Automatic Differentiation (Autograd)

Automatic differentiation computes derivatives algorithmically by applying the chain rule to elementary operations, enabling efficient gradient computation without manual derivation

Manual

Derive gradients by hand using calculus

- ✓ Full control
- ✓ Symbolic form
- ✗ Error-prone
- ✗ Time-consuming
- ✗ Not scalable

Numerical

Approximate using finite differences

- ✓ Easy to implement
- ✗ Approximation errors
- ✗ Slow (O(n) evaluations)
- ✗ Numerical instability

Automatic (AD)

Compute exact derivatives automatically

- ✓ Exact derivatives
- ✓ Efficient O(1) overhead
- ✓ Scalable to complex models
- ✓ No manual derivation

PyTorch Example

```
import torch # Create tensors with gradient
tracking x = torch.tensor([2.0],
requires_grad=True) w = torch.tensor([0.5],
requires_grad=True) b = torch.tensor([1.0],
requires_grad=True) # Forward pass (build
computation graph) y = torch.sigmoid(w * x + b)
```

Key Features

1. **Dynamic Graph:** Built during forward pass
2. **Chain Rule:** Applied automatically layer by layer

```
loss = (y - 1)**2 # Backward pass (compute gradients) loss.backward() # Access gradients print(w.grad) # ∂loss/∂w print(b.grad) # ∂loss/∂b
```

Efficient: Reuses intermediate values

4 **Flexible:** Supports arbitrary operations

💡 **Why It Matters:** Autograd eliminates the need for manual gradient derivation, making deep learning accessible and enabling rapid experimentation with complex architectures.

🔥 **PyTorch**
Dynamic graphs

🧮 **TensorFlow**
Eager execution

🌊 **JAX**
Functional AD