Appendices

Contents

A Nomenclature

Table 2: Nomenclature

Symbol	Description
Model Parameters and Tensors	
p_i	Parameter tensor.
$\text{layer}(p)$	Function mapping a parameter p to its layer identifier.
$\text{vec}(\cdot)$	Operation that flattens a tensor into a vector.
Gradients and Grouping	
g_i	Gradient of parameter p_i.
$\nabla f(p)$	Gradient of the objective function f with respect to p.
$g_t \in \mathbb{R}^n$	Full (raw) gradient at iteration t (vector of dimension n).
n	Total number of gradient components.
m	Number of disjoint groups into which g_t is partitioned.
j	Group index, $j \in \{1, 2, \ldots, m\}$.
G_j	jth group (subset) of gradient components from g_t.
$\lvert G_j \rvert$	Number of components in group G_j.
μ_j	Empirical mean of the components in group G_j.
σ_j^2	Empirical variance of the components in group G_j.
ϵ	Stability constant: small positive constant added in normalization ($\sigma_j + \epsilon$) to avoid division by zero.
$\tilde{g}_t$	Normalized gradient at iteration t, defined as $[\tilde{g}_t]_i = \frac{[g_t]_i - \mu_j}{\sigma_j + \epsilon}$ for $i \in G_j$.
ρ_j	Variance contraction factor for group G_j, defined as $\rho_j = \frac{\sigma_j^2}{(\sigma_j + \epsilon)^2}$.
ρ	Global constant such that $\rho_j \leq \rho < 1$ for all groups.
Layer-Wise Grouping (if applicable)	
$\mathcal{G}_l$	Set of parameters (or corresponding gradient indices) in layer l.
v_l	Flattened gradient vector for layer l.
$\lvert v_l \rvert$	Number of elements in v_l.
$v_l^{(j)}$	jth subgroup of v_l (when partitioning a layer's gradients).
$\mu_l^{(j)}$	Mean of subgroup j in layer l.
$\sigma_l^{(j)}$	Standard deviation of subgroup j in layer l.
$\tilde{v}_l$	Concatenated normalized gradient vector for layer l.
Optimization Variables and Algorithm Parameters	
w_t	Current model parameters (iterate) at iteration t.
f or F	Objective function (convex: f; non-convex: F).
η	Step size (learning rate).
T	Total number of iterations.
D	Diameter of the feasible set $\mathcal{X}$.
$\mathcal{X}$	Feasible set of model parameters (subset of $\mathbb{R}^d$).
G	Uniform upper bound on the gradient norm.
Theoretical Analysis Parameters	
δ_t	Alignment error vector defined by $\tilde{g}_t = \nabla f(w_t) + \delta_t$.
ε	Alignment error bound: a scalar such that $\lvert\langle \nabla f(w_t), \delta_t \rangle\rvert \leq \varepsilon \, \lVert \nabla f(w_t) \rVert^2$. specifically, it is bounded as $\lvert\langle \nabla f(w_t), \delta_t \rangle\rvert \leq \varepsilon \, \lVert \nabla f(w_t) \rVert^2$, with $\varepsilon = \frac{\epsilon}{\sigma_{\min}} C(\tau)$.
α	Constant in $(0, 1)$ controlling the ratio between ϵ and σ_j to ensure proper scaling.
$C(\tau)$	Constant that depends on the sub-Gaussian parameter τ of the gradient components.
$\Theta(\cdot)$	Tight asymptotic bound.
$O(\cdot)$	Big-O notation.
Regret Analysis	
$R(T)$	Cumulative regret over T iterations.
$R_{SGD}(T)$	Regret bound for standard SGD.
$R_{Milo}(T)$	Regret bound for Normalized SGD with Layer-Wise Grouping (Milo).

B Convergence Proof for Convex Settings

In this section we present the main convergence proof. We begin by introducing the definitions, assumptions, and lemmas that will be used throughout.

B.1 Proof Sketch of Theorem

Under standard L-smoothness, bounded-gradient, and variance-contraction/moment conditions, we obtain the following.

Proof Sketch of Theorem B. We outline the main steps under Assumptions B.1–B.4 and Lemmas B.4, B.2.

1. Smoothness Step. By L–smoothness,

$$f(w_{t+1}) \le f(w_t) + \langle \nabla f(w_t),\, w_{t+1} - w_t \rangle + \tfrac{L}{2}\|w_{t+1} - w_t\|^2.$$

Using the update $w_{t+1} = w_t - \eta \tilde{g}_t$ gives

$$f(w_{t+1}) \le f(w_t) - \eta \langle \nabla f(w_t), \tilde{g}_t \rangle + \tfrac{L\eta^2}{2}\|\tilde{g}_t\|^2.$$

2. Variance-Contraction Bound. By Assumption B.3 and Lemma B.4, each group satisfies $\|[\tilde{g}_t]_{G_j}\|^2 \le \rho\,|G_j|$. Summing yields

$$\|\tilde{g}_t\|^2 \le \rho\, n.$$

3. Alignment Control. Decompose $\tilde{g}_t = \nabla f(w_t) + \delta_t$. Lemma B.2 gives

$$|\langle \nabla f(w_t), \delta_t \rangle| \le \varepsilon \|\nabla f(w_t)\|^2,$$

so

$$\langle \nabla f(w_t), \tilde{g}_t \rangle \ge (1-\varepsilon)\|\nabla f(w_t)\|^2.$$

4. One-Step Progress. Combine the above bounds:

$$f(w_{t+1}) \le f(w_t) - \eta(1-\varepsilon)\|\nabla f(w_t)\|^2 + \tfrac{L\eta^2}{2}\rho\, n.$$

5. Telescoping Sum. Summing over $t = 1 \dots T$ and using $f(w_{T+1}) \ge f(w^*)$:

$$\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \le \frac{f(w_1) - f(w^*)}{\eta(1-\varepsilon)} + \frac{L\eta T}{2(1-\varepsilon)}\rho\, n.$$

6. Learning-Rate Choice. Setting $\eta = \Theta(1/\sqrt{T})$ balances the two terms and yields

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 = O\big(T^{-1/2}\big).$$

7. Regret Comparison. Standard SGD regret is $O(GD\sqrt{T})$. Replacing G by $\sqrt{\rho n}$ gives Milo

$$R_{\text{Milo}}(T) = O\big(D\sqrt{\rho n\, T}\big) < O(DG\sqrt{T}) = R_{\text{SGD}}(T),$$

showing improved dependence on the variance.

B.2 Definitions and Assumptions

Definition B.1 (Convex Function)**.** *A function $f : \mathbb{R}^d \to \mathbb{R}$ is* convex *if for all $x, y \in \mathbb{R}^d$ and all $\lambda \in [0,1]$,*

$$\lambda f(x) + (1-\lambda) f(y) \ge f\big(\lambda x + (1-\lambda) y\big). \tag{B.1}$$

Remark B.1. *Notice that for a convex function f, one can lower bound f by a hyperplane at any point of its domain. In particular, if f is differentiable, then for all $x, y \in \mathbb{R}^d$,*

$$f(y) \ge f(x) + \nabla f(x)^T (y - x). \tag{B.2}$$

Assumption B.1 (Smoothness)**.** *The function $f : \mathbb{R}^d \to \mathbb{R}$ is differentiable and L-*smooth*; that is, for all $x, y \in \mathbb{R}^d$,*

$$\|\nabla f(x) - \nabla f(y)\|_2 \leq L\|x - y\|_2. \tag{B.3}$$

Often, f is given as a finite sum $f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ with individual smoothness constants L_i; we denote $L_{\max} = \max\{L_1, \ldots, L_n\}$.

[See, e.g., [Nesterov, 1983] for the smoothness condition.]

Assumption B.2 (Bounded Gradient Norm)**.** *For every $x \in \mathcal{X}$ (the feasible set), the gradient is bounded:*

$$\|\nabla f(x)\|_2 \leq G, \tag{B.4}$$

with $G > 0$.

[This is standard in stochastic optimization; see, e.g., [Bottou et al., 2018].]

Assumption B.3 (Variance Contraction via Normalization)**.** *Let $g_t = \nabla f(w_t)$ be the gradient at iteration t. Partition g_t into disjoint groups $\{G_j\}_{j=1}^{m}$ (with $\sum_{j=1}^{m} |G_j| = n$). For each group G_j, define the empirical mean and variance as*

$$\mu_j = \frac{1}{|G_j|}\sum_{i \in G_j} [g_t]_i, \quad \sigma_j^2 = \frac{1}{|G_j|}\sum_{i \in G_j} \Big([g_t]_i - \mu_j\Big)^2. \tag{B.5}$$

The normalized gradient is then defined by

$$[\tilde{g}_t]_i = \frac{[g_t]_i - \mu_j}{\sigma_j + \epsilon}, \quad i \in G_j, \tag{B.6}$$

Because the mean is subtracted in Equation (B.6) (centering the data), the variance of the normalized gradient in G_j becomes

$$\tilde{\sigma}_j^2 = \frac{\sigma_j^2}{(\sigma_j + \epsilon)^2}. \tag{B.7}$$

Define the variance contraction factor as

$$\rho_j = \frac{\sigma_j^2}{(\sigma_j + \epsilon)^2}. \tag{B.8}$$

We assume that there exists a global constant $\rho \in [0, 1)$ such that

$$\rho_j \leq \rho, \quad \forall\, j. \tag{B.9}$$

[See, e.g., [Wu and He, 2018] for the effects of group normalization on variance contraction.]

Assumption B.4 (Moment Conditions)**.** *For each parameter tensor p with flattened gradient $v = \mathrm{vec}(g(p))$, assume:*

1. *(**Large Group Size**) Each group G_j (obtained by partitioning v) satisfies $|G_j| \geq N_0$ so that the empirical estimates*

$$\mu_j = \frac{1}{|G_j|}\sum_{i \in G_j} v_i, \quad \sigma_j^2 = \frac{1}{|G_j|}\sum_{i \in G_j} (v_i - \mu_j)^2 \tag{B.10}$$

   concentrate around the true moments.

2. *(**Bounded Higher-Order Moments**) There exists a constant M_k and some $k \geq 3$ such that*

$$\mathbb{E}[|v_i - \mu_j|^k] \leq M_k. \tag{B.11}$$

3. *(**Symmetry**) The distribution of v_i for $i \in G_j$ is symmetric (or nearly so) around μ_j.*

[Moment conditions as in [Vershynin, 2018].]

B.3 Auxiliary Lemmas

Lemma B.1 (Concentration of Empirical Means)**.** *Let $v_1, \dots, v_N$ be independent random variables in a group G_j with true mean μ and assume $|v_i - \mu| \le B$ almost surely. Then, for any $\delta > 0$,*

$$\Pr\left(\left| \frac{1}{N} \sum_{i=1}^{N} v_i - \mu \right| \ge \delta \right) \le 2 \exp\left(-\frac{2N\delta^2}{B^2} \right). \tag{B.12}$$

A similar bound holds for the empirical variance.

[This is an application of Hoeffding's inequality; see [Hoeffding, 1963].]

Lemma B.2 (Formal Alignment of Normalized Gradient)**.** *Assume that for a given group G_j the gradient components can be expressed as*

$$v_i = \mu_j + \sigma_j z_i, \quad i \in G_j, \tag{B.13}$$

where z_i are independent sub-Gaussian random variables with zero mean, unit variance, and sub-Gaussian parameter τ. Define the normalized elements by

$$\tilde{v}_i = \frac{v_i - \mu_j}{\sigma_j + \epsilon}. \tag{B.14}$$

Then,

$$\tilde{v}_i = \frac{\sigma_j}{\sigma_j + \epsilon} \, z_i. \tag{B.15}$$

Define the scaling factor $P_j := \frac{\sigma_j}{\sigma_j + \epsilon}$. If for a fixed $\alpha \in (0, 1)$ it holds that $\epsilon \le \alpha \, \sigma_j$ (i.e. $P_j \ge \frac{1}{1+\alpha}$), then

$$1 - P_j \le \frac{\epsilon}{\sigma_j}. \tag{B.16}$$

Now, writing the decomposition

$$\tilde{g}_t = \nabla f(w_t) + \delta_t, \tag{B.17}$$

the above results (with appropriate concentration bounds) imply that

$$\left| \langle \nabla f(w_t), \delta_t \rangle \right| \le \varepsilon \, \|\nabla f(w_t)\|^2, \tag{B.18}$$

with $\varepsilon = \frac{\epsilon}{\sigma_{\min}} \, C(\tau)$ (here $\sigma_{\min} = \min_j \sigma_j$ and $C(\tau)$ is a constant depending on τ).

[See, e.g., [Duchi et al., 2011] for related analysis on adaptive and normalized gradient methods.]

Lemma B.3 (Robustness for Small σ_j with Explicit ϵ)**.** *Assume that for each group G_j either:*

1. *$\sigma_j \ge \sigma_{\min} > 0$, or*
2. *If $\sigma_j < \sigma_{\min}$, then $|v_i - \mu_j| \le \beta \, \sigma_{\min}$ for all $i \in G_j$,*

for some constants $\sigma_{\min} > 0$ and $\beta > 0$. Then, by choosing

$$\epsilon \le \gamma \, \sigma_{\min}, \tag{B.19}$$

for a given $\gamma \in (0, 1)$, it follows that for every $i \in G_j$,

$$|[\tilde{g}_t]_i| \le \frac{\beta}{\gamma}. \tag{B.20}$$

Lemma B.4 (Bounded Variance Contraction)**.** *Let $\epsilon > 0$ be such that for every group G_j (and for all iterations) it holds that*

$$\epsilon \ge c \, \sigma_j, \tag{B.21}$$

for some constant $c > 0$. Then, the variance contraction factor satisfies

$$\rho_j = \frac{\sigma_j^2}{(\sigma_j + \epsilon)^2} \le \left(\frac{1}{1 + c} \right)^2 < 1. \tag{B.22}$$

[This result is motivated by normalization techniques in [Ioffe and Szegedy, 2015].]

B.4 Proof of Convergence for Convex Settings

We now proceed with the convergence proof. The update rule for Milo is given (after normalization) by

$$w_{t+1} = w_t - \eta \, \tilde{g}_t, \tag{B.23}$$

where η is the learning rate and the normalized gradient $\tilde{g}_t$ is defined in eq. Equation (B.6). Throughout the proof we also use the following decomposition:

$$\tilde{g}_t = \nabla f(w_t) + \delta_t, \tag{B.24}$$

where δ_t represents the alignment error (as introduced in Lemma B.2).

Step 1: Smoothness Inequality. By Assumption B.1 (Smoothness), for any $x, y \in \mathbb{R}^d$ we have

$$f(y) \leq f(x) + \langle \nabla f(x), y - x \rangle + \frac{L}{2}\|y - x\|^2. \tag{B.25}$$

Taking $x = w_t$ and $y = w_{t+1} = w_t - \eta\, \tilde{g}_t$ yields

$$f(w_{t+1}) \leq f(w_t) - \eta \langle \nabla f(w_t), \tilde{g}_t \rangle + \frac{L\eta^2}{2}\|\tilde{g}_t\|^2. \tag{B.26}$$

Here:

- $\nabla f(w_t)$ is the true gradient at w_t,
- $\tilde{g}_t$ is the normalized gradient from Equation (B.6), and
- $\langle \nabla \mathbf{f}(\mathbf{w_t}), \tilde{\mathbf{g}}_\mathbf{t} \rangle$: The inner product measuring the alignment between the true gradient and the normalized gradient.
- $\mathbf{L}$: The Lipschitz constant of ∇f, governing the curvature of f, as in Assumption B.1.

Step 2: Bounding the Norm of $\tilde{g}_t$. Within each group G_j, by the variance contraction property in Assumption B.3 and Lemma B.4,

$$\|[\tilde{g}_t]_{G_j}\|^2 = \frac{|G_j|\,\sigma_j^2}{(\sigma_j + \epsilon)^2} \leq \rho\,|G_j|. \tag{B.27}$$

Summing over all groups, we obtain

$$\|\tilde{g}_t\|^2 \leq \sum_j \rho\,|G_j| = \rho\, n. \tag{B.28}$$

Step 3: Handling the Alignment Error. We decompose the normalized gradient as

$$\tilde{g}_t = \nabla f(w_t) + \delta_t. \tag{B.29}$$

Then, by Lemma B.2 (Formal Alignment of Normalized Gradient), under the moment conditions (Assumption B.4) and the condition $\epsilon \leq \alpha\, \sigma_j$ for all j, we have

$$|\langle \nabla f(w_t), \delta_t \rangle| \leq \varepsilon\, \|\nabla f(w_t)\|^2, \tag{B.30}$$

with $\varepsilon = \frac{\epsilon}{\sigma_{\min}}\, C(\tau)$. Consequently,

$$\langle \nabla f(w_t), \tilde{g}_t \rangle \geq (1 - \varepsilon)\|\nabla f(w_t)\|^2. \tag{B.31}$$

Step 4: Inserting the Bounds. Substitute the alignment bound and the norm bound into the smoothness inequality:

$$f(w_{t+1}) \leq f(w_t) - \eta(1 - \varepsilon)\|\nabla f(w_t)\|^2 + \frac{L\eta^2}{2}\rho\, n. \tag{B.32}$$

Step 5: Telescoping. Summing from $t = 1$ to T yields

$$\sum_{t=1}^{T} \Big[f(w_t) - f(w_{t+1}) \Big] \geq \eta(1 - \varepsilon) \sum_{t=1}^{T} \|\nabla f(w_t)\|^2 - \frac{L\eta^2 T}{2}\rho\, n. \tag{B.33}$$

Since the sum telescopes to $f(w_1) - f(w_{T+1})$ and $f(w_{T+1}) \geq f(w^*)$ (where w^* is an optimal point), we have

$$f(w_1) - f(w^*) \geq \eta(1 - \varepsilon) \sum_{t=1}^{T} \|\nabla f(w_t)\|^2 - \frac{L\eta^2 T}{2}\rho\, n. \tag{B.34}$$

Dividing both sides by $\eta(1 - \varepsilon)T$ gives

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \leq \frac{f(w_1) - f(w^*)}{\eta(1 - \varepsilon)T} + \frac{L\eta}{2(1 - \varepsilon)}\rho\, n. \tag{B.35}$$

Step 6: Choosing the Learning Rate. By setting

$$\eta = \Theta\Big(\frac{1}{\sqrt{T}}\Big), \tag{B.36}$$

the two terms on the right-hand side are balanced, leading to an overall convergence rate of

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 = O\Big(\frac{1}{\sqrt{T}}\Big). \tag{B.37}$$

Step 7: Regret Bound. Standard regret analysis for SGD yields

$$R_{\text{SGD}}(T) \leq \frac{D^2}{2\eta} + \frac{\eta G^2 T}{2}. \tag{B.38}$$

For Milo, if we denote by $\tilde{G}$ the effective bound on $\|\tilde{g}_t\|$ (with $\tilde{G} \leq \sqrt{\rho\, n}$), then the regret can be bounded as

$$R_{\text{Milo}}(T) \leq \frac{D^2}{2\eta} + \frac{\eta \tilde{G}^2 T}{2}. \tag{B.39}$$

Choosing $\eta = \frac{D}{\tilde{G}\sqrt{T}}$ gives

$$R_{\text{Milo}}(T) \leq D\,\tilde{G}\,\sqrt{T} < R_{\text{SGD}}(T) \leq D\,G\,\sqrt{T}, \tag{B.40}$$

thus demonstrating the advantage of variance reduction.

B.5 Discussion

The final convergence bound depends explicitly on the alignment error ε, the variance contraction factor ρ, and the learning rate η:

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \leq \frac{f(w_1) - f(w^*)}{\eta(1-\varepsilon)T} + \frac{L\eta}{2(1-\varepsilon)}\rho\, n. \tag{B.41}$$

Lemmas B.1 and B.2 ensure that with sufficiently large group sizes and proper moment conditions (Assumption B.4), the empirical estimates concentrate and the normalized gradient remains well aligned with the true gradient. By choosing ϵ relative to the minimum variance (as in Lemma B.3), even cases with very small σ_j are robustly handled. This completes the proof of the $O(1/\sqrt{T})$ convergence rate and confirms that Milo yields improved regret bounds compared to standard SGD.

B.6 Conclusion

We have presented a detailed, self-contained convergence proof for Milo under convex settings. By introducing the necessary definitions, assumptions, and auxiliary lemmas inline, and then integrating them into the convergence analysis, we demonstrate that under appropriate conditions the alignment error is controlled, leading to an $O(1/\sqrt{T})$ convergence rate with improved regret bounds over standard SGD. The explicit dependencies on parameters (such as ϵ, ρ, and ε) bridge the gap between theory and empirical performance.

B.7 Discussion and Practical Relevance

The final convergence bound explicitly depends on parameters such as ε (which captures the alignment error), ρ (the variance contraction factor), and η. Explicitly, we have

$$\frac{1}{T}\sum_{t=1}^{T} \|\nabla f(w_t)\|^2 \leq \frac{f(w_1) - f(w^*)}{\eta(1-\varepsilon)T} + \frac{L\eta}{2(1-\varepsilon)}\rho\, n. \tag{B.42}$$

Our concentration lemmas (e.g., Lemma B.1) guarantee that for sufficiently large group sizes, the empirical estimates μ_j and σ_j^2 deviate from the true moments by at most an error that decays exponentially in $|G_j|$. This justifies the approximate unbiasedness assumption and confirms that, under practical conditions (as seen in our empirical experiments), the constants ε and ρ remain small. Similarly, by ensuring that ϵ is chosen relative to the minimum variance $\sigma_{\min}$ via Lemma B.3, even edge cases where σ_j is very small are handled gracefully, ensuring stable normalization.

B.8 Discussion of Key Technical Aspects

B.8.1 Unbiasedness of the Normalized Gradient

Since the normalization is a nonlinear transformation, in general we cannot expect

$$\mathbb{E}[\tilde{g}_t] = \nabla f(w_t) \tag{B.43}$$

to hold exactly. In our analysis, we assume approximate unbiasedness is achieved when:

1. The group sizes $|G_j|$ are large so that the empirical estimates μ_j and σ_j^2 converge to the true moments.
2. The gradient components within each group come from a symmetric (or nearly symmetric) distribution.

Under these conditions, any bias induced by the normalization is of order $O(\delta)$ (with δ small). Future work will provide more explicit concentration bounds on this bias.

B.8.2 Justification of the Variance Contraction Factor

We require that

$$\rho_j = \frac{\sigma_j^2}{(\sigma_j + \epsilon)^2} \leq \rho < 1. \tag{B.44}$$

A sufficient condition is that, for some constant $c > 0$, we have

$$\epsilon \geq c\,\sigma_j \quad \text{for all } j. \tag{B.45}$$

Then,

$$\sigma_j + \epsilon \geq (1 + c)\sigma_j \quad \Rightarrow \quad \frac{\sigma_j}{\sigma_j + \epsilon} \leq \frac{1}{1 + c}, \tag{B.46}$$

and thus

$$\rho_j \leq \left(\frac{1}{1 + c}\right)^2 \equiv \rho. \tag{B.47}$$

We include this formally as Lemma B.4.

B.8.3 Alignment Between Normalized and True Gradients

A key step in our convex analysis is the approximation

$$\langle \nabla f(w_t), \tilde{g}_t \rangle \approx \|\nabla f(w_t)\|_2^2. \tag{B.48}$$

Define the error vector δ_t by

$$\tilde{g}_t = \nabla f(w_t) + \delta_t. \tag{B.49}$$

Then,

$$\langle \nabla f(w_t), \tilde{g}_t \rangle = \|\nabla f(w_t)\|_2^2 + \langle \nabla f(w_t), \delta_t \rangle. \tag{B.50}$$

Under the assumption that the normalized gradient is nearly aligned with $\nabla f(w_t)$ (which holds if the deviations within each group are small relative to μ_j), we can show that

$$|\langle \nabla f(w_t), \delta_t \rangle| \leq \varepsilon \|\nabla f(w_t)\|_2^2, \tag{B.51}$$

for some $\varepsilon = O(\delta)$ with $\delta \ll 1$. A formal lemma with full details is deferred to future extensions.

B.8.4 Handling Small Denominators and the Role of ϵ

The normalization divides by $\sigma_j + \epsilon$. In cases where σ_j is very small, the update may be dominated by ϵ. To prevent unwanted distortion, one can:

1. Assume a lower bound $\sigma_j \geq \sigma_{\min} > 0$ or,
2. Choose ϵ sufficiently small relative to the typical scale of σ_j,
3. Or analyze the two regimes separately.

We provide a preliminary lemma showing that, if σ_j is extremely small but the deviations $|[g_t]_i - \mu_j|$ are also small, then the normalized gradient remains bounded.

Lemma B.5 (Robustness for Small σ_j)**.** *Assume that for each group G_j, either*

1. *$\sigma_j \geq \sigma_{\min} > 0$, or*
2. *When $\sigma_j < \sigma_{\min}$, the deviations satisfy $|[g_t]_i - \mu_j| \leq \beta\,\sigma_{\min}$ for some $\beta > 0$.*

Then, by choosing $\epsilon \leq \gamma\,\sigma_{\min}$ for a sufficiently small $\gamma > 0$, we have

$$|[\tilde{g}_t]_i| \leq \frac{\beta}{\gamma}, \quad \textit{for } i \in G_j, \tag{B.52}$$

ensuring that the update remains bounded.

C Convergence Proof for Non-Convex Settings

In this section, we present a detailed convergence analysis for Milo when the objective function is *non-convex*. Although nonconvexity precludes guarantees of reaching a global optimum, under standard smoothness and boundedness assumptions we can establish a sublinear rate in terms of the (squared) gradient norm.

Proof Sketch of Theorem C. Under Assumptions C.1, C.2, and the variance contraction/moment conditions (Assumptions B.3, B.4), we outline the main steps.

1. Descent Lemma. By L–smoothness of F, for $w_{t+1} = w_t - \eta \tilde{g}_t$:

$$F(w_{t+1}) \leq F(w_t) - \eta \langle \nabla F(w_t), \tilde{g}_t \rangle + \frac{L\eta^2}{2} \|\tilde{g}_t\|^2.$$

2. Alignment Bound. Decompose $\tilde{g}_t = \nabla F(w_t) + \delta_t$. From Lemma B.2,

$$\langle \nabla F(w_t), \tilde{g}_t \rangle \geq (1 - \varepsilon) \|\nabla F(w_t)\|^2.$$

3. Norm Control. Using variance contraction (Lemma B.4),

$$\|\tilde{g}_t\|^2 \leq \rho\, n.$$

4. One-Step Inequality. Combine the above into the descent:

$$F(w_{t+1}) \leq F(w_t) - \eta(1 - \varepsilon) \|\nabla F(w_t)\|^2 + \tfrac{L\eta^2}{2} \rho\, n.$$

Rearrange:

$$\eta(1 - \varepsilon) \|\nabla F(w_t)\|^2 \leq F(w_t) - F(w_{t+1}) + \tfrac{L\eta^2}{2} \rho\, n.$$

5. Telescoping Sum. Summing for $t = 0 \dots T - 1$ and using $F(w_T) \geq F^*$:

$$\sum_{t=0}^{T-1} \|\nabla F(w_t)\|^2 \leq \frac{F(w_0) - F^*}{\eta(1 - \varepsilon)} + \frac{L\eta T}{2(1 - \varepsilon)} \rho\, n.$$

6. Rate via Step-Size. Choose $\eta = \min\{1/L, c/\sqrt{T}\}$ for constant $c > 0$. Then both terms scale as $O(T^{-1/2})$, yielding

$$\min_{t<T} \|\nabla F(w_t)\|^2 = O(T^{-1/2}).$$

□

C.1 Assumptions and Preliminaries

Let $F : \mathbb{R}^d \to \mathbb{R}$ be a (possibly) non-convex function. In addition to the variance reduction ingredients from the convex analysis, we assume the following:

Assumption C.1 (Smoothness for Non-Convex Functions)**.** *F is differentiable and L-smooth; that is, for all $x, y \in \mathbb{R}^d$,*

$$\|\nabla F(x) - \nabla F(y)\|_2 \leq L \|x - y\|_2. \tag{C.1}$$

Assumption C.2 (Bounded Gradient)**.** *For every $x \in \mathbb{R}^d$, the gradient is bounded:*

$$\|\nabla F(x)\|_2 \leq G, \tag{C.2}$$

with $G > 0$.

We also assume that the grouping and normalization procedure is defined exactly as in the convex case. That is, for iteration t we have

$$g_t = \nabla F(w_t), \tag{C.3}$$

which is partitioned into groups $\{G_j\}_{j=1}^m$ with

$$\mu_j = \frac{1}{|G_j|} \sum_{i \in G_j} [g_t]_i, \quad \sigma_j^2 = \frac{1}{|G_j|} \sum_{i \in G_j} \Big([g_t]_i - \mu_j \Big)^2, \tag{C.4}$$

and the normalized gradient is defined as

$$[\tilde{g}_t]_i = \frac{[g_t]_i - \mu_j}{\sigma_j + \epsilon}, \quad i \in G_j. \tag{C.5}$$

We assume that Assumptions B.3 and B.4 from the convex proof (regarding variance contraction via normalization and moment conditions) hold unchanged here.

C.2 Proof of Convergence

The Milo update is given by

$$w_{t+1} = w_t - \eta\, \tilde{g}_t. \tag{C.6}$$

Since F is L-smooth (Assumption C.1), we have the descent lemma:

$$F(w_{t+1}) \le F(w_t) + \langle \nabla F(w_t), w_{t+1} - w_t \rangle + \frac{L}{2}\|w_{t+1} - w_t\|_2^2. \tag{C.7}$$

Substituting $w_{t+1} = w_t - \eta\, \tilde{g}_t$ into Equation (C.7), we obtain

$$F(w_{t+1}) \le F(w_t) - \eta\, \langle \nabla F(w_t), \tilde{g}_t \rangle + \frac{L\eta^2}{2}\|\tilde{g}_t\|_2^2. \tag{C.8}$$

Decomposition and Alignment. We write the normalized gradient as

$$\tilde{g}_t = \nabla F(w_t) + \delta_t, \tag{C.9}$$

where δ_t is the error due to the nonlinearity of the normalization. Under the moment conditions (Assumption B.4) and by using Lemma B.2 (Formal Alignment of Normalized Gradient), we can bound the deviation:

$$\langle \nabla F(w_t), \tilde{g}_t \rangle \ge (1 - \varepsilon)\|\nabla F(w_t)\|_2^2, \tag{C.10}$$

for some ε (dependent on $\epsilon/\sigma_{\min}$).

Bounding the Norm of $\tilde{g}_t$. As in the convex setting (see Lemma B.4), the group normalization yields

$$\|\tilde{g}_t\|_2^2 \le \rho\, n, \tag{C.11}$$

where n is the total number of parameters and $\rho \in [0, 1)$ is the variance contraction factor.

Combining the Bounds. Substitute the alignment bound Equation (C.10) and the norm bound Equation (C.11) into Equation (C.8):

$$F(w_{t+1}) \le F(w_t) - \eta\,(1 - \varepsilon)\|\nabla F(w_t)\|_2^2 + \frac{L\eta^2}{2}\rho\, n. \tag{C.12}$$

Rearrange Equation (C.12) to obtain an inequality on the gradient norm:

$$\eta\,(1 - \varepsilon)\|\nabla F(w_t)\|_2^2 \le F(w_t) - F(w_{t+1}) + \frac{L\eta^2}{2}\rho\, n. \tag{C.13}$$

Telescoping over Iterations. Summing Equation (C.13) over $t = 0, \ldots, T-1$ results in:

$$\eta\,(1 - \varepsilon) \sum_{t=0}^{T-1} \|\nabla F(w_t)\|_2^2 \le F(w_0) - F(w_T) + \frac{L\eta^2}{2}\rho\, nT. \tag{C.14}$$

Since $F(w_T) \ge F^*$ (with $F^* = \min_w F(w)$ or a known lower bound), we have:

$$\eta\,(1 - \varepsilon) \sum_{t=0}^{T-1} \|\nabla F(w_t)\|_2^2 \le F(w_0) - F^* + \frac{L\eta^2}{2}\rho\, nT. \tag{C.15}$$

Dividing both sides of Equation (C.15) by $\eta\,(1 - \varepsilon)T$ yields:

$$\frac{1}{T}\sum_{t=0}^{T-1} \|\nabla F(w_t)\|_2^2 \le \frac{F(w_0) - F^*}{\eta\,(1 - \varepsilon)T} + \frac{L\eta\,\rho\, n}{2(1 - \varepsilon)}. \tag{C.16}$$

Choosing the Learning Rate. By selecting the step-size as

$$\eta = \min\Big\{\frac{1}{L},\ \frac{c}{\sqrt{T}}\Big\}, \tag{C.17}$$

for some constant $c > 0$, the two terms on the right-hand side of Equation (C.16) balance and both decay as $O(1/\sqrt{T})$. Consequently, we obtain

$$\min_{t=0,\ldots,T-1} \|\nabla F(w_t)\|_2^2 = O\Big(\frac{1}{\sqrt{T}}\Big), \tag{C.18}$$

which establishes that Milo converges (in terms of the squared gradient norm) at a sublinear rate despite nonconvexity.

C.3 Summary of the Proof

1. **Smoothness:** Using L-smoothness we derived
$$F(w_{t+1}) \leq F(w_t) - \eta \langle \nabla F(w_t), \tilde{g}_t \rangle + \frac{L\eta^2}{2} \|\tilde{g}_t\|_2^2.$$
2. **Alignment:** The error decomposition $\tilde{g}_t = \nabla F(w_t) + \delta_t$ combined with Lemma B.2 gives
$$\langle \nabla F(w_t), \tilde{g}_t \rangle \geq (1 - \varepsilon) \|\nabla F(w_t)\|_2^2.$$
3. **Norm Bound:** Lemma B.4 ensures that $\|\tilde{g}_t\|_2^2 \leq \rho\, n$.
4. **Telescoping:** Summing over iterations and rearranging leads to
$$\frac{1}{T} \sum_{t=0}^{T-1} \|\nabla F(w_t)\|_2^2 \leq \frac{F(w_0) - F^*}{\eta\,(1-\varepsilon)T} + \frac{L\eta\,\rho\,n}{2(1-\varepsilon)}.$$
5. **Rate:** With the choice of step-size in Equation (C.17), we conclude that
$$\min_{t=0,\ldots,T-1} \|\nabla F(w_t)\|_2^2 = O\Big(\frac{1}{\sqrt{T}}\Big).$$

C.4 Discussion

The analysis shows that even in the non-convex setting, the normalized gradients provided by Milo yield a controlled descent of the objective F. The key technical tools are:

- **Smoothness:** Ensuring that the objective does not change too abruptly.
- **Variance Contraction:** The grouping and normalization (via Assumption B.3) reduce the effective noise in the gradient, leading to a tighter bound on $\|\tilde{g}_t\|$.
- **Alignment Control:** Lemma B.2 guarantees that the error introduced by normalization, δ_t, is sufficiently small relative to the true gradient.

Together, these factors allow us to conclude that Milo reaches an approximate stationary point (where $\|\nabla F(w_t)\|$ is small) at a rate of $O(1/\sqrt{T})$, even when the objective function is non-convex.

C.5 Conclusion

We have provided a detailed, self-contained convergence proof for Milo in the non-convex setting. By leveraging the same normalization strategy that yields variance contraction in the convex case and carefully controlling the alignment error, we show that Milo achieves an $O(1/\sqrt{T})$ convergence rate in terms of the squared gradient norm. This result provides theoretical support for the empirical success of Milo in both convex and non-convex scenarios.

D Experimentation

D.1 Experiment Hardware

Table 3: Static Hardware Configuration

Property	Value
Platform	Windows-11-10.0.22631-SP0
Processor	AMD Ryzen 9 9900X
Physical CPU cores	12
Logical CPU threads	24
Total system memory	61.6 GB
Disk capacity used	931.4 GB
CUDA available	True
Compute type	GPU
GPU count	1
GPU model	NVIDIA GeForce RTX 5070 Ti,
GPU memory	16.0 GB

D.2 Libraries Used

We rely on the following open-source software (version, license, URL, citation):

- **PyTorch** (v2.6.0; BSD 3-clause License; https://pytorch.org) [Paszke et al., 2019]
- **torchvision** (v0.21.0; BSD 3-clause License; https://github.com/pytorch/vision) [PyTorch Contributors, 2023]
- **seaborn** (v0.13.2; BSD License; https://seaborn.pydata.org) [Waskom, 2021]
- **matplotlib** (v3.10.0; PSF License; https://matplotlib.org) [Hunter, 2007]
- **NumPy** (v2.1.3; BSD License; https://numpy.org) [van der Walt et al., 2011]
- **NovoGrad** optimizer (preprint; Apache-2.0; https://arxiv.org/abs/1905.11286) [Ginsburg et al., 2019]
- **pandas** (v2.2.3; BSD License; https://pandas.pydata.org) [McKinney, 2010]
- **JSON module** (Python 3.10 stdlib; PSF License; https://docs.python.org/3/library/json.html) [Crockford, 2006]
- **Optuna** (v4.3.0; MIT License; https://optuna.org) [Akiba et al., 2019]
- **scikit-learn** (v1.6.1; BSD License; https://scikit-learn.org) [Pedregosa et al., 2011]
- **SciPy** (v1.15.1; BSD License; https://scipy.org) [Virtanen et al., 2020]
- **itertools** (Python 3.10 stdlib; PSF License; https://docs.python.org/3/library/itertools.html) [Python Software Foundation, 2023]
- **Gymnasium** (v1.1.0; MIT License; https://gymnasium.farama.org) [Brockman et al., 2016]
- **Transformers** (v4.30.0; Apache-2.0; https://github.com/huggingface/transformers) [Wolf et al., 2020]
- **train-llm-from-scratch** (MIT License; https://github.com/FareedKhan-dev/train-llm-from-scratch) [FareedKhan-dev, 2025]
- **NovoGrad-pytorch** (MIT License; https://github.com/lonePatient/NovoGrad-pytorch) [lonePatient, 2019]

D.3 Datasets

We conduct all experiments on the following publicly released datasets:

- **MNIST** (Public Domain; http://yann.lecun.com/exdb/mnist/) [LeCun et al., 1998]
- **CIFAR-10** (2009 Technical Report; MIT License; https://www.cs.toronto.edu/~kriz/cifar.html) [Krizhevsky and Hinton, 2009]
- **CIFAR-100** (2009 Technical Report; MIT License; https://www.cs.toronto.edu/~kriz/cifar.html) [Krizhevsky and Hinton, 2009]
- **The Pile** (Apache 2.0 License; https://pile.eleuther.ai/) [Gao et al., 2020]

D.4 Error Bar Computation

Error bars in our experiments quantify the uncertainty in each metric across multiple runs. They are computed as follows:

1. Calculation

- **Standard Error of the Mean (SEM):**

$$\mathrm{SEM} = \frac{\mathrm{SD}}{\sqrt{n}}$$

  where SD is the sample standard deviation and n is the number of independent runs.

- **95% Confidence Interval (CI):**

$$\mathrm{CI} = t_{0.975,\,n-1} \times \mathrm{SEM},$$

  with $t_{0.975,\,n-1}$ the two-sided critical value from the Student's t-distribution.

- For each epoch or step, we aggregate the metric values over n runs, compute the mean, SD, SEM, and then plot the mean $\pm$ CI as error bands.

2. Sources of Variability

- *Run-to-run stochasticity*: random initialization, data shuffling.
- *Optimizer/model differences*: varying update rules or architectures.
- *Data subsampling*: different train/validation/test splits.
- *Hardware noise*: floating-point precision and device variability.

3. Assumptions

- **Normality**: metric distributions are approximately Gaussian.
- **Independence**: each run is statistically independent.
- **Sufficient sample size**: $n > 1$ for reliable SEM and CI estimates.
- **Consistency**: identical hyperparameters and data splits across runs.

4. Implementation The function `calculate_statistics()` computes mean, SD, SEM, and CI for each metric, and `plot_seaborn_style_with_error_bars()` renders the resulting error bands in the figures.

D.5 Experiment 1: Loss Map Predictions

The objective of this experiment is to evaluate multiple optimizers on different loss maps to provide an understanding of their behavior in complex loss environments. In each optimization problem, the optimizers were run 5 times and their outputs were averaged, however, since it is a static environment there was never any variation in their behavior.

D.5.1 Himmelblau Function and Its Gradient

The Himmelblau function is defined as:

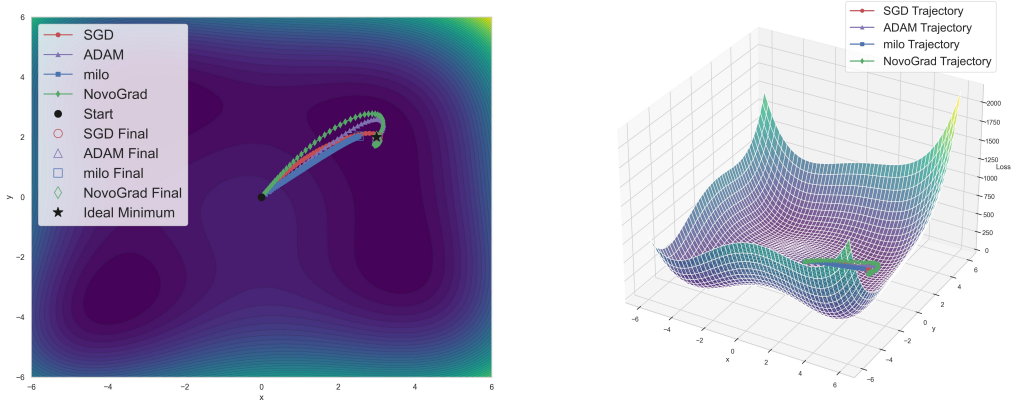
$$f(x_1, x_2) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2. \tag{D.1}$$

For the analytic gradient, we define:

$$u = x_1^2 + x_2 - 11, \qquad v = x_1 + x_2^2 - 7. \tag{D.2}$$

Then, the partial derivatives are:

$$\frac{\partial f}{\partial x_1} = 4x_1\, u + 2v, \qquad \frac{\partial f}{\partial x_2} = 2u + 4x_2\, v. \tag{D.3}$$



(a) Himmelblau Loss 2D (b) Himmelblau Loss 3D

Figure 6: Himmelblau Loss Function: 2D Contour vs. 3D Loss Map

Experimental Setup

- **Domain:** $[-6, 6] \times [-6, 6]$
- **Initial Point:** $(0, 0)$
- **Iterations:** 150

Observations and Analysis

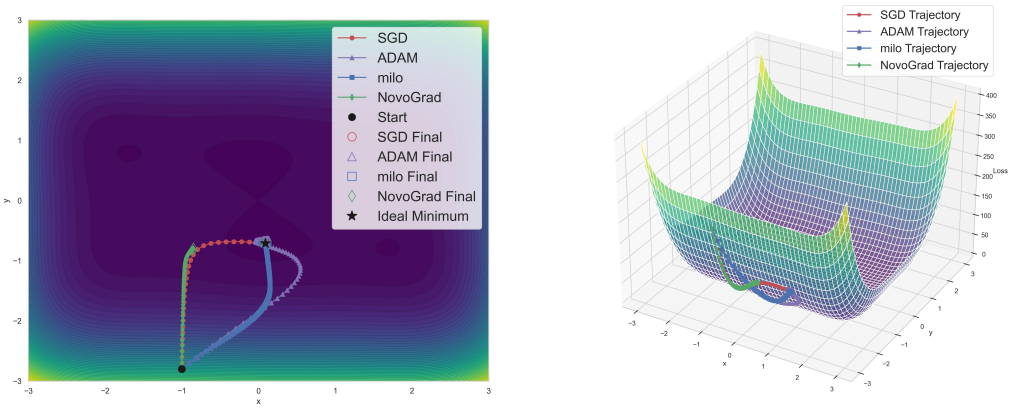
- **SGD:** The SGD trajectory follows a generally follows a more direct path, quickly descending into a low-loss valley, curving towards the end, and stabilizing near the minimum.
- **Adam:** Adam has a less direct path, overshooting and eventually curving around to reach the objective.
- **NovoGrad:** NovoGrad, despite taking normalized steps, overshoots with similar behavior to Adam, eventually curving around to reach the objective.
- **Milo:** Milo takes normalized steps, taking a more conservative path since the large gradients are scaled down, following a uniform trajectory toward the basin.

Key Takeaway: All four optimizers ultimately approach a global minimum, but SGD and Milo exhibit a more stable and direct convergence compared to Adam and NovoGrad.

D.5.2 Camel6 Function

The Camel6 function is defined by:

$$f(x_1, x_2) = \left(4 - 2.1\, x_1^2 + \frac{x_1^4}{3}\right) x_1^2 + x_1 x_2 + \left(-4 + 4\, x_2^2\right) x_2^2. \tag{D.4}$$



(a) Camel6 Loss 2D

(b) Camel6 Loss 3D

Figure 7: Camel6 Loss Function: 2D Contour vs. 3D Loss Map

Experimental Setup

- **Domain:** $[-3, 3] \times [-3, 3]$
- **Initial Point:** $(-1, -2.8)$
- **Iterations:** 150

Observations and Analysis

- **SGD:** The trajectory takes consistent steps, taking a largely indirect path to the local minima and eventually converging to the global.
- **Adam:** Similar to SGD Adam takes an indirect path to the local minima, then shifts to a smaller step size with a more direct path to the eventual global minima.
- **NovoGrad:** Novograd follows a similar path as SGD, however is comes to an early stop after it reaches the local minima, never reaching the global.

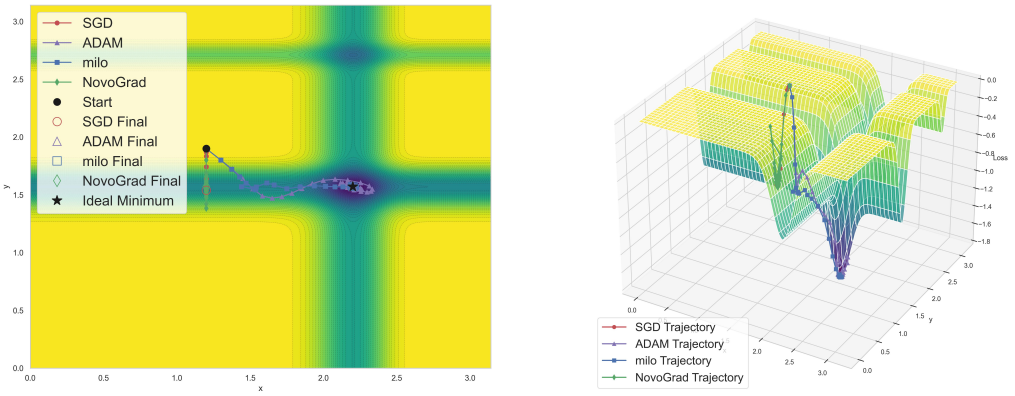
- **Milo:** With normalized updates, Milo takes consistent steps, initialy following the same path as Adam, but deviating towards the global minima for a more direct convergence than the other optimizers.

Key Takeaway: In the Camel6 landscape, Milo's consistent step sizes help it maintain stable progress even when facing plateaus, with SGD and Adam taking more indirect paths. NovoGrad never reaches convergence in this problem, instead settling in the local minima it quickly reached.

D.5.3 Michalewicz Function

For a d-dimensional input x and a parameter m (typically $m = 10$), the Michalewicz function is:

$$f(x) = -\sum_{i=1}^{d} \sin(x_i) \left[\sin\left(\frac{i\, x_i^2}{\pi}\right)\right]^{2m}. \tag{D.5}$$



(a) Michalewicz Loss 2D (b) Michalewicz Loss 3D

Figure 8: Michalewicz Loss Function: 2D Contour vs. 3D Loss Map

Experimental Setup

- **Domain:** $[-10, 10] \times [-10, 10]$
- **Initial Point:** $(2.4, -4.3)$
- **Iterations:** 150

Observations and Analysis

- **SGD:** For the Michalewicz experiment, SGD quickly became stuck in the local minima within the basin, never reaching the global.
- **Adam:** Adam's adaptive step size helps it overcome the local traps, reaching the global minima, however it takes a more indirect path and briefly overshoots the goal.
- **NovoGrad:** NovoGrad, similar to SGD, also becomes quickly stuck in the local minima within the basin.
- **Milo:** Milo successfully reaches the global minima, making consistent and direct progress towards the goal with no overshooting.

Key Takeaway: De Jong's function, with its many local minima, highlights Milo's strength in sustaining movement when the gradient is very small. Adam adapts well but again overshoots the goal before circling back. Additionally, it is shown that SGD and NovoGrad can become trapped in shallow basins.

D.5.4 De Jong Function 5

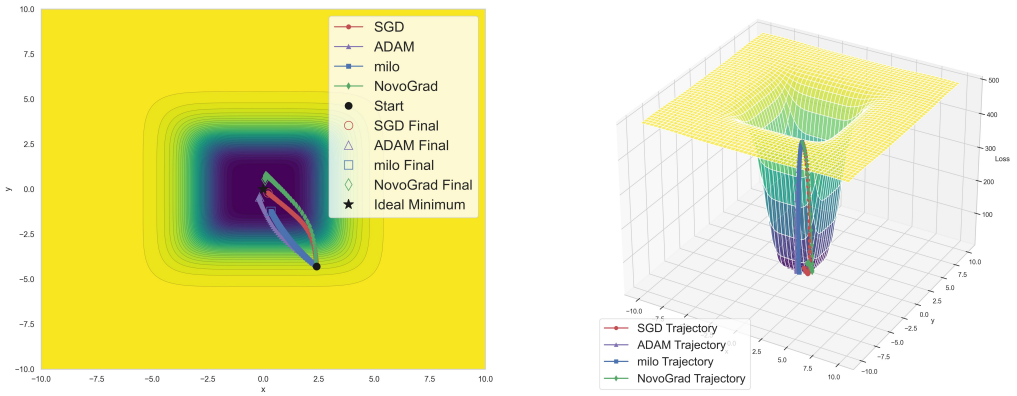
De Jong's fifth function is a multimodal test function defined for a 2-dimensional input vector $x = (x_1, x_2)$. It is constructed using a fixed grid of constants and is known for its complex landscape with many local minima. The function is given by:

$$f(x) = \left[0.002 + \sum_{i=1}^{25} \frac{1}{i + (x_1 - a_{1i})^6 + (x_2 - a_{2i})^6} \right]^{-1}, \tag{D.6}$$

where the constants a_{1i} and a_{2i} form a 5×5 grid of values from the set $\{-32, -16, 0, 16, 32\}$, specifically:

$$a_{1i} = \text{tile}([-32, -16, 0, 16, 32], 5),$$
$$a_{2i} = \text{repeat}([-32, -16, 0, 16, 32], 5).$$

This function is highly sensitive to small changes in x, making it useful for evaluating the performance of optimization algorithms on rugged search landscapes.



(a) DeJong Loss 2D

(b) DeJong Loss 3D

Figure 9: DeJong's Fifth Function Loss Function: 2D Contour vs. 3D Loss Map

Experimental Setup

- **Domain:** $[0, \pi] \times [0, \pi]$
- **Initial Point:** $(1.2, 1.9)$
- **Iterations:** 250

Observations and Analysis

- **SGD:** Due to the steep descent structure of the function, SGD curves around the edge until it corrects towards the global minima, reaching it.
- **Adam:** Adam's adaptive updates help it follow a smoother descent, reaching the final global minima smoothly.
- **NovoGrad:** NovoGrad follows a similar path around the edge as SGD, eventually overshooting slightly before reaching the global minima.
- **Milo:** Milo's normalized update rule maintains a consistent step size even when the gradients are small, allowing steady progress; however, it stops right as the bottom of the basin is reached, not continuing further as the other optimizers do.

Key Takeaway:

D.5.5 Experiment 1 Analysis

Optimizer Comparison

- **Early Stage Behavior:**
  - **SGD:** May take large initial steps but risks overshooting and stalling on plateaus.

  - **Adam:** Adapts the step size effectively, resulting in a rapid descent and smoother convergence, but also commonly overshoots.
  - **NovoGrade:** NovoGrad has similar behavior as Adam and SGD, however it displays issues with reaching the global minima when there are many local minima as well.
  - **Milo:** Uses normalized updates to maintain consistent step sizes, ensuring steady progress even when gradients are small.

- **Navigation of Plateaus and Local Minima:**
  - SGD may get trapped in local minima without additional momentum or careful scheduling.
  - Adam can sometimes reduce its step size too quickly, causing temporary plateaus.
  - NovoGrad performs well when in environments with few local minima, but shows issues with handling higher dimensionality functions.
  - Milo excels at escaping regions with vanishing gradients by enforcing a constant update magnitude.

- **Final Convergence and Stability:**
  - Adam and Milo tend to converge to lower loss values than SGD and NovoGrad.
  - Overall, Milo demonstrates both robust exploration in complex landscapes and stable convergence.

Practical Takeaways

- **Function-Specific Behavior:**
  - In multimodal functions (Himmelblau, Michalewicz), normalization helps maintain progress.
  - In functions with shallow basins (Michalewicz), Milo's constant step size aids in escaping local traps.

- **Optimizer Selection:**
  - No single optimizer is universally best. Milo offers a balanced approach when gradient scales vary widely and consistent progress is desired.

Overall the experiments reveal that while SGD, Adam, NovoGrad, and Milo each have their strengths, the proposed Milo optimizer, with its normalized, group-based updates, can offer stable and consistent progress in challenging optimization landscapes. In particular, Milo shows promise in escaping flat regions and avoiding overshooting due to its consistent step size. However, as shown with DeJong, the consistent step size can potentially lead to it stopping earlier then desired.

Table 4: Key Experimental Settings, Experiments 2 - 4

Setting	**Specification**
Data splits	Validation: 15%, Test: 10%
Batch size	128
Epochs	10
Runs per optimizer	5
Hyperparameter tuning	5 trials per optimizer using log-uniform and uniform sampling over specified parameter grids
Base learning rates	`LOGISTIC`, `MULTILAYER`: 0.05; `DEEPCNN`, `RESNET18`: 0.005
Optimizers	Milo, Milo_LW, SGD, AdaGrad, AdamW, NovoGrad
Models & Datasets	`LOGISTIC`, `MULTILAYER`: MNIST; `RESNET18`: CIFAR-100
Transforms	MNIST: flatten to vector; CIFAR: tensor only (no augmentations)

D.6 Hyperparameter Tuning:

Table 5: Hyperparameter search spaces for all optimizers

Optimizer	Search Space
SGD	`lr`: log-uniform $[0.005, 0.1]$; `momentum`: uniform $[0.45, 0.99]$; `weight_decay`: log-uniform $[0.0005, 0.01]$
ADAGRAD	`lr`: log-uniform $[0.005, 0.1]$; `lr_decay`: uniform $[0.0, 0.1]$; `weight_decay`: log-uniform $[0.0005, 0.01]$; `eps`: log-uniform $[10^{-10}, 10^{-6}]$
ADAMW	`lr`: log-uniform $[0.005, 0.1]$; `weight_decay`: log-uniform $[0.0005, 0.01]$; `betas`: $\{(0.9, 0.98), (0.95, 0.99)\}$; `eps`: log-uniform $[10^{-9}, 10^{-6}]$
NovoGrad	`lr`: log-uniform $[0.005, 0.1]$; `betas`: $\{(0.9, 0.98), (0.95, 0.99)\}$; `weight_decay`: log-uniform $[0.0005, 0.01]$
Milo	– (default settings)
Milo_{LW}	– (default settings)

Table 6: Default learning rates and optimizer parameter settings

Component	Default Setting
Base learning rate	
LOGISTIC	0.05
MULTILAYER	0.05
DEEPCNN	0.005
RESNET18	0.005
Optimizer defaults	
SGD	`momentum`=0.9; `nesterov`=True; `weight_decay`=0.0001
ADAGRAD	`lr_decay`=0; `weight_decay`=0.0; `eps`=1e-10
ADAMW	`betas`=(0.9, 0.999); `eps`=1e-8; `weight_decay`=0.01
Milo	`layer_wise`=False; `scale_aware`=True; `scale_factor`=0.2; `momentum`=0.9; `use_cached_mapping`=False; `foreach`=True
Milo_{LW}	`layer_wise`=True; `scale_aware`=True; `scale_factor`=0.2; `momentum`=0.9; `use_cached_mapping`=True; `foreach`=True
NovoGrad	`betas`=(0.9, 0.99); `weight_decay`=0.001; `grad_averaging`=True

D.7 Experiment 2: Logistic Regression

This experiment evaluates a convex multi-class logistic regression using the MNIST Dataset ([LeCun et al., 1998]), just as was done in the study that proposed Adam. The goal is to compare optimizers with respect to training loss, convergence speed, and test accuracy. The model uses L2 regularization with a single linear layer mapping a 784-dimensional input to 10 classes. Visual results can be seen in Figure 3.

Table 7: Logistic Regression Model Architecture Layers

Layer	Description
Input Layer	Flattened input vector of dimension 784 (28×28)
Linear	Single fully connected layer mapping 784 to 10 (number of classes)

Table 8: Logistic Regression Training Results

Optimizer	Epoch Group	Loss	Accuracy (%)	F1 Score
Milo	1–4	0.2930	92.42	0.9233
	5–8	0.2732	93.07	0.9298
	9–10	0.2714	93.19	0.9310
Milo_{LW}	1–4	0.2594	92.87	0.9277
	5–8	0.2312	93.62	0.9354
	9–10	0.2258	93.75	0.9367
SGD	1–4	0.3076	91.47	0.9135
	5–8	0.2818	92.23	0.9212
	9–10	0.2773	92.34	0.9224
ADAGRAD	1–4	0.4866	88.03	0.8784
	5–8	0.4720	88.29	0.8810
	9–10	0.4687	88.33	0.8814
ADAMW	1–4	0.3103	91.48	0.9135
	5–8	0.2593	92.83	0.9273
	9–10	0.2490	93.18	0.9309
Novograd	1–4	0.3774	89.87	0.8973
	5–8	0.2928	91.83	0.9172
	9–10	0.2827	92.13	0.9202

Table 9: Average Runtime and Memory Increase for Logistic Regression Experiment (5 runs)

Optimizer	Avg Runtime (s)	Avg Memory (MB)
MILO	32.51	63.5
MILO_{LW}	32.66	2.0
SGD	29.71	0.0
ADAGRAD	29.76	0.0
ADAMW	30.03	16.4
NOVOGRAD	32.16	4.1

D.7.1 Statistical Analysis

Table 10: Pairwise Significance Tests for Validation Loss

Optimizer A	Optimizer B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	0.350833	0.304008	MILO_{LW}	8.13499e-08	***
MILO	SGD	0.350833	0.306714	SGD	3.62048e-06	***
MILO	ADAMW	0.350833	0.287661	ADAMW	9.31980e-07	***
MILO	ADAGRAD	0.350833	0.481010	MILO	9.90140e-11	***
MILO	NOVOGRAD	0.350833	0.309194	NOVOGRAD	4.23780e-06	***
MILO_{LW}	SGD	0.304008	0.306714	MILO_{LW}	1.32372e-01	
MILO_{LW}	ADAMW	0.304008	0.287661	ADAMW	9.06983e-05	***
MILO_{LW}	ADAGRAD	0.304008	0.481010	MILO_{LW}	2.49227e-13	***
MILO_{LW}	NOVOGRAD	0.304008	0.309194	MILO_{LW}	1.67816e-02	*
SGD	ADAMW	0.306714	0.287661	ADAMW	1.66218e-08	***
SGD	ADAGRAD	0.306714	0.481010	SGD	2.60849e-11	***
SGD	NOVOGRAD	0.306714	0.309194	SGD	2.46024e-02	*
ADAMW	ADAGRAD	0.287661	0.481010	ADAMW	4.96532e-11	***
ADAMW	NOVOGRAD	0.287661	0.309194	ADAMW	9.13035e-09	***
ADAGRAD	NOVOGRAD	0.481010	0.309194	NOVOGRAD	1.79728e-11	***

Table 11: Pairwise Significance Tests for Validation F1-Score

Optimizer A	Optimizer B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	0.912861	0.916955	MILO_{LW}	1.75279e-03	**
MILO	SGD	0.912861	0.914048	SGD	2.41945e-01	
MILO	ADAGRAD	0.912861	0.876378	MILO	5.42161e-10	***
MILO	ADAMW	0.912861	0.919612	ADAMW	5.82021e-05	***
MILO	NOVOGRAD	0.912861	0.913075	NOVOGRAD	7.63541e-01	
MILO_{LW}	SGD	0.916955	0.914048	MILO_{LW}	1.56224e-02	*
MILO_{LW}	ADAGRAD	0.916955	0.876378	MILO_{LW}	2.24596e-10	***
MILO_{LW}	ADAMW	0.916955	0.919612	ADAMW	9.94675e-03	**
MILO_{LW}	NOVOGRAD	0.916955	0.913075	MILO_{LW}	1.93199e-03	**
SGD	ADAGRAD	0.914048	0.876378	SGD	3.79011e-10	***
SGD	ADAMW	0.914048	0.919612	ADAMW	4.33298e-04	***
SGD	NOVOGRAD	0.914048	0.913075	SGD	2.51483e-01	
ADAGRAD	ADAMW	0.876378	0.919612	ADAMW	8.47333e-09	***
ADAGRAD	NOVOGRAD	0.876378	0.913075	NOVOGRAD	9.06177e-08	***
ADAMW	NOVOGRAD	0.919612	0.913075	ADAMW	3.71358e-05	***

D.8 Experiment 3: Multilayer Neural Networks

The objective of this experiment is to test the optimizer on a non-convex problem by using a multilayer perception. For this study, our model choice was driven by the previous Adam publication, using two fully connected layers with 1000 ReLU units each, with dropout and weight decay on a minibatch of 128, on the MNIST image dataset ([LeCun et al., 1998]).

Table 12: Multilayer NN Model Architecture Layers

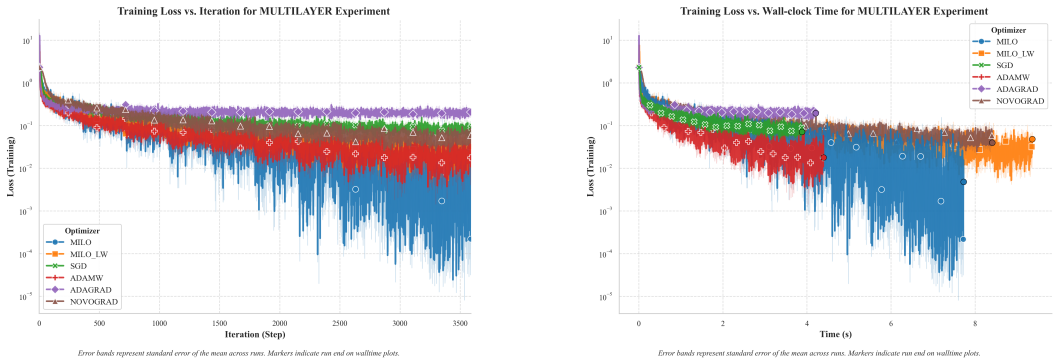
Layer	**Description**
Linear	Fully connected layer: input dimension 784 to hidden dimension 128
ReLU	Activation function
Dropout	Dropout layer with probability 0.2
Linear	Fully connected layer: 128 to 10 (number of classes)

Table 13: Multilayer Network Training Results

Optimizer	Epoch Group	Loss	Accuracy (%)	F1 Score
Milo	1–4	0.0777	98.24	0.9822
	5–8	0.0161	99.66	0.9966
	9–10	0.0050	99.90	0.9990
Milo_{LW}	1–4	0.1034	96.95	0.9692
	5–8	0.0372	98.88	0.9887
	9–10	0.0215	99.36	0.9936
SGD	1–4	0.1695	95.32	0.9528
	5–8	0.0921	97.63	0.9762
	9–10	0.0756	98.16	0.9815
ADAGRAD	1–4	0.2204	93.98	0.9391
	5–8	0.2013	94.53	0.9447
	9–10	0.1968	94.66	0.9461
ADAMW	1–4	0.0791	97.58	0.9756
	5–8	0.0259	99.14	0.9913
	9–10	0.0144	99.51	0.9951
Novograd	1–4	0.1655	95.24	0.9519
	5–8	0.0637	98.18	0.9817
	9–10	0.0461	98.72	0.9871

Table 14: Average Runtime and Memory Increase for Multilayer Experiment (5 runs)

Optimizer	Avg Duration (s)	Avg Memory (MB)
MILO	34.37	18.4
MILO_{LW}	36.05	0.0
SGD	30.28	0.0
ADAGRAD	30.74	0.0
ADAMW	30.87	0.0
NOVOGRAD	34.24	0.0



(a) MLP Train Loss

(b) MLP Train Walltime Loss

Figure 10: Comparison of MLP training cost by iterations and wall time.

D.8.1 Statistical Analysis

Table 15: Pairwise Significance Tests for Validation Loss (Final Epoch)

Optimizer A	Optimizer B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	0.225943	0.150747	MILO_{LW}	1.51986e-04	***
MILO	SGD	0.225943	0.104657	SGD	1.10944e-04	***
MILO	ADAMW	0.225943	0.121495	ADAMW	2.48203e-05	***
MILO	ADAGRAD	0.225943	0.215592	ADAGRAD	3.16927e-01	
MILO	NOVOGRAD	0.225943	0.0872035	NOVOGRAD	2.96707e-05	***
MILO_{LW}	SGD	0.150747	0.104657	SGD	4.70080e-04	***
MILO_{LW}	ADAMW	0.150747	0.121495	ADAMW	1.64005e-03	**
MILO_{LW}	ADAGRAD	0.150747	0.215592	MILO_{LW}	1.31936e-05	***
MILO_{LW}	NOVOGRAD	0.150747	0.0872035	NOVOGRAD	2.06945e-05	***
SGD	ADAMW	0.104657	0.121495	SGD	1.71513e-02	*
SGD	ADAGRAD	0.104657	0.215592	SGD	2.16145e-05	***
SGD	NOVOGRAD	0.104657	0.0872035	NOVOGRAD	9.02238e-04	***
ADAMW	ADAGRAD	0.121495	0.215592	ADAMW	7.75432e-07	***
ADAMW	NOVOGRAD	0.121495	0.0872035	NOVOGRAD	4.28650e-04	***
ADAGRAD	NOVOGRAD	0.215592	0.0872035	NOVOGRAD	1.16408e-06	***

Table 16: Pairwise Significance Tests for Validation F1-Score (Final Epoch)

Optimizer A	Optimizer B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	0.973116	0.968024	MILO	9.10593e-04	***
MILO	SGD	0.973116	0.971065	MILO	5.32615e-02	
MILO	ADAMW	0.973116	0.972800	MILO	8.14991e-01	
MILO	ADAGRAD	0.973116	0.941102	MILO	6.55427e-06	***
MILO	NOVOGRAD	0.973116	0.973147	NOVOGRAD	9.80252e-01	
MILO_{LW}	SGD	0.968024	0.971065	SGD	1.56224e-02	**
MILO_{LW}	ADAMW	0.968024	0.972800	ADAMW	6.94322e-03	**
MILO_{LW}	ADAGRAD	0.968024	0.941102	MILO_{LW}	3.03333e-05	***
MILO_{LW}	NOVOGRAD	0.968024	0.973147	NOVOGRAD	2.48226e-03	**
SGD	ADAMW	0.971065	0.972800	ADAMW	1.83324e-01	
SGD	ADAGRAD	0.971065	0.941102	SGD	3.85788e-05	***
SGD	NOVOGRAD	0.971065	0.973147	NOVOGRAD	8.87216e-02	
ADAMW	ADAGRAD	0.972800	0.941102	ADAMW	1.98429e-06	***
ADAMW	NOVOGRAD	0.972800	0.973147	NOVOGRAD	8.11459e-01	
ADAGRAD	NOVOGRAD	0.941102	0.973147	NOVOGRAD	3.13313e-06	***

D.9 Experiment 4: Convolutional Neural Networks (CNNs)

This experiment evaluates a deep Residual Network, specifically ResNet-18 [He et al., 2016], on the CIFAR-100 dataset [Krizhevsky and Hinton, 2009] to test optimizer performance in a finer-grained classification setting. Compared to CIFAR-10, CIFAR-100 has 100 target classes, which stresses both the representation capacity of the network and the optimizer's ability to generalize. We use the standard ResNet-18 architecture, modifying only the final linear layer to output 100 logits. Input images are augmented with 4-pixel zero-padding, then per-channel normalized to zero mean and unit variance. All training runs use a batch size of 128.

Table 17: Deep CNN Model Architecture Layers

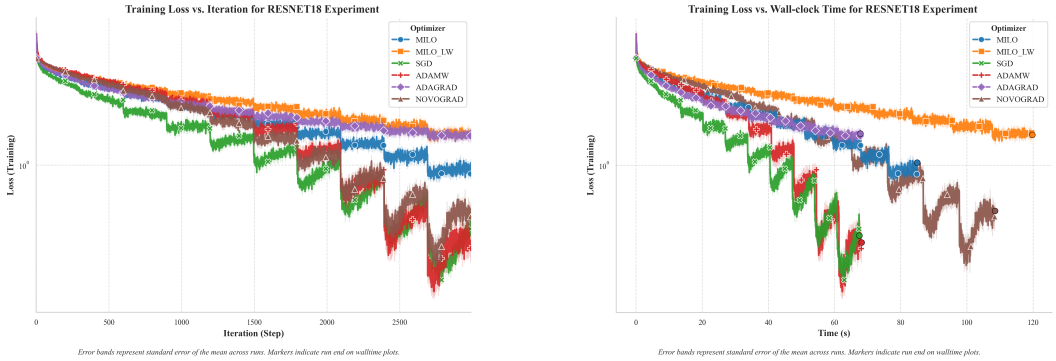
Layer	Description
Conv2d	Input: 3 channels, Output: 32 channels, Kernel size: 3, Padding: 1
ReLU	Activation function
Conv2d	Input: 32 channels, Output: 32 channels, Kernel size: 3, Padding: 1
ReLU	Activation function
MaxPool2d	Pooling layer with kernel size 2
Conv2d	Input: 32 channels, Output: 64 channels, Kernel size: 3, Padding: 1
ReLU	Activation function
Conv2d	Input: 64 channels, Output: 64 channels, Kernel size: 3, Padding: 1
ReLU	Activation function
MaxPool2d	Pooling layer with kernel size 2
Flatten	Converts feature maps into a 1D vector
Linear	Fully connected layer: 4096 (i.e. $64 \times 8 \times 8$) to 128
ReLU	Activation function
Linear	Fully connected layer: 128 to 10 (number of classes)

Table 18: ResNet-18 Training Results

Optimizer	Epoch Group	Loss	Accuracy (%)	F1 Score
Milo	1–4	2.6884	33.03	0.3130
	5–8	1.4598	61.59	0.6113
	9–10	0.7754	80.63	0.8064
Milo_{LW}	1–4	3.0691	23.40	0.2103
	5–8	2.0448	44.88	0.4389
	9–10	1.5241	57.17	0.5651
SGD	1–4	2.6914	34.53	0.3335
	5–8	1.2595	63.90	0.6395
	9–10	0.6575	81.21	0.8144
ADAGRAD	1–4	2.7093	32.94	0.3112
	5–8	1.7878	55.53	0.5470
	9–10	1.4423	65.83	0.6533
ADAMW	1–4	3.2084	22.69	0.2104
	5–8	1.1366	68.65	0.6847
	9–10	0.2509	92.53	0.9258
NOVOGRAD	1–4	3.1196	24.28	0.2205
	5–8	1.1359	66.98	0.6673
	9–10	0.3902	87.82	0.8780

Table 19: Average Runtime and Memory Increase for ResNet-18 Experiment (5 runs)

Optimizer	Avg Duration (s)	Avg Memory (MB)
MILO	148.53	59.4
MILO_{LW}	182.82	2.0
SGD	130.56	14.3
ADAGRAD	130.69	0.0
ADAMW	131.10	14.3
NOVOGRAD	172.25	4.1



(a) ResNet18 Train Loss

(b) ResNet18 Walltime Loss

Figure 11: Comparison of ResNet18 training cost by iterations and wall time.

D.9.1 Statistical Analysis

Table 20: Pairwise Significance Tests for Validation Loss

Opt. A	Opt. B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	2.09638	2.11451	MILO	0.428833	
MILO	SGD	2.09638	2.16893	MILO	0.044138	*
MILO	ADAGRAD	2.09638	2.48136	MILO	4.41550e-07	***
MILO	ADAMW	2.09638	3.38785	MILO	1.17780e-06	***
MILO	NOVOGRAD	2.09638	2.29087	MILO	0.000535	***
MILO_{LW}	SGD	2.11451	2.16893	MILO_{LW}	0.081725	
MILO_{LW}	ADAGRAD	2.11451	2.48136	MILO_{LW}	1.24390e-06	***
MILO_{LW}	ADAMW	2.11451	3.38785	MILO_{LW}	6.11940e-06	***
MILO_{LW}	NOVOGRAD	2.11451	2.29087	MILO_{LW}	0.001571	**
SGD	ADAGRAD	2.16893	2.48136	SGD	9.18010e-06	***
SGD	ADAMW	2.16893	3.38785	SGD	6.27030e-07	***
SGD	NOVOGRAD	2.16893	2.29087	SGD	0.009368	**
ADAGRAD	ADAMW	2.48136	3.38785	ADAGRAD	8.92350e-06	***
ADAGRAD	NOVOGRAD	2.48136	2.29087	NOVOGRAD	0.000613	***
ADAMW	NOVOGRAD	3.38785	2.29087	NOVOGRAD	6.09120e-07	***

Table 21: Pairwise Significance Tests for Validation F1-Score

Opt. A	Opt. B	Mean A	Mean B	Better	p-value	Sig.
MILO	MILO_{LW}	0.481343	0.441000	MILO	0.000321	***
MILO	SGD	0.481343	0.454009	MILO	0.003989	**
MILO	ADAGRAD	0.481343	0.365256	MILO	1.58180e-07	***
MILO	ADAMW	0.481343	0.411299	MILO	0.000352	***
MILO	NOVOGRAD	0.481343	0.492262	NOVOGRAD	0.113014	
MILO_{LW}	SGD	0.441000	0.454009	SGD	0.050683	
MILO_{LW}	ADAGRAD	0.441000	0.365256	MILO_{LW}	1.69300e-06	***
MILO_{LW}	ADAMW	0.441000	0.411299	MILO_{LW}	0.024678	*
MILO_{LW}	NOVOGRAD	0.441000	0.492262	NOVOGRAD	3.15430e-06	***
SGD	ADAGRAD	0.454009	0.365256	SGD	6.67760e-07	***
SGD	ADAMW	0.454009	0.411299	SGD	0.005071	**
SGD	NOVOGRAD	0.454009	0.492262	NOVOGRAD	0.000213	***
ADAGRAD	ADAMW	0.365256	0.411299	ADAMW	0.003599	**
ADAGRAD	NOVOGRAD	0.365256	0.492262	NOVOGRAD	4.01170e-08	***
ADAMW	NOVOGRAD	0.411299	0.492262	NOVOGRAD	0.000330	***

D.10 Experiment 5: Reinforcement Learning

Reinforcement Learning (RL) has become increasingly vital across various domains in recent years. RL enables agents to learn optimal behavior across a variety of environments through trial and error. This results in agents being more adaptable than traditional methods in complex environments. It plays a role in autonomous systems, sequential decision-making without supervision, and LLMS through Reinforcement Learning from Human Feedback (RLHF). Due to these growing uses, this study explores the performance of our optimizer in a three-dimensional space with stable consistent points, and randomly generated points. The purpose of this experiment is to study the optimizer's influence on reward-based actions and its adaptability. This experiment involved simulating a boxed environment, providing the agent with the observation of its current location and goal, and simulating 1000 episodes with 500 steps each. Each agent was trained 5 individual times, given 5 runs per training regimen, resulting in the final results being the averaged prediction of each agent across 25 runs, with the success rate being the rate the agent correctly reached the goal. The following results were observed.

Table 22: RL Experiment Training Parameters

Parameter	**Value**
Learning Rate	0.0005
Gamma (Discount Factor)	0.99
Episodes	1000
Log Interval	250 episodes
Initial Exploration	0.2
Final Exploration	0.01
Exploration Decay	0.998
Gradient Clip	0.5
Distance Threshold	0.05
Random Goal	True
Max Steps per Episode	500
Start Position	[0.0, 0.0, 0.0]
Goal Position	[1, 1, 1]

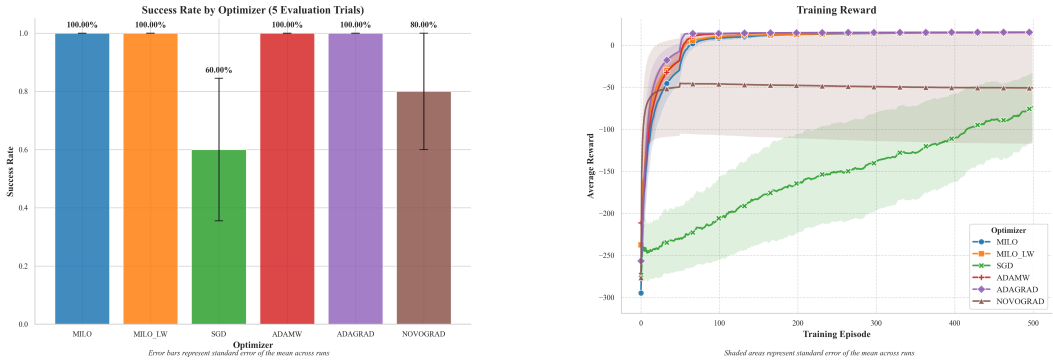
Table 23: Policy Network Architecture

Component	**Description**
Input Embedding	A linear layer mapping the input (observation) to 256 dimensions, followed by LayerNorm and ReLU activation.
Residual Blocks	4 residual blocks; each block consists of: • Linear ($256 \rightarrow 128$) • LayerNorm + ReLU • Dropout (0.1) • Linear ($128 \rightarrow 256$) • LayerNorm • Skip connection
Output Layers	A sequential bottleneck: Linear ($256 \rightarrow 128$) + LayerNorm + ReLU, Linear ($128 \rightarrow 64$) + LayerNorm + ReLU, and Linear ($64 \rightarrow$ output dimension), with final outputs scaled via `tanh` (multiplied by 0.1).

D.10.1 Set Result:

(a) 3D Agent Tractories (b) 2D Agent Tractories

Figure 12: 3D and 2D visuals of set RL agent trajectories.

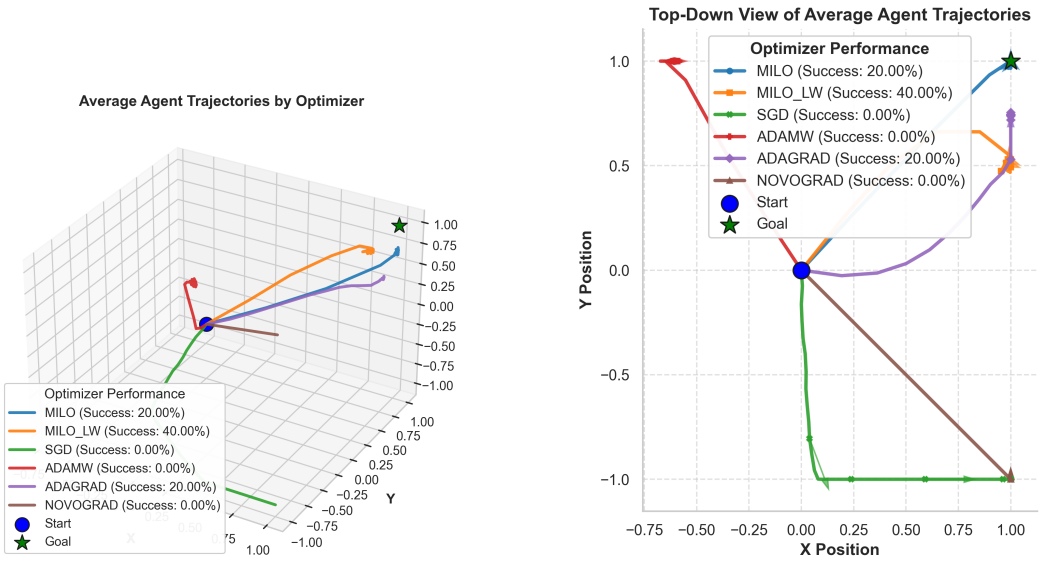


(a) RL Set Success Rates (b) Reward Comparision

Figure 13: Comparison of set success rates by optimizer and average reward by episode.

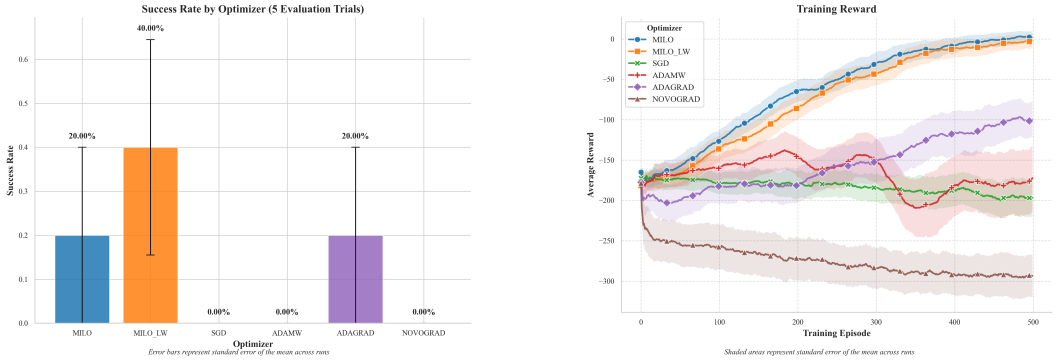
As shown in Figure 12 and Figure 13, each of the optimizers has a high success rate of reaching the set goal with no random initialization or generation of the goal. The optimizers with the highest performance proved to be AdamW, AdaGrad, Milo, and Milo_{LW}, each achieving a 100% accuracy. Overall, the set experiment demonstrates that each of the models can effectively learn the patterns generated in this 3D space.

D.10.2 Randomized Result:



(a) 3D Agent Tractories (b) 2D Agent Tractories

Figure 14: 3D and 2D visuals of set RL agent trajectories.



(a) RL Randomized Success Rates (b) Reward Comparison

Figure 15: Comparison of randomized success rates by optimizer and average reward by episode.

The results of the randomly generated start and goal differ drastically from the set, with each optimizer performing worse than before. However, each optimizer retains the relative performance shown previously regarding their performance against one another. As shown in Figure 14a, Figure 14b, and Figure 15a, Milo_{LW} has the highest accuracy, with a final success rate of 40%, being a large improvement over the success rates of the other optimizers. This performance is also shown in Figure 15b, with Milo_{LW} demonstrating a higher reward than the other optimizers.

D.11 Experiment 6: Large Language Model

LLMs have grown significantly in popularity in recent years, with many innovations made in model architecture and optimization. This study explores the performance of our optimizer in a pretraining setting, using a small-scale Transformer model trained from scratch on a processed subset of The Pile dataset [Gao et al., 2020]. The model weights are randomly initialized at the start of training, and no pre-trained parameters are used at any stage. This setup allows us to evaluate optimizer behavior in a controlled language modeling task, where both model capacity and training steps are deliberately constrained to enable rapid comparison across runs.

To support this investigation, we adapt the open-source training framework of FareedKhan-dev (2025), which implements a simplified version of the Transformer architecture introduced by Vaswani et al. (2017). This architecture includes core components such as multi-head self-attention, layer normalization, and position-wise feed-forward networks, implemented natively in PyTorch. We extend the original implementation to support multiple optimizers and perform controlled training and evaluation across several runs. Visual results can be seen in Figure 4.

Table 24: Training Parameters

Parameter	Value	Description
VOCAB_SIZE	50304	Number of unique tokens in the vocabulary.
CONTEXT_LENGTH	64	Maximum sequence length for the model.
N_EMBED	64	Dimension of the embedding space.
N_HEAD	4	Number of attention heads in each transformer block.
N_BLOCKS	1	Number of transformer blocks in the model.
T_BATCH_SIZE	32	Number of samples per training batch.
T_CONTEXT_LENGTH	16	Context length for training batches.
T_TRAIN_STEPS	3000	Total number of training steps.
T_EVAL_STEPS	500	Frequency (in steps) for evaluation.
T_EVAL_ITERS	125	Number of iterations for evaluation.
T_LR_DECAY_STEP	1500	Step at which the learning rate decays.
T_LR	0.05	Initial learning rate for training.
T_LR_DECAYED	0.005	Learning rate after decay.
DEVICE	cuda	Device used for training.

Table 25: Model Architecture Layers

Layer	Description
Token Embedding	Converts input token indices into embeddings of dimension N_EMBED.
Positional Embedding	Adds positional information to the token embeddings for a maximum sequence length of CONTEXT_LENGTH.
Transformer Block	Comprises a multi-head self-attention mechanism (with N_HEAD heads) and a feed-forward network, incorporating residual connections and layer normalization.
Final Linear Projection	Projects the transformer output to logits over the vocabulary (transforming from N_EMBED to VOCAB_SIZE).

Table 26: Transformer Experiment Configuration

Component	Details
Model	Transformer
Vocabulary Size	50304
Embedding Dimension	64
Attention Heads	4
Transformer Blocks	1
Training Batch Size	32
Training Context Length	16
Total Training Steps	3000
Evaluation Frequency	Every 500 steps (125 iterations)
Learning Rate	Initial: 0.05, decays to 0.005 at step 1500

The results in this experiment, as shown in Figure 4a and Figure 4b, demonstrate that Milo and Milo$_{\text{LW}}$ demonstrate comparable performance to some of the industry standard used for training LLMs such as Adam and AdamW, with Milo$_{\text{LW}}$ and Milo achieving the lowest recorded loss and fastest reduction of loss as well. Additionally, since this was a more complex task, the wall time of each optimizer was logged as well, with Milo being comparable to Adam and AdamW, whereas Milo$_{\text{LW}}$ and NovoGrad took a longer training time.

D.12 Experiment 7: Ablation Study

To assess the contributions of Milo's design choices, we conduct a comprehensive ablation study across three model architectures (MLP, DeepCNN, ResNet-18) and two datasets (MNIST, CIFAR-10). Our goals are to isolate the effects of key hyperparameters—group size, learning rate, clipping norm, and scale-aware blending—and evaluate their impact on training stability, convergence speed, and sensitivity to hyperparameter selection.

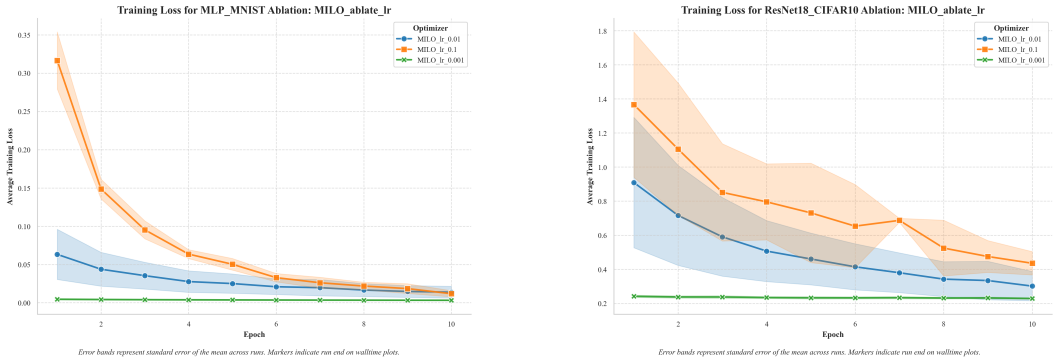
Experimental Setup. We evaluate two optimizer variants: standard Milo and its layer-wise variant Milo_{LW}. For each model–dataset pair, we establish a baseline configuration (see `config.py`) and systematically vary one hyperparameter at a time. Parameters under test include:

- **Learning Rate (`lr`):** {0.0001, 0.001, 0.005, 0.01}
- **Momentum (`momentum`):** {0.0, 0.5, 0.9}
- **Weight Decay (`weight_decay`):** {0, 1e-4, 1e-3}
- **Clipping Norm (`clip_norm`):** {None, 1.0, 5.0}
- **Epsilon (`eps`):** {1e-8, 1e-5, 1e-2}
- **Scale-Aware Blending (`scale_aware`, `scale_factor`):** {True, False}, {0.1,0.2,0.5} (Milo_{LW} only)

Each configuration is run for RUNS_PER_OPTIMIZER independent trials to ensure statistical robustness. Results are recorded under structured directories and exported as CSV metrics and loss/accuracy curves.

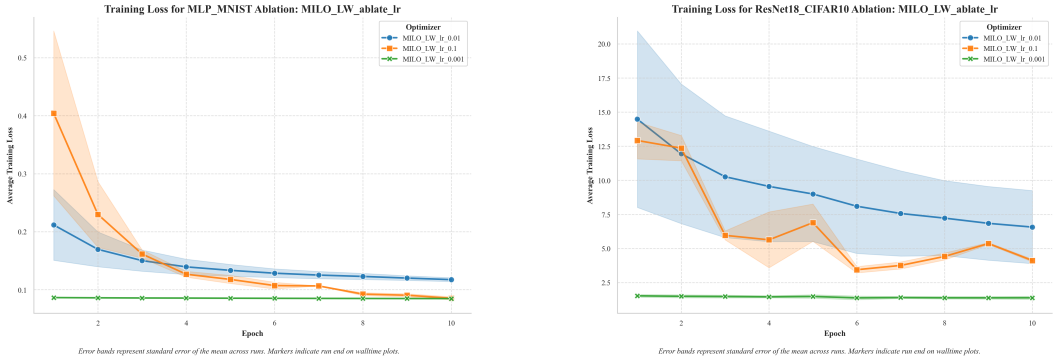
Metrics. We measure:

- *Convergence Speed*: epochs to reach fixed loss threshold.
- *Stability*: variance in final loss and accuracy across seeds.
- *Hyperparameter Sensitivity*: performance variation relative to baseline.



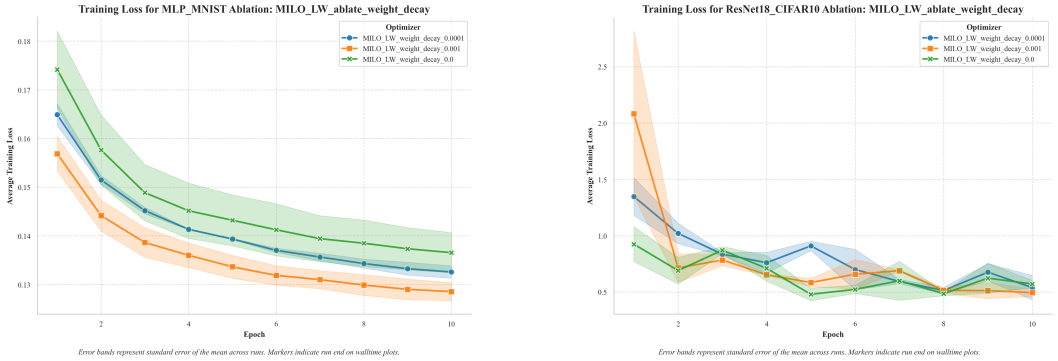
(a) Milo MLP Train Loss (b) Milo ResNet-18 Train Loss

Figure 16: Comparison of Milo LR Ablation in MLP and ResNet-18.



(a) Milo MLP Train Loss (b) Milo ResNet-18 Train Loss

Figure 17: Comparison of Milo_LW LR Ablation in MLP and ResNet-18.



(a) Milo MLP Train Loss

(b) Milo ResNet-18 Train Loss

Figure 18: Comparison of Milo_LW Weight Decay Ablation in MLP and ResNet-18.

Key Findings.

- **Learning Rate** remains the most critical factor: optimal values vary by task and model, with Milo and Milo_{LW} exhibiting similar sensitivity profiles.
- **Weight Decay** and **Epsilon** show moderate, task-dependent effects; safe defaults (0 and 10^{-5}) yield robust performance.
- **Momentum** and **Clipping Norm** have marginal impact on final accuracy but can improve early stability.
- **Scale-Aware Blending**: Contrary to our hypothesis, blending (Milo_{LW}) did not consistently outperform standard Milo. In MLP/MNIST and DeepCNN/CIFAR-10, the best Milo_{LW} (scale_factor=0.5) still trailed standard Milo by 0.2–0.5% accuracy.

Implications. These results suggest that Milo's group normalization and adaptive components are robust across a wide range of hyperparameters, reducing the tuning burden relative to conventional optimizers. However, the scale-aware blending mechanism may require further refinement or alternative blending strategies to realize its theoretical benefits in practice.