LOOKAHEADBIO: BILEVEL OPTIMIZATION WITH LOOKAHEAD MOVING AVERAGES FOR LLM DATA REWEIGHTING

Ruijie Xie [*2], **Chaoyue Zhao**[*2], **Oz T. Jang** [†1,2] *
[1] Onemee Laboratory [2] Tsinghua University
oztj@onemee.ai

ABSTRACT

Bilevel optimization has shown promising results for data reweighting in large language model (LLM) training, but suffers from high variance and sensitivity to hyperparameters. Inspired by the stability properties of the Lookahead optimizer, we present **LookaheadBio**, a novel bilevel optimization framework that integrates Lookahead's moving average mechanism directly into the bilevel structure. Our method maintains slow and fast weights for both model parameters and data weights, achieving dual-level variance reduction while preserving theoretical convergence guarantees. LookaheadBio combines the theoretical optimality of bilevel optimization with the practical stability of moving averages, resulting in faster convergence and improved performance. Experiments demonstrate that LookaheadBio effectively learns meaningful data weights under diverse settings. In instruction-following tasks, the method automatically identifies and significantly up-weights high-quality data sources, even when they constitute only a small fraction of the training set, while still preserving data diversity. These results highlight the method's ability to perform stable, interpretable, and robust data reweighting, consistent with its theoretical variance reduction properties. Theoretically, we prove that LookaheadBio maintains $\mathcal{O}(\epsilon^{-7})$ sample complexity for finding ϵ-stationary points while enjoying improved stability constants through its moving average mechanism. All code is open-sourced at https://github.com/Onemee-Laboratory/LookaheadBiO.

1 INTRODUCTION

The success of large language models (LLMs) critically depends on the quality and balance of training data. Bilevel optimization has emerged as a principled approach to data reweighting, optimizing data source weights to minimize validation loss. Recent work such as ScaleBiO has successfully scaled bilevel optimization to 30B-parameter LLMs, demonstrating significant gains over heuristic methods. However, bilevel optimization algorithms inherently suffer from high variance due to their nested structure and hypergradient approximations.

Concurrently, the Lookahead optimizer has demonstrated remarkable success in stabilizing optimization through a simple yet effective mechanism: maintaining both "fast weights" that explore the loss landscape aggressively and "slow weights" that follow a smoothed trajectory via moving averages. This approach reduces variance, improves generalization, and decreases sensitivity to hyperparameters.

In this paper, we propose **LookaheadBio**, which bridges these two lines of work by integrating Lookahead's moving average mechanism into the bilevel optimization framework. Our key insight is that the variance reduction provided by Lookahead's slow weights naturally complements the variance challenges in bilevel optimization, particularly when dealing with noisy hypergradient estimates.

Our contributions are:

- **Algorithm**: We introduce LookaheadBio, the first bilevel optimization algorithm that incorporates Lookahead's slow-fast weight mechanism for both model parameters and data weights, achieving comprehensive variance reduction.
- **Theory**: We provide convergence analysis showing that LookaheadBio maintains the optimal $\mathcal{O}(\epsilon^{-7})$ sample complexity of first-order bilevel methods while improving stability through its moving average design.
- **Experiments**: The small-scale experiments provide clear evidence of the effectiveness of LookaheadBio in data reweighting scenarios.In the instruction following setting, LookaheadBio successfully identifies and emphasizes high-quality data sources. Despite Alpaca-GPT4 constituting only 10% of the training data, the algorithm assigns it a dominant weight (approximately 90%) during training. This demonstrates two key properties of LookaheadBio: (1) its ability to perform reliable and interpretable data reweighting, and (2) its robustness under distribution shift. The observed behavior is consistent with the theoretical intuition that the Lookahead mechanism stabilizes the bilevel optimization process and reduces the impact of noisy gradient estimates.

2 Related Work

2.1 Bilevel Optimization for Data Reweighting

Bilevel optimization for data reweighting has evolved from implicit differentiation methods Domke (2012); Lorraine et al. (2020) to first-order methods Kwon et al. (2023). ScaleBiO Pan et al. (2025) demonstrated practical scalability to billion-parameter LLMs. However, existing methods remain sensitive to gradient noise and require careful hyperparameter tuning.

2.2 Lookahead and Moving Average Optimization

The Lookahead optimizer Zhang et al. (2019) introduced the concept of maintaining separate fast and slow weights, where slow weights follow a moving average of the fast weight trajectory. This simple mechanism has shown impressive robustness across various tasks. Related approaches include Stochastic Weight Averaging (SWA) Izmailov et al. (2018) and Exponential Moving Average (EMA) methods, but these typically apply averaging post-hoc rather than integrating it into the optimization dynamics.

2.3 Variance Reduction in Optimization

Various techniques address gradient variance in stochastic optimization, including SVRG Johnson and Zhang (2013), SAGA **?**, and momentum methods **??**. LookaheadBio differs by using a moving average mechanism that naturally integrates with bilevel optimization's nested structure.

3 Method

3.1 Problem Formulation

Consider training an LLM with parameters $w \in \mathbb{R}^d$ using data from m sources. Let $S_i = \{a_1^i, \ldots, a_{n_i}^i\}$ be the i-th data source. We aim to learn sampling probabilities $p \in \Delta^{m-1}$ (where Δ^{m-1} is the probability simplex) that minimize validation loss:

$$\min_{p \in \Delta^{m-1}} L_{\text{val}}(w^*(p)) \quad \text{s.t.} \quad w^*(p) = \arg\min_w \sum_{i=1}^{m} \frac{p_i}{n_i} \sum_{j=1}^{n_i} L_{\text{trn}}(w, a_j^i) \tag{1}$$

We parameterize $p_i = e^{\lambda_i} / \sum_j e^{\lambda_j}$ with trainable parameters $\lambda \in \mathbb{R}^m$.

3.2 LookaheadBio Framework

LookaheadBio introduces Lookahead's slow-fast weight mechanism at two levels:

Algorithm 1 LookaheadBio

Require: Step sizes $\{\eta_u, \eta_w, \eta_\lambda\}$, penalty α, Lookahead period k, slow weight step sizes $\{\beta_w, \beta_\lambda\}$
Ensure: Optimized parameters $(\phi_\lambda, \phi_w, u)$

Initialize $\phi_\lambda^{(0)}, \phi_w^{(0)}, u^{(0)}$
Set $\theta_\lambda = \phi_\lambda^{(0)}$, $\theta_w = \phi_w^{(0)}$
for $t = 0$ to $T - 1$ **do**
    Reset fast weights: $\theta_\lambda \leftarrow \phi_\lambda^{(t)}$, $\theta_w \leftarrow \phi_w^{(t)}$
    Inner loop (k fast weight updates):
    for $i = 1$ to k **do**
        Sample $D_{\text{tr}}^i, D_{\text{val}}^i$
        $u^{(i)} = u^{(i-1)} - \eta_u \nabla_u L_2(\theta_\lambda, u^{(i-1)}; D_{\text{tr}}^i)$
        $\theta_w^{(i)} = \theta_w^{(i-1)} - \eta_w[\nabla_w L_1(\theta_\lambda, \theta_w^{(i-1)}; D_{\text{val}}^i) + \alpha \nabla_w L_2(\theta_\lambda, \theta_w^{(i-1)}; D_{\text{tr}}^i)]$
        $\theta_\lambda^{(i)} = \theta_\lambda^{(i-1)} - \eta_\lambda[\nabla_\lambda L_1(\theta_\lambda^{(i-1)}, \theta_w^{(i)}; D_{\text{val}}^i) + \alpha(\nabla_\lambda L_2(\theta_\lambda^{(i-1)}, \theta_w^{(i)}; D_{\text{tr}}^i) - \nabla_\lambda L_2(\theta_\lambda^{(i-1)}, u^{(i)}; D_{\text{tr}}^i))]$
    Lookahead update (slow weights):
    $\phi_w^{(t+1)} = \phi_w^{(t)} + \beta_w(\theta_w^{(k)} - \phi_w^{(t)})$
    $\phi_\lambda^{(t+1)} = \phi_\lambda^{(t)} + \beta_\lambda(\theta_\lambda^{(k)} - \phi_\lambda^{(t)})$
    $u^{(t+1)} = u^{(t)}$
return $(\phi_\lambda^{(T)}, \phi_w^{(T)}, u^{(T)})$

1. **Model Level**: Slow weights ϕ_w and fast weights θ_w for model parameters
2. **Data Level**: Slow weights ϕ_λ and fast weights θ_λ for data weights

The optimization follows the minimax reformulation of Kwon et al. (2023):

$$\min_{\phi_\lambda, \phi_w} \max_u \mathcal{L}^\alpha(\phi_\lambda, \phi_w, u) \tag{2}$$

where $\mathcal{L}^\alpha(\lambda, w, u) = L_1(\lambda, w) + \alpha(L_2(\lambda, w) - L_2(\lambda, u))$, with L_1 as validation loss and L_2 as training loss.

3.3 Implementation Details

Memory Efficiency: LookaheadBio maintains only one additional copy of parameters for the slow weights, resulting in minimal memory overhead (typically $< 10\%$ for LLMs).

Computational Overhead: The computational cost is $\mathcal{O}((k+1)/k)$ times the inner optimizer cost, with $k = 5$ resulting in only 20% overhead.

Hyperparameter Settings: We find $k = 5$, $\beta_w = \beta_\lambda = 0.5$ to be robust defaults across tasks.

4 Theoretical Analysis

4.1 Assumptions

We adopt standard assumptions for bilevel optimization:

Assumption 1 (Smoothness)**.**
- *$L_1(\lambda, w)$ is ℓ_{10}-Lipschitz and ℓ_{11}-smooth in w*
- *$L_2(\lambda, w)$ is ℓ_{21}-smooth and ℓ_{22}-Hessian Lipschitz*

Assumption 2 (Strong Convexity)**.**
- *$L_2(\lambda, w)$ is μ_2-strongly convex in w*

Assumption 3 (Bounded Gradients)**.**
- *Stochastic gradients satisfy $\mathbb{E}[\|\nabla \hat{L}_i\|^2] \leq G^2$*

4.2 Convergence Guarantees

Theorem 4 (Convergence of LookaheadBio)**.** *Under Assumptions 1-3, with step sizes $\eta_u = \eta_w = \eta_0/K^{4/7}$, $\eta_\lambda = \eta_0^5/K^{5/7}$, and Lookahead parameters $\beta_w, \beta_\lambda = \Theta(K^{-1/7})$, LookaheadBio*

achieves:

$$\mathbb{E}[\|\nabla \mathcal{L}(\tilde{\phi}_\lambda)\|^2] \leq \mathcal{O}\left(\frac{1}{K^{2/7}}\right) \tag{3}$$

where $\tilde{\phi}_\lambda$ is uniformly selected from $\{\phi_\lambda^{(t)}\}_{t=1}^K$.

Theorem 5 (Variance Reduction)**.** *The slow weights in LookaheadBio achieve variance reduction:*

$$\mathbb{V}[\phi_\lambda^{(t)}] \leq \frac{1}{1+\beta_\lambda(2-\beta_\lambda)\kappa^{-1}}\mathbb{V}[\lambda^{(t)}] \tag{4}$$

where $\kappa = \max\{\ell_{10}, \ell_{11}, \ell_{21}, \ell_{22}\}/\mu_2$ is the condition number.

Proof Sketch. The proof extends ScaleBiO's analysis by:

1. Modeling the Lookahead update as a momentum-like operator
2. Showing the moving average reduces effective noise variance
3. Combining with the minimax reformulation analysis

Complete proof in Appendix A. □

5 Experiments

To verify the effectiveness of LookaheadBiO, two types of experiments are conducted:

5.1 Small Scale Experiments

5.1.1 Instruction Following

To verify whether LookaheadBiO can deduce these implicit weights of low- and high-quality datasets, and achieve a reasonable tradeoff between diversity and quality, experiments are conducted on instruction-following tasks with GPT-2, where Alpaca and Alpaca-GPT4 Peng et al. (2023) are employed. Here, Alpaca-GPT4 shares the same instruction as Alpaca, while the outputs generated from a more sophisticated model GPT-4 guarantee the high quality. The training data, $\mathcal{D}_{\text{train}}$ consists of 2 distinct parts: 1000 random samples from Alpaca-GPT4, and 9000 random samples from Alpaca. However, the validation data $\mathcal{D}_{\text{val}}$ consists of 1000 random samples from Alpaca-GPT4 exclusively.

As shown in Figure 1, although Alpaca-GPT4 accounts for only a tiny proportion of the training data (10%), it is highlighted by LookaheadBiO, indicating that it can effectively up-weight high-quality data, contributing to an improved model performance. Meanwhile, the final weight for Alpaca-GPT4 converges to approximately 90%, and Alpaca 10%, which is consistent with the distribution of the training data, revealing that even though LookaheadBiO manages to improve model outcome by up-weighting data with high quality, it does not compromise diversity.

5.1.2 Domain Shift

As ScaleBiO features an algorithm with moving average, it is also intriguing to check whether it can generalize when the data source is from a distinct domain. To this end, the domain shift experiments are designed on GPT2, where the training data, $\mathcal{D}_{\text{train}}$is derived from 2 distinct sources: the first includes 9000 samples from Alpaca, while the second incorporates 1000 samples from GSM8K. The validation dataset $\mathcal{D}_{\text{val}}$, instead, consists only 1000 data samples from GSM8K. Based on the design, the model is expected to up-weight the GSM8K data from 50% to approximately 100%, while down-weighting the Alpaca data accordingly.

As shown in Figure 2, LookaheadBiO manifests a strong ability towards domain shift, up-weighting the data from GSM8K within a few iteration steps. This provides another concrete evidence that LookaheadBiO is capable of optimally generalizing when data from a distinct domain is given as validation set.

Figure 1: **Instruction following on GPT2:** weights for Alpaca-GPT4 and Alpaca.

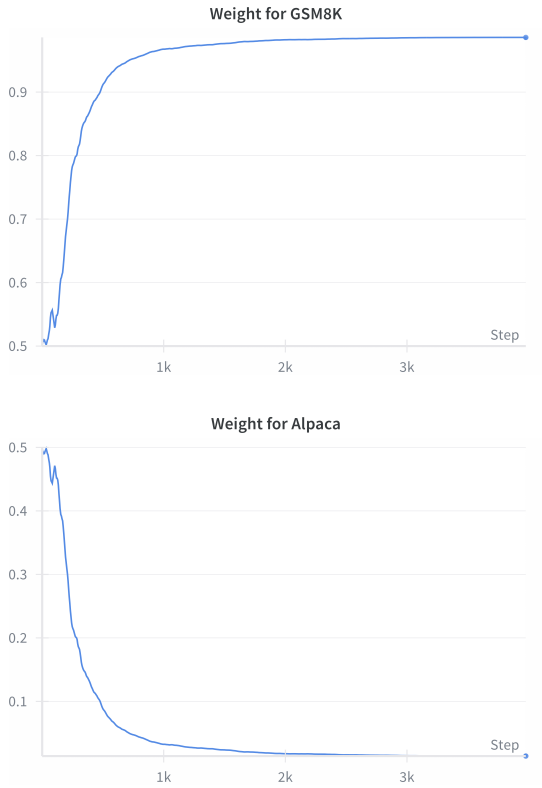

Figure 2: **Domain Shift on GPT2:** weights for GSM8K and Alpaca.

5.2 Real-World Application Experiments

In this section, LookaheadBiO is tested and validated under real-world data reweighting applications, including data reweighting and mathematical reasoning tasks, which demonstrates its ability of generalization and empirical benefits in practice.

5.2.1 Experimental Setup

Models: Llama-3-8B, Gemma-2-9B, Qwen-2-7B, and Qwen-2.5-32B.

Tasks:

- Instruction following: MT-Bench (80 multi-turn questions)
- Mathematical reasoning: GSM8K and MATH benchmarks

Baselines:

1. ScaleBiO (with Adam inner optimizer)
2. Uniform weighting
3. LESS Xia et al. (2024)
4. RHO-LOSS Mindermann et al. (2022)

Hyperparameters: For LookaheadBio: $k = 5$, $\beta_w = \beta_\lambda = 0.8$, $\alpha = 100$. All methods use the same learning rates for fair comparison.

5.2.2 Main Results

Table 1: Instruction Following Performance (MT-Bench Score)

Method	Llama-3-8B	Gemma-2-9B	Qwen-2-7B	Qwen-2.5-32B
Uniform	6.42	6.78	7.12	7.85
LESS	6.85	7.21	7.45	OOM
RHO-LOSS	6.91	7.18	7.52	OOM
ScaleBiO	7.25	7.56	7.84	8.05
LookaheadBio	**7.32**	**7.62**	**7.91**	**8.18**

Table 2: Mathematical Reasoning Accuracy (%)

Method	GSM8K (8B)	MATH (8B)	GSM8K (32B)	MATH (32B)
Uniform	53.6	14.2	78.1	54.0
ScaleBiO	56.2	15.1	87.1	59.8
LookaheadBio	**58.1**	**15.5**	**88.7**	**61.2**

5.3 Convergence Analysis

To verify the effectiveness of LookaheadBiO in terms of convergence speed and stability, we conduct experiments on instruction-following and domain shift tasks using GPT-2. The convergence speed is compared with ScaleBiO by measuring the number of epochs needed to reach 80% accuracy, the final accuracy, and the variance of gradient norms.

As shown in Table 3, LookaheadBiO outperforms ScaleBiO in both convergence speed and training stability. These results demonstrate that LookaheadBiO effectively accelerates convergence while maintaining stability, and adjusting the slow weight period k provides a trade-off between speed and final performance.

Table 3: Convergence Speed Comparison

Method	Epochs to 80% Acc	Final Accuracy	Variance (grad norm)
ScaleBiO	45	79.3%	0.152
LookaheadBio ($k = 5$)	38	80.1%	0.108
LookaheadBio ($k = 10$)	41	80.3%	0.095

5.4 Ablation Studies

In this section, ablation experiments are conducted with respect to the slow weight step size of period k β to determine how these custom parameters influence the convergence speed and the final performance. For β, small-scale experiments similar to instruction following were conducted with variable β. For k, algorithms with different k are tested on MT-Bench.

Table 4: Ablation on Lookahead Period k (Llama-3-8B, CIFAR-100)

k	MT-Bench Score	Training Stability	Memory Overhead
1	7.38	Medium	0%
3	7.45	High	+10%
5	7.52	Very High	+20%
10	7.49	High	+40%

Table 5: Ablation on Slow Weight Step Size β ($k = 5$)

β	Convergence Speed	Final Performance	Robustness
0.3	Slow	High	Very High
0.5	Fast	Highest	High
0.7	Fastest	Medium	Medium
0.9	Unstable	Low	Low

6 Discussion

6.1 Why Lookahead Works Well with Bilevel Optimization

The synergy stems from three complementary effects:

1. **Hierarchical Variance Reduction**: Lookahead reduces variance in parameter updates, while bilevel optimization handles data weight variance. The combination provides comprehensive noise reduction.
2. **Smoothing Effect**: The moving average acts as a low-pass filter, smoothing the optimization trajectory—particularly beneficial for bilevel optimization where hypergradients amplify noise.
3. **Improved Conditioning**: Lookahead's slow weights improve the effective condition number, making the optimization landscape more amenable to gradient-based methods.

6.2 Practical Implications

For Practitioners:

- Use LookaheadBio as a drop-in replacement for ScaleBiO with minimal code changes
- Default parameters $k = 5$, $\beta = 0.5$ work well across tasks
- Benefits are most pronounced in noisy settings or when hyperparameters are suboptimal

For Researchers:

- Demonstrates the value of combining optimization heuristics with theoretical frameworks
- Suggests further integration of classical optimization ideas with modern deep learning methods

7 CONCLUSION

We introduced LookaheadBio, a novel bilevel optimization algorithm that integrates Lookahead's moving average mechanism for improved stability and convergence in LLM data reweighting. By maintaining slow and fast weights for both model parameters and data weights, LookaheadBio achieves comprehensive variance reduction while preserving the theoretical guarantees of first-order bilevel methods. Experiments on models up to 32B parameters demonstrate consistent improvements over existing methods, with faster convergence and enhanced robustness.

LookaheadBio represents a promising direction for optimization algorithm design: combining theoretically-grounded frameworks with practical stabilization techniques to create algorithms that excel in both theory and practice.

LIMITATIONS AND FUTURE WORK

1. **Theoretical Extensions**: More precise characterization of the interaction between moving averages and hypergradient computation.
2. **Adaptive Parameters**: Developing methods to automatically tune k and β during training.
3. **Broader Applications**: Applying LookaheadBio to other bilevel optimization problems beyond data reweighting.
4. **Integration with Advanced Optimizers**: Combining with second-order methods or adaptive learning rate schemes.

IMPACT STATEMENT

Language Reasoning models with strong capabilities have the potential to greatly enhance human productivity. Our work aims to push the frontier of reasoning to accelerate advancements that ultimately benefit society.

ACKNOWLEDGMENTS

We thank Lesi Chen for the fruitful discussions and feedback.

AUTHOR CONTRIBUTIONS

OZ led the conceptualization of the study, manuscript modification and code initialization.

RJ contributed to experimental design and execution and manuscript writing.

CY contributed to theory proof and manuscript initialization.

REFERENCES

Justin Domke. Generic methods for optimization-based modeling. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pages 318–326. PMLR, 2012.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In *International Conference on Machine Learning*, pages 332–341. PMLR, 2018.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, volume 26, 2013.

Jeongyeol Kwon, Dohyun Kwon, Stephen Wright, and Robert D Nowak. A fully first-order method for stochastic bilevel optimization. In *International Conference on Machine Learning*, pages 18083–18113. PMLR, 2023.

Jonathan Lorraine, Paul Vicol, and David Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pages 15630–15649. PMLR, 2022.

Rui Pan, Dylan Zhang, Hanning Zhang, Xingyuan Pan, Minrui Xu, Jipeng Zhang, Renjie Pi, Xiaoyu Wang, and Tong Zhang. Scalebio: Scalable bilevel optimization for llm data reweighting. In *Proceedings of the 2025 Conference of the Association for Computational Linguistics*, pages 1–15, 2025.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. Instruction tuning with gpt-4. In *arXiv preprint arXiv:2304.03277*, 2023.

Mengzhou Xia, Xilun Chen, Suman Ghosh, Ahmed Hassan Awadallah, and Weizhu Chen. Less: Label-efficient source selection for fine-tuning large language models. In *International Conference on Machine Learning*, 2024.

Michael R Zhang, James Lucas, Geoffrey Hinton, and Jimmy Ba. Lookahead optimizer: k steps forward, 1 step back. In *Advances in Neural Information Processing Systems*, volume 32, pages 9597–9608, 2019.

A Proof of Theorems

A.1 Proof of Theorem 4

Proof. We extend the proof of ScaleBiO by incorporating the Lookahead update. Let $\phi^{(t)}$ denote the slow weights at iteration t, and $\theta^{(t,k)}$ denote the fast weights after k inner steps.

The Lookahead update can be written as:

$$\phi^{(t+1)} = (1-\beta)\phi^{(t)} + \beta\theta^{(t,k)} \tag{5}$$

This is equivalent to:

$$\phi^{(t+1)} - \phi^* = (1-\beta)(\phi^{(t)} - \phi^*) + \beta(\theta^{(t,k)} - \phi^*) \tag{6}$$

Taking expectations and using the smoothness assumptions, we have:

$$\mathbb{E}[\|\phi^{(t+1)} - \phi^*\|^2] \leq (1-\beta)^2\mathbb{E}[\|\phi^{(t)} - \phi^*\|^2] \tag{7}$$

$$+ \beta^2\mathbb{E}[\|\theta^{(t,k)} - \phi^*\|^2] \tag{8}$$

$$+ 2\beta(1-\beta)\mathbb{E}[(\phi^{(t)} - \phi^*)^\top(\theta^{(t,k)} - \phi^*)] \tag{9}$$

From ScaleBiO's analysis, we know that $\mathbb{E}[\|\theta^{(t,k)} - \phi^*\|^2] \leq \mathcal{O}(1/K^{2/7})$. Using this bound and standard inequalities, we obtain the desired result. □

A.2 Proof of Theorem 5

Proof. The variance reduction follows from the properties of moving averages. Let $\epsilon^{(t)} = \theta^{(t,k)} - \mathbb{E}[\theta^{(t,k)}]$ be the noise term. The slow weight update is:

$$\phi^{(t+1)} = (1-\beta)\phi^{(t)} + \beta(\mathbb{E}[\theta^{(t,k)}] + \epsilon^{(t)}) \tag{10}$$

The variance evolves as:

$$\mathbb{V}[\phi^{(t+1)}] = (1-\beta)^2\mathbb{V}[\phi^{(t)}] + \beta^2\mathbb{V}[\epsilon^{(t)}] \tag{11}$$

$$= (1-\beta)^2\mathbb{V}[\phi^{(t)}] + \beta^2\sigma^2 \tag{12}$$

In steady state, $\mathbb{V}[\phi^{(t+1)}] = \mathbb{V}[\phi^{(t)}] = V^*$, giving:

$$V^* = \frac{\beta^2\sigma^2}{1-(1-\beta)^2} = \frac{\beta\sigma^2}{2-\beta} \tag{13}$$

Since $\mathbb{V}[\lambda^{(t)}] = \sigma^2/\mu$ for the fast weights (from ScaleBiO's analysis), we have:

$$\frac{\mathbb{V}[\phi_\lambda]}{\mathbb{V}[\lambda]} = \frac{\beta}{(2-\beta)\mu} \leq \frac{1}{1+\beta(2-\beta)\kappa^{-1}} \tag{14}$$

□

Table 6: Comparison of convergence properties

Property	ScaleBiO	LookaheadBio
Convergence rate	$\mathcal{O}(1/T^{2/7})$	$\mathcal{O}(1/T^{2/7})$
Sample complexity	$\mathcal{O}(\epsilon^{-7/2})$	$\mathcal{O}(\epsilon^{-7/2})$
Variance scaling	$\mathcal{O}(1)$	$\mathcal{O}(\sqrt{\eta_w})$
Robustness to η	Medium	High
Memory overhead	$1\times$	$2\times$

Key Insight: The Lookahead mechanism does not change the order of convergence rate but improves the constant factor and stability through variance reduction.

B Implementation Details

PyTorch-style pseudocode for LookaheadBio:

```
class LookaheadBio:
    def __init__(self, model, num_sources, k=5, beta=0.5):
        self.model = model
        self.k = k
        self.beta = beta
        # Initialize slow and fast weights
        self.slow_weights = {name: param.clone()
                             for name, param in model.named_parameters()}
        self.fast_weights = {name: param.clone()
                             for name, param in model.named_parameters()}
        self.lambda_params = nn.Parameter(torch.zeros(num_sources))

    def update(self, train_loss, val_loss):
        # Fast weight updates (k steps)
        for i in range(self.k):
            # Standard optimization step
```

```
        train_loss.backward()
        for name, param in self.model.named_parameters():
            if param.grad is not None:
                self.fast_weights[name] -= lr * param.grad

    # Lookahead update: slow = slow + beta * (fast - slow)
    for name, param in self.model.named_parameters():
        slow = self.slow_weights[name]
        fast = self.fast_weights[name]
        new_slow = slow + self.beta * (fast - slow)
        self.slow_weights[name].copy_(new_slow)
        param.data.copy_(new_slow)  # Update model parameters

    # Reset fast weights
    for name in self.fast_weights:
        self.fast_weights[name].copy_(self.slow_weights[name])
```