# Controlled Improvement and Error Analysis

## ECBS5200 — Week 2

**A model is only as good as your ability to understand why it works.**

# Where we left off

| Model | Accuracy | Macro F1 | Zero-F1 classes |
|---|---|---|---|
| Majority class | 23.0% | ~0.003 | 112 |
| TF-IDF + LogReg | 54.2% | 0.132 | 70 |
| **Your fine-tuned encoder** | **~56%** | **~0.20** | **~47** |

Your model rescued ~23 classes from zero.

But **47 classes still get F1 = 0.** Can we do better?

# This week

**Two skills:**

1. **Diagnostic reasoning** — given a model's metrics and training curves, figure out what happened during training
2. **Controlled improvement** — change one thing at a time and measure the effect

The lecture covers both. The lab tests the first. The homework tests both.

# Today's plan

**Block 1 — Lecture (after the quiz)**

- Training curves: how to read them
- Class imbalance: why your model ignores 47 classes
- Class weighting: the biggest lever for rare classes
- Error analysis: where does the model fail?
- Controlled experiments: changing things systematically
- When to trust your numbers

**Block 2 — Lab**

- You receive 4 pre-trained models
- Your job: figure out what produced each one

# Training Curves

**The single most diagnostic artifact you can look at.**

# What is a training curve?

Two lines plotted over epochs:

- **Train loss** (blue): what the optimizer directly minimizes
- **Val loss** (red): performance on data the model never trained on

They start high. They should both decrease. **What happens next** is the diagnostic.

# Three regimes

**Undertrained** — both curves still dropping when training stopped.
The model wasn't done learning. Train more.

**Well-trained** — both curves have flattened. Val loss is at its minimum.
This is the sweet spot.

**Overtrained / data-starved** — train loss near zero, val loss rising.
The model memorized its training data. On small datasets, predictions degrade.
On large datasets, the model may become overconfident without losing accuracy.

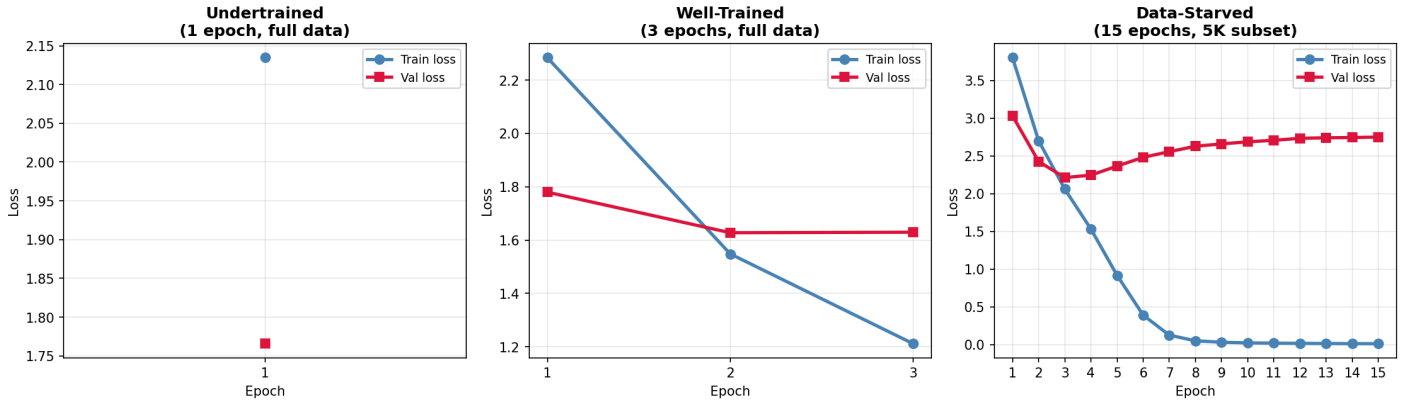
Nakkiran et al. (2019) showed this story is incomplete — look up **"deep double descent"** in the readings.

# Reading the curves

**Look for:**

1. Is train loss still dropping? → **Undertrained**
2. Have both lines flattened? → **Well-trained**
3. Train loss near zero, val loss rising? → **Overtrained or data-starved**

**In the lab, the training curve is your primary diagnostic tool.**



# Data-starved vs overtrained: a crucial distinction

Both produce rising val loss. But the causes — and fixes — are different.

| | Overtrained | Data-starved |
|---|---|---|
| **Data size** | Enough data, too many epochs | Not enough data, any epoch count |
| **Fix** | Early stopping | More data |
| **Prediction quality** | May or may not degrade | Hits a low ceiling |

A 149M-parameter model on 5,000 examples hits a low ceiling, but not because it "overfitted" — it simply never had enough signal.

**Rising val loss means worse calibration. Not necessarily worse predictions.**

Mallinar et al. (2022) call this **"tempered overfitting"** — the model gets overconfident, not wrong.

# Worked example: read this curve

Suppose you see:

- **Epoch 1:** Train loss 2.8, val loss 2.9
- **Epoch 2:** Train loss 1.4, val loss 1.6
- **Epoch 3:** Train loss 0.9, val loss 1.5
- **Epoch 4:** Train loss 0.6, val loss 1.55
- **Epoch 5:** Train loss 0.4, val loss 1.6

**What regime? What's the best checkpoint? What would you try next?**

# The paper: Zhang et al. 2017

**"Understanding Deep Learning Requires Rethinking Generalization"** (ICLR 2017)

The experiment that broke classical intuition:

- Take a standard network. Train it on **real labels** → generalizes.
- Same network, same data, **random labels** → memorizes perfectly. Train loss → 0.

If the model can memorize random noise, why doesn't it memorize real data and fail to generalize?

**Standard capacity measures (VC dimension, Rademacher complexity) are too loose to explain this.** They predict these models should always overfit. They don't.

`readings/week2/zhang2017_rethinking_generalization.pdf`

# The Class Imbalance Problem

**Before we fix it, let's see how bad it really is.**

# Your class distribution

The largest class: **13,333 training examples.**
The smallest class: **5 training examples.**

That's a **2,666:1 ratio.**

67 of your 113 classes have fewer than 100 training examples. These are the "tail."

This isn't unusual — it's the norm. Customer complaints, medical diagnoses, fraud detection, rare species identification. **Real-world classification is almost always long-tailed.**

# What cross-entropy actually does

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N} \log p(y_i \mid x_i)$$

Each example contributes equally. But classes contribute **proportionally to their frequency.**

| Class | Examples | Share of total loss |
|---|---|---|
| "Incorrect info on your report" | 13,333 | **~23%** |
| "Lost or stolen money order" | 5 | **~0.009%** |

Getting a rare class right changes the loss by **0.009%**. The optimizer's rational response: **ignore it entirely.**

# Class weighting: make the loss care

**Multiply each example's loss by a weight based on class frequency.**

Rare classes → higher weight → bigger gradient → model pays attention.

But how much weight? This is where people go wrong.

| Approach | Max weight | Result |
|---|---|---|
| No weighting | 1.0 | 47 classes at F1 = 0 |
| **Sqrt-inverse** | **3.6** | **Works — rescues 10 classes** |
| sklearn `balanced` | 128 | **Crashes training** |
| Raw inverse | 2,666 | Don't even try |

Focal loss (Lin et al. 2017) attacks the same problem differently — reshaping the loss to down-weight *easy* examples rather than up-weighting *rare* ones. Both approaches are in the readings.

# The paper: Cui et al. 2019 — Effective Number of Samples

**"Class-Balanced Loss Based on Effective Number of Samples"** (CVPR 2019)

Key idea: the **effective number** of samples for a class is less than the actual count.

$$E_n = \frac{1 - \beta^n}{1 - \beta}$$

As you add more examples for a class, each new one overlaps more in feature space. Diminishing returns.

$\beta$ controls assumed overlap: $\beta \to 0$ means every sample counts fully; $\beta \to 1$ means high overlap (diminishing returns). This gives a principled spectrum from no reweighting to full inverse.

**Our sqrt-inverse weighting is a related heuristic** — same intuition (diminishing returns), simpler formula.

`readings/week2/cui2019_class_balanced_loss.pdf`

# What the weights look like

| Class | Examples | Weight |
|---|---|---|
| "Incorrect info on your report" | 13,333 | **0.06** |
| "Problem with a purchase on your statement" | 3,905 | **0.12** |
| "Lost or stolen money order" | 5 | **3.23** |
| "Shopping for a line of credit" | 4 | **3.61** |

The rarest class gets **60x** the weight of the most common.

Not 2,000x. That would break training.

# The accuracy-F1 paradox

| | Accuracy | Macro F1 | Zero-F1 classes |
|---|---|---|---|
| **Without weighting** | 55.4% | 0.199 | 49 |
| **With weighting** | 50.2% | 0.216 | 39 |

Accuracy **drops 5 points.** Macro F1 **rises.** 10 classes rescued from zero.

**This is not a failure. This is a trade-off.**

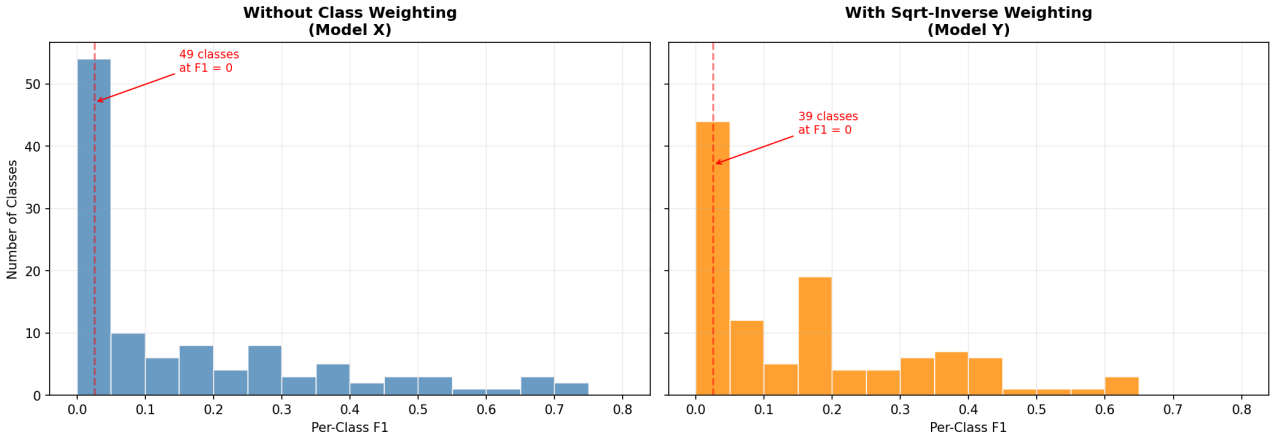
# Why the paradox happens

**Without weighting:** Big spike at F1 = 0. Head classes at 0.5–0.7.

**With weighting:** Spike shrinks. Head classes drop slightly. New bars in 0.1–0.3 range.

The model doesn't make every class better. It makes some worse and others **possible.**

Kang et al. (2020) showed this cost may come from **distorting the representation** to help the classifier. See the readings.



# Metrics disagreement: which number do you trust?

**Accuracy** = "what fraction did I get right?" Dominated by head classes.

**Macro F1** = "average F1 across all classes." Every class counts equally.

They optimize for **different objectives**. Accuracy rewards predicting common classes. Macro F1 rewards covering all classes.

When they disagree, the question isn't "which is right?" — it's **"what do you care about?"**

A spam filter needs accuracy. A medical triage system needs macro F1.

# The paper: Menon et al. 2021 — Logit Adjustment

**"Long-Tail Learning via Logit Adjustment"** (ICLR 2021)

Shows that logit adjustment by **log class priors** is **Fisher consistent for balanced error** — and unifies several prior imbalance-handling heuristics.

Applied post-hoc or during training. The key insight: the optimal classifier for **balanced error** is different from the optimal classifier for **accuracy**.

You literally cannot maximize both.

`readings/week2/menon2021_logit_adjustment.pdf`

# Error Analysis

## Where exactly does the model fail — and why?

# Per-class F1 by frequency tier

| Tier | Training examples | Typical F1 range |
|---|---|---|
| **Head** (6 classes) | ≥ 2,000 | 0.5 – 0.7 |
| **Mid** (~40 classes) | 100 – 1,999 | 0.2 – 0.4 |
| **Tail** (~67 classes) | < 100 | Many at 0.0 |

Performance tracks class frequency. Class weighting narrows the gap but can't close it.

**5 training examples is not enough to learn a decision boundary.** Fang et al. (2021) showed that beyond a critical imbalance ratio, minority-class classifiers collapse toward each other in the final layer — the model cannot tell them apart, not because it hasn't learned features, but because the decision boundaries become geometrically indistinguishable.

# Confusion matrix: where do errors land?

When a tail class gets misclassified, where does the error usually go?

**(a)** Head classes — rare examples get swamped by the majority

**(b)** Semantically similar mid-tier classes — the model can't distinguish fine-grained categories

**(c)** Other tail classes

**You'll answer this empirically in the homework.**

# Hard-example inspection

The 5–10 **most confident wrong predictions** tell you more than any aggregate metric.

What to look for:

- Semantic overlap between true and predicted class
- Ambiguous text that could genuinely go either way
- Systematic patterns (always confuses X with Y)

Some "errors" are actually label ambiguity, not model failure.

# The Diagnostic Mindset

Given metrics + training curves, you should be able to answer:

- **Did the model converge?** → training curve
- **Is it overfitting?** → train-val loss divergence (but check: tempered or catastrophic?)
- **Does it handle rare classes?** → per-class F1, zero-F1 count
- **Is the accuracy-F1 gap explained?** → class weighting? (Menon: different optimal classifiers)
- **Are improvements real?** → compare to noise floor

**Numbers don't speak for themselves.**

# The Lab: Diagnostic Forensics

**4 pre-trained ModernBERT models.** Same architecture. Same data. Different configs.

For each model you get:

- Model weights (you evaluate on the val set)
- Training log (per-epoch train loss, val loss, accuracy, F1)

**You don't know what config produced each model.**

Your job: evaluate, examine, hypothesize, then bring your evidence to me.

# Controlled Experiments

## Changing things systematically

# The principle

**Change one variable. Hold everything else constant. Measure the effect.**

Most ML work in practice: change three things at once, number goes up, declare victory.

**You will not do that.**

# The mixing knobs warning

**Real example from building this course:**

We changed the learning rate AND the scheduler in the same run. F1 improved by 2 points.

Was it the learning rate? The scheduler? Both?

**We couldn't tell.** We had to throw away the result and rerun two separate experiments.

Two separate experiments cost 2x the compute. But the confounded experiment cost **3x** — the original run plus both reruns.

**Confounded experiments don't save time. They waste it.**

# The experiment template

| Field | What to write |
|---|---|
| **Variable changed** | Exactly ONE thing |
| **Held constant** | List everything else |
| **Prediction** | What you expect — write this BEFORE running |
| **Result** | What actually happened |
| **Meaningful?** | Is this difference larger than noise? |

**Write the prediction before you run.** That's the field that makes you think.

# Learning rate and schedulers

**LR recap:** 2e-5 for fine-tuning. Too high → catastrophic forgetting. Too low → barely adapts.

An interesting finding: ModernBERT handled lr=1e-3 without collapsing. The safe range is wider than the textbooks suggest.

**Schedulers:** We tested cosine vs linear on this task. The difference was **less than 0.5 F1 points** — within noise. Not every knob matters.

# Batch size

**Smaller batches = more optimizer steps per epoch.**

With 113 classes and batch = 32, most batches contain zero examples of any given tail class.

In our experiments: **the effect was modest and task-dependent.**

This is an empirical question, not a settled principle. **Test it on your data.**

# Early stopping

**Train for more epochs than you think you need. Stop when the target metric stops improving.**

The tension on long-tail tasks:

- Val loss may start rising (model less calibrated)
- Macro F1 may still be improving (model better at rare classes)

**Use macro F1 for stopping** — it's what you care about. But don't over-trust small movements.

Patience: 2–3 epochs. **Best checkpoint ≠ last checkpoint.**

# Common training bugs

Before you go into the lab, one more diagnostic skill: **recognizing when code is broken.**

**The softmax-before-CrossEntropyLoss bug:**

```
# WRONG — loss stuck at ~4.5, model never learns
loss = F.cross_entropy(F.softmax(logits, dim=-1), labels)

# CORRECT — CrossEntropyLoss applies softmax internally
loss = F.cross_entropy(logits, labels)
```

`cross_entropy` expects **raw logits**. Applying softmax first compresses the input into [0,1], which compresses gradients to near zero. Loss gets stuck. Model doesn't learn.

**You'll encounter this in the lab.** The training curve is the tell.

# Val set reliability

Your val set: **6,430 examples** across **113 classes.**

Some classes have hundreds of val examples. Some have very few.

- **~20 classes** have ≤ 5 val examples (F1 is essentially random)
- **~10 classes** have exactly 1 val example (F1 is a literal coin flip)

That means **~18% of your macro F1 average is noise.** A 1-point improvement could be real — or it could be coin flips landing differently.

**This doesn't mean metrics are useless. It means you need to read them carefully.**

# Seeds and reproducibility

What changes with a different random seed:

1. Classification head initialization
2. Data shuffle order
3. Dropout masks

Same config, different seed → **different number. That's normal.**

**Explore** with 1 seed. **Confirm** with 3. Report **mean ± std.**

# Your homework this week

**You train your own models.**

- Reproduce the class weighting trade-off on full data
- Test whether batch size matters
- Implement early stopping on macro F1
- Error analysis: confusion matrix, per-class deep dive
- Discover how noisy your validation metrics really are
- Write your memo (5 sections, prompts in the notebook)

**Due:** Wednesday morning before Week 3 class. HTML via Moodle.

# Week 2 Reading List

| Paper | Year | Key idea |
|---|---|---|
| **Zhang et al. — Rethinking Generalization** | **2017** | **DNNs memorize yet generalize** |
| Lin et al. — Focal Loss | 2017 | Reshape loss to focus on hard examples |
| **Cui et al. — Class-Balanced Loss** | **2019** | **Effective number of samples** |
| Nakkiran et al. — Deep Double Descent | 2019 | The U-curve has a second descent |
| Kang et al. — Decoupling | 2020 | Representations are fine; classifiers are biased |
| **Menon et al. — Logit Adjustment** | **2021** | **You can't maximize accuracy and balanced error** |
| Fang et al. — Minority Collapse | 2021 | Minority-class classifiers collapse in geometry |
| Mallinar et al. — Tempered Overfitting | 2022 | Benign vs tempered vs catastrophic |

All PDFs in `readings/week2/`. **Bold = covered in lecture. Pick 2–3 others that connect to your memo.**

# Next week

We try a **completely different approach** to this same task.

⬇️

# REVEAL BELOW

⬇️

**Do not advance past this slide until students have written their hypotheses.**

# Lab Reveal

| Model | Configuration | Key evidence |
|---|---|---|
| **Charlie** | 1 epoch, full data (undertrained) | Only 1 epoch in logs, both curves still descending |
| **Bravo** | 15 epochs, 5K subset (data-starved) | Most epochs but worst metrics. 157 batches/epoch vs 1808 |
| **Delta** | 3 epochs, full data (vanilla baseline) | Highest accuracy, matches Week 1 numbers |
| **Alpha** | 3 epochs, full data, class weighting | Lower accuracy, higher F1, fewer zero-F1 classes |