Optimizers are crucial in training neural networks as they adjust the weights to minimize the loss function. Here are some common optimizers used in neural networks:

**1. Gradient Descent (GD)**

* **Batch Gradient Descent**: Computes the gradient using the entire dataset. It's accurate but slow and requires a lot of memory.
* **Stochastic Gradient Descent (SGD)**: Computes the gradient using a single sample. It's faster and more memory-efficient but introduces noise, leading to more fluctuations in the loss function.
* **Mini-batch Gradient Descent**: A compromise between batch GD and SGD, computing the gradient on a small batch of samples. It balances the efficiency and stability of the updates.

**2. Momentum**

* **Momentum**: Enhances SGD by adding a fraction of the previous update to the current update. It helps accelerate gradients in the right direction, leading to faster converging.
* **Nesterov Accelerated Gradient (NAG)**: A variant of momentum that looks ahead by calculating the gradient of the loss function at the anticipated next point, leading to more accurate updates.

**3. Adaptive Learning Rate Methods**

* **AdaGrad**: Adapts the learning rate for each parameter based on the past gradients. Parameters with larger gradients get smaller updates and vice versa. It's useful for sparse data but may decrease the learning rate too much.
* **RMSprop**: Modifies AdaGrad by using a moving average of squared gradients to normalize the gradient. It helps resolve the diminishing learning rate issue of AdaGrad.
* **Adam (Adaptive Moment Estimation)**: Combines the ideas of momentum and RMSprop by computing adaptive learning rates for each parameter. It uses running averages of both the gradients and the squared gradients. Adam is efficient and works well in practice.
* **Adadelta**: An extension of AdaGrad that restricts the window of accumulated past gradients to a fixed size. It addresses the diminishing learning rate issue without needing an external learning rate.
* **Adamax**: A variant of Adam based on the infinity norm, which provides more stable updates when the gradients are large.
* **Nadam (Nesterov-accelerated Adam)**: Combines Adam and NAG, incorporating the Nesterov momentum into Adam's update rule for potentially better convergence.

**4. Advanced Optimizers**

* **SGD with Warm Restarts**: Introduces periodic restarts to the learning rate, encouraging the optimizer to escape local minima and explore more.
* **L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb–Shanno)**: An optimization algorithm suitable for batch training with a memory-efficient implementation of the BFGS algorithm, which approximates the Hessian matrix.

**Practical Tips**

* **Default Choices**: Adam is a popular choice due to its adaptive nature and efficiency. It's often a good starting point.
* **Experimentation**: It's beneficial to experiment with different optimizers, learning rates, and schedules based on the specific problem and dataset.
* **Learning Rate Schedulers**: Using learning rate schedulers (e.g., ReduceLROnPlateau, ExponentialDecay) can help improve performance by adjusting the learning rate dynamically during training.

Each optimizer has its strengths and weaknesses, and the choice often depends on the specific characteristics of the problem, the architecture of the neural network, and the nature of the data.

**Adam Optimizer**

   