Efficient Optimization, Post Training and Inference

Jiawei Zhang

October 16, 2025

WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

Outline

- Optimal First-Order Algorithm for Stochastic Constrained Optimization
- Understanding GRPO as Adaptive Gradient Descent
- Efficient Lossless Inference of Diffusion Language Model

Background

- Consider a generic optimization problem:

$$\min_{x \in \mathbb{R}^d} f(x).$$

- Suppose f is bounded below and ℓ-Lipschitz smooth, i.e.,

$$\|\nabla f(x) - \nabla f(x')\| \leq \ell \|x - x'\|.$$

- Two simple but powerful algorithms: gradient descent and stochastic gradient descent:
    - **GD**: $x^{t+1} = x^t - c_t \nabla f(x^t)$
    - **SGD**: $x^{t+1} = x^t - c_t g^t$, where g^t is an unbiased estimator of $\nabla f(x^t)$ with variance σ.
    - First-order algorithms: well-suited for large-scale problems, simple steps, especially SGD using small batch.

Iteration and Sample Complexity

- For nonconvex problems, Iteration complexity is the number of iterations needed to obtain an ϵ-stationary solution:
  - Deterministic case: $\|\nabla f(x)\| \leq \epsilon$;
  - Stochastic case: $\mathbb{E}\|\nabla f(x)\| \leq \epsilon$
- For GD, we use $c_t = 1/\ell$ and iteration complexity is $\mathcal{O}(\ell/\epsilon^2)$.
- For SGD, we choose $c_t = \epsilon^2/\ell\sigma^2$ and the sample and iteration complexity are $\mathcal{O}(\ell\sigma^2/\epsilon^4)$.

Natural Open Problems

- We have some natural open questions:
- For constrained optimization problems, can we still achieve $\mathcal{O}(1/\epsilon^2)$ iteration complexity using first-order methods?
- For stochastic constrained optimization, can we still achieve $\mathcal{O}(1/\epsilon^4)$ sample and iteration complexity?
- Suppose the Lipschitz smoothness constants are different x, can we use adaptive stepsize to accelerate?
- These questions are just very natural extension of textbook results.

My Previous Works for Deterministic Constrained Optimization

- Consider constrained optimization

$$\min_{x \in X, Ax=b} f(x)$$

  where X is a convex, closed set and easy to project,
- We can design a first-order algorithm, only requiring gradient iteration and projection to X with $\mathcal{O}(1/\epsilon^2)$ iteration complexity.
- This is shown in [Zhang and Luo 2020, proximal],[Zhang and Luo, 2022, global] and [Zhang et al, 2022, iteration]
- I further work on the problem $\min_{x \in X, g_i(x) \le 0} f(x)$, where $g_i, i \in [m]$ is convex, X is convex, closed and easy to project.
- We show it can be solved by first-order method with $\mathcal{O}(1/\epsilon^2)$ iteration complexity, only requiring gradient iteration and projection on X.

Stochastic Constrained Optimization

- In the first part of this talk, we design first-order method for stochastic constrained optimization
- We show the optimal sample complexity.
- We also discuss the optimal sample complexity under a variance reduction technique.

Stochastic Smoothed Primal-Dual Algorithms for Nonconvex Optimization with Linear Inequality Constraints

Ruichuan Huang, Jiawei Zhang, and Ahmet Alacaoglu

We solve

$$\begin{aligned} \min_{\mathbf{x}\in X} \quad & \mathbb{E}_{\xi\sim\Xi}[f(\mathbf{x},\xi)] \\ \text{s.t.} \quad & \mathbb{E}_{\zeta\sim Q}[A(\zeta)\mathbf{x}-\mathbf{b}] = 0 \end{aligned}$$

where $X = H\mathbf{x} \leq h$, for some matrix H and vector h.

- Without loss of generality, we assume X is projection-friendly
- Otherwise, we introduce slack variables to rewrite the constraints to be $\{x, w \mid Ax = b, Hx + w = h, w \geq 0\}$ and then the inequality constraints are projection-friendly

The First Type of Stochastic Oracle

- Assume that for any x, we can pick i.i.d. sample ξ and
- compute the unbiased estimation of $\nabla f, Ax - b$:

$$\mathbb{E}_{\xi}[\widehat{\nabla} f(\mathbf{x};\xi)] = \nabla f(\mathbf{x}),$$
$$\mathbb{E}_{\zeta}[v(\zeta)] = E_{\zeta} A(\zeta) x - b,$$

- Assume that the variance of estimator is bounded by σ^2.
- From Arjevani et al. [2023], for unconstrained nonconvex stochastic optimization, the lower bound of sample complexity under oracle 1 is $\Omega(\epsilon^{-4})$
- Open problem: Can we achieve this lower bound for constrained problems?

The Second Type of Stochastic Oracle

- Sometimes, we can access stronger oracles.
- We assume that for any i.i.d. sample ξ, we access the unbiased estimation of $\nabla f, Ax - b$ at two different x^1, x^2:

$$\mathbb{E}_\xi[\widehat{\nabla} f(\mathbf{x}^i; \xi)] = \nabla f(\mathbf{x}^i),$$
$$\mathbb{E}_\zeta[v(x^i, \zeta)] = Ax^i - b,$$

  and the variance of the estimator is assumed to be bounded by σ^2.
- Commonly used assumption for variance reduction, sometimes impractical, e.g., in federated learning
- Lower bound under this oracle: $\Omega(1/\epsilon^3)$
- Open problem: Can we achieve this for constrained optimization?

An Example: Federated Learning

- Federated learning: m agents, a central server, objective function of agent i: $f_i(x) := \mathbb{E}_{\xi \sim P_i} f(x; \xi)$
- We minimize $f(x) = \sum_i f_i(x)$
- A consensus constrained reformulation: $\min_{x_i - x_0 = 0, i \in [m]} \sum_i f_i(x_i)$
- agent i controls x_i and the server controls x_0
- At any time, agent i only has some probability p to be active, i.e., connected to the server
- If connected we can estimate $x_i - x_0$, otherwise $0 * x_i - 0 * x_0$

Continue

- Hence for any $x = (x_0, x_1, \cdots, x_n)$, we have an unbiased estimate of $x_0 - x_i$
- The first oracle is more practical
- At different times, random sample–the status of connection to server is different
- At time t and $t+1$, the connection is different
- Hence, hard to satisfy oracle 2

Comparison of Methods

Reference	Constraint	Oracle	Complexity	Loops	Method
Alacaoglu & Wright [2024]	$Ax = b$	2	$\widetilde{O}(\varepsilon^{-3})$	1	ALM
Alacaoglu & Wright[2024]	$\mathbb{E}[c(x,\zeta)] = 0$, and $x \in X$ where X is easy to project	2	$\widetilde{O}(\varepsilon^{-5})$	1	Penalty
Lu et al. [2024]	$c(x) = 0$, and $x \in X$ where X is easy to project	2	$O(\varepsilon^{-3})$	1	Penalty
Li et al. [2024]	$\mathbb{E}[c(x,\zeta)] = 0$, and $x \in X$ where X is easy to project	2	$O(\varepsilon^{-5})$	2	Penalty*
This work	$Ax = b$, and $x \in X$ is polyhedral	1	$O(\varepsilon^{-4})$	1	ALM
This work	$\mathbb{E}_\zeta[A(\zeta)x - b(\zeta)] = 0$, and $x \in X$ is polyhedral	1	$O(\varepsilon^{-4})$	1	ALM
This work	$Ax = b$, and $x \in X$ is polyhedral	2	$O(\varepsilon^{-3})$	1	ALM

* *This method is referred to as a penalty method because the penalty parameter is taken to infinity to ensure feasibility, and dual updates do not contribute in achieving feasibility.*

Our Contribution

- We design first-order, single-loop algorithms with constant batch size to solve nonconvex optimization problems with stochastic constraints
- The algorithm is Lagrangian based, does not require large penalty
- optimal sample complexity $\mathcal{O}(\varepsilon^{-4})$ under the first stochastic oracle
- optimal sample complexity $\mathcal{O}(\varepsilon^{-3})$ under the second stochastic oracle

Algorithm Design

One influential idea is to use the augmented Lagrangian(AL) method to solve the problem:

$$L_\rho(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + \langle A\mathbf{x} - \mathbf{b}, \mathbf{y} \rangle + \frac{\rho}{2}\|A\mathbf{x} - \mathbf{b}\|^2$$

Then the ALM iteration proceeds for $k = 1, 2, \cdots$ by updating

$$\begin{aligned} \mathbf{x}_{k+1} &\approx \arg\min_{\mathbf{x} \in X} L_\rho(\mathbf{x}, \mathbf{y}_k) \\ \mathbf{y}_{k+1} &= \mathbf{y}_k + \eta(A\mathbf{x}_{k+1} - \mathbf{b}) \end{aligned}$$

However, the ALM may not converge if f is nonconvex

Algorithm Design

- First recall the deterministic case. Consider the deterministic constrained opt:

$$\min_{x \in X, Ax=b} f(x)$$

- Let $K(x, y, z) = L_\rho(x, y) + \frac{\mu}{2}\|x - z\|^2$
- Dual ascent: $y_{t+1} = y_t + \alpha(Ax_t - b)$;
- $x_{t+1} = \mathrm{Proj}_X(x_t - \nabla_x K(x_t, y_{t+1}, z_t))$;
- $z_{t+1} = z_t + \beta(x_{t+1} - z_t)$.

Interpretation of the Algorithm

- The algorithm can be regarded as an Inexact GD on the Moreau envelope
- The Moreau envelope of our problem is given as

$$\Psi(\mathbf{z}_t) = \min_{\mathbf{x}\in X, A\mathbf{x}=\mathbf{b}} \left\{ f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{z}_t\|^2 \right\}$$

- By Danskin's Theorem, the gradient of $\Psi(z)$ is given by

$$\nabla_z \Psi(z) = \mu(z - \bar{x}(z))$$

  where $\bar{x}(z) = \arg\min_{\mathbf{x}\in X, A\mathbf{x}=\mathbf{b}} \left\{ f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{z}_t\|^2 \right\}$
- The update of $\mathbf{x}$ and $\mathbf{y}$ can be viewed as an approximation to $\bar{x}(z)$.

The Potential Function

In the deterministic case (Zhang and Luo [2020]), we can use the following potential function:

$$V_t = K(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t) - 2d(\mathbf{y}_t, \mathbf{z}_t) + 2\Psi(\mathbf{z}_t), \text{ where we use}$$

$$K(\mathbf{x}, \mathbf{y}, \mathbf{z}) = L_\rho(\mathbf{x}, \mathbf{y}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{z}\|^2,$$

$$d(\mathbf{y}, \mathbf{z}) = \min_{\mathbf{x} \in X} L_\rho(\mathbf{x}, \mathbf{y}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{z}\|^2,$$

$$\Psi(\mathbf{z}) = \min_{\mathbf{x} \in X, A\mathbf{x} = \mathbf{b}} \left\{ f(\mathbf{x}) + \frac{\mu}{2}\|\mathbf{x} - \mathbf{z}\|^2 \right\}.$$

- The gradient projection step can reduce K
- The dual ascent step can approximately reduce $-d$
- The update of z can approximately reduce $\psi(z)$ (Moreau envelope gradient descent).

The potential function V_t is decreasing, then we can show that the convergence of the algorithm. Denote $\mathbf{x}^*(\mathbf{y}, \mathbf{z}) = \arg\min_{\mathbf{x} \in X} K(\mathbf{x}, \mathbf{y}, \mathbf{z})$, we have

$$V_t - V_{t+1} \geq \frac{1}{4\tau}\|\mathbf{x}_t - \mathbf{x}_{t+1}\|^2 + \frac{\eta}{2}\|A\mathbf{x}^*(\mathbf{y}_{t+1}, \mathbf{z}_t) - \mathbf{b}\|^2 + \frac{\mu}{3\beta}\|\mathbf{z}_t - \mathbf{z}_{t+1}\|^2.$$

Algorithm Design

A natural idea is to directly extend the algorithm to stochastic case.

Stochastic Smoothed and Linearized ALM

Initialize: $\mathbf{x}_0 = \mathbf{z}_0 \in X$, $\mathbf{y}_0 \in \mathbb{R}^m$, and $\rho \geq 0$.

For $t = 0$ to $T - 1$:

1. Sample i.i.d. ζ_t;
2. $\mathbf{y}_{t+1} = \mathbf{y}_t + \eta v(\mathbf{x}_t, \zeta_t)$, where $v(\mathbf{x}_t, \zeta_t)$ is an unbiased estimator of $(A\mathbf{x}_t - \mathbf{b})$
3. Sample $\xi_t \in \Xi$ i.i.d. and generate gradient estimate:

$$G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t, \xi_t) = \nabla_{\mathbf{x}} L_\rho(\mathbf{x}_t, \mathbf{y}_{t+1}; \xi_t) + \mu(\mathbf{x}_t - \mathbf{z}_t)$$

4. $\mathbf{x}_{t+1} = \mathrm{proj}_X(\mathbf{x}_t - \tau G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t, \xi_t))$
5. $\mathbf{z}_{t+1} = \mathbf{z}_t + \beta(\mathbf{x}_{t+1} - \mathbf{z}_t)$

End For

Two Challenges

- The projection breaks unbiased structure:
  - Well-known that without constraint X, SGD step is an unbiased estimator of gradient descent for K
  - However, the projection step can break the unbiased structure.
  - We do not have a descent lemma for K
- Bounded variance of $\nabla_x K$ requires boundedness of dual variable y
  - The variance is proportional to $\|y\|$
  - Need boundedness of y, usually challenging in nonconvex and stochastic settings

First Challenge: Using Moreau Envelope

- To address the challenge, we consider the Moreau envelope of K

$$\varphi_{1/\lambda}(\mathbf{x}, \mathbf{y}, \mathbf{z}) = \min_{\mathbf{u} \in X} \left\{ K(\mathbf{u}, \mathbf{y}, z) + \frac{\lambda}{2} \|\mathbf{u} - \mathbf{x}\|^2 \right\}.$$

- By Davis and Drusvyatskiy [2019], if the variance for gradient estimation is bounded, we have descent lemma for $\varphi_{1/\lambda}$
- Therefore, we replace K by $\varphi_{1/\lambda}$ and consider the potential function

$$\varphi_{1/\lambda}(\mathbf{x}_t, \mathbf{y}_t, \mathbf{z}_t) - 2d(\mathbf{y}_t, \mathbf{z}_t) + 2\Psi(\mathbf{z}_t)$$

- We can show a descent lemma of this potential function if we can guarantee bounded variance, i.e., bounded $\mathbf{y}^t$

Controlling $\|y^t\|$

- To control $\|y^t\|$, we propose a pulling back approach
- For some lower bound M_y, we set $y^{t+1} = 0$ if $\|y^{t+1}\| \geq M_y$.
- This can guarantee the boundedness of y^t
- Interestingly, we can show that the potential function is non-increasing after this pulling back step.
- Therefore, we incorporate this in our algorithm.

Our Algorithm

Stochastic Smoothed and Linearized ALM

Initialize: $\mathbf{x}_0 = \mathbf{z}_0 \in X$, $\mathbf{y}_0 \in \mathbb{R}^m$, and $\rho \geq 0$.
For $t = 0$ to $T-1$:

1. Sample i.i.d. ζ_t;
2. $\mathbf{y}_{t+1} = \mathbf{y}_t + \eta v(\mathbf{x}_t, \zeta_t)$, where $v(\mathbf{x}_t, \zeta_t)$ is an unbiased estimator of $(A\mathbf{x}_t - \mathbf{b})$
3. If $\|y^{t+1}\| \geq M_y$, set $y^{t+1} = 0$;
4. Sample $\xi_t \in \Xi$ i.i.d. and generate gradient estimate:

$$\mathbb{E}_{\xi_t}[G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t, \xi_t)] = \nabla_{\mathbf{x}} L_\rho(\mathbf{x}_t, \mathbf{y}_{t+1}) + \mu(\mathbf{x}_t - \mathbf{z}_t)$$

5. $\mathbf{x}_{t+1} = \text{proj}_X(\mathbf{x}_t - \tau G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t, \xi_t))$
6. $\mathbf{z}_{t+1} = \mathbf{z}_t + \beta(\mathbf{x}_t - \mathbf{z}_t)$

End For

Theoretical Guarantee

Theorem

Under oracle 1, run our Algorithm(Dual-Safeguarded Stochastic ALM) with η, τ, β *chosen as* $\Theta(1/\sqrt{T})$, *we have that* $\mathbb{E}\|\nabla\Psi(\mathbf{z}_{t^*})\| \leq \varepsilon$ *where* t^* *is selected uniformly at random from* $\{1, \ldots, T\}$ *with* $T = \Theta(\varepsilon^{-4})$. *The sample complexity is* $O(\varepsilon^{-4})$.

Variance Reduction

- Under oracle2, we can use variance reduction technique to reduce the sample complexity
- We replace the gradient projection step of K by a variance reduced SGD called STORM
- Then we can prove the $\mathcal{O}(1/\epsilon^3)$ sample complexity
- This means our framework is flexible to combine different techniques for stochastic optimization

Stochastic Smoothed and Linearized ALM with STORM

Algorithm: STORM-based Stochastic ALM

Input: $\rho \geq 0$, $N = T^{1/6}$
Initialize: $\mathbf{x}_0 = \mathbf{z}_0 \in X$, $\mathbf{y}_0 \in \mathbb{R}^m$, $\widehat{\nabla} f_0 = \frac{1}{N} \sum_{i=1}^{N} \nabla f(\mathbf{x}_0, \zeta_i)$

For $t = 0$ to $T - 1$:

1. $\mathbf{y}_{t+1} = \mathbf{y}_t + \eta(A\mathbf{x}_t - \mathbf{b})$
2. Compute: $G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t) = \widehat{\nabla} f_t + A^\top \mathbf{y}_{t+1} + A^\top (A\mathbf{x}_t - \mathbf{b}) + \lambda(\mathbf{x}_t - \mathbf{z}_t)$
3. $\mathbf{x}_{t+1} = \mathrm{proj}_X \left(\mathbf{x}_t - \tau G(\mathbf{x}_t, \mathbf{y}_{t+1}, \mathbf{z}_t)\right)$
4. $\mathbf{z}_{t+1} = \mathbf{z}_t + \beta(\mathbf{x}_t - \mathbf{z}_t)$
5. Sample $\xi_{t+1} \sim \Xi$ i.i.d., update gradient:

$$\widehat{\nabla} f_{t+1} = \nabla f(\mathbf{x}_{t+1}, \xi_{t+1}) + (1 - \alpha) \left(\widehat{\nabla} f_t - \nabla f(\mathbf{x}_t, \xi_{t+1}) \right)$$

Theoretical Results

Theorem

Under oracle 2, run our Algorithm(STORM-based Stochastic ALM) with η, τ, β *chosen as* $\Theta(T^{-1/3})$, *we have that* $\mathbb{E}\|\nabla\Psi(\mathbf{z}_{t^*})\| \leq \varepsilon$ *where* t^* *is selected uniformly at random from* $\{1, \ldots, T\}$ *with* $T = \Theta(\varepsilon^{-3})$. *The sample complexity is* $O(\varepsilon^{-3})$.

Why GRPO Needs Normalization: A Local-Curvature Perspective on Adaptive Gradients

Cheng Ge*, Heqi Yin*, Hao Liang, and Jiawei Zhang

LLM Reasoning

- Large language model (LLM) have achieved remarkable success.
- LLM reasoning, e.g., LLM for maths, coding has become an active research topic.
- Reinforcement learning (RL) and policy optimization provide powerful tools for post-training LLMs to improve reasoning capabilities.
- Among existing approaches, Group Relative Policy Optimization (GRPO) demonstrates strong empirical performance.
- Main message

GRPO $\Leftrightarrow$ gradient descent with adaptive stepsizes

GRPO adapts to local curvature/local Lipschitz smoothness.

The Evolution of LLM Training for Reasoning

Traditional Approaches:

- Supervised Fine-Tuning (SFT)
- Reinforcement Learning from Human Feedback (RLHF) with learned reward models
- PPO with value function (critic model)

Challenges:

- SFT: Cannot capture reasoning processes
- RLHF: Reward model bias
- PPO: High memory and computational resources

Problem Setting

- Aim to train a policy π_θ to generate the next token in an LLM.
- Initial state: question $q \in \mathcal{Q} = \{q_1, \ldots, q_n\}$.
- Objective

$$J(\theta) := \frac{1}{n}\sum_{i=1}^{n} J_i(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{o\sim\pi_\theta(\cdot|q_i)}[r(o, q_i)] \tag{1}$$

- For a question q, the policy gradient:

$$\sum_{t=0}^{T} \mathbb{E}_{\pi_\theta(a_t|s_t)}\left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) \cdot Q^{\pi_\theta}(s_t, a_t)\right],$$

  where $Q^\pi(s, a)$ is the Q-value function.
- Need a critic model to approximate Q.
- Memory-inefficient, inspiring critic-free methods.

Critic-Free Methods

- In many reasoning tasks, rewards are only available after the final token, e.g., whether the answer is correct
- The environment can be viewed as a one-step MDP/bandit.

$$\begin{aligned} \text{response} \quad & o \sim \pi_\theta(\cdot|q) \\ \text{reward} \quad & r(q, o) \in \{0, 1\} \end{aligned}$$

- Focus on the one-step MDP (bandit) setting.

Policy Gradient, REINFORCE, and GRPO

- **Objective Function:**

$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} J_i(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{o\sim\pi_\theta(\cdot|q_i)}[r(q_i, o)]$$

- Given current iterate θ_{t-1}, sample $q \in \mathcal{Q}$, an answer $o \sim \pi_\theta(o \mid q)$ and compute the (unbiased) stochastic gradient

$$\hat{g}_t := \nabla \log \pi_{\theta_{t-1}}(o \mid q) \cdot r(q, o)$$

- **Stochastic policy gradient**

$$\theta_t = \theta_{t-1} - \eta \hat{g}_t$$

- Unbiased, but high variance
- Variance-reduction methods such as REINFORCE are therefore required

REINFORCE

- Sample K responses: $\{o_1, \ldots, o_K\}$
- Compute rewards: $\{r_1, \ldots, r_K\}$ with $r_k = r(o_k)$.
- Calculate baseline:

$$\bar{r} = \frac{1}{K} \sum_{k=1}^{K} r_k.$$

- Advantage function: $A_k = r_k - \bar{r}$
- Update parameters:

$$\theta_t = \theta_{t-1} + \eta \sum_{k=1}^{K} \nabla_\theta \log \pi_{\theta_{t-1}}(o_k | q_i) \cdot A_k.$$

- **Remark:** If we have infinitely many samples, reinforce and stochastic policy gradient methods are equivalent, both equivalent to full policy gradient method

Group Relative Policy Optimization (GRPO)

- GRPO further normalizes the advantages.
- Given prompt/question q, compute the sample variance of rewards:

$$\sigma_r^2 = \frac{1}{K} \sum_{k=1}^{K} (r_k - \bar{r})^2.$$

- Define normalized advantages: $A_k = \frac{r_k - \bar{r}}{\sigma_r}$.
- Policy update:

$$\theta_t = \theta_{t-1} + \eta \sum_{k=1}^{K} \nabla_\theta \log \pi_{\theta_{t-1}}(o_k|q) \cdot A_k.$$

- GRPO rescales updates by the reward standard deviation.
- **Open Question:**

Why does this normalization yield such strong empirical performance?

Additional Notation and Assumptions

- **Assumption 1**: Unique correct answer per question

$$r(q_i, o_j) = \begin{cases} 1 & \text{if } j = a_i^* \text{ (correct answer)} \\ 0 & \text{otherwise} \end{cases}$$

- **Success probability**. The probability of generating the correct answer is

$$\pi_\theta^*(i) = \pi_\theta(o_{a_i^*} \mid q_i).$$

Deterministic Version

- The baseline reward $\bar{r}$ is already known to reduce variance.
- Main focus: effect of standard deviation (STD) normalization.
- To isolate this effect, we consider the **deterministic versions** of PG, REINFORCE, and GRPO.
- Specifically, we assume access to infinitely many samples from $\pi_{\theta_{t-1}}(o \mid q_i)$, allowing us to compute the full gradient of $J(\theta)$.
- Reinforce is equivalent to full policy gradient method.
- The update of REINFORCE and GRPO simplify and can be compared directly.

Algorithm Comparison: REINFORCE vs GRPO

REINFORCE (Vanilla PG)

1: **Input:** learning rate η, initial parameters θ_0
2: **for** $t = 1$ to T **do**
3: **for** each question i **do**
4: $\theta_t \leftarrow \theta_{t-1} + \frac{\eta}{N} \nabla J_i(\theta_{t-1})$
5: **end for**
6: **end for**
7: **Return:** π_{θ_T}

Update form:

$$\theta_t \leftarrow \theta_{t-1} + \eta\, \pi^*_\theta(i)(1 - \pi^*_\theta(i))\, x_{i,a_i}.$$

GRPO (With Normalization)

1: **Input:** learning rate η, initial parameters θ_0
2: **for** $t = 1$ to T **do**
3: **for** each question i **do**
4: $\theta_t \leftarrow \theta_{t-1} + \frac{\eta}{N} \frac{\nabla J_i(\theta_{t-1})}{\sqrt{\pi^*_{\theta_{t-1}}(i)(1-\pi^*_{\theta_{t-1}}(i))}}$
5: **end for**
6: **end for**
7: **Return:** π_{θ_T}

Key Difference: GRPO normalizes each gradient by the STD .

Log-Linear Parametrization and GRPO Updates

Policy parametrization:

$$\pi_\theta(o_j \mid q_i) = \frac{\exp\left(x_{i,j}^\top \theta\right)}{\sum_{l=1}^{K} \exp\left(x_{i,l}^\top \theta\right)},$$

where $x_{i,j} \in \mathbb{R}^d$ is the feature vector for the pair (q_i, o_j).

REINFORCE update:

$$\theta_t \leftarrow \theta_{t-1} + \eta\left[\pi^*_{\theta_{t-1}}(i)\left(1 - \pi^*_{\theta_{t-1}}(i)\right)x_{i,a_i} - \pi^*_{\theta_{t-1}}(i)\sum_{j\neq a_i} \pi_{\theta_{t-1}}(o_j \mid q_i)\, x_{i,j}\right].$$

GRPO update:

$$\theta_t \leftarrow \theta_{t-1} + \eta\Bigg[\sqrt{\pi^*_{\theta_{t-1}}(i)\left(1 - \pi^*_{\theta_{t-1}}(i)\right)}\, x_{i,a_i} - \sqrt{\frac{\pi^*_{\theta_{t-1}}(i)}{1-\pi^*_{\theta_{t-1}}(i)}} \sum_{j\neq a_i} \pi_{\theta_{t-1}}(o_j \mid q_i)\, x_{i,j}\Bigg].$$

Observation: GRPO adaptively rescales the gradient via the local variance.

Core Discovery: Variance = Local Curvature

Theorem (Local Smoothness Bound)

Under log-linear policy parametrization, for any question i and $\theta \in \mathbb{R}^d$:

$$\boxed{\|\nabla^2 J_i(\theta)\| \;\leq\; 4X_{\max}^2 \cdot \underbrace{\pi_\theta^*(i)\big(1 - \pi_\theta^*(i)\big)}_{\textit{Reward variance on } q_i}}$$

where $X_{\max} = \max_{i \in [n]} \|X_i\|$ is the maximum feature matrix norm.

Corollary (Global Smoothness)

For all $i \in [n]$ and $\theta \in \mathbb{R}^d$,

$$\|\nabla^2 J_i(\theta)\| \leq X_{\max}^2.$$

Thus, $J_i(\theta)$ is globally $X_{\max}^2$-smooth.

Key Insight: The local curvature of $J_i(\theta)$ scales directly with the reward variance. Hence, GRPO adaptively adjusts step sizes to match local Lipschitz smoothness, while REINFORCE uses a fixed step size.

Local Curvature Stability

Lemma (Non-uniform Local Smoothness)

Under Assumption 1, for all $i \in [n]$ *and* $\theta \in \mathbb{R}^d$, $J_i(\theta)$ *is*

$$\frac{5}{2} X_{\max}^2 \cdot \sqrt{\pi_\theta^*(i)\big(1 - \pi_\theta^*(i)\big)}$$

smooth over the ball $B\left(\theta,\ \frac{1}{X_{\max}} \cdot \sqrt{\pi_\theta^*(i)\big(1 - \pi_\theta^*(i)\big)}\ \right)$.

Interpretation:

- Curvature remains bounded within a neighborhood whose radius scales with $\sqrt{\text{variance}}$.
- With step size $\eta = \frac{1}{2X_{\max}^2}$, GRPO updates remain inside this stable region.
- Local smoothness guarantees hold throughout training.

Implication

Normalization in GRPO automatically constrains updates to regions where curvature estimates are valid, enhancing stability.

An Additional Assumption

- Based on the previous results, we can establish the convergence rate of GRPO in the single-question setting.
- To extend to multiple questions, we introduce a technical condition
- **Assumption 2 (Gradient Orthogonality).** For any $i \neq j$,

$$\nabla J_i(\theta)^\top \nabla J_j(\theta) = 0.$$

The gradients corresponding to different questions are mutually orthogonal.

Convergence Analysis: REINFORCE

Theorem (Convergence of REINFORCE)

Under Assumptions 1–2 and with step size $\eta = \frac{1}{X_{\max}^2}$*, the following holds:*

$$J_i(\theta_t) - J_i(\theta_{t-1}) \;\le\; -\frac{1}{2X_{\max}^2}\,\|\nabla J_i(\theta_{t-1})\|^2.$$

Moreover,

$$\sum_{t=1}^{T} \|\nabla J_i(\theta_t)\|^2 \;\le\; 2(1 - \pi_{\theta_0}^*(i))\, X_{\max}^2,$$

and the iteration complexity satisfies

$$\min_{t \in [T]} \|\nabla J_i(\theta_t)\|^2 \;\le\; \frac{2(1 - \pi_{\theta_0}^*(i))\, X_{\max}^2}{T}.$$

Implication: REINFORCE achieves a convergence rate that does not depend on the reward variance during training.

Convergence Analysis: GRPO

Theorem (Convergence of GRPO)

Under Assumptions 1–2 and with step size $\eta = \frac{1}{2X_{\max}^2}$, *we have*

$$J_i(\theta_t) - J_i(\theta_{t-1}) \;\leq\; -\frac{3}{8X_{\max}^2\, C_i(t)} \,\|\nabla J_i(\theta_{t-1})\|^2,$$

where $C_i(t) \;\leq\; \sqrt{\pi^*_{\theta_t}(i)\big(1 - \pi^*_{\theta_t}(i)\big)}$. *Moreover,*

$$\sum_{t=1}^{T} \|\nabla J_i(\theta_t)\|^2 \;\leq\; 2(1 - \pi^*_{\theta_0}(i))X_{\max}^2 \cdot \frac{8}{3T} \sum_{t=0}^{T-1} C_i(t),$$

and

$$\min_{t\in[T]} \|\nabla J_i(\theta_t)\|^2 \;\leq\; \frac{2(1 - \pi^*_{\theta_0}(i))X_{\max}^2}{T} \cdot \frac{8}{3T} \sum_{t=0}^{T-1} C_i(t).$$

Convergence Analysis: GRPO

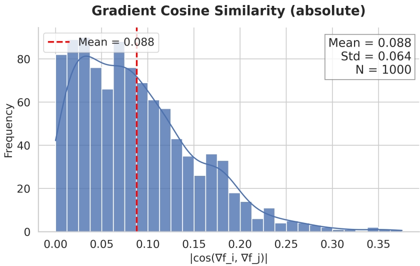
Recall $C_i(t) \;\le\; \sqrt{\pi^*_{\theta_t}(i)\big(1-\pi^*_{\theta_t}(i)\big)}$., and

$$\min_{t\in[T]} \|\nabla J_i(\theta_t)\|^2 \;\le\; \frac{2(1-\pi^*_{\theta_0}(i))X^2_{\max}}{T} \cdot \frac{8}{3T}\sum_{t=0}^{T-1} C_i(t).$$

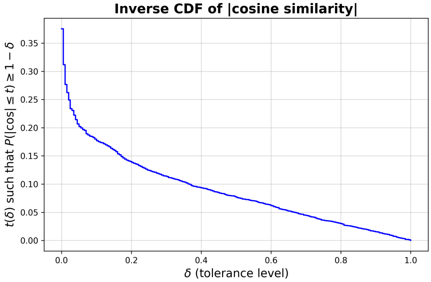
Implications:

- The factor $\frac{8}{3T}\sum_t C_i(t) < 1$ in most practical cases, implying faster convergence than REINFORCE.
- When $\pi^*_\theta(i) \ll \frac{1}{2}$ (hard problems), GRPO accelerates learning significantly.
- When $\pi^*_\theta(i) \approx 1$ (near-perfect policy), gradients are already small, so GRPO and REINFORCE behave similarly.

Empirical Validations



(a) Gradient Cosine Similarities
Mean $|\cos(\theta)| = 0.088 \pm 0.064$



(b) Inverse CDF of Cosine Similarities
90% of gradient pairs $|\cos(\theta)| < 0.15$

CurvatureâĂŞVariance Correlation

Time Lag	**Pearson Correlation**	**Significance**
Same iteration	0.342	$p < 0.01$
Different iterations	-0.028	$p = 0.18$ (n.s.)

Experimental Setup

Model Configuration

- Base model: `Qwen2.5-Math-1.5B`
- Fine-tuning via LoRA (rank $= 16$, $\alpha = 32$)
- $K = 8$ generations per prompt

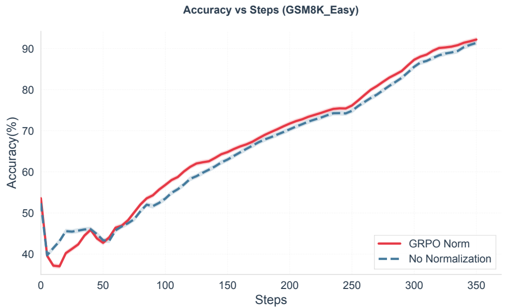
Dataset Stratification

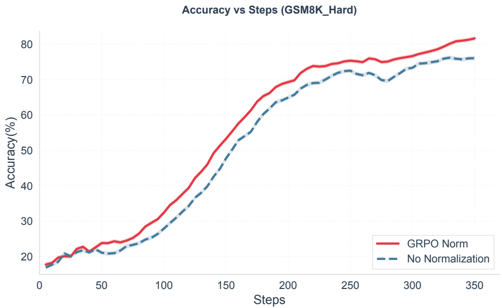
- GSM8K training set partitioned by difficulty
- Easy: 4,695 examples
- Hard: 1,909 examples
- Difficulty defined by solution complexity

Normalization Variants

- $\mathcal{N}_{\text{std}}$: Standard GRPO (with variance normalization)
- $\mathcal{N}_{\text{no-std}}$: GRPO without normalization

GSM8K Results: Easy vs. Hard Questions





Easy Subset: Low-Variance Regime

Hard Subset: High-Variance Regime

- Final accuracy: $\mathcal{N}_{\text{std}}$ (92%) $>$ $\mathcal{N}_{\text{no-std}}$ (91%).
- Normalization yields small but consistent gains.

- Final accuracy: $\mathcal{N}_{\text{std}}$ (81%) $\gg$ $\mathcal{N}_{\text{no-std}}$ (76%).
- GRPO significantly outperforms the

Remarks

- For hard questions, the initial accuracy is around 20%, far below 50%.
  $\Rightarrow$ GRPO accelerates learning substantially in the early phase.
- For easy questions, the initial accuracy is close to 50%.
  $\Rightarrow$ GRPO offers little speedup over REINFORCE in this regime.
- As accuracy becomes high, gradients diminish in magnitude.
  $\Rightarrow$ All methods progress slowly, with GRPO still slightly outperforming REINFORCE.

Future Problems

- Today we studied stochastic constrained optimization and adaptive step sizes guided by local Lipschitz smoothness.
- Natural extensions include:
  - Relaxing the gradient orthogonality assumption: prompts may exhibit correlated gradients with heterogeneous smoothness.
  - Designing adaptive step-size rules for more general stochastic constrained problems.
  - Extending to distributed optimization, where each agent may have very different smoothness properties.

Free Draft-and-Verification: Toward Lossless Parallel Decoding for Diffusion Large Language Models

Shutong Wu, and Jiawei Zhang

Overview

1. Diffusion Large Language Models (DLLMs): Preliminaries and Challenges
2. Our Solution to the Dilemma of Inference Efficiency and Performance
3. Experiments on Math Reasoning and Code Generation Tasks
4. Conclusion and Future Work

Preliminaries

1. Autoregressive Language Modeling
   - $p_\theta(x) = \prod_{i=1}^{L} p_\theta(x^i | x^{<i})$
   - the i-th token x_i conditional on all previous token $x^{<i}$
   - usually parameterized by causal-attention Transformers
   - GPT, Gemini, Llama, Qwen, DeepSeek, etc.

Preliminaries

❷ (Masked) Diffusion Language Modeling

- forward process: progressively replaces each unmasked token in the original sequence x independently to a special mask token **m**
- probability of being masked controlled by a noise schedule α_t
- α_t monotonically decreasing *w.r.t.* $t \in [0,1]$; $\alpha_0 = 1, \alpha_1 = 0$
- $q(x_t|x_0) = \prod_{i=1}^{L} q(x_t^i|x_0^i) = \prod_{i=1}^{L} \text{Cat}\left(x_t^i; \alpha_t x_0^i + (1-\alpha_t)\mathbf{m}\right)$
- x_t^i: the i-th token at time level t
- once a token is masked at $s \in [0,1]$, it will remain masked at $\forall t \in [s,1]$

Preliminaries

❷ (Masked) Diffusion Language Modeling

- reverse process: recover the original sequence from an all-mask sequence
- for $s < t$, we have $q(x_s|x_t, x_0) = \prod_{i=1}^{L} q(x_s^i|x_t^i, x_0^i)$
- for each token,

$$
\begin{aligned}
q(x_s^i|x_t^i, x_0^i) &= q(x_t^i|x_s^i, x_0^i)q(x_s^i|x_0^i)/q(x_t^i|x_0^i) \\
&= \begin{cases} \mathsf{Cat}(x_s^i; x_t^i) & \text{if } x_t^i \neq \mathbf{m} \\ \mathsf{Cat}(x_s^i; \frac{(1-\alpha_t)\mathbf{m}+(\alpha_s-\alpha_t)x_0^i}{1-\alpha_t}) & \text{if } x_t^i = \mathbf{m} \end{cases}
\end{aligned}
$$

- train a model f_θ to estimate x_0 from x_t (usually with ELBO as objective), and induce the reverse process for each token as

$$
\begin{aligned}
p_\theta(x_s^i|x_t^i) &= q(x_s^i|x_t^i, x_0^i = f_\theta(x_t^i, t)) \\
&= \begin{cases} \mathsf{Cat}(x_s^i; x_t^i) & \text{if } x_t^i \neq \mathbf{m} \\ \mathsf{Cat}(x_s^i; \frac{(1-\alpha_t)\mathbf{m}+(\alpha_s-\alpha_t)f_\theta^i(x_t,t)}{1-\alpha_t}) & \text{if } x_t^i = \mathbf{m} \end{cases}
\end{aligned}
$$

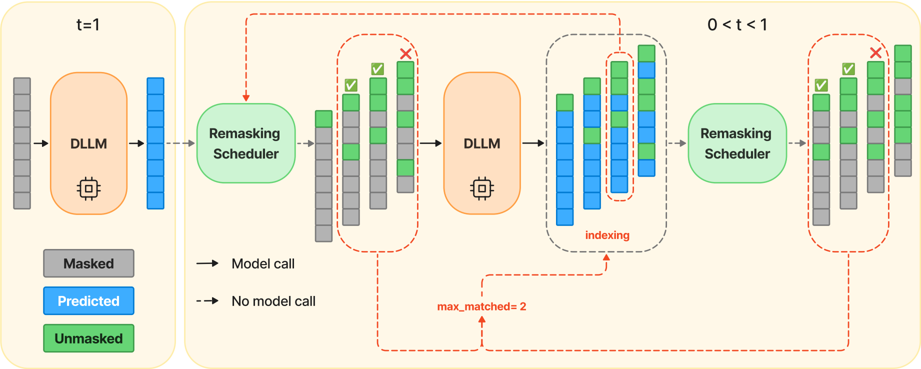
- once a token is unmasked at t, it will remain unchanged at $\forall s \in [0, t]$
- static decoding: decode token with highest probability at each step

Preliminaries

❷ (Masked) Diffusion Language Modeling

- usually parameterized by bidirectional-attention Transformers
- comparable performance with AR LLMs
- challenges:
    - good performance requires more decoding steps (usually equal to the sequence length)
    - slower decoding due to the bidirectional attention
    - parallel decoding: decode multiple tokens at each step, but usually with non-trivial performance drop

FreeDave: Lossless Parallel Decoding



1. Our solution: self-verifiable lossless parallel decoding
2. Multiple parallel-decoded candidates from the estimated distribution at current step
3. At next step, batch forward and decode one more step on each candidate, and compare with the previous candidates to verify correctness

FreeDave: Lossless Parallel Decoding

Static Decoding

Prompt: When did Albert Einstein obtain his doctoral degree?

Step 0: <mask> <mask> <mask> <mask> <mask> <mask> <mask> <mask>

Step 1: <mask> <mask> <mask> doctoral <mask> <mask> <mask> <mask>

Step 2: <mask> <mask> <mask> doctoral degree <mask> <mask> <mask>

Step 3: <mask> <mask> his doctoral degree <mask> <mask> <mask>.

Step 4: <mask> <mask> his doctoral degree <mask> 1905 <mask>.

Step 5: <mask> <mask> his doctoral degree in 1905 <mask>

Step 6: He <mask> his doctoral degree in 1905 <mask>

Step 7: He obtained his doctoral degree in 1905 <mask>

Step 8: He obtained his doctoral degree in 1905.

FreeDave: Lossless Parallel Decoding

FreeDave Decoding

Prompt: When did Albert Einstein obtain his doctoral degree?

Step 0: <mask> <mask> <mask> <mask> <mask> <mask> <mask> <mask>

V-Step 1: <mask> <mask> <mask> doctoral <mask> <mask> <mask> <mask>

D-Step 1: <mask> <mask> <mask> doctoral <mask> <mask> <mask> <mask>

<mask> <mask> <mask> doctoral degree <mask> <mask> <mask>

<mask> <mask> his doctoral degree <mask> <mask> <mask>

<mask> <mask> his doctoral degree <mask> 1905 <mask>

V-Step 2: <mask> <mask> <mask> doctoral degree <mask> <mask> <mask> ✅

<mask> <mask> his doctoral degree <mask> <mask> <mask> ✅

<mask> <mask> his doctoral degree <mask> 1905 <mask> ✅

<mask> <mask> his doctoral degree in 1905 <mask>

D-Step 2: <mask> <mask> his doctoral degree in 1905 <mask>

He <mask> his doctoral degree in 1905 <mask>

He obtained his doctoral degree in 1905 <mask>

He obtained his doctoral degree in 1905.

V-Step 3: He <mask> his doctoral degree in 1905 <mask> ✅

He obtained his doctoral degree in 1905 <mask> ✅

He obtained his doctoral degree in 1905. ✅

He obtained his doctoral degree in 1905.

Finish: He obtained his doctoral degree in 1905.

● candidate token ● target token w/ predicted distribution ● unmasked token

FreeDave: Lossless Parallel Decoding

1. Theoretically, we can prove that FreeDave generates the same sequence as static decoding that decodes one token with the highest probability at each step.
2. Additionally, with a large enough draft size, FreeDave is guaranteed to find the optimal path with the fewest decoding steps.

Experiments on Math Reasoning Tasks

Table: Detailed comparison of performance and efficiency with static decoding, parallel decoding, and FreeDave decoding for different DLLMs on MATH500.

Model	Sampling	Acc (%) ↑	Throughput over time (#tokens/s) ↑	Throughput over NFEs (#tokens) ↑
Dream-7B-Instruct	Static	40.00	23.02	0.91
	Parallel	37.00 (-3.00)	16.73 (0.73)	1.88 (2.07×)
	FreeDave	**40.20 (+0.20)**	**30.00 (1.30×)**	**2.63 (2.89×)**
TraDo-4B-Instruct	Static	74.20	7.26	0.26
	Parallel	68.80 (-5.40)	**18.94 (2.61×)**	0.61 (2.35×)
	FreeDave	**76.40 (+2.20)**	16.36 (2.25×)	**0.67 (2.58×)**
TraDo-8B-Instruct	Static	76.40	7.10	0.28
	Parallel	74.00 (-2.40)	**16.11 (2.27×)**	0.60 (2.14×)
	FreeDave	**77.60 (+1.20)**	15.99 (2.25×)	**0.66 (2.36×)**

Experiments on Math Reasoning Tasks

Table: Detailed comparison of performance and efficiency with static decoding, parallel decoding, and FreeDave decoding for different DLLMs on GSM8K.

Model	**Sampling**	**Acc (%) ↑**	**Throughput over time (#tokens/s) ↑**	**Throughput over NFEs (#tokens) ↑**
Dream-7B-Instruct	Static	79.61	20.99	0.83
	Parallel	68.16 (-11.45)	16.61 (0.79×)	1.81 (2.18×)
	FreeDave	**80.21 (+0.60)**	**27.39 (1.30×)**	**2.34 (2.82×)**
TraDo-4B-Instruct	Static	**91.58**	4.41	0.15
	Parallel	89.08 (-2.50)	9.82 (2.23×)	0.35 (2.33×)
	FreeDave	91.05 (-0.53)	**10.03 (2.27×)**	**0.39 (2.60×)**
TraDo-8B-Instruct	Static	92.72	3.41	0.12
	Parallel	92.34 (-0.38)	6.17 (1.81×)	0.23 (1.92×)
	FreeDave	**92.80 (+0.08)**	**6.92 (2.03×)**	**0.28 (2.33×)**

Experiments on Math Reasoning Tasks

Table: Detailed comparison of performance and efficiency with static decoding, parallel decoding, and FreeDave decoding for different DLLMs on AIME2024.

Model	Sampling	Acc (%) ↑	Throughput over time (#tokens/s) ↑	Throughput over NFEs (#tokens) ↑
Dream-7B-Instruct	Static	**6.67**	22.82	0.94
	Parallel	3.33 (-3.34)	16.09 (0.71×)	1.92 (2.04×)
	FreeDave	3.33 (-3.34)	**24.08 (1.06×)**	**3.55 (3.78×)**
TraDo-4B-Instruct	Static	10.00	11.38	0.41
	Parallel	10.00 (+0.00)	20.52 (1.80×)	0.75 (1.83×)
	FreeDave	**13.30 (+3.30)**	**26.07 (2.29×)**	**1.04 (2.54×)**
TraDo-8B-Instruct	Static	13.33	15.39	0.51
	Parallel	10.00 (-3.33)	24.00 (1.56×)	0.86 (1.67×)
	FreeDave	**16.66 (+6.66)**	**29.62 (1.92×)**	**1.18 (2.31×)**

Experiments on Code Generation Tasks

Table: Detailed comparison of performance and efficiency with static decoding, parallel decoding, and FreeDave decoding for different DLLMs on MBPP.

Model	Sampling	Acc (%) ↑	Throughput over time (#tokens/s) ↑	Throughput over NFEs (#tokens) ↑
Dream-7B-Instruct	Static	46.20	15.49	0.62
	Parallel	37.40 (-8.80)	15.36 (0.99×)	1.70 (2.74×)
	FreeDave	**46.40 (+0.20)**	**20.32 (1.31×)**	**1.80 (2.90×)**
TraDo-4B-Instruct	Static	**57.40**	1.63	0.06
	Parallel	49.40 (-8.00)	**4.28 (2.63×)**	0.14 (2.33×)
	FreeDave	56.60 (-0.80)	4.19 (2.57×)	**0.15 (2.50×)**
TraDo-8B-Instruct	Static	63.20	1.80	0.07
	Parallel	57.00 (-6.20)	3.67 (2.04×)	0.13 (1.86×)
	FreeDave	**63.60 (+0.40)**	**3.86 (2.14×)**	**0.15 (2.14×)**

Experiments on Code Generation Tasks

Table: Detailed comparison of performance and efficiency with static decoding, parallel decoding, and FreeDave decoding for different DLLMs on HumanEval.

Model	**Sampling**	**Acc (%)** $\uparrow$	**Throughput over time (#tokens/s)** $\uparrow$	**Throughput over NFEs (#tokens)** $\uparrow$
Dream-7B-Instruct	Static	54.88	17.85	0.72
	Parallel	35.37 (-19.51)	15.72 (0.88×)	1.77 (2.46×)
	FreeDave	**56.09 (+1.21)**	**24.40 (1.37×)**	**2.14 (2.97×)**
TraDo-4B-Instruct	Static	59.76	4.33	0.17
	Parallel	57.32 (-2.44)	7.36 (1.70×)	0.26 (1.53×)
	FreeDave	**60.98 (+1.22)**	**8.74 (2.02×)**	**0.38 (2.24×)**
TraDo-8B-Instruct	Static	**68.90**	2.69	0.12
	Parallel	65.24 (-3.66)	**4.57 (1.70×)**	0.22 (1.83×)
	FreeDave	**68.90 (+0.00)**	4.28 (1.59×)	**0.26 (2.17×)**

Conclusion and Future Work

1. FreeDave: bring more speedup, but also overcome the challenge of performance degradation at the same time.
   - No modification or extra training required
   - No extra modules
   - Self-verifiable, seamless integration with existing DLLMs
   - Compatible with caching techniques
2. When using a very large number of draft steps, a model forward call on a batch of inputs will take a longer time
   - Trade-off between time and NFE
   - Tensor Parallelism or Data Parallelism? Extra communication cost?

References I

Ahmet Alacaoglu and Stephen J Wright. Complexity of single loop algorithms for nonlinear programming with stochastic objective and constraints. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4627–4635. PMLR, 02–04 May 2024. URL https://proceedings.mlr.press/v238/alacaoglu24a.html.

Yossi Arjevani, Yair Carmon, John C Duchi, Dylan J Foster, Nathan Srebro, and Blake Woodworth. Lower bounds for non-convex stochastic optimization. *Mathematical Programming*, 199(1):165–214, 2023.

Damek Davis and Dmitriy Drusvyatskiy. Stochastic model-based minimization of weakly convex functions. *SIAM Journal on Optimization*, 29(1):207–239, 2019.

Zichong Li, Pin-Yu Chen, Sijia Liu, Songtao Lu, and Yangyang Xu. Stochastic inexact augmented lagrangian method for nonconvex expectation constrained optimization. *Computational Optimization and Applications*, 87(1):117–147, 2024.

Zhaosong Lu, Sanyou Mei, and Yifeng Xiao. Variance-reduced first-order methods for deterministically constrained stochastic nonconvex optimization with strong convergence guarantees. *arXiv preprint arXiv:2409.09906*, 2024.

Jiawei Zhang and Zhi-Quan Luo. A proximal alternating direction method of multiplier for linearly constrained nonconvex minimization. *SIAM Journal on Optimization*, 30(3): 2272–2302, 2020.