled with the two-phase switching optimizer. It is also observed that this combination reaches its peak accuracy with the fewest number of steps.

Keywords Meta-learning · MAML · Fine tuning · Two-phase switching optimization · SDAGD

1 Introduction

Adapting to the scarcity of samples during training is an essential trait to improve the learning capability of a deep learning model on specific tasks. Meta-learning [1–3] is an alternative solution to train the network with fewer examples to achieve accurate task performance using metadata. It applies metadata using a two-loops mechanism to guide the training efficiently to learn the patterns with the least number of training samples. As a result, meta-learning [3, 4] is also known as"learning to learn" mechanism that can enable the new model design to rapidly learn new tasks or adapt to new environments with a few training examples. This great stride of meta-learning has been made in the field of few-shot learning from emerging models and training methods such as Siamese networks [5], matching networks [6] and memory augmented networks [7].

One of the variants in meta-learning is known as Model Agnostic Meta-Learning (MAML). MAML [8] was created with the goal of teaching the base network to be more versatile and adaptive to more than one tasks. This method can be used in classification, regression and in reinforcement learning. MAML conducts the training procedure using two loops, which are known as the inner loop and the outer training loop. Both the training loops try to achieve good performance using different objectives. In the inner loop, MAML tries to guide the model in such a way that the lowest value for the training loss is achieved for a particular task. In the outer loop, the objective function is used to attain the optimum parameters that can generalize to similar sets of tasks [8]. The main flaw of MAML [9] is that the performance and stability of this method varies greatly when the number of parameters is substantially increased from a base network. This problem is tackled and solved in the approach known as MAML++ [9]. MAML++ accounts for the instability of MAML by calculating the loss on the target set at every inner loop. These losses are then weighted and accumulated which is used to update the meta weights of base network. However,

✉ Saad Mahmud
shuvro.mahmud79@gmail.com

[1] The Department of Electrical and Computer Engineering, Curtin University Malaysia, Miri, Malaysia

MAML++ still preserved certain drawbacks such as large memory consumption, longer training speed and poor optimization for larger size of network parameters.

In this paper, a novel meta-learning method called one-step MAML is developed and applied in the field of meta-learning. One-step MAML is a variation of MAML where training is conducted using inner and outer loops. One-step MAML is trained with the aim of achieving an effective accuracy score in presence of limited amount of data. It attempts to tackle the vanishing and exploding gradient problem that is typically seen during MAML training. Traditionally, during the outer loop, MAML [8] tries to unroll through the multiple training steps performed per task during the inner loop to derive the gradient for the meta weights. This typically develops into a vanishing and exploding gradient problem. One-step MAML performs only one training step during the inner loop per task and updates the meta weight. This relinquishes the need for storing the inner loop gradients in memory for it to be unrolled later. This way the memory usage during training is also lowered. One-step MAML is then combined with the SDAGD optimizer that utilizes a two-phase switching strategy to obtain faster convergence and reduce overfitting. The combination and modification of these techniques results into a new and improved meta-learning method that is used for the purpose of achieving a better accuracy score.

2 Related work

Conventional training of neural architecture [10] has a lot of limitations which needs to be tackled before a neural network can perform well in real life conditions. One such limitation [11, 12] is the requirement of huge amounts of diverse data, before a model can perform well for the purpose that it was trained for. Various objectives [13] might lack such copious proportion of data and thus learning to complete that objective becomes increasingly difficult for various models. Traditional deep learning models [2] suffer from restrictive performance in the sense that it might only perform effectively on one specific task. These models might be significantly less effective if it encounters a task that has a slightly different aim than the one it was trained on. Meta-learning [14] is a field that tries to address these barriers toward performance, by creating a scenario where models train with the aim of learning meta-data that affects the performance for a given objective. It can be sub-classified into three categories, i.e., Metric-based approach, Model-based approach and Optimization-based approach. Each of these approaches is discussed in detail in the following sub-sections.

2.1 Metric-based meta-learning

Metric-based methods [1] are based upon the same core ideology that various statistical models use in machine learning. The performance of such techniques is evaluated upon the 'metric' used and the accuracy of the calculated value. The metrics taken under consideration can be any model hyper-parameter or even the loss function. Identification of the ideal criterion to be optimized is heavily influenced by the problem statement that the model is trying to overcome. These methods conceptualize the input data and the correlation between them on a n-dimensional feature space and performs the updates based on that.

Koch et al. [5] devised a mechanism using Siamese Networks to utilize a metric that calculates a distance value based on the similarity of the inputs. The aim of this model is to perform training on image classification with as few as one input image, formally known as 'one-shot learning'. In Siamese Networks, two images are passed as input at the same time. These inputs undergo operations that extracts the features in the images and converts them into image embeddings. These operations are run parallelly, and the two embeddings, belonging to the two inputs, are then used to compute the distance metric. For one-shot learning, training is conducted with the purpose of learning the similarities and differences between the two images. During inference, the model compares the new input to its already learned embeddings which leads to identification of the target class.

Vinyals et al. [6] incorporates the concepts of attention and meta-learning and introduces 'Matching Networks'. Like Siamese networks, Matching Networks also converts input features into a vector embedding. This time however, the computations for calculating the two embeddings are different. One function is used to arrive at the embedding for the support set used during training, while another different function is used to transform the test sample. Attention is used to analyze these two separate representations and then classification is done based on that. On the other hand, Snell et al. [15] demonstrated a technique of representing the target classes as vectors. These vectors are known as 'prototype vectors', and each target class will have an associated prototype vector referring to it. The vectors generated are multi-dimensional in nature. When an input is passed to the model, several distance values are computed based on its embedding vector and the prototype vectors, which are then passed through a Softmax layer to identify the correct class.

Guo et al. [16] proposes a metric-based meta-learning method to tackle the problem of deep learning models over-fitting during the training process due to insufficient training data. The first stage of this method combines two

different attention modules to extract the features embedded in the inputs. The output of the attention modules is then passed to an ensemble of different models. The outputs of the ensemble mechanism are then used to calculate the distance which is the desired metric for the optimization process. The ensemble section of the process is the one responsible for the reduction of over-fitting. This method was then tested in a one-shot and a multi-shot setting, where it was displayed to have better results compared to other meta-learning methods such as Siamese networks [5], Matching networks [6] and Prototypical networks [15].

Ji et al. [17] contributes toward solving the problem of Zero-Shot Learning (ZSL) by proposing a novel model named Unseen Prototype Learning model. At first, an auto-encoder is used to convert an image to its semantic representation. From this representation, the original image is rebuilt using the decoder part of the auto-encoder. This method is then coupled with triplet loss which aids to lower the classification bias. After the training process, the model was tested on four different ZSL tasks and was shown to have high testing scores compared to various other methods.

2.2 Model-based meta-learning

Model-based meta-learning [2] uses features of the model architecture to improve the learning flexibility with a few training steps (e.g., MetaNet, MANN). This is often done using recurrent neural networks with internal or external memory. These mechanisms do not modify or deal with the probability distribution of the model outputs. Rather, these methods are mostly developed to boost the speed at which the models reach the optimum parameters. This is achieved through careful design of the model architecture or through the assistance of another neural network.

Santoro et al. [7] approached this concept of accelerated learning and proposed the architecture named 'Memory Augmented Neural Network' (MANN). MANNs are similar to RNNs or LSTM models in the sense that they also utilize the conventions of memory. For Meta-learning however, MANN performs the operations related to storing in memory at a much quicker pace than RNN models. The data flow for this network is designed in such a way that target labels for the previous step are passed alongside the input for the current step and then the label for the current step is made available to the model in the next step. This process is repeated throughout the entire training. This causes the model to retain all the features in memory to be accessed later.

Munkhdalai and Yu [18] tackled this prospect using their proposed MetaNet. Traditionally, model parameters are updated using a gradient-based optimizer. This, however, is a relatively slow process. MetaNet [18] demonstrates that weights of a network can also be derived as an output of another neural network. Weights obtained through this process is referred to as 'fast weights'. During the inference stage, the model uses a concatenation of the 'slow' and 'fast' weights to perform the testing.

Li et al. [19] demonstrates a method that attempts to merge various benefits of generative adversarial networks with the aim of solving the classification bias problem in Zero-Shot Learning. First, the source image and the target images are converted to their semantic representation. From these representation, four types of features are created which are source features, target features, fake features, and cycle features. These features are then assessed, and the training loss is generated. The cycle features assist to retain the essential features of the data. This method was displayed to have highly effective performance on both traditional Zero-Shot Learning and generalized Zero-Shot Learning tasks.

2.3 Optimization-based meta-learning

Optimization-based approach [20–22] optimizes the model parameters explicitly for fast learning, so that the model can be good at learning with a few examples (e.g., LSTM Meta-learner, MAML). It focuses on the backward phase of the model training, where the gradients are calculated through differentiation. Traditional optimization methods were not developed with the aim of few-shot learning or to account for training time. Optimization-based methods of meta-learning attempts to solve this problem while maintaining a relatively effective model performance.

Ravi and Larochelle [23] presented the way they tackled the concept of innovating the optimization procedure, through 'meta-learners'. Meta-learners have a high resemblance with LSTM networks, in the sense that the operation for obtaining the new parameters is similar to the forget gate of the LSTM module. Meta-learners have the benefit of remembering previous gradients and thus can observe the trend between them. This enables the methodology to have a more efficient gradient calculation. Furthermore, this method also has the ability of determining how much of previous gradients to take into consideration and how much of it can be deleted.

Similar to MetaNet [18], Finn et al. [8] also demonstrated the use of 'fast weights' to perform the training step. Unlike MetaNet, this method utilizes the gradient descent optimizer to determine the fast weights. This method was named as Model Agnostic Meta-Learning (MAML) [8]. Through this method, optimum parameters can be obtained that used to perform more than one task. Training process with MAML is incorporated with two loops, where ideal parameters are obtained for a specific task in the inner loop and in the outer loop the parameters

obtained can be equated to perform well on all the tasks. One of the disadvantages of MAML is that it computationally heavy, as the gradients obtained from it are of a higher order, due to the back-propagation step must reverse through multiple inner steps. Nichol et al. [24] provides a solution to this problem, through their implemented method known as 'REPTILE'. REPTILE was created with the aim of achieving reducing the amount of total training steps required to achieve the maximum performance. The training process of REPTILE is very similar to MAML, apart from the operation to calculate the gradients and to change the model weights.

3 One-step MAML using SDAGD

One of popular techniques used in optimization-based meta-learning is known as MAML [8]. MAML [8] attempts to reach faster convergence toward the optimum parameters by using a two-loop training strategy. However, MAML [8] faces the problem of instability in presence of large model parameter size. To tackle this issue, this study proposes a novel meta-learning strategy named one-step MAML. One-step MAML is developed with the aim of achieving effective accuracy scores in presence of limited data. It is a variation of MAML [8] that conducts training using two training loops namely the inner and outer loop. The inner loop of one-step MAML is run once per task, and the model parameters are updated based on the support set of that task. The support set consists of 'n' different types of data and 'k' samples of each type of data. After the forward pass with the support set, the model is run again with the query set, and the loss obtained on the query set is recorded. In the outer loop, the query losses for all the tasks are added together, and the resultant loss is used to update the model parameter.

Figure 1 demonstrates the training process of the proposed one-step MAML method. The training process contains two loops, the inner and outer loop. The inner loop runs once for every different task. For example, if we set the number of tasks to five, the inner loop will be run 5 times for every outer loop. For every task, the model is run with the support set from that task. The support loss is calculated which is used to update the model. The target loss for that task is calculated and added to the summation of all the task losses. In the outer loop, that summation of losses is used to calculate the gradient and update the model.

In the inner loop of the process, the parameter updates can be defined by the equation:

$$\theta = \theta - \alpha \nabla_{\theta} \mathcal{L}_{S_b}(f_{\theta_{b-1}}), \tag{1}$$

where α refers to the learning rate of the model, b refers to the task number, S_b refers to the support set for that task, $f_{\theta_{b-1}}$ refers to the model equipped with initial weights for task b, and $\mathcal{L}_{S_b}$ is the loss obtained for support set b. In the outer loop of the process, parameter update is performed with the accumulation of all the query losses which is represented by,

$$\theta = \theta - \beta \nabla_{\theta} \sum_{b=1}^{B} \mathcal{L}_{T_b}(f_{\theta_{b-1}}) \tag{2}$$

where β represents the learning rate used in the outer loop, B refers to the total task number, T_b is the target set or the query set for that task.

Algorithm 1 shows the basic flow of the one-step MAML procedure. At first, the BERT Tiny model is initialized with its pre-trained weights. Then, a number of batches of tasks is sampled from the entire dataset. For each task, the gradient is calculated based on the loss for the support set of that task and then the model weights are updated. After this update, process is done for all the tasks, the model is updated again based on the loss derived from the summation of all the query set losses.

One-step MAML is performed in an iterative process to update the weight of neural network using inner and outer loops. It neither changes the model structure nor does it interact with the format of the model inputs. Due to this, it makes one-step MAML suitable to be used in fields such as natural language processing, image classification and other machine learning tasks. One-step MAML accounts for the stability issues present in MAML by using a summation of query set losses rather one query set loss. This stops the fluctuation of the gradients due to a single loss value. Summation of the query set losses, in the outer loop, is also implemented in MAML++ [9]. MAML++ [9] multiplies the query loss from each inner loop with a vector to assign its importance. One-step MAML does not utilize multiple inner loops, and thus, such vectors are not utilized when calculating the summation. Furthermore, since one-step MAML uses only one inner step per task, less parameters are stored in memory during the training process as compared to MAML and MAML++ that uses multiple inner loops per task.

Algorithm 1: One-step MAML

Require: $p(\tau)$: distribution over tasks
Require: α, β: step size hyper-parameters
Initialize the pretrained BERT Tiny weights;
while *not done* **do**
  Sample batch of tasks $\tau_i \sim p(\tau)$;
  for *all* τ_i **do**
    Evaluate $\nabla_\theta \mathcal{L}_{S_{\tau_i}}(f_{\theta_b})$;
    Update the meta weight using one step, $\theta = \theta - \alpha \nabla_\theta \mathcal{L}_{S_{\tau_i}}(f_{\theta_{\tau_{i-1}}})$;
  end
  Update the meta weight, $\theta = \theta - \beta \nabla_\theta \sum_{\tau_i \sim p(\tau)} \mathcal{L}_{T_{\tau_i}}(f_{\theta_{\tau_{i-1}}})$;
end

Traditional optimizing methods [25] face the issue of having slow convergence rates as well as failing to arrive at optimal minima. One optimizer that tackles this problem is known as Stochastic Diagonal Approximate Gradient Descent (SDAGD) [26]. SDAGD differs from other optimizer such that it uses back-propagation in two-phases. The first phase of SDAGD [26] attempts to arrive at the global minima at a relatively fast rate. It tries to achieve this by identifying the local search regions and formulating the largest step required for each of those regions. In phase-2, SDAGD [26] utilizes the Newton method to arrive at the desired value at a quick pace. To reduce the convergence time, SDAGD considers first order differentials. After the Hessian is calculated, the diagonals are disregarded in accordance to the weights. These steps also help prevent over-fitting during the training process. The resultant weight update for the next training step results to [26]:

$$\theta_{k+1} = \theta_k + [\mu_k + \bar{H}(\theta_k)]^{-1} J(\theta_k), \tag{3}$$

where μ_k denotes the step length relative to search regions. It can be expressed as $\mu_k \approx \|g(w_k)\|/R_k$. In this equation, $g(w_k)$ refers to the loss derivative, and thus, $\|g(w_k)\|$ denotes the value after the derivative was normalized. R_k refers to the search area radius at step k.

The use of SDAGD optimizer within one-step MAML could improve the gradient calculation using the adaptive step-length computation. It is proved in [26] to be outperformed the conventional weight updating rules such as SGD and ADAM. Comparison of the parameter update step in both methods, it can be seen that the gradient of the loss equates to $\alpha \nabla_\theta \mathcal{L}_{S_b}(f_{\theta_{b-1}}) = [\mu_k + \bar{H}(\theta_k)]^{-1} J(\theta_k)$. as shown in Algorithm 1.

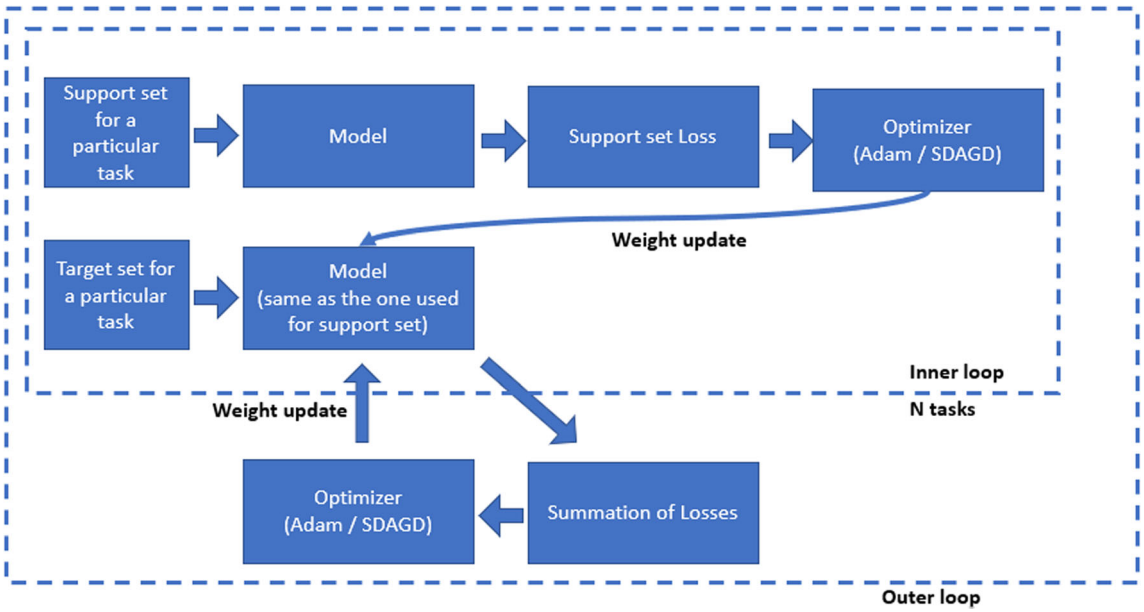

Fig. 1 One-step MAML training process

Table 1 Evaluation losses observed on several benchmark NLP datasets using the fine-tuning algorithms

Training method	Optimizer	Datasets						
		MNLI-M	MNLI-MM	RTE	QQP	STS-2	QNLI	Average
Training w/o MAML [12]	Adam	1.0300	1.0250	0.6669	0.5060	0.5027	0.5272	0.7096
MAML++ [9]		1.1138	1.1028	0.6909	0.6547	0.6910	0.6947	0.8247
One-step MAML		0.9727	0.9835	0.6634	0.5705	0.5150	0.5278	0.7054
One-step MAML	SDAGD [26]	0.9065	0.9154	0.7392	0.6785	0.4905	0.4736	0.7006

Table 2 Evaluation accuracy on several benchmark NLP datasets using the fine-tuning algorithms

Training method	Optimizer	Datasets						
		MNLI-M	MNLI-MM	RTE	QQP	STS-2	QNLI	Average
Training w/o MAML [12]	Adam	0.4764	0.4672	0.59928	0.7294	0.7694	0.7525	0.63228
MAML++ [9]		0.3184	0.3056	0.5270	0.6318	0.54243	0.5015	0.4711
One-step MAML		0.5743	0.5547	0.6131	0.7443	0.7809	0.7539	0.6702
One-step MAML	SDAGD [26]	0.6018	0.5846	0.6209	0.7628	0.7878	0.7825	0.6901

4 Experimental setup

The model training was conducted using five datasets from the Glue Benchmark [27], these datasets are MNLI [28], RTE [29], QQP [30], STS-2 [31] and QNLI [32] datasets. For training, a limit of maximum 10000 sentence pairs were selected. During testing, there was no limit applied, and the entire test dataset were used. For datasets, that have less than 10000 samples, the entire set was used for training. The Glue Benchmark [27] is designed to have sentence pair tasks that represents various scenarios when it comes to understanding the English language. These datasets cover a large range of complexity when it comes to the English language.

Data processing was used to convert the datasets such that it fits a few-shot learning scheme. For each outer loop, a certain number of tasks is selected. For the experiments discussed in these paper, number of tasks was set to five. For each task, the number of classes n and the number of samples per class k is selected. Therefore, each batch contains $n \times k$ sentence pairs. For example, the MNLI dataset [28] has three classes, and the value of k was set to five. Thus, each batch would have 15 different input samples. For all the experiments discussed in this paper, the value of k is assigned to 5.

The measurement metrics used for these datasets is accuracy, which is calculated as follows, [27],

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \tag{4}$$

where TP and TN refer to true positive and true negatives, and TN and FN refer to true negative and false negative. Essentially, the numerator refers to the amount of samples correctly classified, and the denominator refers to the total value.

The loss metric used for datasets training is cross entropy loss. It is calculated using the following equation [33]:

$$\text{loss}(x, \text{class}) = -\log\left(\frac{\exp(x[\text{class}])}{\sum_j \exp(x[j])}\right), \tag{5}$$

where x refers to the activation function score, class refers to the target class. The loss function can also be expressed as $\text{loss}(x, \text{class}) = -x[\text{class}] + \log(\sum_j \exp(x[j]))$.

5 Result and discussion

In this experiment, four different training methods were implemented, and their performances were compared, i.e., training without meta-learning, training with MAML++, One-step MAML with ADAM optimizer and one-step MAML with SDAGD optimizer. The model used to execute the training procedures is the BERT Tiny Model. These methods were trained and tested on five benchmark GLUE datasets. Table 1 demonstrates the lowest validation loss values obtained after the experiments were conducted on the different datasets. For some datasets, only a portion of the training data is used, for example, only 10000 samples were used from the MNLI dataset. This was done

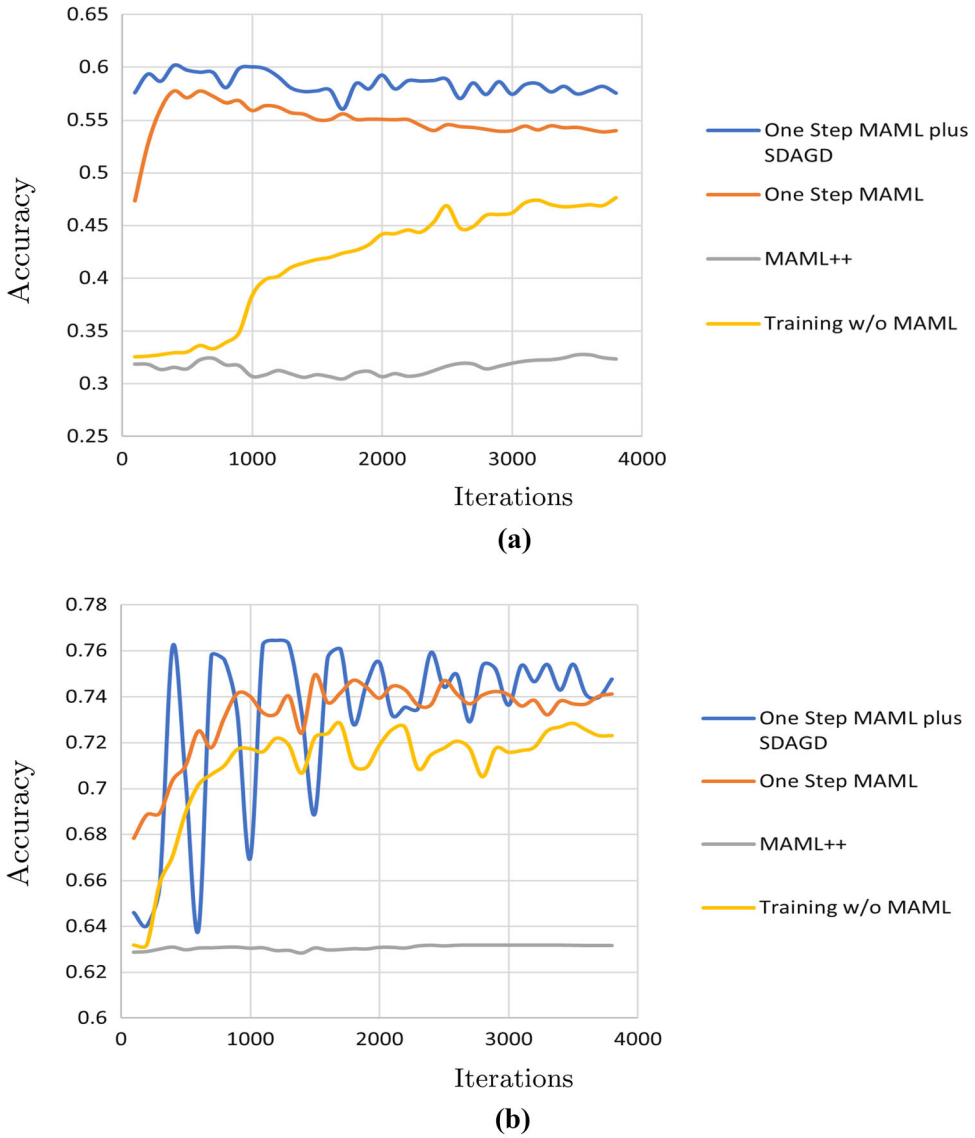

Fig. 2 Evaluation accuracy curves **a** MNLI dataset and **b** QQP dataset

to create a scenario where sufficient data are not present and to observe the model's performance on limited data. From this table alone, it is observed that the lowest losses obtained is for most of the case of using one-step MAML optimized using SDAGD. The only exceptions being for the QQP dataset, where only the one-step MAML has lower losses.

Comparing the validation accuracy obtained for the same experiments in Table 2, it is seen that the best scores are also for the one-step MAML method combined with a SDAGD optimizer. Figure 2 displays the highest accuracy values obtained for the experiments conducted. We observe an increment in scores for all the datasets, with the highest increase being for the MNLI dataset. For the MNLI dataset, we observe an increase of almost 13% for the matched and 12% for the mismatched category. One-step MAML seems to demonstrate an increment in accuracy, and addition of the SDAGD optimizer boosts the accuracy by another 2% to 3%.

Two evaluation curves using MNLI and QQP dataset are shown in Fig. 2 to demonstrate the change of accuracy values obtained from the testing datasets after every inner and outer loop of training process is completed at each step. The weights of neural network are trained using meta-learning approaches to obtain the best optimum parameters along with the training iterations. As demonstrated in Fig. 2, one iteration after MAML training refers to the evaluation result after both completion of inner and outer loops training. On the other hand, the performance of the network is also compared with the training without fine-tuning using MAML algorithms. In the case of training where meta-learning methods are not applied, one complete iteration refers to the execution of the algorithm on one batch of training set.

Figure 2a displays the evaluation accuracy curves obtained for the MNLI dataset [28]. From the four experiments conducted, it seems that the combination of one-step MAML and the SDAGD optimizer yields the highest accuracy. It can be observed from the graph that the accuracy for this case is higher than the others at the beginning of the plot. This means that for the case of the MNLI dataset, this combination is able to arrive at the optimal solution using several steps. It can be also noted that one-step MAML optimized using SDAGD reaches its highest accuracy at around 500 steps, which is relatively much faster than the other MAML algorithms.

Another example of the evaluation accuracy curve is displayed in Fig. 2b, which is for the QQP dataset [30]. Similar to the case of MNLI, one-step MAML with SDAGD optimizer can achieve the best performance for the QQP dataset. Unlike the MNLI dataset, this method does not start with the highest accuracy from the very first few steps. It is observed that the curve slope of the one-step MAML optimized using SDAGD starts increasing until around 1.5k steps and then keeps oscillating around a constant value. This is an indication that using this method, the model is able to grasp the features of the MNLI dataset quicker than the QQP dataset. Again, as observed for the MNLI dataset, this method converges faster than the other methods displayed in the graph.

Qualitative analysis displays the performance of the model in action. For example, in a sample input from the MNLI dataset [28], text A is"The new rights are nice enough", and text B is"Everyone really likes the newest benefits". The label for this sentence pair is neutral, as it can be clearly seen by viewing the two sentences. The model trained with one-step MAML and SDAGD correctly classifies this as neutral and obtains a loss of 9.205×10^{-2}. In another sample input from the same dataset, text a is "You and your friends are not welcome here, said Severn", and text b is"Severn said the people were not welcome there". The model trained with the same method as the previous example, correctly classifies this and gets a loss of 6.707×10^{-2}.

In the meta-learning methods, the differences between MAML and one-step MAML might give us a clearer picture for the reason behind the better performance of one-step MAML. MAML [8] and MAML++ [9] holds in memory all the gradients calculated during the inner loop which is later unrolled during the outer loop. This causes a problem of exploding and vanishing gradients when dealing with models with large number parameters. MAML++ tries to accommodate this by taking a summation of query losses which is weighted by an importance vector, then calculating the gradient based on that sum during the outer loop. However, MAML++ continues to face the same issue when it is used to train very large models. One-step MAML adopts the summation process of the query losses discounting the importance vector, since this method only takes one gradient step per inner loop and the importance vector's purpose is to assign value to the multiple inner steps. On top of this, one-step MAML updates the meta weights during the inner loop itself, to avoid the problem of exploding and vanishing gradients during the backpropagation process of the outer loop. This variations in combination could attribute to the more effective performance of the one-step MAML.

6 Conclusion

The task in natural language processing differs a lot in terms of complexities due to the high magnitude of variations present in words and sentences. Hence, maintaining a high accuracy score becomes crucial for language learning generalization. In this paper, a new meta-learning strategy named as one-step MAML is proposed to perform gradient updates using two loops mechanism. During the inner loop, weights are updated once per task and during the outer loop, weights are updated based on summation of all the target set losses. This study demonstrates the performance of four different methods, i.e., training without meta-learning, training with MAML++, One-step MAML with ADAM optimizer and one-step MAML with SDAGD optimizer. The performance for these methods is recorded and compared for five GLUE datasets. It is observed that one-step MAML performs best when combined with the SDAGD optimizer. This combination obtains the highest accuracy and lowest loss for all the datasets. Most significant jump in accuracy is observed for the MNLI dataset, One-step MAML with SDAGD optimizer achieves an accuracy of 60.18% which is around 13% higher than simply training without MAML. It was also demonstrated that using this process the model can converge much faster. From the accuracy curve, it was displayed that One-step MAML with SDAGD optimizer reaches peak accuracy with only around 700 steps. Qualitative analysis is also done to view the sentence pairs that were correctly classified.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Vilalta R, Drissi Y (2002) A perspective view and survey of meta-learning. Artif Intell Rev 18(2):77–95
2. Vanschoren J (2018) Meta-learning: a survey, arXiv preprint arXiv:1810.03548
3. Hospedales T, Antoniou A, Micaelli P, Storkey A (2020) Meta-learning in neural networks: a survey, arXiv preprint arXiv:2004.05439
4. Thrun S, Pratt L (1998) Learning to learn: introduction and overview. In: Thrun S (ed) Learning to learn. Springer, Boston, MA, pp 3–17. https://doi.org/10.1007/978-1-4615-5529-2_1
5. Koch G, Zemel R, Salakhutdinov R (2015) Siamese neural networks for one-shot image recognition. In: ICML deep learning workshop, vol 2. Lille
6. Vinyals O, Blundell C, Lillicrap T, Wierstra D et al (2016) Matching networks for one shot learning. In: Lee D, Sugiyama M, Luxburg U, Guyon I, Garnett R (eds) Advances in neural information processing systems, vol 29, pp 3630–3638. https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf
7. Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T (2016) Meta-learning with memory-augmented neural networks. In: International conference on machine learning, pp 1842–1850
8. Finn C, Abbeel P, Levine S (2017) Model-agnostic meta-learning for fast adaptation of deep networks. In: International conference on machine learning, pp 1126–1135. PMLR
9. Antoniou A, Edwards H, Storkey A (2018) How to train your MAML, arXiv preprint arXiv:1810.09502
10. Marcus G (2018) Deep learning: a critical appraisal, arXiv preprint arXiv:1801.00631
11. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
12. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805
13. Altae-Tran H, Ramsundar B, Pappu AS, Pande V (2017) Low data drug discovery with one-shot learning. ACS Central Sci 3(4):283–293
14. Li X, Sun Z, Xue J-H, Ma Z (2021) A concise review of recent few-shot meta-learning methods. Neurocomputing 456:463–468
15. Snell J, Swersky K, Zemel R (2017) Prototypical networks for few-shot learning. In: Advances in neural information processing systems, pp 4077–4087
16. Guo N, Di K, Liu H, Wang Y, Qiao J (2021) A metric-based meta-learning approach combined attention mechanism and ensemble learning for few-shot learning. Displays 70:102065
17. Ji Z, Cui B, Yu Y, Pang Y, Zhang Z (2021) Zero-shot classification with unseen prototype learning. Neural Comput Appl 33:1–11. https://doi.org/10.1007/s00521-021-05746-9
18. Munkhdalai T, Yu H (2017) Meta networks. In: International conference on machine learning, pp 2554–2563. PMLR
19. Li X, Zhang D, Ye M, Li X, Dou Q, Lv Q (2021) Bidirectional generative transductive zero-shot learning. Neural Comput Appl 33(10):5313–5326
20. Finn C, Rajeswaran A, Kakade S, Levine S (2019) Online meta-learning. In: International conference on machine learning, pp 1920–1930. PMLR
21. Wang Y, Yao Q, Kwok JT, Ni LM (2020) Generalizing from a few examples: a survey on few-shot learning. ACM Comput Surv (CSUR) 53(3):1–34
22. Baxter J (1998) Theoretical models of learning to learn. In: Learning to learn, pp 71–94. Springer
23. Ravi S, Larochelle H (2017) Optimization as a model for few-shot learning. In: *ICLR*
24. Nichol A, Achiam J, Schulman J (2018) On first-order meta-learning algorithms, arXiv preprint arXiv:1803.02999
25. Tan HH, Lim KH, Harno HG (2017) Stochastic diagonal approximate greatest descent in convolutional neural networks. In: 2017 IEEE international conference on signal and image processing applications (ICSIPA), pp 451–454
26. Tan HH, Lim KH (2020) Two-phase switching optimization strategy in deep neural networks. In: IEEE transactions on neural networks and learning systems, pp 1–10
27. Wang A, Singh A, Michael J, Hill F, Levy O, Bowman SR (2018) Glue: A multi-task benchmark and analysis platform for natural language understanding, arXiv preprint arXiv:1804.07461
28. Williams A, Nangia N, Bowman S (2018) A broad-coverage challenge corpus for sentence understanding through inference, In: Proceedings of the 2018 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (Long Papers), pp 1112–1122. Association for Computational Linguistics
29. Dagan I, Glickman O, Magnini B (2005) The pascal recognising textual entailment challenge. In: Machine learning challenges workshop, pp 177–190. Springer
30. Sharma L, Graesser L, Nangia N, Evci U (2019) Natural language understanding with the quora question pairs dataset, arXiv preprint arXiv:1907.01041
31. Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng AY, Potts C (2013) Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 conference on empirical methods in natural language processing, pp 1631–1642
32. Rajpurkar P, Zhang J, Lopyrev K, Liang P (2016) Squad: 100,000+ questions for machine comprehension of text, arXiv preprint arXiv:1606.05250
33. Mannor S, Peleg D, Rubinstein R (2005) The cross entropy method for classification. In: Proceedings of the 22nd international conference on machine learning, ICML '05, (New York, NY, USA), pp 561–568. Association for Computing Machinery