PUBLIC LECTURE 5 CHEATSHEET

Gradient estimates and the training clock

Choose how much data measures each step, then name the unit in which training progress is counted.

Backprop measures a slope. Optimization chooses which data measure it and how far the parameters move.

1 · Objective and vanilla update

For N examples and parameters $\boldsymbol{\theta}$:

$$\ell_{i(\boldsymbol{\theta})} = \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), y_i), \quad \mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell_{i(\boldsymbol{\theta})}.$$

Full-batch gradient descent uses

$$\boldsymbol{g}_t = \nabla\mathcal{L}(\boldsymbol{\theta}_t), \quad \boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}_t.$$

$\eta > 0$ is the learning rate. The gradient points uphill; subtracting it moves locally downhill.

2 · Full batch, SGD, minibatch

Method	B	Updates / epoch	Noise
full batch	N	1	none for empirical loss
SGD	1	N	highest
minibatch	$1 < B < N$	$\lceil \frac{N}{B} \rceil$	intermediate

Minibatch estimator:

$$\hat{\boldsymbol{g}}_t = \tfrac{1}{B_t}\textstyle\sum_{i\in\mathcal{B}_t} \nabla\ell_{i(\boldsymbol{\theta}_t)}.$$

3 · Name the clock

If N examples are processed in batches of size B:

$$K = \lceil \tfrac{N}{B} \rceil \quad \text{iterations per epoch.}$$

EPOCH one pass through the data
ITERATION one minibatch loop body
UPDATE one executed parameter change

Usually iteration = update. Gradient accumulation and skipped optimizer steps are important exceptions.

4 · Why a sampled gradient can work

Draw I_t uniformly from $\{1, \ldots, N\}$ and use

$$\hat{\boldsymbol{g}}_t = \nabla\ell_{I_t}(\boldsymbol{\theta}_t).$$

Holding $\boldsymbol{\theta}_t$ fixed,

$$\mathbb{E}[\hat{\boldsymbol{g}}_t \mid \boldsymbol{\theta}_t] = \frac{1}{N}\sum_i \nabla\ell_{i(\boldsymbol{\theta}_t)} = \nabla\mathcal{L}(\boldsymbol{\theta}_t).$$

The estimator is **unbiased**: correct at its centre across repeated draws.

What unbiased does not mean

One sampled gradient need not be close to the full gradient and one valid SGD step need not reduce the full loss. It only satisfies

$$\mathbb{E}[\Delta\boldsymbol{\theta}_t \mid \boldsymbol{\theta}_t] = -\eta\nabla\mathcal{L}(\boldsymbol{\theta}_t).$$

Larger minibatches preserve the same centre while averaging away some sample disagreement.

5 · Batch size is a systems-and-statistics knob

Smaller B
more updates per epoch
noisier estimates
less parallel work / update
may regularize implicitly

Larger B
fewer updates per epoch
less estimator spread
more parallelism / update
may need rate retuning

Compare experiments by examples processed or optimizer updates, not only by epochs.

"SGD" in libraries often means the minibatch-SGD family. The dataloader's batch size tells you which member you ran.

PUBLIC LECTURE 5 CHEATSHEET

Step size, geometry, and diagnosis

An exact gradient can still make poor progress when one scalar learning rate must serve very different directions.

1 · The learning rate scales every coordinate

$$\Delta\boldsymbol{\theta}_t = -\eta\hat{\boldsymbol{g}}_t.$$

Too small: safe but slow. Too large: overshoots, oscillation, or divergence. A useful rate makes early progress without destroying stability.

Diagnostic: plot training loss against optimizer updates and inspect the actual rate used at each update.

2 · Ravines create a speed-stability trade-off

A transparent anisotropic loss is

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{2}(\theta_1 - 1)^2 + 50(\theta_2 + 0.5)^2,$$

$$\nabla\mathcal{L} = (\theta_1 - 1, 100(\theta_2 + 0.5)).$$

One direction is $100\times$ steeper. A rate large enough to move quickly along θ_1 can repeatedly cross the narrow valley in θ_2; a stable small rate can crawl along the valley floor.

Concrete first step

At $\boldsymbol{\theta}_0 = (-1.4, 0.35)$, $\boldsymbol{g}_0 = (-2.4, 85)$. With $\eta = 0.0195$:

$$\Delta\boldsymbol{\theta}_0 = (0.0468, -1.6575), \quad \boldsymbol{\theta}_1 = (-1.3532, -1.3075).$$

The step is downhill locally yet crosses the optimum $\theta_2^\star = -0.5$ in the steep direction.

3 · Fix geometry before blaming the optimizer

- ✓ Standardize input features when scales differ strongly.
- ✓ Inspect loss contours or per-parameter update scales on small problems.
- ✓ Check exploding gradients, invalid targets, and reduction factors.
- ✓ Retune η after changing batch size or normalization.

4 · A disciplined batch-size comparison

Hold the model, data order, objective reduction, and total example budget fixed. For each B log:

- **A** examples processed
- **B** optimizer updates
- **C** wall-clock time and device utilization
- **D** training and validation metrics
- **E** gradient or update norms

If B changes, do not assume the old learning rate remains optimal.

5 · Minimal PyTorch loop

```
for xb, yb in loader:
    opt.zero_grad(set_to_none=True)
    pred = model(xb)
    loss = loss_fn(pred, yb)   # usually batch mean
    loss.backward()            # gradient estimate
    opt.step()                 # one update
```

`optimizer.zero_grad()` is essential because PyTorch accumulates gradients. Record the loss reduction (`mean` versus `sum`).

6 · Symptom → first check

Symptom	First check
loss explodes	η, invalid values, reduction scale
loss zigzags	feature scale / curvature; reduce η
loss flat	zeroed gradients, dead units, too-small η
very noisy	shuffle, batch size, outliers, logging window
run is slow	examples per second and updates per second

Report the batch size, loss reduction, learning rate, update count, and example budget. Otherwise the optimization experiment is underspecified.