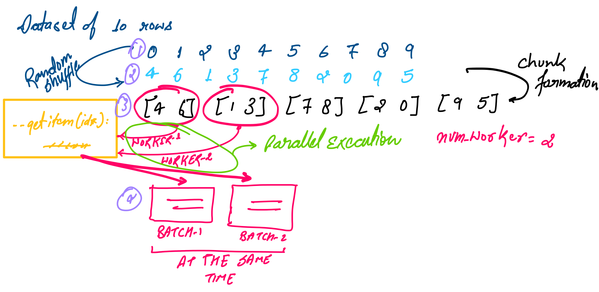
# 6.4 NOTE ABOUT PARALLELIZATION

03 September 2025 02:32 AM Avinash Yadav




Imagine the entire data loading and training process for one epoch with ***num_workers=4***:

## Assumptions:

- **Total samples:** 10,000
- **Batch size:** 32
- **Workers (*num_workers*):** 4
- **Approximately 312 full batches per epoch** (*10000 / 32 ≈ 312*)

## Workflow:

### 1. Sampler and Batch Creation (Main Process):

- Before training starts for the epoch, the DataLoader's sampler generates a shuffled list of all 10,000 indices. These are then grouped into 312 batches of 32 indices each. All these batches are queued up, ready to be fetched by workers.

### 2. Parallel Data Loading (Workers):

- At the start of the training epoch, you run a training loop like:

```
for batch_data, batch_labels in dataloader:
    # Training logic
```

- Under the hood, as soon as you start iterating over dataloader, it dispatches the first four batches of indices to the four workers:
  - Worker #1 loads batch 1 (indices [batch_1_indices])
  - Worker #2 loads batch 2 (indices [batch_2_indices])
  - Worker #3 loads batch 3 (indices [batch_3_indices])
  - Worker #4 loads batch 4 (indices [batch_4_indices])

  Each worker:
  - Fetches the corresponding samples by calling ***__getitem__(index)*** on the dataset for each index in that batch.
  - Applies any defined transforms and passes the samples through ***collate_fn*** to form a single batch tensor.

### 3. First Batch Returned to Main Process:

- Whichever worker finishes first sends its fully prepared batch (e.g., batch 1) back to the main process.
- As soon as the main process gets this first prepared batch, it yields it to your training loop, so your code for ***batch_data***, ***batch_labels*** in ***dataloader:receives (batch_data, batch_labels)*** for the first batch.

### 4. Model Training on the Main Process:

- While you are now performing the forward pass, computing loss, and doing backpropagation on the first batch, the other three workers are still preparing their batches in parallel.
- By the time you finish updating your model parameters for the first batch, the DataLoader likely has the second, third, or even more batches ready to go (depending on processing speed and hardware).

### 5. Continuous Processing:

- As soon as a worker finishes its batch, it grabs the next batch of indices from the queue.
- For example, after Worker #1 finishes with batch 1, it immediately starts on batch 5. After Worker #2 finishes batch 2, it takes batch 6, and so forth.
- This creates a pipeline effect: at any given moment, up to 4 batches are being prepared concurrently.

### 6. Loop Progression:

- Your training loop simply sees:

```
for batch_data, batch_labels in dataloader:
    # forward pass
    # loss computation
    # backward pass
    # optimizer step
```

- Each iteration, it gets a new, ready-to-use batch without long I/O waits, because the workers have been pre-loading and processing data in parallel.

### 7. End of the Epoch:

- After ~312 iterations, all batches have been processed. All indices have been consumed, so the DataLoader has no more batches to yield.
- The epoch ends. If ***shuffle=True***, on the next epoch, the sampler reshuffles indices, and the whole process repeats with workers again loading data in parallel