Popular Optimizers in Deep Learning (GD, SGD, Adam, etc.)

High-Level Summary

Optimizers are algorithms that **update the model's parameters** (weights, biases) to minimize the loss.

They differ in **speed, stability, memory use, and convergence behavior**.

Detailed Explanation

🔷 1. Gradient Descent (Batch GD)

- **What it is**: Uses the **entire dataset** to compute gradients before updating weights.
- **Update rule**:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta)$$

- **Why used**: Simple, stable for convex functions.
- **Limitations**: Very slow for large datasets (one update per full dataset pass).
- **When to use**: Small datasets (fits in memory). Rare in modern deep learning.

🔷 2. Stochastic Gradient Descent (SGD)

- **What it is**: Updates parameters using **one sample at a time**.
- **Update rule**:

$$\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(x_i, y_i; \theta)$$

- **Why used**: Much faster than GD, introduces **noise** that helps escape local minima.

- **Limitations**: Very noisy, loss oscillates.
- **When to use**: Streaming data, online learning.

🔷 3. Mini-Batch SGD (default in practice)

- **What it is**: Updates using a **small batch** of samples (e.g., 32, 64).
- **Why used**:
  - More efficient than GD.
  - Smoother than pure SGD.
  - Well-suited for GPUs.
- **When to use**: Standard choice for deep learning training.

🔷 4. SGD with Momentum

- **What it is**: Adds a **velocity term** to smooth updates, accumulates past gradients.
- **Update rules**:

$$v_t = \beta v_{t-1} + (1 - \beta)\nabla_\theta \mathcal{L}$$

$$\theta \leftarrow \theta - \eta v_t$$

- **Why used**: Speeds up convergence, prevents oscillations.
- **When to use**: Deep CNNs (ResNet, VGG).

🔷 5. RMSProp

- **What it is**: Adapts learning rate per parameter by dividing by a moving average of squared gradients.
- **Update rule**:

$$s_t = \beta s_{t-1} + (1 - \beta)(\nabla_\theta \mathcal{L})^2$$

$$\theta \leftarrow \theta - \eta \frac{\nabla_\theta \mathcal{L}}{\sqrt{s_t + \epsilon}}$$

- **Why used**: Handles non-stationary problems well, stabilizes learning.
- **When to use**: RNNs, unstable training tasks.

🔷 6. Adam (Adaptive Moment Estimation)

- **What it is**: Combines **Momentum + RMSProp** (first + second moment estimates).
- **Update rules**:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1)\nabla_\theta \mathcal{L}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2)(\nabla_\theta \mathcal{L})^2$$

  Bias correction:

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

  Update:

$$\theta \leftarrow \theta - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

- **Why used**:
  - Fast convergence.
  - Works well with little tuning.
  - Handles sparse gradients.
- **When to use**: Default for NLP, Transformers, GANs.

🔷 7. AdamW

- **What it is**: Adam with **decoupled weight decay** (fixes over-regularization issue).

- **Why used**: More stable than Adam, better generalization.
- **When to use**: Transformers (BERT, GPT, ViTs).

Analogy

- **GD**: Like checking the **entire class's exams** before adjusting teaching.
- **SGD**: Like checking **one student's exam** after each question.
- **Mini-batch SGD**: Like checking a **small group** of exams before changing teaching.
- **Momentum**: Like a ball rolling downhill — builds speed in the right direction.
- **RMSProp**: Like adjusting your stride size based on the terrain.
- **Adam**: Combines momentum (rolling ball) + adaptive stride (terrain-aware).

Comparison Table

Optimizer	Idea	Pros	Cons	When to Use
GD	Full dataset update	Stable, exact	Slow, memory heavy	Small datasets
SGD	One sample update	Fast, helps escape minima	Noisy updates	Online learning
Mini-batch SGD	Small batch	Efficient, stable	Needs batch tuning	Deep learning default
Momentum SGD	Adds velocity	Faster convergence	Needs momentum tuning	Deep CNNs
RMSProp	Scales lr by past gradients	Handles exploding/vanishing gradients	May generalize poorly	RNNs
Adam	Momentum + RMSProp	Fast, works out of box	May overfit, bad generalization	NLP, GANs
AdamW	Adam + weight decay	Better generalization	Slightly slower	Transformers, SOTA models

Key Takeaway

- Use **Mini-batch SGD + Momentum** for CNNs.
- Use **Adam/AdamW** for Transformers, NLP, GANs.
- Use **RMSProp** for RNNs or unstable cases.
- Use **GD** only for small datasets.
