Gradient Norm (L2) in Neural Networks

Why compute global gradient norm?

- Each parameter (weights, biases, etc.) has its own gradient tensor.
- What matters for training stability is the **overall size of the gradient update** across the entire model.
- The **global L2 norm** treats all gradients as one big vector and measures its length.
- This is useful for:
  - Detecting **exploding/vanishing gradients**
  - Applying **gradient clipping** (to prevent unstable updates)

Formula

If gradients are split across parameters $g^{(1)}, g^{(2)}, \ldots, g^{(n)}$:

$$\text{Global L2 norm} = \sqrt{\sum_{i=1}^{n} \|g^{(i)}\|_2^2}$$

This is equivalent to flattening all gradients into a single vector and taking its L2 norm.

Example

Suppose two parameter tensors have gradients:

- Param 1: `[3, 4]` → L2 norm = $\sqrt{3^2 + 4^2}$ = 5
- Param 2: `[6, 8]` → L2 norm = $\sqrt{6^2 + 8^2}$ = 10

Incorrect way (just summing norms):

5 + 10 = 15

Correct global L2 norm:

Flatten all gradients → `[3, 4, 6, 8]`

$$\sqrt{3^2 + 4^2 + 6^2 + 8^2} = \sqrt{125} \approx 11.18$$

✅ This is the true magnitude of the gradient vector.

PyTorch Code Snippet

```
total_norm = 0
for p in model.parameters():
    if p.grad is not None:
        param_norm = p.grad.data.norm(2)   # L2 norm for one param
        total_norm += param_norm.item()**2
total_norm = total_norm**0.5   # Global gradient L2 norm
```

📌 **Key Point:**

We do this because the optimizer applies all parameter updates together — the global norm tells us the **true step size** being taken.
