University of Tehran

Faculty of New Sciences and Technologies Department

of Mechatronics

Practice of Artificial Neural Networks

By:

Omid Moradi

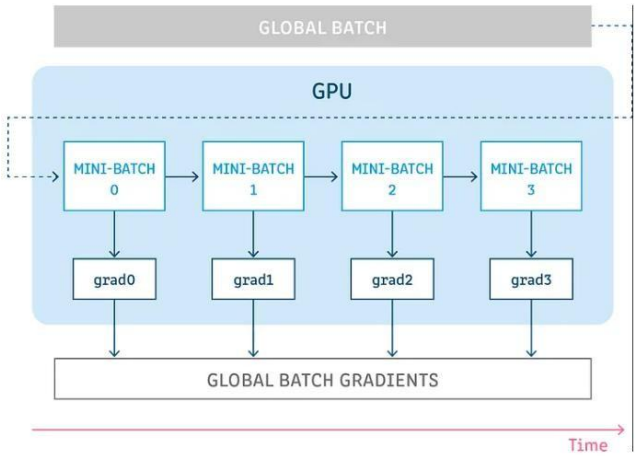
Lecturer:

Dr. Masoudnia

Fall 2024

Q1- When GPU memory is limited, you can accumulate gradients over multiple smaller mini-batches before performing a weight update. This approach effectively simulates using a larger batch size while keeping memory usage low. Explain it and provide a sample code to do so.

Gradient accumulation means running a configured number of steps without updating the model variables while accumulating the gradients of those steps and then using the accumulated gradients to compute the variable updates.



Gradient accumulation

Running some steps without updating any of the model variables is the way we logically split the batch of samples into a few mini-batches. The batch of samples that is used in every step is effectively a mini-batch, and all the samples of those steps combined are effectively the global batch.

By not updating the variables at all those steps, we cause all the mini-batches to use the same model variables for calculating the gradients. This is mandatory to ensure the same gradients and updates are calculated as if we were using the global batch size.

```python
import torch
import torch.optim as optim
import torch.nn as nn

model = MyModel()
optimizer = optim.Adam(model.parameters(), lr=1e-4)
criterion = nn.CrossEntropyLoss()

large_batch_size = 128
mini_batch_size = 32

accumulation_steps = large_batch_size // mini_batch_size

for epoch in range(num_epochs):
    optimizer.zero_grad()
    for i, (inputs, targets) in enumerate(data_loader):
        outputs = model(inputs)
        loss = criterion(outputs, targets)
        loss = loss / accumulation_steps
        loss.backward()
        if (i + 1) % accumulation_steps == 0:
            optimizer.step()
            optimizer.zero_grad()
```

Q2- Compare different batch normalization techniques, including standard BN, Layer Normalization, Instance Normalization and Group Normalization, discussing their advantages and limitations compared to each other.

1. Batch Normalization (BN)

Batch Normalization is applied across the mini-batch, normalizing the activations over the batch dimension and each feature channel. For each batch, BN calculates the mean and variance of activations, then normalizes them by subtracting the batch mean and dividing by the batch standard deviation.

Advantages:

- **Internal Covariate Shift Reduction:** Helps stabilize training by keeping input distributions to each layer consistent.
- **Improved Learning:** Allows for higher learning rates, speeding up convergence.
- **Regularization Effect:** Introduces slight regularization by adding noise to the learning process through mini-batch variance.

Disadvantages:

- **Mini-batch Dependency:** The performance of BN degrades with very small batch sizes, as the estimates of batch statistics become unreliable.
- **Not Ideal for Sequence Models:** In tasks like NLP or reinforcement learning, where batch sizes are small or non-existent, BN is less effective.

Best For: Convolutional neural networks (CNNs) with large batch sizes, such as image classification tasks.

---

2. Layer Normalization (LN)

Layer Normalization is applied across the features of each individual data point rather than across a minibatch. It computes the mean and variance for each input data point independently, normalizing all features within the same layer.

Advantages:

- **Independent of Batch Size:** LN can handle small batch sizes or even single-instance mini-batches effectively.
- **Good for Recurrent Networks:** Works well with sequential models such as RNNs or transformers, where long-term dependencies are critical.

Disadvantages:

- **Less Effective for CNNs:** LN does not capture spatial correlations in convolutional layers, leading to suboptimal performance in CNN architectures.
- **Slower Training:** Compared to BN, the regularization effect is weaker, potentially leading to slower training.

Best For: Sequence models (e.g., RNNs, transformers) in NLP, where inputs vary over time and batch sizes are small.

---

3. Instance Normalization (IN)

Instance Normalization is applied independently to each sample and each feature channel. It computes the mean and variance per channel for each individual data point, effectively normalizing each image in a minibatch separately.

Advantages:

- **Effective for Style Transfer:** IN normalizes each image independently, making it well-suited for tasks like style transfer, where preserving unique characteristics of each image is important.
- **Removes Image-Specific Biases:** Helps in scenarios where the global mean and variance of the image are not crucial, such as generative models.

Disadvantages:

- **Poor for Discriminative Tasks:** IN lacks the ability to generalize as well in tasks like classification, where batch-wide statistics are useful.
- **Less Batch-Level Representation:** IN removes useful cross-batch information, which could otherwise aid in learning.

Best For: Image generation tasks, such as style transfer, where preserving individual sample characteristics is important.

---

4. Group Normalization (GN)

Group Normalization splits the feature channels into groups, and normalizes the activations within each group. This technique strikes a balance between instance-level and batch-level normalization, offering more flexibility.

Advantages:

- **Independent of Batch Size:** Like LN, GN is independent of batch size, making it effective even for small-batch or mini-batch training.
- **Works Well for Vision Tasks:** GN can improve performance in vision tasks when batch normalization isn't ideal, such as in object detection or segmentation models with small batches.
- **Better Generalization:** It combines the benefits of both batch-wide and instance-level statistics, leading to more robust learning in a variety of scenarios.

Disadvantages:

- **Group Size Hyperparameter:** Choosing the right number of groups is crucial, and improper group sizes can degrade performance.
- **Less Effective for Small Tasks:** In tasks with very few channels, GN may not offer significant improvements over other methods.

Best For: Vision tasks, especially when working with small batch sizes where Batch Normalization struggles.