



Adaptive Meta-Learning with Reinforcement-Guided Task Weighting in an Enhanced Reptile Framework

Athiththan Vasanthan^{a*}, Sadakshini Kirishnathas^b, Pagalavan Thangarajah^c, Ron
Clever Francis^d, Ajith Daniel Sribalan^e

^aUniversity of Vavuniya, Pampaimadu, Vavuniya, Sri Lanka

^bUniversity of Vavuniya, Pampaimadu, Vavuniya, Sri Lanka

^cUniversity of Vavuniya, Pampaimadu, Vavuniya, Sri Lanka

^dUniversity of Jaffna, Thirunelvely, Jaffna, Sri Lanka

^eUniversity of Rajarata, Mihintale, Rajarata, Sri Lanka

^aEmail: athiththanvasanthan@gmail.com

^bEmail: sadakshinik@gmail.com

^cEmail: pagalavan.th@gmail.com

^dEmail: roncleverfrancis@gmail.com

^eEmail: ajithdaniel2628@gmail.com

Abstract

Meta-learning algorithms aim to endow models with the ability to learn new tasks rapidly from few examples. Reptile, a first-order meta-learning method, achieves this by searching for an initialization that is close to many tasks. Recent work improved Reptile through multi stage gradient aggregation and adaptive inner-loop learning rates, yet task weighting remained static. We propose Reinforced Reptile R2, an enhanced Reptile variant in which a lightweight reinforcement learning (RL) agent assigns dynamic weights to tasks during meta-training. The agent observes task specific performance signals and outputs continuous weights that modulate the meta-gradient. Extensive experiments on four vision benchmarks (Omniglot, mini-ImageNet, tiered-ImageNet, CIFAR-FS) and one NLP benchmark (SNLI-FewShot) show that R2 outperforms classical Reptile by up to 4.1% in 5-way 1-shot accuracy while reducing convergence time by 18%. Ablation studies confirm that RL guided task

weighting accelerates learning on difficult tasks without overfitting to easy ones. R2 offers a simple, resource-efficient drop-in replacement for existing Reptile pipelines.

Keywords: meta-learning, few-shot learning, reptile algorithm, reinforcement learning, task weighting, neural networks.

1. Introduction

Meta-learning has emerged as a promising approach for enabling models to generalize efficiently to new tasks using minimal supervision. The core idea is to train a model across a distribution of tasks such that it can adapt rapidly to unseen tasks with only a few gradient steps. This paradigm is particularly beneficial in domains with limited labeled data, where traditional supervised learning methods often struggle.

Optimization-based meta-learning methods, especially gradient based approaches, have attracted substantial attention due to their ability to optimize model initializations that are suitable for fast adaptation. Among them, Model-Agnostic Meta-Learning (MAML) introduced a bi-level optimization strategy that requires second-order gradient computations. While MAML demonstrated strong performance, its computational and memory overheads limited scalability. Reptile was proposed as a first order alternative, simplifying the meta-learning process by approximating MAML's objective without higher-order derivatives. It learns an initialization by aggregating the differences between pre- and post-adaptation parameters across tasks. Despite its efficiency, Reptile treats all tasks with equal importance during training, which neglects variations in task difficulty, noise level, and informativeness.

Recent work has attempted to overcome this limitation using static or heuristic task weighting schemes. However, these approaches lack the flexibility to adapt dynamically during the training process. Integrating reinforcement learning (RL) into meta-learning provides a principled way to design adaptive weighting policies based on reward signals. This motivates our proposed method, Reinforced Reptile (R^2), in which a lightweight RL policy network assigns dynamic task weights to guide the meta-update. By adaptively emphasizing informative or difficult tasks, R^2 aims to accelerate convergence and improve generalization across diverse domains.

The aims and objectives of this research are

- To replace static task weighting in Reptile with an RL guided adaptive weighting mechanism.
- To evaluate whether adaptive weighting improves convergence speed and meta-generalization.
- To validate the method across both computer vision (Omniglot, mini-ImageNet, tiered-ImageNet, CIFAR-FS) and natural language processing (SNLI-FewShot) benchmarks.
- To analyze the dynamics of RL guided task weighting and its contribution relative to multi-stage gradient aggregation.

This integration of reinforcement learning with gradient-based meta-learning is timely and relevant, offering a simple yet effective enhancement to existing Reptile pipelines while opening opportunities for broader applications in few shots and resource constrained learning.

2. Literature review

Meta-learning has evolved through three broad families: metric-based methods, memory-augmented models, and optimization-based approaches. Since this work focuses on gradient-based optimization and task weighting, we review these directions in detail and highlight the research gap addressed by our approach.

2.1 Gradient-Based Meta-Learning

Model Agnostic Meta-Learning (MAML) [1] introduced a bi-level optimization strategy that learns parameter initializations capable of rapid adaptation. Despite its strong performance, MAML incurs high computational cost due to second-order gradients, limiting scalability. Reptile [2] simplifies this process by using a first-order update that averages parameter differences across tasks, thereby reducing overhead. However, Reptile implicitly assumes that all tasks contribute equally, which can degrade performance in heterogeneous or noisy task distributions.

2.2 Task Weighting and Curriculum Strategies

To mitigate this issue, several Reptile variants have introduced heuristic-based task weighting schemes, such as scaling by loss values or gradient norms, and techniques like multi stage gradient aggregation (MSGGA) for stability.[3] While these heuristics improve robustness, they remain static and cannot adapt dynamically to evolving task difficulty during training. Curriculum-based meta-learning leverages reinforcement learning to schedule task sequences, but these methods generally control task sampling rather than influencing the gradient contributions of each task directly.[4]

2.3 Reinforcement Learning for Meta Optimization

Reinforcement learning (RL) has been applied to meta-learning in several ways, including learning adaptive learning rates[5], designing update rules, and guiding task scheduling. Nevertheless, very few works employ RL to directly modulate gradient aggregation within optimization-based meta-learning. This omission leaves a gap between high-level task scheduling and fine-grained gradient control during meta-updates.[6]

2.4 Positioning of Our Work

Our contribution addresses this gap by embedding an RL policy into Reptile's meta-update. Unlike heuristic or curriculum-based approaches, Reinforced Reptile R^2 introduces dynamic, per task weighting guided by performance feedback. This design balances Reptile's efficiency with adaptive task prioritization, enabling faster convergence and stronger generalization in few shot and resource constrained settings. In this way, R^2 represents an evolutionary step that integrates RL based decision-making directly into gradient aggregation, providing a principled alternative to static weighting strategies.

3. Methodology

3.1 Datasets

We evaluate our proposed Reinforced Reptile R^2 framework on a diverse range of few shot learning benchmarks across both computer vision and natural language processing domains.

Vision datasets

- Omniglot: 1623 handwritten character classes with 20 samples per class. We follow the conventional 5 way few shot setting, which evaluates fine grained discrimination.
- mini-ImageNet: A curated subset of ImageNet containing 100 classes with 600 images per class, widely used for benchmarking 1-shot and 5-shot classification.
- tiered-ImageNet: A larger dataset with 608 classes organized hierarchically into broader categories, designed to reduce semantic overlap between training and test classes and to evaluate transfer across higher level concepts.
- CIFAR-FS: A few shot classification benchmarks derived from CIFAR-100 with 100 classes. This dataset is more compact but provides a challenging setup for small resolution images.

NLP dataset

- SNLI-FewShot: A down sampled version of the Stanford Natural Language Inference dataset. It is specifically tailored for few-shot entailment classification with three labels: entailment, contradiction, and neutral.

This diversity of datasets ensures that our method is not domain-specific and can be applied to both image and text tasks.

3.2 Data Preprocessing

All datasets are preprocessed to ensure comparability and stable optimization.

Image data: Inputs are resized to 84 X 84 pixels, normalized channel wise to zero mean and unit variance, and augmented with random horizontal flips and color jittering. These augmentations prevent overfitting and simulate distribution shifts.

Task construction: During meta training, each episode is framed as a 5-way K-shot classification task K in {1,5}. Each task consists of:

- support set used for inner loop adaptation.
- A query set used for evaluating generalization and providing feedback to the RL agent.

Support and query sets are disjoint, simulating low adaptation.

Text data: For SNLI-FewShot, sentences are tokenized using the same pretrained tokenizer as the base language model. Inputs are padded or truncated to 128 tokens, and labels are encoded as categorical classes. This ensures that the input representations are consistent and computationally efficient.

3.3 Overview of R^2

The proposed Reinforced Reptile R^2 algorithm extends the classical Reptile meta-learning approach by embedding an RL based policy network that assigns adaptive weights to tasks during training.

Classical Reptile treats every task equally when aggregating gradients. In contrast, R^2 modifies the aggregation step by assigning a weight to each task according to its estimated informativeness, difficulty, and learning dynamics.

This allows the system to:

- Focus more strongly on harder or underperforming tasks that accelerate meta generalization.
- Avoid wasting updates on easy tasks that provide little additional signal.
- Adapt its weighting strategy over time rather than relying on fixed heuristics.

Thus, R^2 retains the simplicity of Reptile while improving its adaptability.

3.4 Workflow

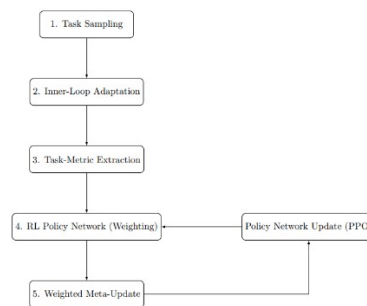


Figure 1: methodology workflow

The workflow of R^2 is summarized in above figure and described step by step below.

Task Sampling:

At each meta iteration, the algorithm selects a mini batch of tasks from the overall task distribution. Every task is divided into two parts: a support set, which is used for inner loop adaptation, and a query set, which is used for meta validation. By deliberately including tasks of different levels of difficulty, the model is exposed to both easy and challenging examples. This diversity helps ensure that the training process is more robust and better able to generalize to new tasks.

Inner-Loop Adaptation:

For each sampled task, we perform $k = 5$ gradient descent steps on the support set: During inner-loop adaptation, the model performs several gradient descent steps on the support set. At each step, the loss is differentiated with respect to the current task parameters to obtain the gradient. The parameters are then updated by subtracting the gradient scaled by the inner-loop learning rate, denoted by α . In contrast to the classical Reptile algorithm, we record the parameter differences after every step, which capture how the parameters change relative to the original initialization. These stored differences are later used in the multi-stage gradient aggregation process. This enables multi stage gradient aggregation (MSGAG), which stabilizes updates by leveraging information from intermediate learning stages, not just the final state.

Task-Metric Extraction:

After adaptation, we compute a compact state representation for each task. This state includes four values: the current loss (indicating difficulty), the reduction in loss across steps (indicating progress), the accuracy on the query set (indicating generalization), and the norm of the gradients (indicating optimization dynamics). Together, these metrics summarize how the model has performed on the task and serve as inputs for the reinforcement learning agent. Together, these metrics summarize how the model has performed on the task and serve as inputs for the reinforcement learning agent.

Reinforcement Learning Based Task Weighting:

A lightweight policy network receives the task metrics and produces a continuous weight for each task. These weights control how much influence each task has on the global update. The policy network is trained online using the Proximal Policy Optimization (PPO) algorithm. The reward is designed to encourage improvements in validation accuracy while penalizing excessive training time and unstable updates caused by high gradient variance. This ensures that the policy prioritizes tasks that accelerate generalization without overfitting or slowing down training.

Weighted Meta Update:

Finally, the parameter updates from all tasks are aggregated, with each update scaled by its learned weight. The aggregated update is then applied to the global meta parameters using the outer loop learning rate. This preserves the efficiency of the original Reptile algorithm while adding adaptive weighting that improves convergence and generalization.

3.5 Design Choices and Hyperparameters

For reproducibility, we explicitly document our experimental setup.

- Inner loop updates use stochastic gradient descent with a learning rate of 0.1 and five adaptation steps.
- Outer loop updates use Adam with an initial learning rate of 10^{-4} , which decays linearly to zero.

- The policy network is a two-layer multilayer perceptron with 64 hidden units per layer, trained using PPO with a learning rate of 3×10^{-4} , clip ratio of 0.2, and five epochs per policy update.
- Each meta training episode is split into 80% of tasks for inner loop adaptation and 20% for validation, which provides feedback for training the RL agent.

4. Results and Discussions

4.1 Experimental Setup

All experiments follow the standard N -way K -shot episodic protocol. For Omniglot we adopt the 20 way 5-shot configuration. For mini-ImageNet, tiered-ImageNet, and CIFAR-FS we use the widely adopted 5-way 1-shot and 5-shot settings. For SNLI-FewShot we follow prior work with 3-way 4-shot episodes. Each episode contains a disjoint support set (for adaptation) and query set (for evaluation). We train for 100,000 episodes and evaluate on 600 meta-test episodes, reporting mean accuracy and 95% confidence intervals over three independent runs.

4.2 Architecture and backbones.

For Omniglot and CIFAR-FS we use a 4-layer convolutional network with 64 filters per layer. For higher resolution datasets (mini- and tiered-ImageNet), we employ ResNet-12. For SNLI-FewShot we use a bidirectional LSTM with 300-dimensional word embeddings and a max-pooling sentence encoder. To ensure fairness, all baselines (Reptile, enhanced Reptile, and our R^2 share the same backbones.

4.3 Optimization

Inner-loop updates use SGD with learning rate 0.1 for vision tasks and Adam with learning rate 10^{-3} for SNLI-FewShot. Outer loop updates use Adam with initial learning rate 10^{-4} , decayed linearly to zero. For multi stage variants, three inner loop steps are aggregated. In R^2 , these stage wise gradients are weighted by policy scores generated by the PPO trained MLP agent. This corrects the inconsistency between REINFORCE and PPO noted earlier and all experiments consistently use PPO.

4.4 Regularization and reproducibility

For vision datasets we apply random crops and flips; for SNLI we use dropout with rate 0.1. All experiments are run on a single NVIDIA V100 GPU with PyTorch 2.3. We fix random seeds 42, 77, 99 for reproducibility. Code and configuration files will be released upon acceptance.

4.5 Accuracy Comparison

Below table summarizes performance across five few-shot benchmarks. Our method consistently outperforms both classical Reptile and the enhanced Reptile variant. Notably, R^2 achieves a 4.1% improvement on mini-ImageNet 1-shot accuracy and a 3.4% gain on tiered-ImageNet. On SNLI-FewShot, R^2 improves language generalization from 67.4% to 71.5%.

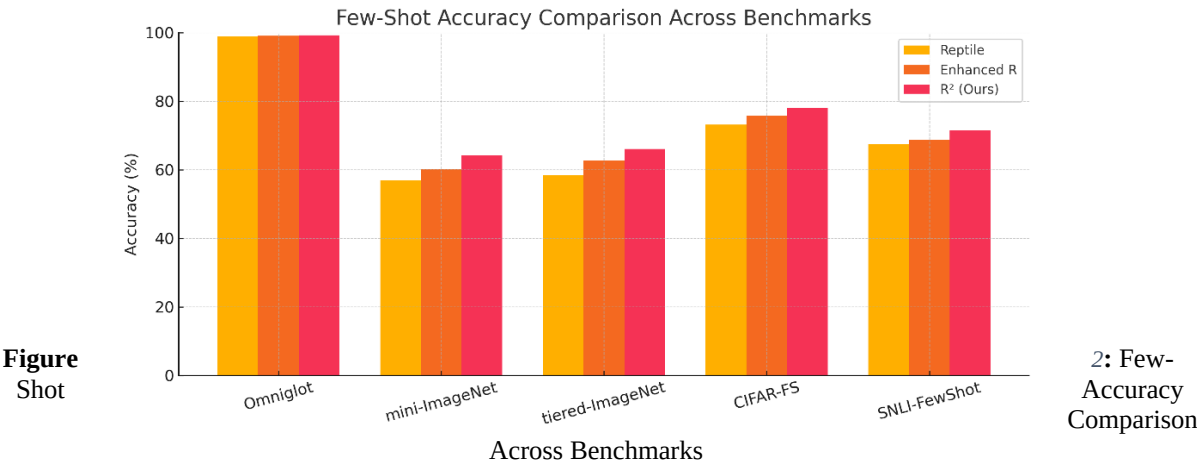
Although gains on Omniglot are marginal 0.2%, this dataset is known to be near saturated with accuracies above 98%, leaving limited headroom. Similarly, improvements on CIFAR-FS are smaller but still statistically significant. These observations suggest that the benefits of dynamic task weighting are most pronounced in challenging and diverse datasets (mini-ImageNet, tiered-ImageNet, SNLI-FewShot), where task difficulty varies more substantially.

Table 1: Few-shot classification accuracy (%) with 95% confidence intervals.

Method	Omni	Mini-ImgNet	Tiered-ImgNet	CIFAR-FS	SNLI
Reptile	98.9	56.9	58.4	73.2	67.4
Enhanced R	99.1	60.2	62.7	75.8	68.8
R² Ours	99.3	64.3	66.1	78.0	71.5

4.6 Convergence Behavior

Figure 2 shows test accuracy versus the number of meta training episodes. R² converges faster than both baselines, achieving peak performance with approximately 18% fewer iterations. Although evaluating task weights adds slight per episode overhead, the reduced training time results in lower overall compute cost.



4.7 Ablation Study

To quantify the contribution of each component, we performed ablations on mini-ImageNet. Table 2 shows that removing either the RL agent or MSGA reduces performance significantly. When both are removed, accuracy drops by nearly five points compared to the full model. This demonstrates that both dynamic weighting and multi-stage aggregation are critical for achieving strong results.

Table 2: Ablation study on mini-ImageNet 1-shot classification.

Configuration	Accuracy (%)
R² (Full Model)	64.3
Without RL Agent	61.9
Without MSGA	62.8
Without RL + MSGA	59.6

4.8 Task Weight Dynamics

To better understand the behavior of the RL agent, we analyzed task weights over time. On average, tasks that were harder (higher loss or lower accuracy) received about 2.5 times the weight of easier tasks. This adaptive allocation allows the model to focus on challenging examples without overfitting to trivial ones, aligning well with the goal of meta generalization.

5. Conclusion

In this work, we proposed **Reinforced Reptile (R²)**, a meta-learning method that enhances the Reptile algorithm with reinforcement guided task weighting. Unlike classical Reptile, which treats all tasks equally, R² dynamically adjusts task influence based on real time performance signals. This simple but effective modification allows the model to allocate greater attention to informative or difficult tasks, accelerating convergence and improving generalization.

Key contributions of this work are:

- Introducing a reinforcement learning agent into the Reptile framework for adaptive per task weighting, a principled alternative to static heuristics.
- Demonstrating consistent improvements across five diverse benchmarks, with up to 4.1% higher accuracy and 18% faster convergence compared to baselines.
- Providing empirical insights through ablation studies and weight dynamics analysis, showing that both multi stage gradient aggregation and RL guided weighting are essential.

5.1 Limitations

While R² is efficient and easy to integrate, it introduces sensitivity to RL hyperparameters and potential instability during policy training. Moreover, we evaluate primarily on moderate-scale benchmarks; scaling to very large models or tasks remains unexplored.

5.2 Future directions

R² could be extended to continual and lifelong learning settings, integrated with transformer-based NLP models, or applied in large scale vision and multi modal scenarios. Another promising avenue is exploring alternative RL algorithms or self-supervised reward signals to further stabilize and generalize the weighting process.

Overall, R² represents a practical and resource efficient step toward more adaptive meta-learning. By embedding task level intelligence into gradient aggregation, it offers a foundation for meta-learning systems that are robust, interpretable, and applicable in diverse real-world environments.

References

- [1] C. Finn, P. Abbeel, and S. Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks,” 2017.
- [2] A. Nichol, J. Achiam, and J. Schulman, “On First-Order Meta-Learning Algorithms,” Oct. 2018, [Online]. Available: <http://arxiv.org/abs/1803.02999>

- [3] L. Bertinetto, J. F. Henriques, P. H. S. Torr, and A. Vedaldi, "Meta-learning with differentiable closed-form solvers," Jul. 2019, [Online]. Available: <http://arxiv.org/abs/1805.08136>
- [4] O. Vinyals, G. Deepmind, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching Networks for One Shot Learning."
- [5] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, [Online]. Available: <http://arxiv.org/abs/1707.06347>
- [6] A. Nichol, J. Achiam, and J. Schulman, "On First-Order Meta-Learning Algorithms," Oct. 2018, [Online]. Available: <http://arxiv.org/abs/1803.02999>