Final Project Report:
Hybrid Optimization Strategies for Non-Convex Landscapes

[Your Name/Group Here]

November 14, 2025

Abstract

This project investigates the critical failure of standard Gradient Descent (GD) in non-convex optimization and develops a series of novel hybrid optimizers to overcome this challenge. The primary motivation is to design an algorithm that avoids getting "stuck" in the numerous local minima that characterize complex, high-dimensional loss landscapes. We first establish a robust testbed by implementing a custom `loss_non_convex_trapped` function, which is explicitly designed to trap naive gradient-based methods. We then implement two baseline optimizers: a standard Gradient Descent (local exploitation) and a custom-built, gradient-free Attention Search (global exploration). The core of this project lies in the design and implementation of four distinct hybrid optimization strategies that combine these two baselines. These strategies range from a simple reactive escape mechanism (Hybrid 1) and deterministic alternation (Hybrid 2) to more advanced methods like GD restarting (Hybrid 3) and a "meta-optimization" approach where Attention Search dynamically selects the optimal learning rate for GD at each step (Hybrid 4). This report details the inner workings of each component, analyzing its trade-offs and comparing our *heuristic-based* hybrid approach to the *data-driven, learned* approach of advanced research like the "Optimus" (Transformer-Based Learned Optimization) paper.

Contents

1 Introduction & Motivation

1.1 Context: The "Optimus" Learned Optimizer

1.2 The Challenge: Non-Convex Optimization

1.3 Our Hypothesis

2 Methodology: Components & Baselines

2.1 The Testbed: `loss_non_convex_trapped`

Crux of the Code

Analysis (Why this is a Good Testbed)

2.2 Baseline 1: Gradient Descent (Local Exploitation)

Analysis (Why this is Better/Worse)

2.3 Baseline 2: Attention Search (Global Exploration)

Crux of the Code

Analysis (Why this is Better/Worse)

3 The Hybrid Optimization Strategies

3.1 Hybrid 1: GD with Reactive Escape

Crux of the Code

Analysis (Why this is Better/Worse)

3.2 Hybrid 2: Alternating Optimization

Crux of the Code

Analysis (Why this is Better/Worse)

3.3 Hybrid 3: Attention-Restarted GD

Crux of the Code

Analysis (Why this is Better/Worse)

3.4 Hybrid 4: Attention as a Meta-Optimizer (for Learning Rate)

Crux of the Code

Analysis (Why this is Better/Worse)

4 Expected Results & Analysis

5 Conclusion