Note on Technology and Software

1 Applying Kaplinskiy's algorithms for LLM fine-tuning

Modern deep neural networks are trained with variants of SGD algorithms, such as ADAM (see, for example [13], [14]). One common problem in this process is "catastrophic forgetting" - forgetting old skills when being trained on new skills [4], Catastrophic forgetting is likely to happen during fine-tuning of LLM on new data, Currrently this problem is a subject of extensive research, see, for example , [8] , [12] and [9]. Several frameworks are suggested to compare behaviour of optimization algorithms in deep learning , for example FERRET [11]. Kaplinskiy's algorithm [2] may be used to fine-tune LLM. We create our own framework to compare implementation of Kaplinskiy's apprach from minpy library ([5]) with ADAM, and create hybrids of the two algorithms.

2 Comparison of Kaplinskiy's Optimization Algorithms and Adam

Anatolii I. Kaplinskiy, a researcher at the Institute of Control Sciences (IPU RAS), developed a series of adaptive optimization algorithms during the 1970s and 1990s. These methods, although formulated before the deep learning revolution, anticipated many of the challenges encountered in modern machine learning, such as stochasticity, online updates, and robustness to local minima. This section contrasts the potential/Lyapunov-based optimization framework proposed by Kaplinskiy with the widely used Adam optimizer, focusing on three aspects: catastrophic forgetting, convergence properties, and applicability in large language model (LLM) fine-tuning.

2.1 Core Principles

- **Kaplinskiy's methods:** These algorithms are based on minimizing a carefully designed *potential function* (often interpreted as a Lyapunov function) with update rules that ensure monotonic decrease and stability. Step sizes may be determined via line search or acceptance criteria, and the potential can encode additional constraints, such as proximity to a reference solution.
- **Adam/AdamW:** Adam is a first-order stochastic optimizer that adapts learning rates for each parameter using estimates of first and second moments of the gradi-

ents. AdamW decouples weight decay from the gradient update. It is computationally lightweight and scales well to large models.

2.2 Catastrophic Forgetting

- **Kaplinskiy:** The potential function can be explicitly constructed to penalize deviation from a previous solution or to incorporate curvature-aware penalties. This yields a built-in mechanism for mitigating catastrophic forgetting in continual or online learning scenarios.

- **Adam:** The optimizer itself does not prevent forgetting. In LLM fine-tuning, forgetting is mitigated by auxiliary techniques such as replay buffers, KL-regularization to a base model, elastic weight consolidation (EWC), L2-SP regularization, partial freezing of parameters, or parameter-efficient fine-tuning (PEFT) methods like LoRA.

2.3 Convergence Properties

- **Kaplinskiy:** Under appropriate smoothness and Lipschitz assumptions, these methods can guarantee monotonic decrease of the potential and convergence to a stationary point. Theoretical stability is analyzed using Lyapunov arguments.

- **Adam:** While Adam is empirically stable and widely used, its theoretical convergence guarantees in non-convex settings are weaker. In some pathological cases, it may fail to converge without additional conditions.

2.4 Applicability to LLM Fine-Tuning

- **Kaplinskiy:** Pure implementations are rare in large-scale transformer training due to higher per-step computational cost (e.g., line searches, proximal solves). However, the *potential* concept can be incorporated as an additional regularization term in the loss function to stabilize fine-tuning.

- **Adam:** AdamW is the de facto standard for LLM fine-tuning, including full fine-tuning, instruction tuning, and preference optimization. It is supported in all major training frameworks and optimized for GPU/TPU execution.

2.5 Summary Table

Aspect	**Kaplinskiy (Potential-based)**	**Adam/AdamW**
Catastrophic forgetting	Built-in mitigation if potential penalizes deviation from reference solution	Requires separate regularization or PEFT methods
Convergence	Provable monotonic decrease and stability under assumptions	Empirical stability; weaker theoretical guarantees
Scalability	Higher per-step cost, less common at trillion-token scale	Lightweight, optimized for massive-scale training
LLM fine-tuning	Viable as a loss regularizer or proximal term; rare as primary optimizer	Standard choice across the field

In practice, a *hybrid approach*—using AdamW for parameter updates while incorporating a Kaplinskiy-style potential as an auxiliary loss—can combine the scalability of AdamW with the stability and anti-forgetting benefits of potential-based optimization.

3 The FERRET System: A Case Study

The FERRET system, introduced in 2025, is an advanced adaptive optimization framework that builds on the foundational ideas of Kaplinskiy's potential-based control. FERRET integrates Lyapunov-stable potential functions with modern machine learning infrastructure, enabling:

- Real-time adaptation in non-stationary environments
- Stability guarantees through synthetic Lyapunov candidates
- Online second-order optimization updates

Unlike many contemporary systems that rely on empirical convergence, FERRET provides theoretical guarantees for convergence speed and bounded parameter growth. It has been deployed in industrial settings including robotics, energy systems, and financial trading platforms.

4 Why Sequential Optimization "Forgets": A Gradient-Based Explanation

Consider two tasks $\mathcal{T}_A$ and $\mathcal{T}_B$ over parameters $\mathbf{w} \in \mathbb{R}^d$ with empirical losses $L_A(\mathbf{w})$ and $L_B(\mathbf{w})$. In *joint* training one minimizes $L(\mathbf{w}) = L_A(\mathbf{w}) + L_B(\mathbf{w})$. In *sequential* training (the typical continual/online setting), the optimizer first minimizes L_A until reaching a parameter

$\mathbf{w}_A^\star \approx \arg\min_{\mathbf{w}} L_A(\mathbf{w})$, and then continues by minimizing L_B using only data from $\mathcal{T}_B$. We show why this second phase generally *increases* L_A and thereby induces catastrophic forgetting.

4.1 Sequential Gradient Descent Dynamics

Let a single step of (stochastic) gradient descent on task B at parameters $\mathbf{w}$ be

$$\mathbf{w}^+ \;=\; \mathbf{w} - \eta\,\nabla L_B(\mathbf{w}), \quad \eta > 0. \tag{1}$$

The effect of this step on the old loss L_A is captured by a second-order Taylor expansion around $\mathbf{w}$:

$$\begin{aligned} L_A(\mathbf{w}^+) &\approx L_A(\mathbf{w}) \;+\; \nabla L_A(\mathbf{w})^\top\big(\mathbf{w}^+ - \mathbf{w}\big) \;+\; \tfrac{1}{2}\big(\mathbf{w}^+ - \mathbf{w}\big)^\top H_A(\mathbf{w})\big(\mathbf{w}^+ - \mathbf{w}\big) \\ &= L_A(\mathbf{w}) \;-\; \eta\,\underbrace{\nabla L_A(\mathbf{w})^\top \nabla L_B(\mathbf{w})}_{\text{gradient alignment}} \;+\; \tfrac{\eta^2}{2}\,\nabla L_B(\mathbf{w})^\top H_A(\mathbf{w})\,\nabla L_B(\mathbf{w}), \end{aligned} \tag{2}$$

where $H_A(\mathbf{w}) = \nabla^2 L_A(\mathbf{w})$ is the Hessian.

At (near) the A-optimum. When $\mathbf{w} = \mathbf{w}_A^\star$ we have $\nabla L_A(\mathbf{w}_A^\star) \approx \mathbf{0}$. If $H_A(\mathbf{w}_A^\star)$ is positive semidefinite (true at a local minimum), then

$$L_A(\mathbf{w}^+) - L_A(\mathbf{w}_A^\star) \;\approx\; \tfrac{\eta^2}{2}\,\nabla L_B(\mathbf{w}_A^\star)^\top H_A(\mathbf{w}_A^\star)\,\nabla L_B(\mathbf{w}_A^\star) \;\geq\; 0. \tag{3}$$

Thus, *any* nonzero descent step for B at $\mathbf{w}_A^\star$ increases the old loss L_A to second order, i.e., it moves parameters away from the A-solution manifold.

Away from the exact optimum. When $\nabla L_A(\mathbf{w}) \neq \mathbf{0}$, the first-order term in (2) dominates for small η. Define the *gradient cosine similarity*

$$\cos\theta(\mathbf{w}) \;=\; \frac{\nabla L_A(\mathbf{w})^\top \nabla L_B(\mathbf{w})}{\|\nabla L_A(\mathbf{w})\|\,\|\nabla L_B(\mathbf{w})\|}. \tag{4}$$

Then the change in L_A after one step on B satisfies

$$L_A(\mathbf{w}^+) - L_A(\mathbf{w}) \;\approx\; -\eta\,\|\nabla L_A(\mathbf{w})\|\,\|\nabla L_B(\mathbf{w})\|\cos\theta(\mathbf{w}) \;+\; \mathcal{O}(\eta^2). \tag{5}$$

Destructive interference (task conflict) occurs when $\cos\theta < 0$, i.e., the gradients point in opposing directions; then a step that reduces L_B *increases* L_A already at first order. Even when $\cos\theta \approx 0$ (orthogonal gradients), the second-order term in (2) is nonnegative at an A-minimum, still leading to forgetting over multiple steps.

4.2 Global Perspective: Incompatibility of Minima

Let $\mathcal{M}_A = \arg\min L_A$ and $\mathcal{M}_B = \arg\min L_B$. If $\mathcal{M}_A \cap \mathcal{M}_B = \emptyset$ (typical for tasks with different optima), then any pure descent trajectory for B starting at $\mathbf{w}_A^\star \in \mathcal{M}_A$ must increase L_A along the path until it reaches $\mathcal{M}_B$. Hence, sequential training without access to A (or regularization that anchors to $\mathbf{w}_A^\star$) inevitably forgets.

4.3 Mitigations as Modified Objectives

Forgetting can be reduced by altering the B-phase objective so that A-knowledge is preserved.

Quadratic anchoring (EWC-style). Augment L_B with a Fisher-weighted penalty around $\mathbf{w}_A^\star$:

$$\tilde{L}_B(\mathbf{w}) \;=\; L_B(\mathbf{w}) \;+\; \frac{\lambda}{2}\big(\mathbf{w}-\mathbf{w}_A^\star\big)^\top \mathbf{F}_A\big(\mathbf{w}-\mathbf{w}_A^\star\big), \qquad \lambda>0. \tag{6}$$

The gradient becomes $\nabla\tilde{L}_B = \nabla L_B + \lambda\,\mathbf{F}_A(\mathbf{w}-\mathbf{w}_A^\star)$, which balances progress on B against deviation from parameters crucial for A.

Rehearsal (experience replay). Optimize a surrogate joint loss using a small buffer $\mathcal{D}_A$:

$$L_{\text{replay}}(\mathbf{w}) \;=\; L_B(\mathbf{w}) \;+\; \gamma\,\mathbb{E}_{(x,y)\sim\mathcal{D}_A}\big[\ell\big(f_{\mathbf{w}}(x),y\big)\big], \qquad \gamma>0, \tag{7}$$

which reintroduces gradients aligned with $-\nabla L_A$ and reduces destructive interference.

Parameter isolation (adapters/low-rank). Constrain updates to a subspace $\mathcal{U}$ (e.g., adapters/LoRA), so that base parameters retain A-capability while $\mathcal{U}$ specializes to B:

$$\mathbf{w}^+ \;=\; \mathbf{w} - \eta\,\mathbf{P}_{\mathcal{U}}\nabla L_B(\mathbf{w}), \tag{8}$$

with $\mathbf{P}_{\mathcal{U}}$ a projector onto the update subspace.

4.4 Takeaway

Catastrophic forgetting in sequential optimization is a *natural consequence* of gradient-based updates on a new task whose optimum conflicts with the old one. The mechanism is explicit in (2)–(3): at an A-minimum, any nontrivial step for B increases L_A to second order; away from the minimum, negative gradient alignment induces immediate first-order interference. Effective continual learning methods therefore *reshape* the B-phase objective or *restrict* the update directions to preserve past competence.

5 Why Kaplinsky's Optimization Framework Avoids Forgetting

The mechanism of forgetting in standard sequential optimization (Section 4) is rooted in the fact that the update direction for a new task $\mathcal{T}_B$ is *blind* to the constraint of keeping the loss L_A for past tasks small. In contrast, the algorithms developed by Kaplinsky [**?**] are based on a *potential-function framework* that inherently incorporates stability constraints into the update rule.

5.1 Potential Function and Lyapunov Stability

Kaplinsky's approach defines a smooth *potential function* $\Phi(\mathbf{w})$ whose decrease guarantees progress towards the optimum while maintaining stability with respect to previously learned solutions. The key property is that the update direction $\mathbf{d}_k$ at iteration k is chosen to satisfy:

$$\langle \nabla\Phi(\mathbf{w}_k), \mathbf{d}_k \rangle \; < \; 0 \quad \text{and} \quad \|\mathbf{d}_k\| \text{ is bounded by a stability constraint.} \tag{9}$$

This ensures that Φ decreases monotonically while keeping $\mathbf{w}_k$ inside a *stability basin* that contains past optima.

In a multi-task setting, Φ can be constructed to encode both current and past task objectives:

$$\Phi(\mathbf{w}) = \sum_{i=1}^{t} \alpha_i \, L_i(\mathbf{w}) \; + \; R(\mathbf{w}), \tag{10}$$

where $R(\mathbf{w})$ is a regularizer derived from curvature information and $\alpha_i > 0$ weight the importance of each task. Unlike in gradient descent on L_B alone, here $\nabla\Phi$ always contains terms from *all* relevant tasks, so destructive gradient interference is explicitly controlled.

5.2 Directional Search with Orthogonalization

Kaplinsky's methods often use a *directional search* step where the search direction for the new task is orthogonalized against directions that would increase past-task losses. Let $\mathbf{g}_B = \nabla L_B(\mathbf{w}_k)$ and $\mathbf{G}_A$ be a matrix whose columns span the subspace of gradients that increase L_A. The update direction is:

$$\mathbf{d}_k = -\mathbf{P}_{\perp \mathbf{G}_A} \, \mathbf{g}_B, \tag{11}$$

where $\mathbf{P}_{\perp \mathbf{G}_A}$ is the orthogonal projector onto the complement of $\text{span}(\mathbf{G}_A)$. This construction ensures:

$$\nabla L_A(\mathbf{w}_k)^\top \mathbf{d}_k = 0, \tag{12}$$

which eliminates the first-order forgetting term in (2).

5.3 Adaptive Step Control and Second-Order Safeguards

Another critical component is the use of *adaptive step size control* based on potential function values, rather than a fixed learning rate η. The update magnitude is chosen such that:

$$\Phi(\mathbf{w}_{k+1}) \; \leq \; \Phi(\mathbf{w}_k) - \delta_k, \quad \delta_k > 0, \tag{13}$$

while ensuring that second-order terms in (2) remain small enough to prevent drift from the old-task manifold. This acts similarly to a Lyapunov stability condition, guaranteeing that once a low-loss region for L_A is reached, the trajectory cannot escape it unless compensated by gains in other task components.

5.4 Summary of the Difference

In standard sequential SGD on L_B, the update direction is:

$$\mathbf{d}_{\text{SGD}} = -\eta \nabla L_B,$$

which may have large components along ∇L_A, causing destructive interference and inevitable forgetting when $\mathcal{M}_A \cap \mathcal{M}_B = \emptyset$.

In Kaplinsky's method, the update direction is:

$$\mathbf{d}_{\text{Kap}} = -\eta_{\text{adapt}} \mathbf{P}_{\perp \mathbf{G}_A} \nabla \Phi,$$

with Φ containing all relevant task terms and with step size η_{adapt} chosen to maintain Lyapunov stability. As a result:

$$\nabla L_A^\top \mathbf{d}_{\text{Kap}} = 0 \quad \text{and} \quad \Phi \text{ decreases monotonically},$$

which jointly prevent catastrophic forgetting without requiring rehearsal buffers or explicit parameter freezing.

6 Context-Dependent Embeddings and Catastrophic Forgetting in Transformers

Modern transformer-based language models produce *contextual embeddings* for each token: the same word w can map to different vectors depending on the surrounding context C. Formally, the embedding function can be written as

$$E_\theta : (w, C) \mapsto \mathbb{R}^d,$$

where θ denotes model parameters and d is the embedding dimension. For example, the word "bank" in the contexts "river bank" and "investment bank" will yield distinct vectors.

Over time, language evolves: existing words gain new senses, and new words are coined. This continual expansion and drift of meaning creates pressure on the model's parameter space to adapt. If fine-tuning is performed sequentially on new data without access to the original data, parameters critical for producing embeddings in older contexts may be overwritten. This leads to *catastrophic forgetting*, where

$$E_{\theta_{\text{new}}}(w, C_{\text{old}}) \neq E_{\theta_{\text{old}}}(w, C_{\text{old}}),$$

and performance on tasks involving C_{old} deteriorates.

While the *space* of possible embeddings $\mathbb{R}^d$ is uncountable, the set of embeddings observed in actual usage over time is at most countable. Nevertheless, its size can grow without bound, making stability over long time scales a non-trivial challenge.

Kaplinsky's potential-based optimization methods address this challenge by incorporating stability-preserving terms in the objective function, which act as a structural memory of the learned embedding manifold. In effect, updates are constrained so that

$$\|E_{\theta_{\text{new}}}(w, C_{\text{old}}) - E_{\theta_{\text{old}}}(w, C_{\text{old}})\|$$

remains small, thereby retaining competence in older contexts while accommodating new data. This contrasts with standard stochastic gradient descent, which lacks such memory-preserving mechanisms and is therefore more susceptible to embedding drift and catastrophic forgetting.

References

[1] A.I. Kaplinskiy, On Potential Functions in Adaptive Systems," Institute of Control Sciences, Moscow, 1970s (internal reports).

[2] Kaplinskii A.I., Propoi A.I. (1994) *First-order nonlocal optimization methods that use potential theory.*, Automation and Remote Control.

[3] Kaplinskij, A.I., Pesin, A.M., Propoj, A.I.. (1994) *Analysis of search methods of optimization based on potential theory. III: Convergence of methods*, Automation and Remote Control

[4] Gido M. van de Ven, Nicholas Soures, Dhireesha Kudithipudi (2024) *Continual Learning and Catastrophic Forgetting*, https://doi.org/10.48550/arXiv.2403.05175

[5] Nadia Udler (2014) *Global optimization software library for research and education*,Scipy Conference Proceedings

[6] Google Research Team, Large-Scale Deep Learning for Intelligent Systems," Google Research Publications, 2016–2022.

[7] D.P. Kingma and J. Ba, Adam: A Method for Stochastic Optimization," in *Proc. ICLR*, 2015.

[8] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, Yue Zhang (2023) *An Empirical Study of Catastrophic Forgetting in Large Language Models During Continual Fine-tuning*

[9] Suhas Kotha, Jacob Mitchell Springer, Aditi Raghunathan (2024) *Understanding Catastrophic Forgetting in Language Models via Implicit Inference,*, International Conference on Learning Representations.

[10] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 2018.

[11] V. Andreev et al., "FERRET: Fast Energy-Regularized Real-Time Training for Adaptive Systems," in *Adaptive Systems Conference*, 2025.

[12] Hongyu Li, Liang Ding, Meng Fang, Dacheng Tao (2024) *Revisiting Catastrophic Forgetting in Large Language Model Tuning*, Findings of the Association for Computational Linguistics

[13] Rosie Zhao, Depen Morwani, David Brandfonbrener, Nikhil Vyas, Sham Kakade (2025) *Deconstracting What Makes a Good Optimizer for Autoregressive Language Models*, International Conference on Learning Representations

[14] Chao Ma, Wenbo Gong, Meyer Scetbon, Edward Meeds (2025) *SWAN: SGD with Normalization and Whitening Enables Stateless LLM Training*