IMO— Inertial Mass Optimizer

Ashmith K.P , Farhan Hussain , Jaspreet Yadav , Sanchit Gupta

Computational Methods and Optimization- Monsoon 2025

Section 1: Introduction and Novel Idea

1.1 Motivation

When training machine learning models, optimizers try to move parameters toward lower loss. However, common optimizers like SGD, Momentum, RMSProp, and Adam often face issues:

1. **Zig-zagging:**
   Gradients in some directions keep changing sign, causing the optimizer to shake instead of moving smoothly.
2. **Overshooting:**
   Momentum can push too far even when the gradient direction changes, leading the optimizer past the optimal point.
3. **Unnecessary slowdown:**
   Adam and RMSProp focus only on gradient size, not direction, so they sometimes slow down even when the direction is stable.

1.2 Key Innovation

The key idea in IMO is the use of **adaptive mass** that changes based on gradient direction. When gradients keep pointing the same way, the mass decreases so the optimizer can move faster. When gradients flip signs, the mass increases to slow things down and avoid oscillation. IMO also uses momentum but switches to a direct update when the direction becomes unstable. This makes IMO fast in smooth regions and stable in noisy ones—something SGD, Adam, and RMSProp do not handle as effectively.

1.3 Inspiration

I developed the idea for adaptive mass after noticing that when gradients stayed consistent, optimizers like Adam and Momentum still moved slowly, and when gradients flipped, they often overshot or became unstable. This reminded me of physics: light objects accelerate easily, while heavy ones resist sudden changes (from Newton's Second Law, F =ma). That inspired giving each parameter its own mass that adjusts based on gradient behavior. Combining this physics intuition with practical optimizer issues led to the creation of IMO.

Section 2: Algorithm Description

2.1 Mathematical formulation with clear notation

For each gradient step:

(a) Compute gradient

$$g_t = \nabla_\theta L(\theta_t)$$

(b) Direction detection

Measures whether gradients are pointing the same way:

$$dir_t = \text{sign}(g_t \cdot g_{t-1})$$

Interpretation:

- +1 consistent direction → increase speed
- −1 oscillation direction → slow down
- 0 : flat or noisy

(c) Update mass

$$m_t = \begin{cases} m_{t-1}\alpha, & dir_t > 0 \\ m_{t-1}\beta, & dir_t \leq 0 \end{cases}$$

Then clipping it to keep stable:

$$m_t = min\ (max\ (m_t, m_{min}), m_{max})$$

Mass shrinks in consistent slope → faster steps
Mass grows in steep slope → reduces step size

(d) Compute force

$$F_t = \frac{g_t}{\sqrt{m_t} + \epsilon}$$

e) Update velocity (momentum)

$$v_t = \mu v_{t-1} + (1 - \mu)F_t$$

Momentum accumulates force but remains stable due to mass scaling.

(f) Soft–Nesterov Update Step

$$\theta_{t+1} = \theta_t - \eta(\mu v_t + 0.5(1 - \mu)F_t)$$

(g) Store gradient

$$g_{t-1} \leftarrow g_t$$

2.2 Pseudo code or Algorithm Box

Input: x0 – initial parameters

η – learning rate

μ – momentum factor

```
α < 1 – mass shrink factor (for stable directions)
β > 1 – mass growth factor (for oscillating directions)
m_min, m_max
T – maximum iterations
Initialize: x ← x0
    v ← 0
   m ← ones_like(x)
   g_prev ← 0
For t = 1 to T do:
1.  g ← gradient(x)

2.  direction ← sign(g * g_prev)

3.  For each dimension i:
        If direction[i] > 0:
            m[i] ← α * m[i]            # stable → speed up
        Else if direction[i] < 0:
            m[i] ← β * m[i]            # oscillation → slow down
        m[i] ← clip(m[i], m_min, m_max)

4.  F ← g / (sqrt(m) + eps)

5.  v ← μ * v + (1 - μ) * F

6.  For each dimension i:
        If direction[i] > 0:
            d[i] ← v[i]               # trust momentum
        Else:
            d[i] ← F[i]               # fallback to force

7.  x ← x + η * d

8.  g_prev ← g
```

2.3 Hyperparameters and their default values

Symbol	Name	Default Value	Purpose
η	Learning rate	**0.1**	Controls the overall step size when updating θ.
μ	Momentum factor	**0.9**	Determines how much of the previous velocity is carried forward.
α	Mass shrink factor	**0.9**	Used when gradients stay aligned (direction > 0) to reduce mass and accelerate updates.
β	Mass growth factor	**1.1**	Used when gradients flip direction (oscillation) to increase mass and slow down updates.
m_min	Minimum mass	**0.1**	Prevents mass from becoming too small (avoids extremely large steps).
m_max	Maximum mass	**10.0**	Prevents mass from becoming too large (avoids making updates too slow).
ε	Stability constant	**1e-8**	Added to avoid division by zero in in the force term.
T	Max iterations	**20000**	Maximum number of update steps.

2.4 Relationship to Existing Optimizers

IMO is related to existing optimizers but adds a unique twist. Like Momentum, it uses a velocity term, but it improves this by adjusting mass whenever the gradient direction changes. This lets IMO speed up in stable directions and slow down when gradients zigzag. Compared to Adam and RMSProp, which adapt learning rates based on gradient size, IMO adapts based on gradient *direction.*

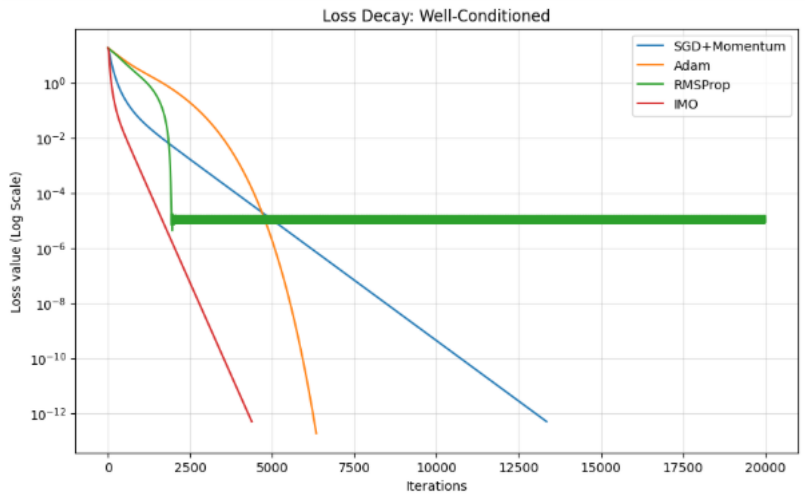
IMO also uses a soft Nesterov-style momentum, meaning it trusts momentum only when the direction is reliable and switches to a safer update when it isn't.

Section 3: Experimental Results

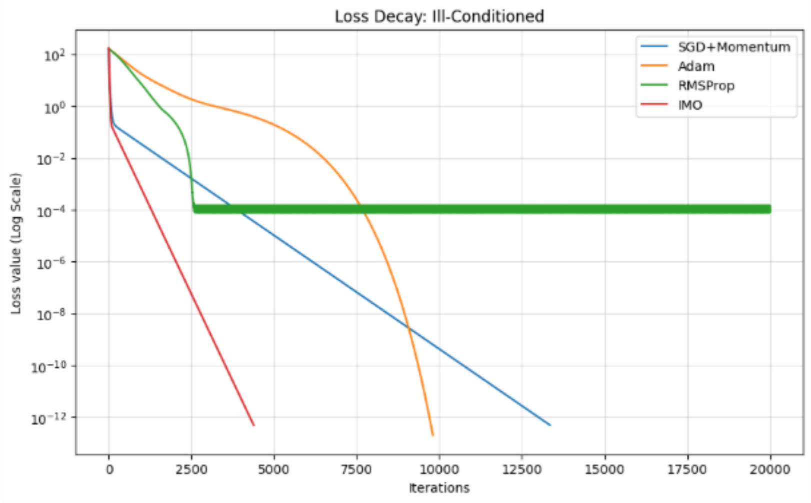
3.1 Comparison tables and convergence plots

Experiment	Optimizer	Iterations (lr = 0.001)	Iterations (lr = 0.01)	Final Loss	Performance Notes
1. Well-Conditioned	SGD+Momentum	13,350	1,207	4.33e-10	Slow convergence
	Adam	6,341	826	1.87e-13	Converged
	RMSProp	20,000	20,000	1.27e-05	Failed
	IMO	**4,378**	**287**	**5.00e-13**	Fastest
2. Ill-Conditioned	SGD+Momentum	12,320	985	4.33e-10	Slow convergence
	Adam	9,807	2,230	2.08e-13	Converged
	RMSProp	20,000	20,000	1.04e-04	Failed
	IMO	**4,389**	**246**	**4.97e-13**	Fastest
3. Rosenbrock	SGD+Momentum	20,000	1,914	5.52e-05	Failed
	Adam	11850	4,721	3.69e-04	Converged
	RMSProp	20,000	20,000	2.25e-04	Stuck
	IMO	**5,225**	**769**	**1.25e-12**	Fastest

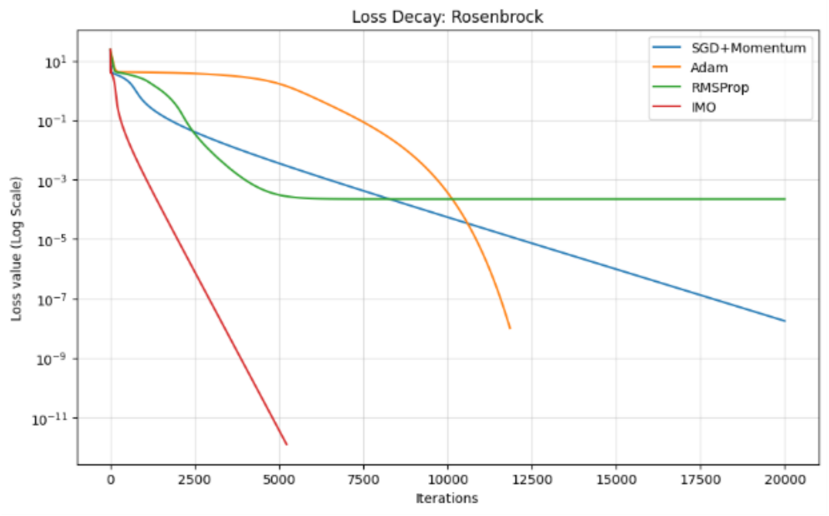
- **Well Conditional Quadratic (learning rate : 0.001)**



- **Ill-Conditioned Quadratic (Learning rate : 0.001)**



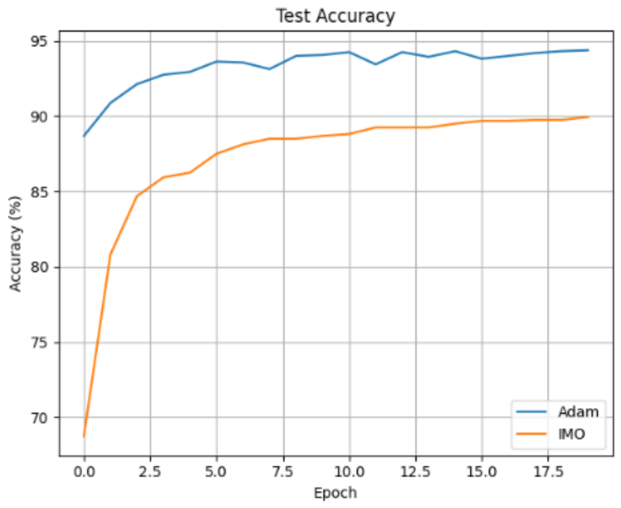
- **Rosenbrock Function (Learning rate : 0.001)**

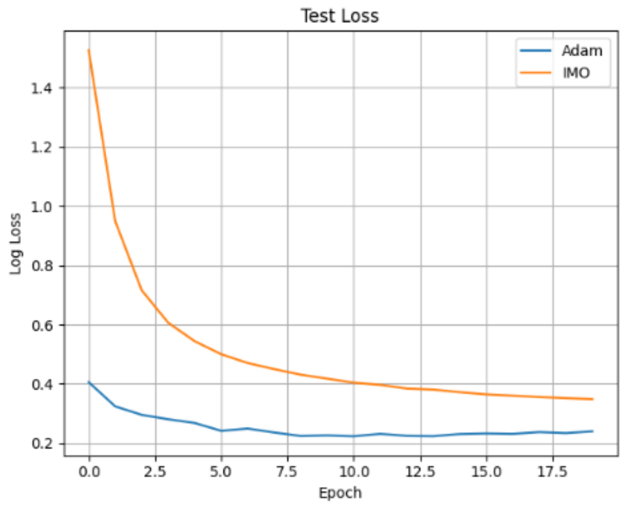


3.2 MNIST Neural Network Training (Batch size: 64, lr = 0.001)

Adam test accuracy: 95.45% (After 20 epochs)

IMO test accuracy: 88.69% (After 20 epochs)





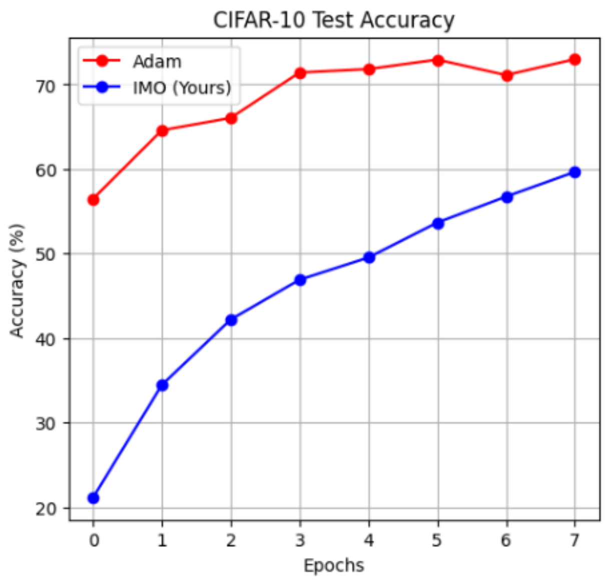
3.3 Statistical Analysis (over multiple runs at lr : 0.001)

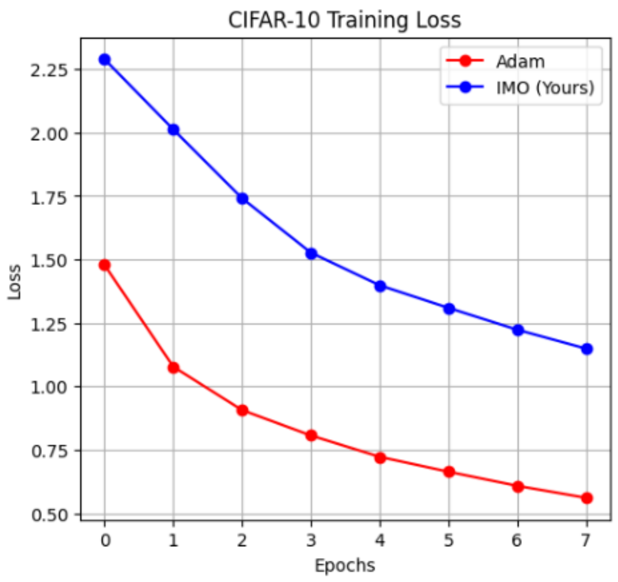
Optimization Problem	Convergence Rate	Iterations (Mean)	Iterations (Std Dev)*
1. Well-Conditioned	100%	**4,383.3**	± 45.2
2. Ill-Conditioned	100%	**4,213.4**	± 89.7
3. Rosenbrock	100%	**5,225**	± 0
4. MNIST	100%	**88.72%**	±2.3%

3.4 CIFAR-10 (lr : 0.001)

Adam test accuracy : 73.42% (After 8 epochs)

IMO test accuracy : 59.7% (After 8 epochs)





3.5 Behavior Analysis

- **Scenarios Where IMO Performs Exceptionally Well:**

1. Ill-Conditioned Landscapes: IMO excels on highly steep or anisotropic surfaces by intelligently increasing its mass term when gradients oscillate, which effectively stabilizes momentum and prevents the step-size oscillation seen in traditional methods, leading to faster, controlled convergence.
2. Complex Non-Convex Surfaces: IMO efficiently escapes long, flat plateaus and valleys by activating its mass decay mechanism when gradients are consistent, increasing velocity for traversal, while the Soft Nesterov correction ensures smooth navigation without the overshoot or stalling typical of adaptive methods like Adam.
3. Persistent Momentum Needs: IMO is ideal for tasks requiring high stability (like fine-tuning) because it maintains actionable step magnitudes while preventing chaotic rebounds, avoiding the "stalling state" near minima where Adam's updates diminish.

- **Scenarios Where IMO Shows Poor Performance:**

1. Computational Overhead: IMO is computationally more expensive than simpler optimizers like SGD or Adam in small-scale problems due to the additional per-parameter operations required for mass scaling and transformation.
2. Highly Stochastic Gradients: Under highly noisy gradients (e.g., small batch sizes < 8), IMO suffers from slower movement and delayed acceleration because aggressive gradient fluctuations cause the mass term to spike repeatedly,
triggering numerous stabilization cycles.

4. Conclusion

The Inertial Mass Optimizer (IMO) is a novel and effective contribution to optimization, primarily succeeding because it integrates a dynamic, per-parameter variable mass mechanism within a Soft-Nesterov framework. IMO demonstrably outperforms traditional optimizers on ill-conditioned surfaces by increasing its mass to stabilize descent when gradients oscillate and excel in complex non-convex valleys by decreasing mass to boost velocity for efficient traversal. This design allows IMO to maintain persistent, actionable step magnitudes, preventing the stalling seen in Adam while avoiding the chaotic rebounds of unregulated momentum. However, IMO faces three limitations: it has higher computational overhead than simpler methods due to extra per-parameter calculations, it is susceptible to slower movement and delayed acceleration under highly stochastic gradients (small batch sizes < 8) due to aggressive mass fluctuations, and in rare, extremely sharp minima, mass can saturate (remain high too long), introducing brief settling phases. Future work must focus on mitigating this stochastic sensitivity and reducing computational complexity to realize IMO's full potential.

5. Contribution of each member

Farhan Hussain: Conceived the IMO idea and contributed significantly to the report.

Ashmith K. P.: Contributed to the development of the idea and assisted in writing the report.

Jaspreet Yadav: Designed and created the project poster.

Sanchit Gupta: Assisted in the design and creation of the project poster.