Gradients Not Cleared before Backward Propagation

Context

In PyTorch, `optimizer.zero_grad()` **clears the old gradients** from last step, `loss_fn.backward()` **does the back propagation**, and `optimizer.step()`**performs weight update** using the gradients.

Problem

If `optimizer.zero_grad()` **is not used before** `loss_fn.backward()`, the gradients **will be accumulated from all** `loss_fn.backward()` **calls** and it will lead to the **gradient explosion**, which fails the training.

Solution

Developers **should not forget to use** `optimizer.zero_grad()` **before** `loss_fn.backward()`.

Existing Stage	Effect
Model Training	Error-prone

Example

Python

```
# PyTorch
# 3. Define a Loss function and optimizer  [...]
# 4. Train the network
for epoch in range(2):  #  training loop multiple times
   running_loss = 0.0
   for i, data in enumerate(trainloader, 0):
       # get the inputs; data is a list of [inputs, labels]
       inputs, labels = data
+       # zero the parameter gradients
+       optimizer.zero_grad()
       # forward + backward + optimize
       outputs = net(inputs)
       loss = criterion(outputs, labels)
       loss.backward()
       optimizer.step()
```