KATHMANDU UNIVERSITY

School of Science

Department of Computer Science and Engineering

Visualization of Optimization Algorithms: Gradient Descent vs Adam

MINI PROJECT SUBMISSION

Course Code: COMP 342

Submitted By:

Prajwal Ghimire, (Roll No: 22)
Saksham Humagain, (Roll No: 25)

Submitted To:

Mr. Dhiraj Shrestha
Department of Computer Science and Engineering

Date of Submission: February 7, 2026

Abstract

Optimization is a critical task in machine learning and computational science. Gradient-based optimization methods such as **Gradient Descent (GD)** and **Adam** are widely used to minimize complex loss functions efficiently. This project implements a 3D visualization of both optimizers on a non-linear loss surface. The animation allows observation of each optimizer's trajectory, convergence speed, and stability, providing intuition on how these algorithms explore and minimize a function.

1 Introduction

Optimization algorithms are fundamental in machine learning, physics simulations, and mathematical modeling. They iteratively update parameters to minimize a loss function.

1.1 Gradient Descent (GD)

Gradient Descent updates parameters in the direction of the negative gradient:

$$\theta_{t+1} = \theta_t - \eta \nabla L(\theta_t) \tag{1}$$

where η is the learning rate. GD is simple to implement but can converge slowly on complex surfaces with multiple minima.

1.2 Adam Optimizer

Adam combines **momentum** and **adaptive learning rates** to achieve faster and smoother convergence:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \nabla L(\theta_t) \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla L(\theta_t))^2 \tag{3}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{4}$$

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{5}$$

Adam often outperforms GD, especially on non-linear or noisy surfaces.

2 Project Objective & Scope

2.1 Objective

The objective of this project is to implement and visualize the behavior of Gradient Descent and Adam on a 3D non-linear loss surface, highlighting differences in convergence patterns and stability.

2.2 Scope

- Define a non-linear loss function with multiple local minima:

```
def loss(x, y):
    return x**2 + y**2 + np.sin(3*x)*np.sin(3*y)
```

- Implement Gradient Descent and Adam optimizers in Python.
- Animate the optimizers' trajectories on a 3D surface using Matplotlib.
- Compare convergence speed, path smoothness, and stability visually.

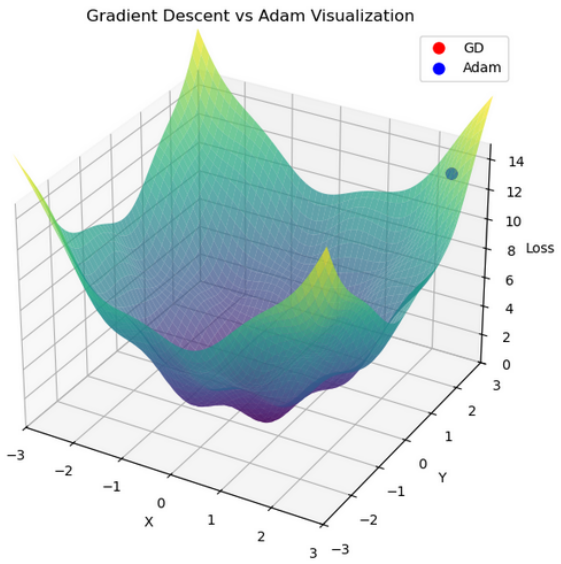

Figure 1: 3D loss surface with initial positions of GD (red) and Adam (blue).

3 Methodology / Implementation

3.1 Gradient and Loss Function

The gradient is computed analytically:

```
def gradient(x, y):
    dx = 2*x + 3*np.cos(3*x)*np.sin(3*y)
    dy = 2*y + 3*np.sin(3*x)*np.cos(3*y)
    return np.array([dx, dy])
```

3.2 Optimizer Implementation

```
def gradient_descent_step(pos, lr=0.03):
    return pos - lr * gradient(*pos)

def adam_step(pos, m, v, t, lr=0.08, beta1=0.9, beta2=0.999, eps=1e-8):
    g = gradient(*pos)
    m = beta1*m + (1-beta1)*g
    v = beta2*v + (1-beta2)*(g**2)
    m_hat = m / (1-beta1**t)
    v_hat = v / (1-beta2**t)
    return pos - lr*m_hat/(np.sqrt(v_hat)+eps), m, v
```

3.3 Animation Implementation

- Dots represent the current positions of the optimizers.
- Dashed lines represent the path (history) of the optimizer on the surface.
- The animation runs for multiple frames to demonstrate convergence.

4 Results / Observations

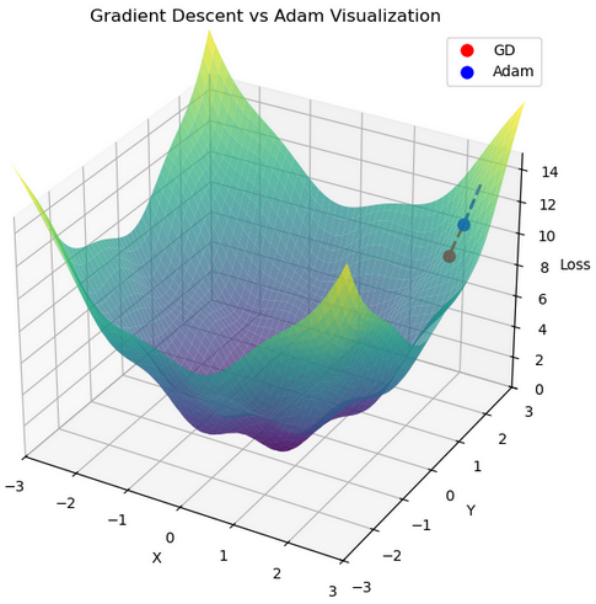

Figure 2: Early frame: GD and Adam starting positions.

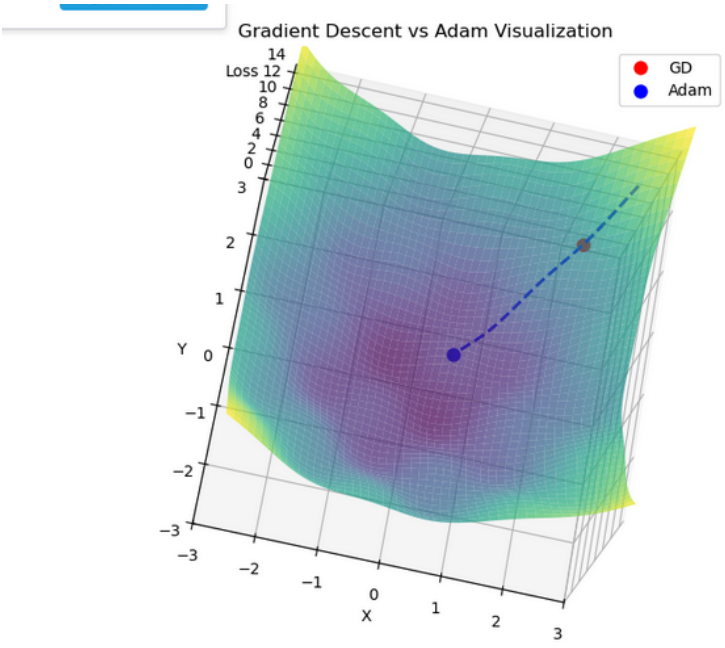

Figure 3: Mid frame: Adam converging faster than GD.

4.1 Observations

- Adam converges faster than GD, especially on non-linear surfaces.

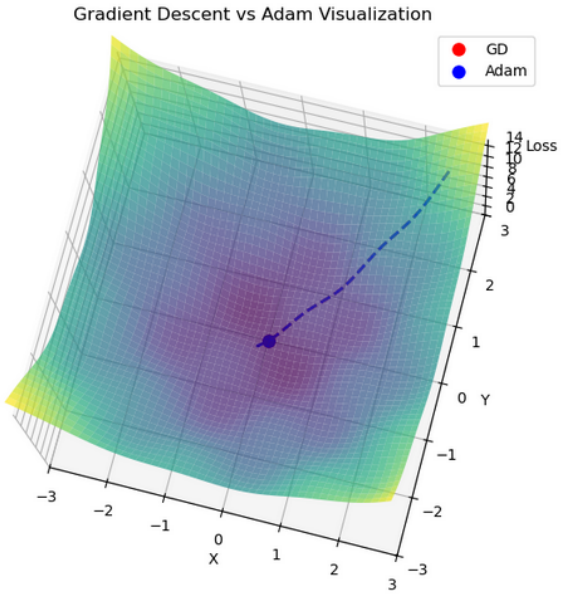

Figure 4: Final frame: Both optimizers near the global minimum.

- GD trajectory may oscillate in shallow regions before reaching minimum.
- Adam's momentum and adaptive learning rates produce smoother trajectories.
- Visualization helps understand optimizer behavior beyond numerical results.

5 Conclusion

This project visually demonstrated how Gradient Descent and Adam traverse a 3D non-linear loss surface. Adam converges faster and smoother due to adaptive learning rates and momentum. Visualizations improve understanding of optimizer behavior, which is important for tuning machine learning models.

5.1 Future Work

- Compare additional optimizers such as RMSProp and AdaGrad.
- Add interactive sliders to adjust learning rates in real time.
- Include 2D contour plots to provide additional intuition.

References

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.
2. Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *ICLR*.
3. Matplotlib Documentation: `https://matplotlib.org/stable/gallery/mplot3d/index.html`
4. NumPy Documentation: `https://numpy.org/doc/stable/`