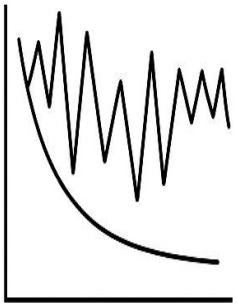
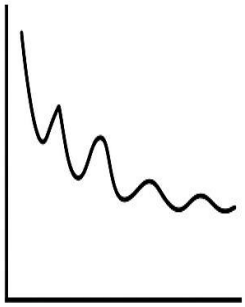
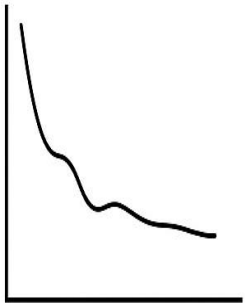
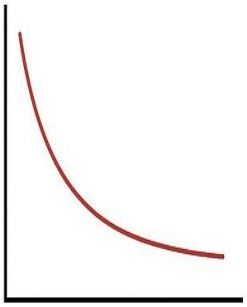
Experimental Analysis of Gradient-Based Optimizers

Gradient Descent	Gradient Descent with Momentum	RMSProp	Adam
			
70	80	90	91
• Simple • Slow convergence	• Faster • Reduces oscillations	• Adaptive • Faster	• Adaptive • Very fast

Comparative Analysis of Gradient-Based Optimizers

This project presents an experimental comparison between four gradient-based optimization algorithms:
- Gradient Descent (GD)
- Gradient Descent with Momentum (Momentum)
- RMSProp
- Adam

The goal is to demonstrate the progression from the basic Gradient Descent to more advanced optimizers, explaining why each improvement was introduced and what advantages and disadvantages it brings.

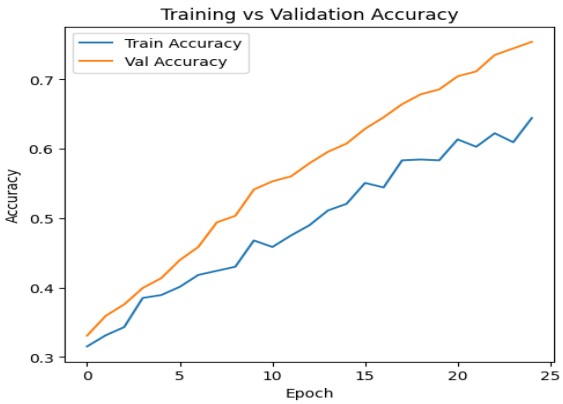
1. Gradient Descent (GD)

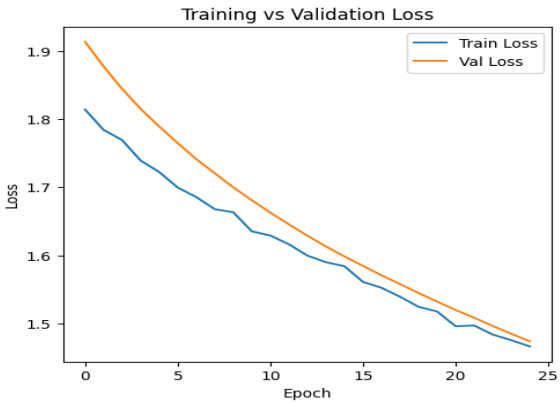
Concept:
Gradient Descent updates model parameters by moving in the opposite direction of the gradient of the loss function with respect to the parameters. The step size is determined by the learning rate.

Advantages:
- Simple and easy to implement.
- Works well for convex optimization problems.

Disadvantages:
- Slow convergence for large datasets or poorly scaled data.
- Can get stuck in local minima or saddle points.
- Sensitive to learning rate choice.

Experimental Results:





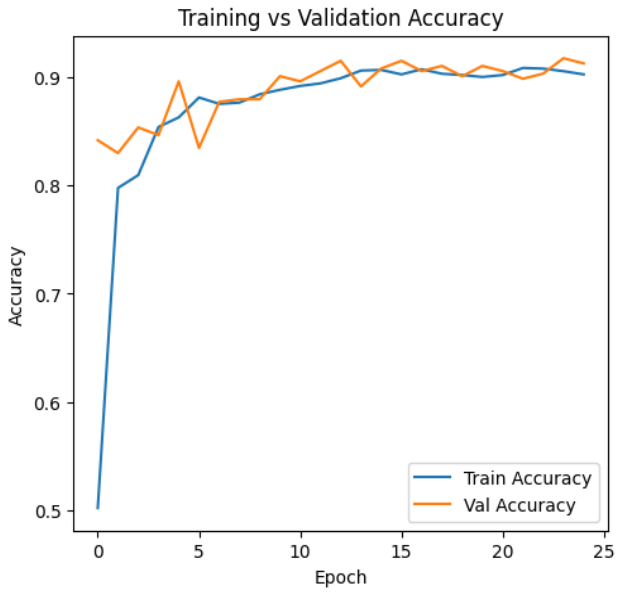
Gradient Descent with Momentum

Why Momentum was introduced:
Standard GD suffers from slow convergence in narrow valleys and oscillates heavily when the cost surface has steep and flat directions. Momentum helps by adding a fraction of the previous update to the current update, allowing the optimizer to build speed in consistent directions.

Advantages:
- Faster convergence in regions with gentle slopes.
- Reduces oscillations in steep directions.
- Helps escape shallow local minima.

Disadvantages:
- Adds one extra hyperparameter (momentum factor β).
- If momentum is too high, it can overshoot the minimum.

Experimental Results:



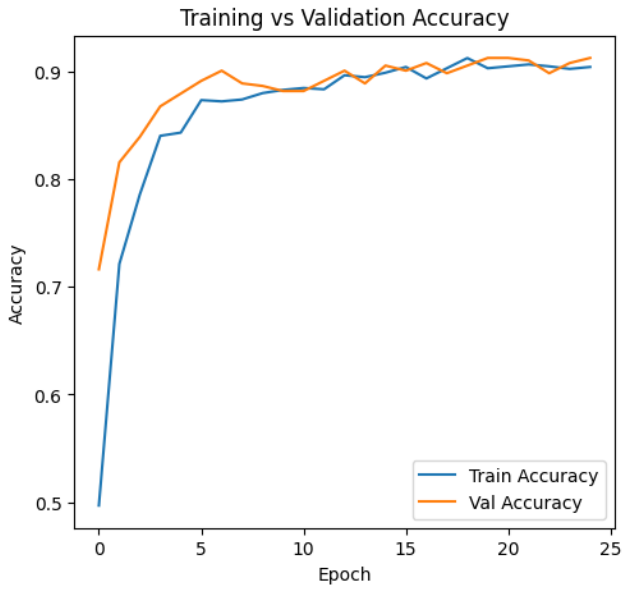
RMSProp

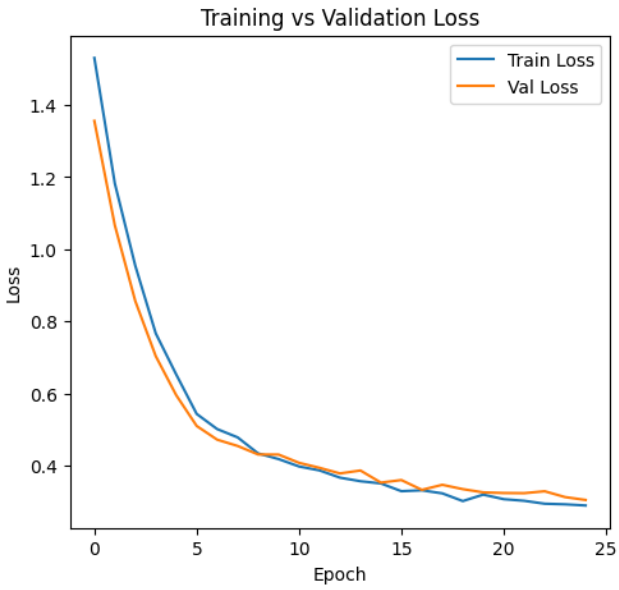
Why RMSProp was introduced:
Momentum improves direction stability but does not adapt the learning rate for each parameter. RMSProp introduces adaptive learning rates by dividing the learning rate by an exponentially decaying average of squared gradients, allowing faster convergence and better handling of varying parameter scales.

Advantages:
- Adapts learning rates individually for each parameter.
- Performs well on non-stationary problems.
- Faster convergence than GD and Momentum for many tasks.

Disadvantages:
- Additional hyperparameter (decay rate β2).
- Can still get stuck in local minima without momentum-like effects.

Experimental Results:





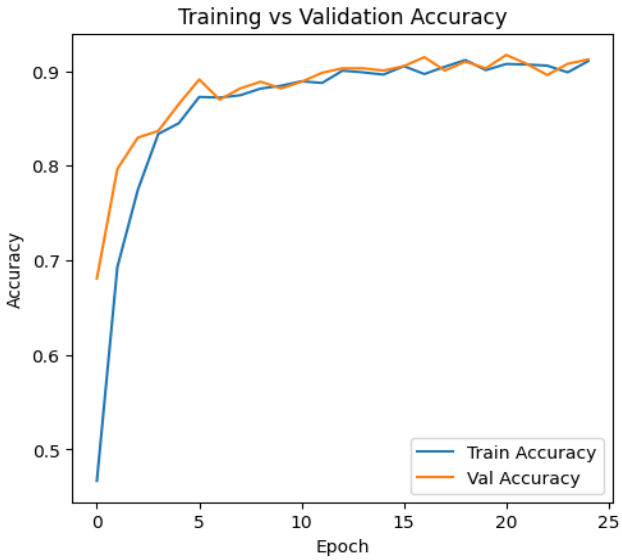
Adam (Adaptive Moment Estimation)

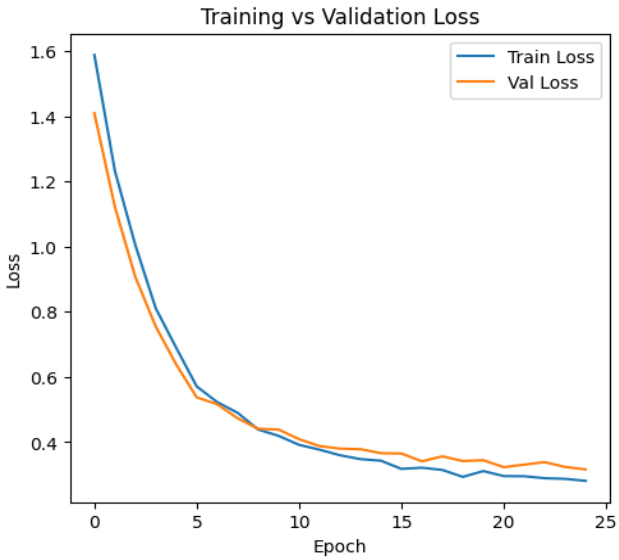
Why Adam was introduced:
Adam combines Momentum (first moment estimation) with RMSProp (second moment estimation) to leverage the strengths of both. It adapts learning rates individually while also using momentum to accelerate convergence.

Advantages:
- Combines benefits of Momentum and RMSProp.
- Works well out-of-the-box with minimal tuning.
- Handles sparse gradients efficiently.
- Robust to noisy datasets.

Disadvantages:
- Can sometimes generalize worse than SGD in certain cases.
- More computationally intensive than plain GD.

Experimental Results:





Summary Table

Optimizer	Adaptivity	Momentum Effect	Convergence Speed	Key Weakness
GD	No	No	Slow	Learning rate sensitivity
Momentum	No	Yes	Faster than GD	Overshooting risk
RMSProp	Yes	No	Fast adaptive	Lacks momentum
Adam	Yes	Yes	Very fast	May generalize worse