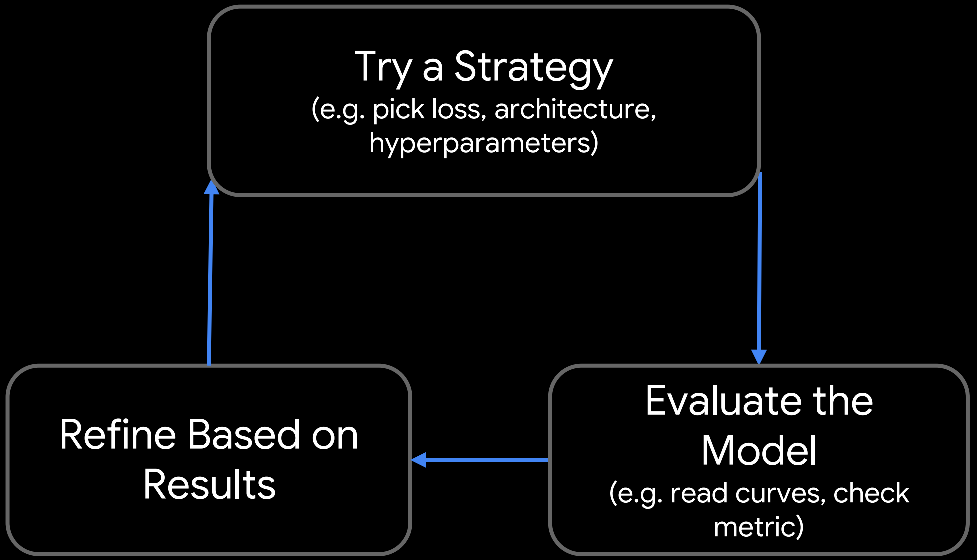
The Art of Training Deep Neural Networks

King Abdullah University of Science and Technology (KAUST)
KAUST Academy

Table of Contents

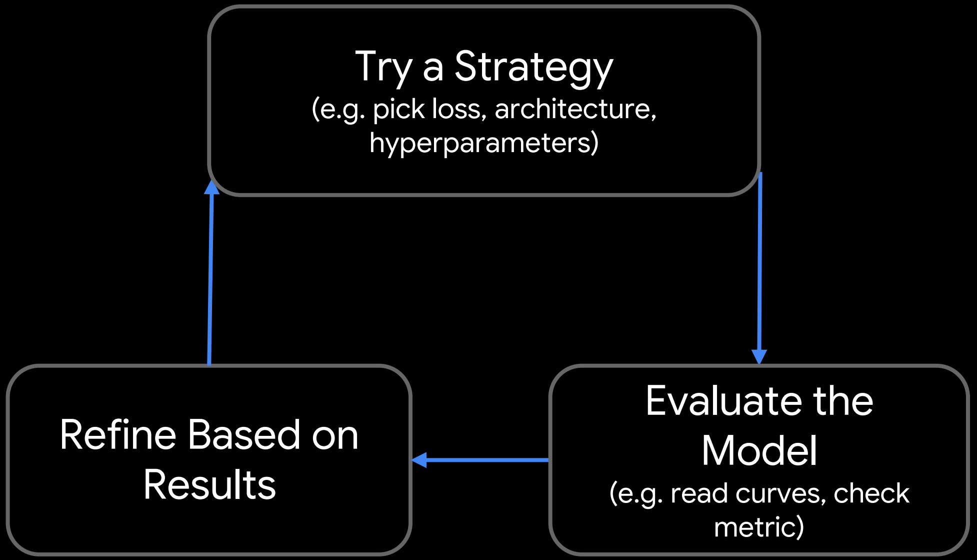
Introduction

- Training AI models is an iterative process of trying, failing, and refining.
- Success depends on empirical experimentation, not just theory.
- Practically, we can offer guidelines on what is likely to work, but there are no guarantees.



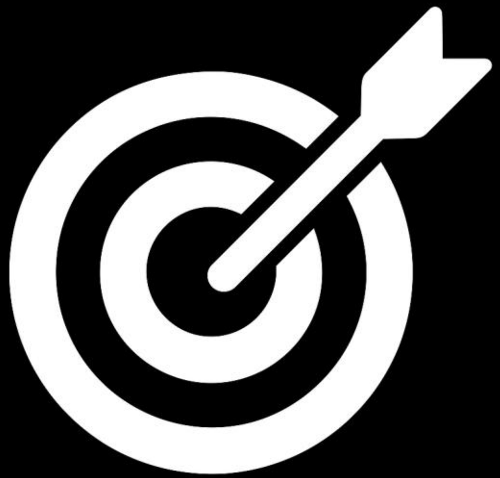
Introduction

- In this session, we'll explore the key components that influence deep learning training.
- What they are, why they matter, and how to optimize them.



Losses & Metrics

What You Optimize Is What You Get.



Loss vs Metric: spot the difference

Role	Used during	Differentiable?	Typical examples
Loss	?	?	Cross-Entropy, MSE, L1, Huber, Focal, Dice-Loss
Metric	?	?	Accuracy, F1, mAP, IoU, AUC-ROC, MAE

Loss vs Metric: spot the difference

Role	Used during	Differentiable?	Typical examples
Loss	Back-prop optimisation	Yes	Cross-Entropy, MSE, L1, Huber, Focal, Dice-Loss
Metric	Validation / reporting	No	Accuracy, F1, mAP, IoU, AUC-ROC, MAE

“Loss is for machines, metrics are for humans.”

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	?	Regression / autoencoders
Huber (Smooth L1) Loss	?	Regression / autoencoders
Dice / IoU Loss	?	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	?	Classification / language models (LLMs)

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	L^1 distance between points/pixels	Regression / autoencoders
Huber (Smooth L1) Loss	?	Regression / autoencoders
Dice / IoU Loss	?	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	?	Classification / language models (LLMs)

Is MAE differentiable? 👀

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	L^1 distance between points/pixels	Regression / autoencoders
Huber (Smooth L1) Loss	?	Regression / autoencoders
Dice / IoU Loss	?	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	?	Classification / language models (LLMs)

Is MAE differentiable? No, but we can use a special type of gradient (subgradient) to minimize it.

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	L^1 distance between points/pixels	Regression / autoencoders
Huber (Smooth L1) Loss	Piecewise L^1 & L^2 (quadratic near 0, linear beyond) distance between points/pixels	Regression / autoencoders
Dice / IoU Loss	?	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	?	Classification / language models (LLMs)

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	L^1 distance between points/pixels	Regression / autoencoders
Huber (Smooth L1) Loss	Piecewise L^1 & L^2 (quadratic near 0, linear beyond) distance between points/pixels	Regression / autoencoders
Dice / IoU Loss	Overlap ratio between masks/boxes	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	?	Classification / language models (LLMs)

Some Popular Losses

Loss	Minimises …	Used for …
Mean Squared Error (MSE)	L^2 distance between points/pixels	Regression / autoencoders
Mean Absolute Error (MAE)	L^1 distance between points/pixels	Regression / autoencoders
Huber (Smooth L1) Loss	Piecewise L^1 & L^2 (quadratic near 0, linear beyond) distance between points/pixels	Regression / autoencoders
Dice / IoU Loss	Overlap ratio between masks/boxes	Segmentation / Detection
Cross-Entropy (CE) / KL divergence	Information (entropy) difference between probability distributions	Classification / language models (LLMs)

Some Popular Losses

Q: Can we optimize for multiple losses at the same time? 👀

Some Popular Composite Losses

Scenario	Composite Loss	Purpose
Detection (Yolo)	BCE_obj + CE_cls + λ·IoU_reg	? + ? + ?
Segmentation (UNet)	CE + λ·Dice	? + ?
Generative (VAE)	Reconstruction_obj + β·KL	? + ?
Generative (GAN)	CE_adv + λ·Perceptual	? + ?

Some Popular Composite Losses

Scenario	Composite Loss	Purpose
Detection (Yolo)	BCE_obj + CE_cls + λ·IoU_reg	Objectness + class + boxes
Segmentation (UNet)	CE + λ·Dice	? + ?
Generative (VAE)	Reconstruction_obj + β·KL	? + ?
Generative (GAN)	CE_adv + λ·Perceptual	? + ?

Some Popular Composite Losses

Scenario	Composite Loss	Purpose
Detection (Yolo)	BCE_obj + CE_cls + λ·IoU_reg	Objectness + class + boxes
Segmentation (UNet)	CE + λ·Dice	Pixels + mask overlap
Generative (VAE)	Reconstruction_obj + β·KL	? + ?
Generative (GAN)	CE_adv + λ·Perceptual	? + ?

Some Popular Composite Losses

Scenario	Composite Loss	Purpose
Detection (Yolo)	BCE_obj + CE_cls + λ·IoU_reg	Objectness + class + boxes
Segmentation (UNet)	CE + λ·Dice	Pixels + mask overlap
Generative (VAE)	Reconstruction_obj + β·KL	Rebuild + Gaussian latent
Generative (GAN)	CE_adv + λ·Perceptual	? + ?

Some Popular Composite Losses

Scenario	Composite Loss	Purpose
Detection (Yolo)	BCE_obj + CE_cls + λ·IoU_reg	Objectness + class + boxes
Segmentation (UNet)	CE + λ·Dice	Pixels + mask overlap
Generative (VAE)	Reconstruction_obj + β·KL	Rebuild + Gaussian latent
Generative (GAN)	CE_adv + λ·Perceptual	Realism + texture & color details

Hyperparameters Tuning

Loss tells us where to climb
Hyper-params decide how fast and which path.

Hyperparameters

What are Hyper-parameters?

⇒ values you set before training (not learned).

Role?

⇒ steer optimisation speed, capacity & generalisation.

How to set them?

⇒ pick sensible starters → train → inspect → iterate.

Hyperparameters Tuning

Hyperparameter	Quick rule-of-thumb
Optimiser	Usually Adam/AdamW works the best.
LR	*CNNs-based*: (1e-3–1e-4)-ish ↔ *Transformers-based*: (1e-5–1e-6)-ish. Usually used with a scheduler (learning rate decay).
Batch size	Vision: (4–32)-ish ↔ Text: (1–16)-ish keep fixed; scale only if needed.
Epochs	Vision ≈ 5–300; NLP ≈ 1-10; LLMs ≈ 1 (up to 3).
Img size / sequence length	prototype small (e.g. img_size ≈ 224*224, seq_len ≈ 256) → upscale when everything else is stable.
Backbone family & size	start tiny → scale up once pipeline is stable (e.g. EfficientNetV2-Small → EfficientNetV2-Large, BERT-base → BERT-large, etc.).

Hyperparameters Tuning

Tip: Scale batch size when adjusting learning rate.

But by how much?

Hyperparameters Tuning

Tip: Scale batch size when adjusting learning rate.

1. SGD: Use linear scaling rule*

$$\mathrm{LR}_{new} = \mathrm{LR}_{old} \times \frac{\text{Batch Size}_{new}}{\text{Batch Size}_{old}}$$

1. Adam/AdamW: Square-root scaling rule*

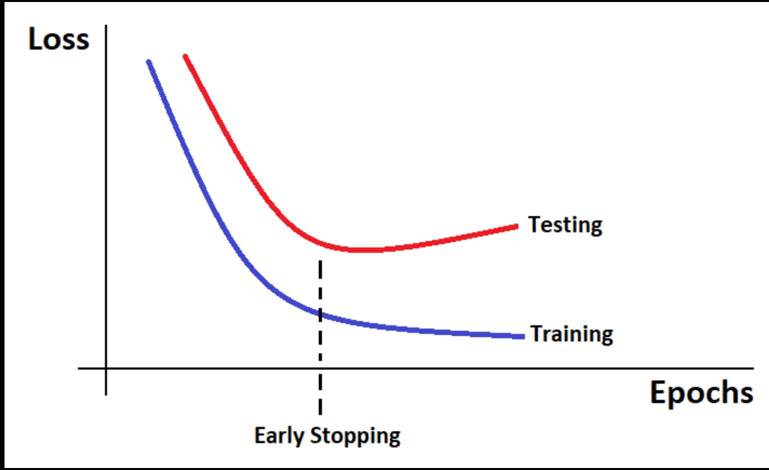
$$\mathrm{LR}_{new} = \mathrm{LR}_{old} \times \sqrt{\frac{\text{Batch Size}_{new}}{\text{Batch Size}_{old}}}$$

---

*Granziol D., Zohren S., Roberts S., "Learning Rates as a Function of Batch Size: A Random Matrix Theory Approach to Neural Network Training," arXiv:2006.09092, 2020.

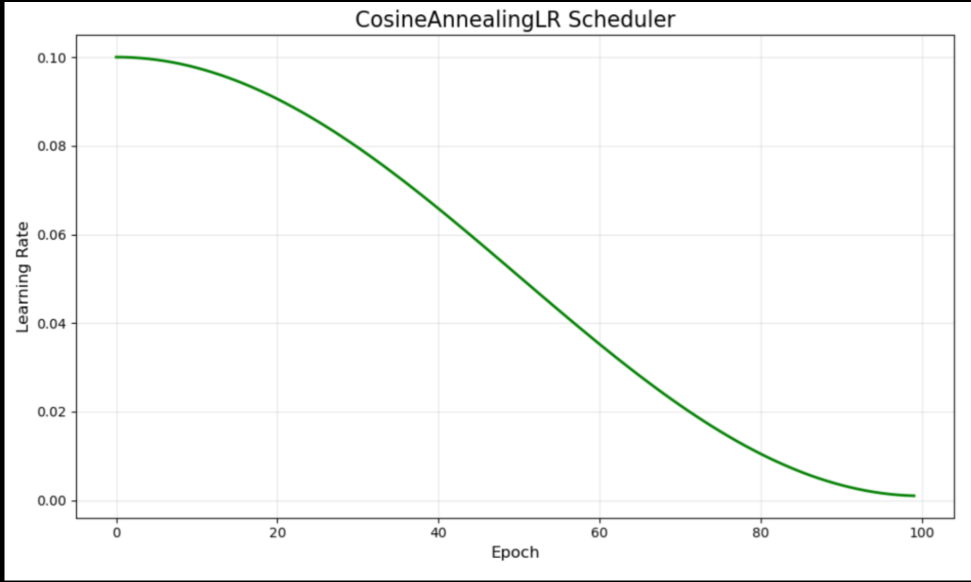
Early Stopping

- **Goal:** stop training just after validation metric flattens or degrades.
- **How:** monitor `val-loss / val-metric` → `patience=N` epochs.
- Helps auto-select optimal epochs value.



Schedulers

- **A scheduler** is the rule that automatically adjusts the learning rate during training.
- They help speed convergence, escape plateaus, and reach a better optimum.
- Most teams now default to cosine decay.



Schedulers Types

Scheduler Type	How It Triggers	Typical Use-case
Step-based	After every optimiser step	When low number of epochs used (e.g. NLP)
Epoch-based	After every epoch	When a high number of epochs used (e.g. Vision, Audio,...etc)

Warm-up

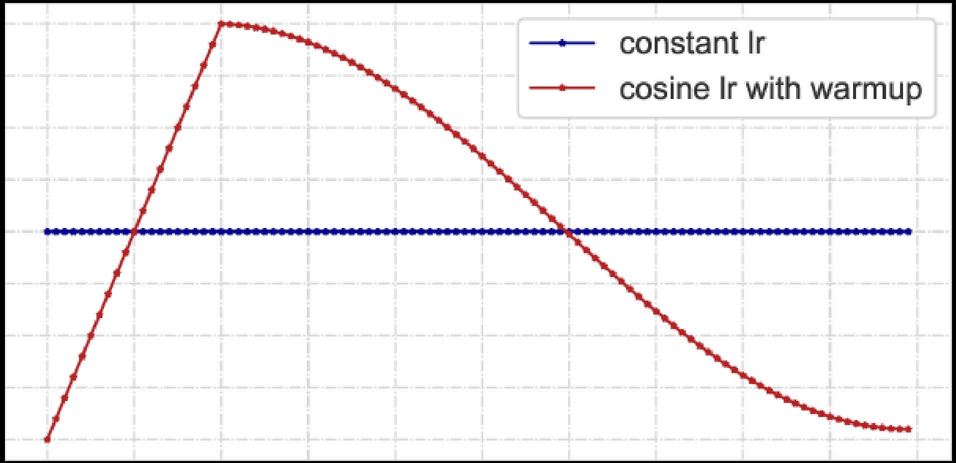
- **Problem:**
  - large initial LR + random weights

    ⇒ gradients explode.
  - large initial LR + Pretrained weights

    ⇒ Forget previous knowledge.

Warm-up

- **Solution:** gradually increase LR from 0 ⟶ base LR over some steps/epochs (or 3-5 % of total steps/epochs).
- Usually used with Transformers.



Early-Stopping vs Fixed Epochs

- **Which one is better:**
  - Many epochs + Early stopping.
  - Fixed epochs + Learning rate scheduler.

Early-Stopping vs Fixed Epochs

- **Which one is better:**
  - Many epochs + Early stopping.
  - Fixed epochs + Learning rate scheduler.

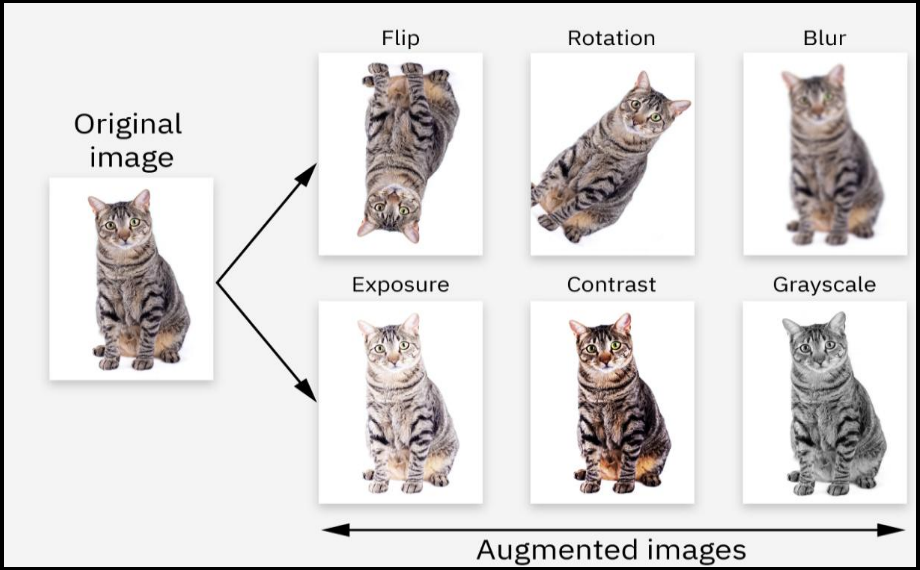
⇒ **Recommended:**

1. Use early stopping initially to discover optimal epoch range.
2. Then, set a fixed number of epochs and apply a scheduler to smoothly decay the learning rate within that range.

Augmentations

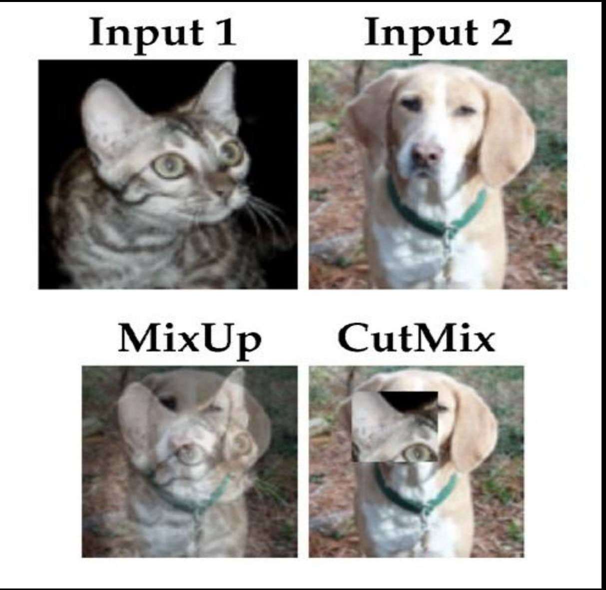
- **Data augmentation** is generating new training examples from existing ones through various transformations.
- Types:

  Random Augmentations: Flip, Crop, Noise,...



Augmentations

- **Data augmentation** is generating new training examples from existing ones through various transformations.
- Types:



Mix Augmentations:

MixUp, CutMix,...

Augmentations

- **Data augmentation** is generating new training examples from existing ones through various transformations.
- **Types:**

  Random Augmentations: Flip, Crop, Noise,...

  Mix Augmentations: MixUp, CutMix,...

- **How to choose?** do error analysis
  $\longrightarrow$ add augs that mimics real mistakes.
- **Note**: heavy augs $\Rightarrow$ add more epochs.

Tuning Order (practical)

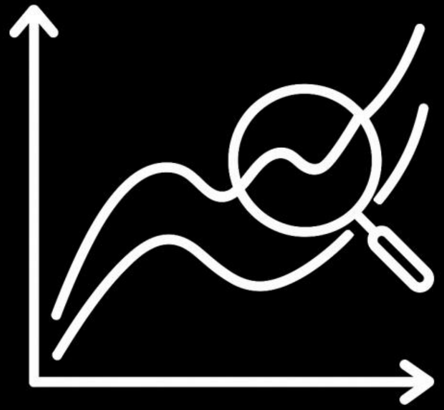
1. **LR & Epochs:** lock batch size unless GPU forces change.
2. **Scheduler:** cosine is usually the best.
3. **Augmentations:** add gradually, re-train.
4. **Model / Input complexity:** scale backbone, image size, sequence length.

Tuning Order (practical)

- Tips:
  - Make one change at a time.
  - Start with small experiments, then scale up.
  - Always ensure the loss behaves normally and check for common bugs (e.g., exploding loss, NaNs, unstable curves).

Reading & Debugging Loss Curves

See the Signal, Catch the Bug

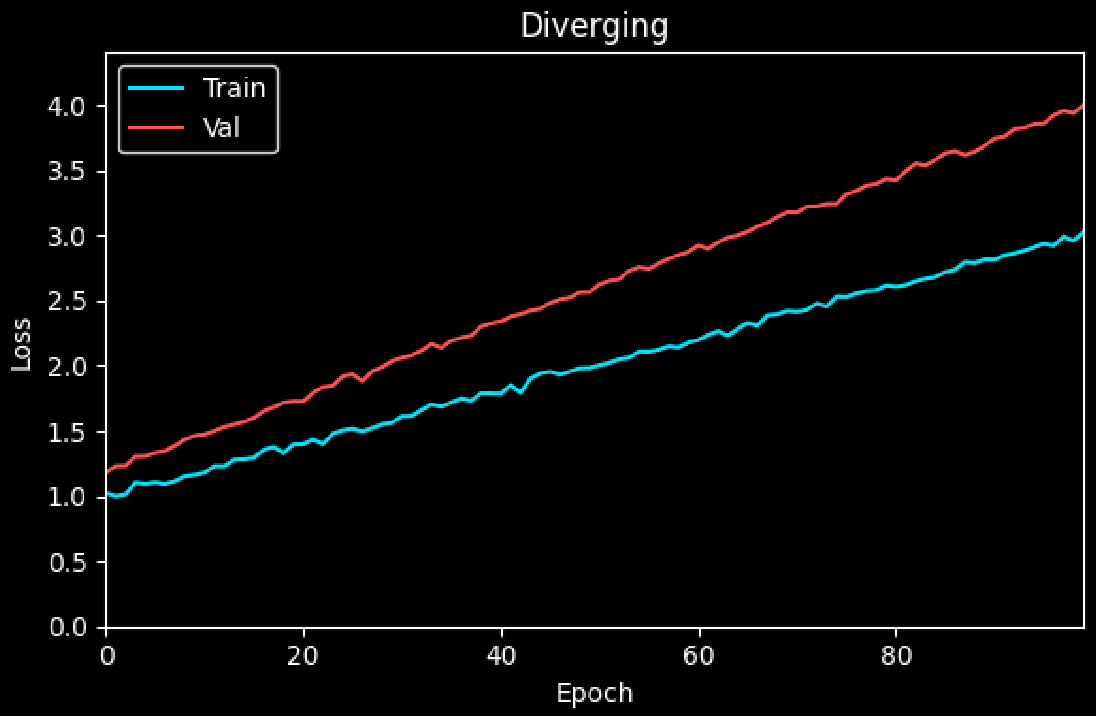


Debugging Loss Curves

- Let’s have a look at some plots...

Case 1: Diverging

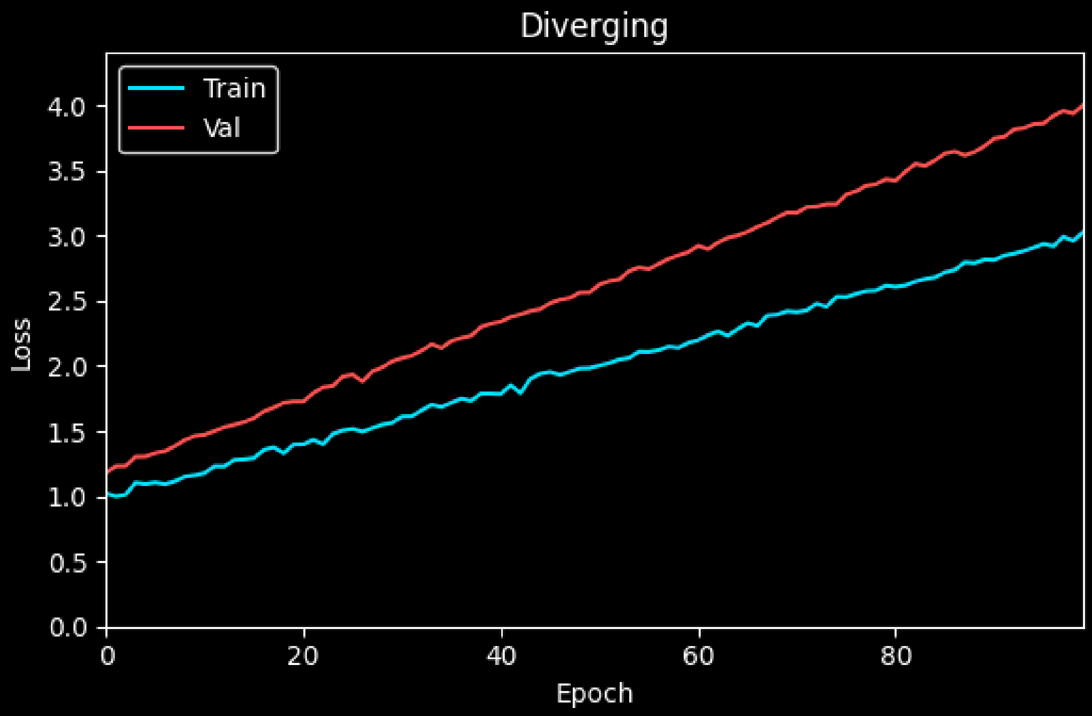
Why did this happen?



Case 1: Diverging

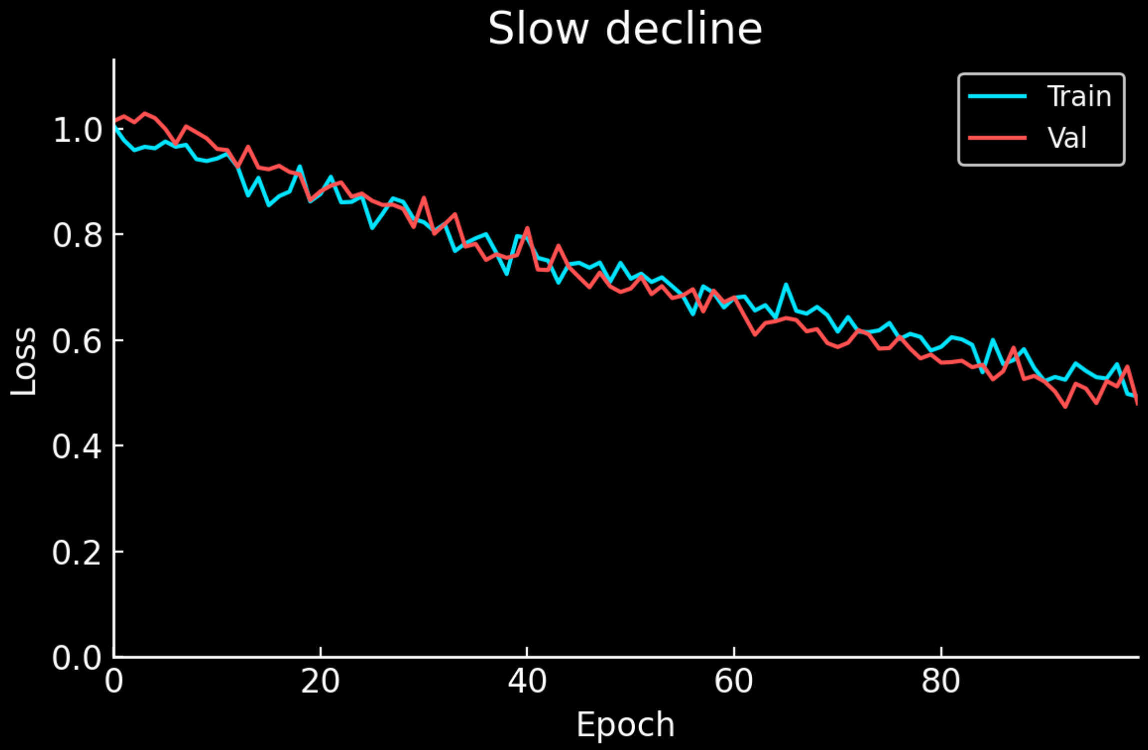
Why did this happen?

- High LR
- Exploding gradient
- Bad data
- No normalization



Case 2: Slow Decline

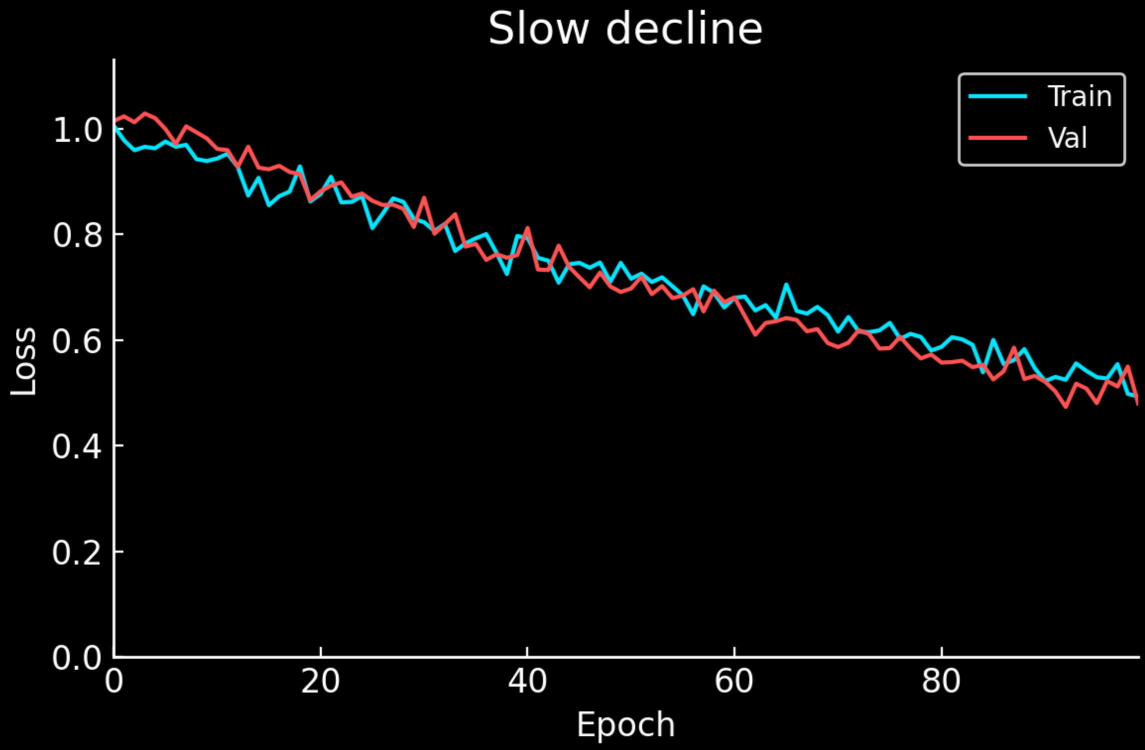
Why did this happen?



Case 2: Slow Decline

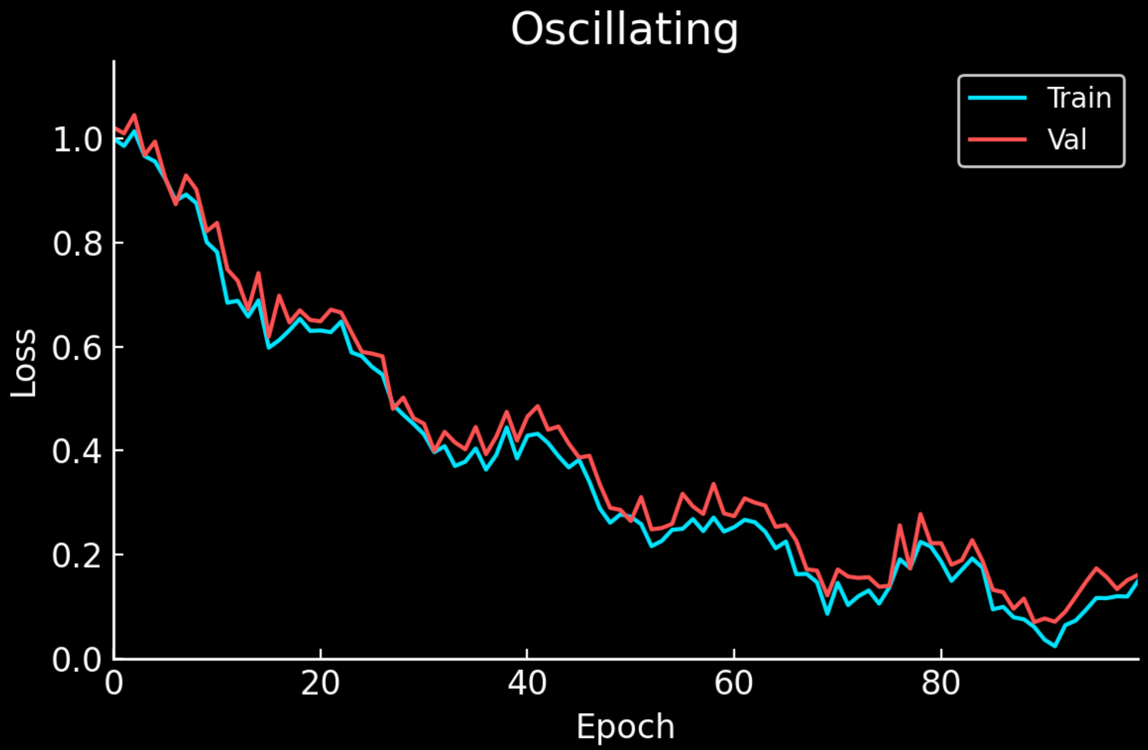
Why did this happen?

- Very small LR
- Vanishing gradients



Case 3: Oscillating

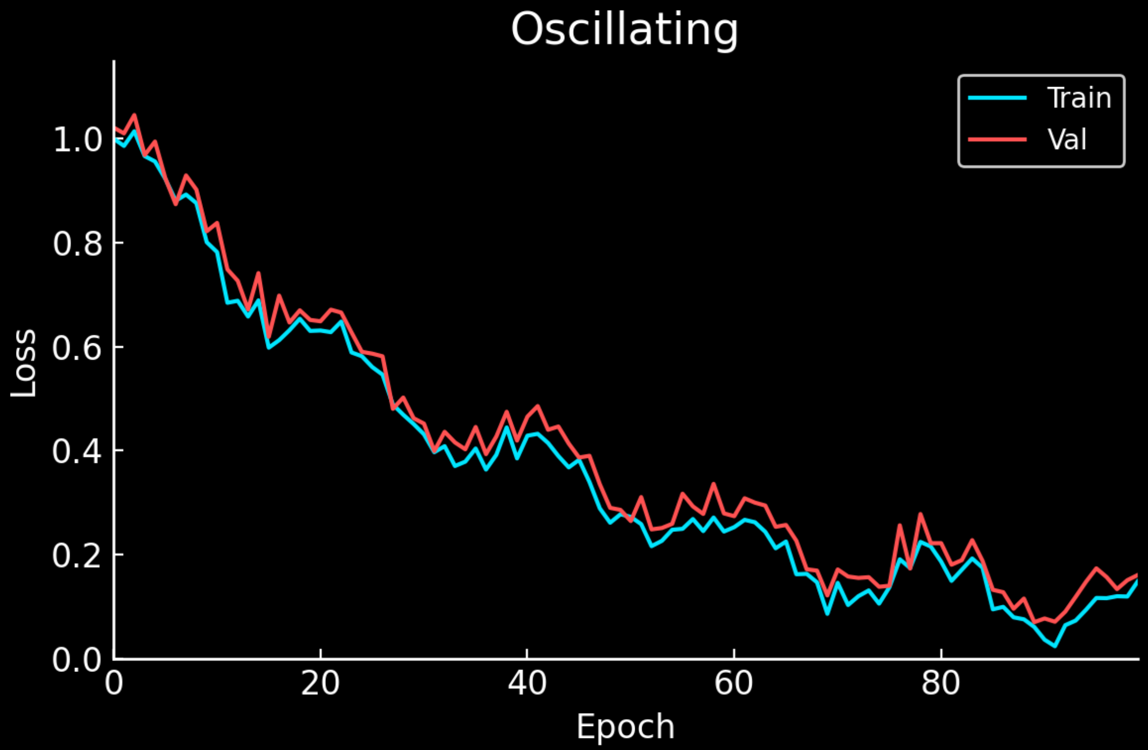
Why did this happen?



Case 3: Oscillating

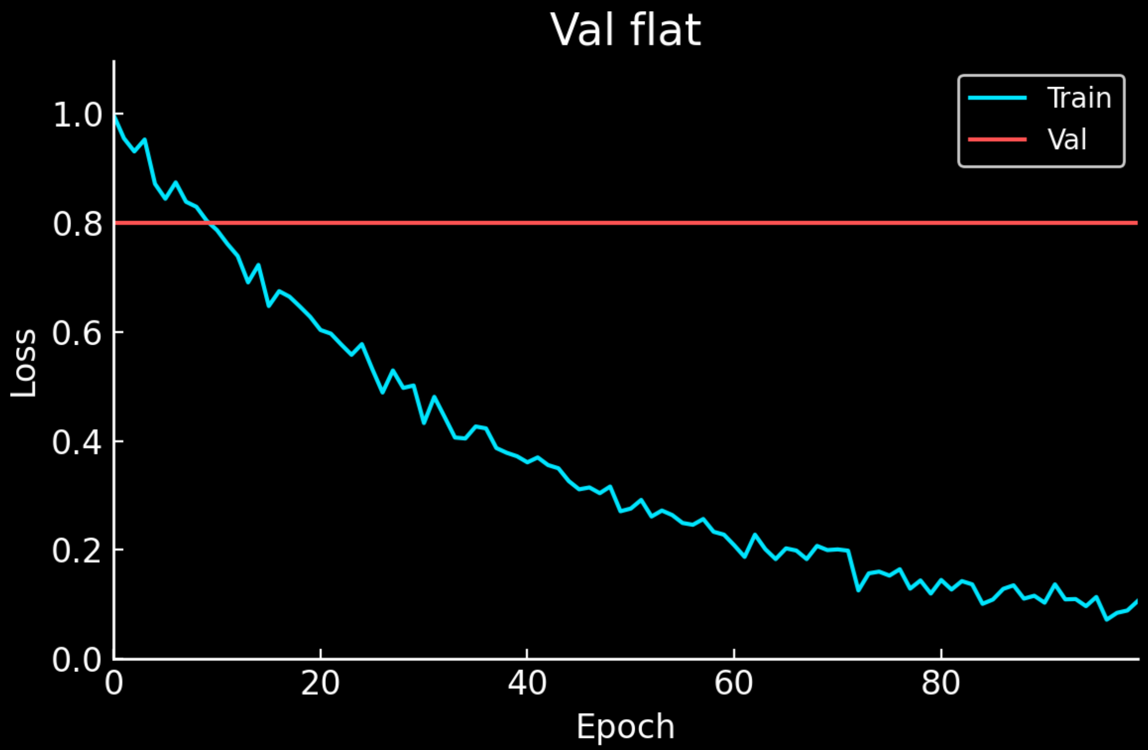
Why did this happen?

- High LR
- Small batch size
- Poor shuffle
- A lot of bad samples



Case 4: Val flat

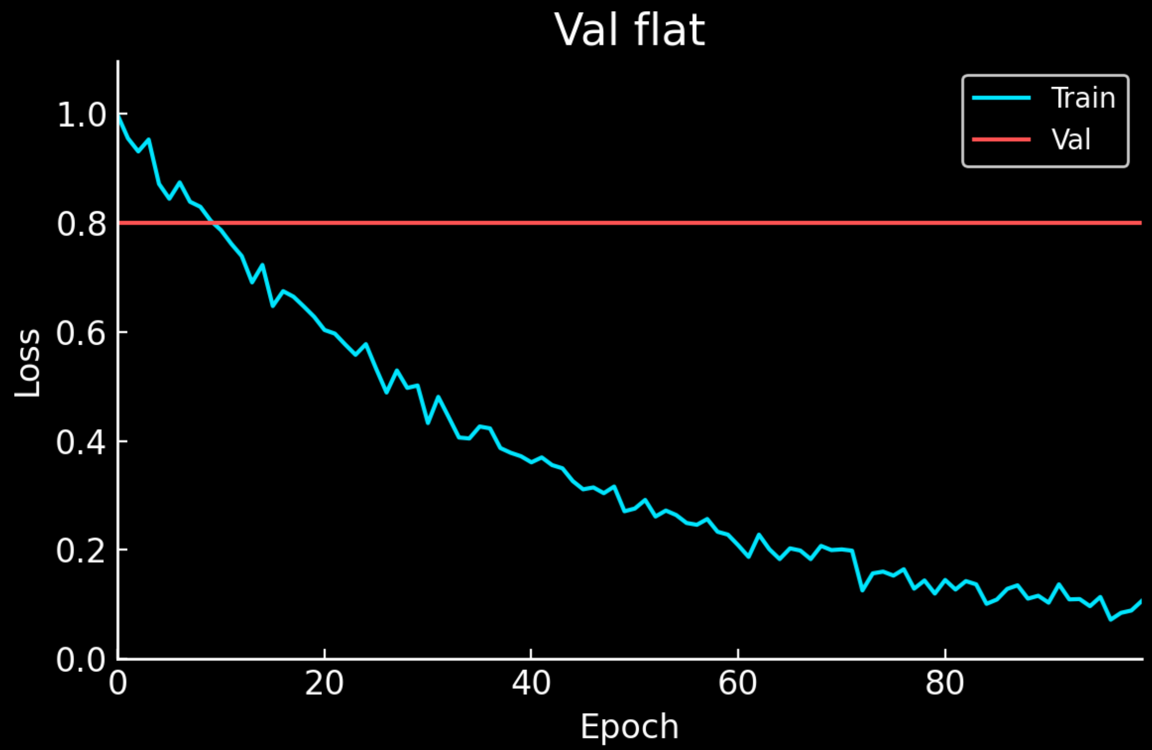
Why did this happen?



Case 4: Val flat

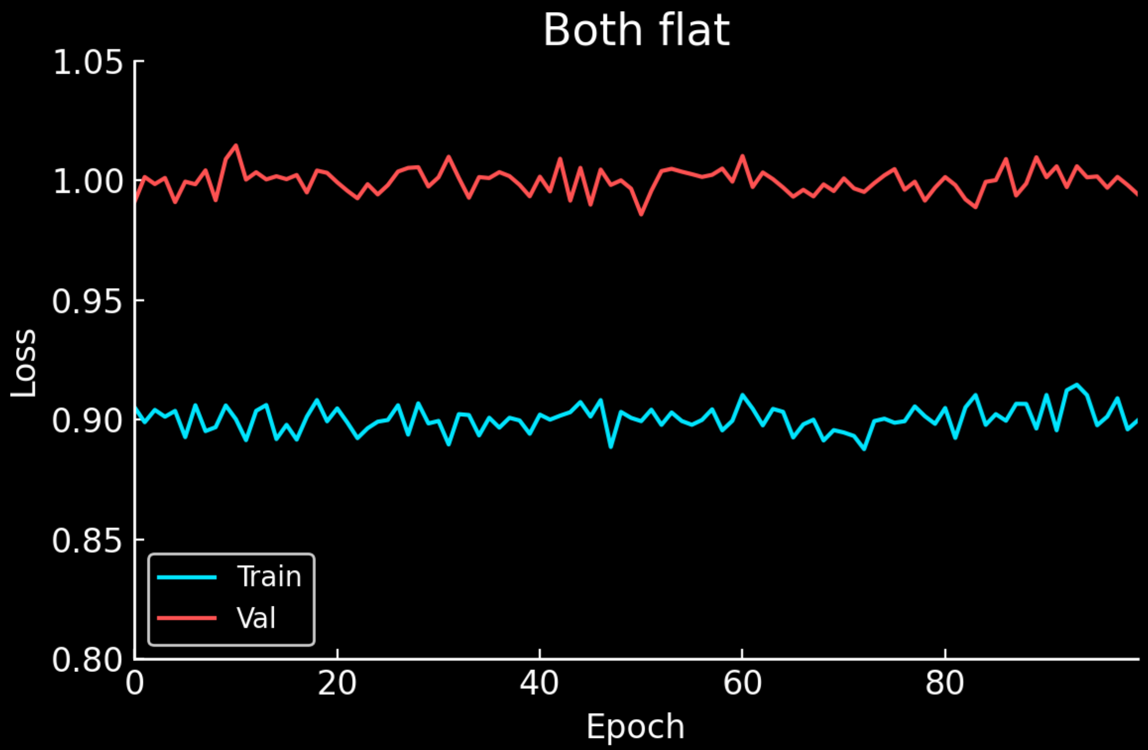
Why did this happen?

- Coding bug in data preparation (e.g. wrong labels)
- Very hard val (distribution shift)
- Significant imbalance



Case 5: Both flat

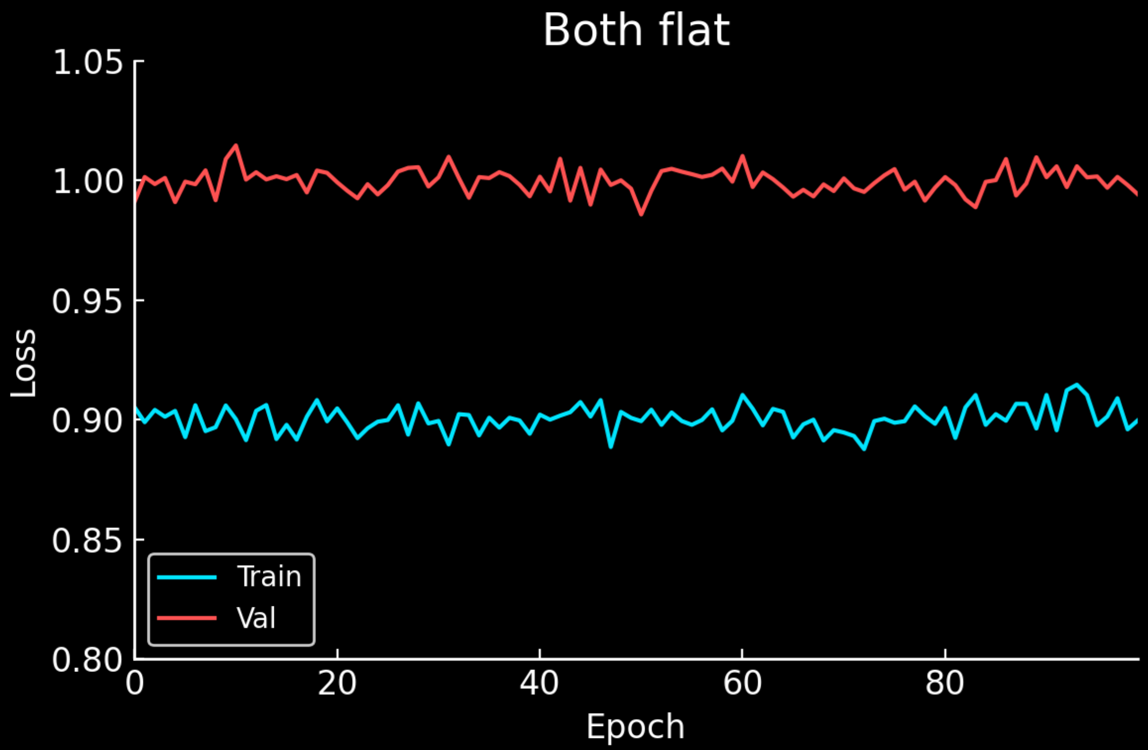
Why did this happen?



Case 5: Both flat

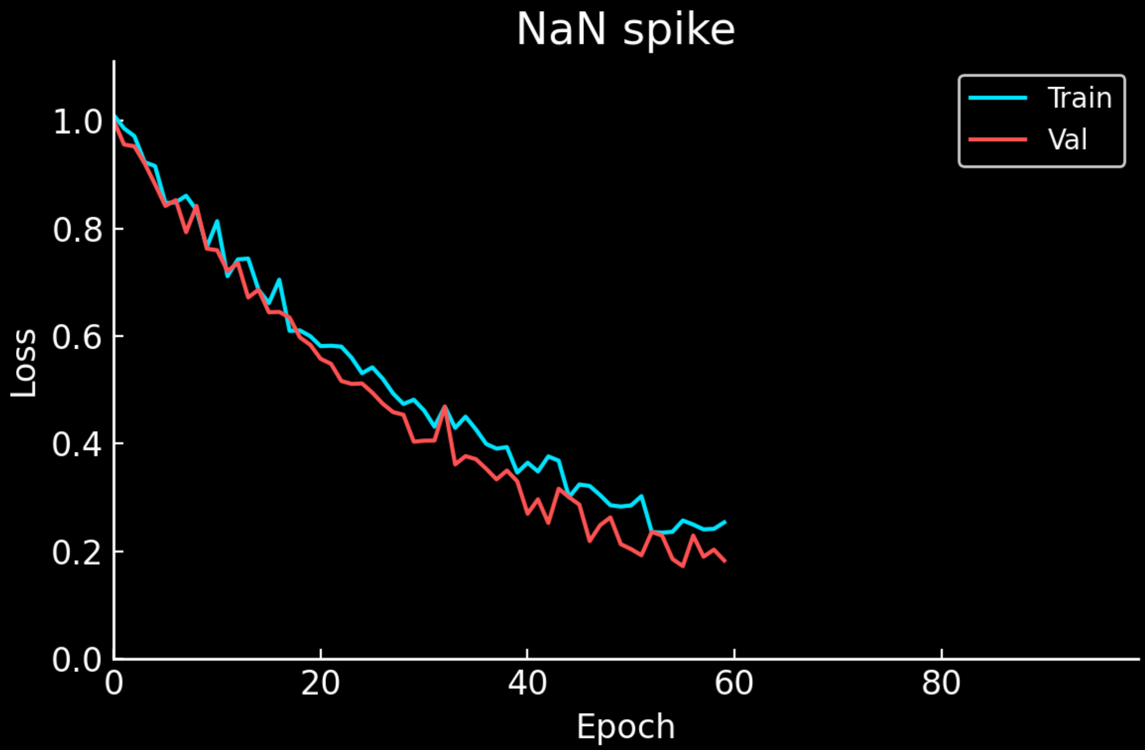
Why did this happen?

- Wrong loss
- Arch mismatch
- Bad labels
- Frozen grads
- Very small LR
- No normalization



Case 6: NaN Loss

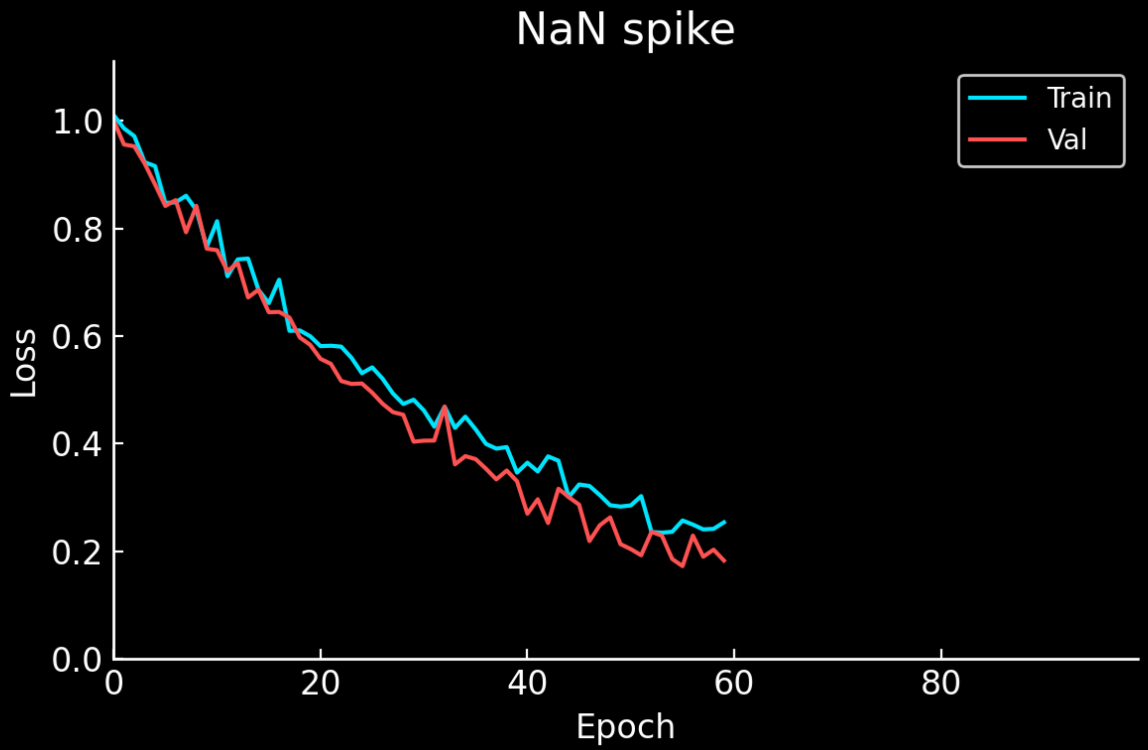
Why did this happen?



Case 6: NaN Loss

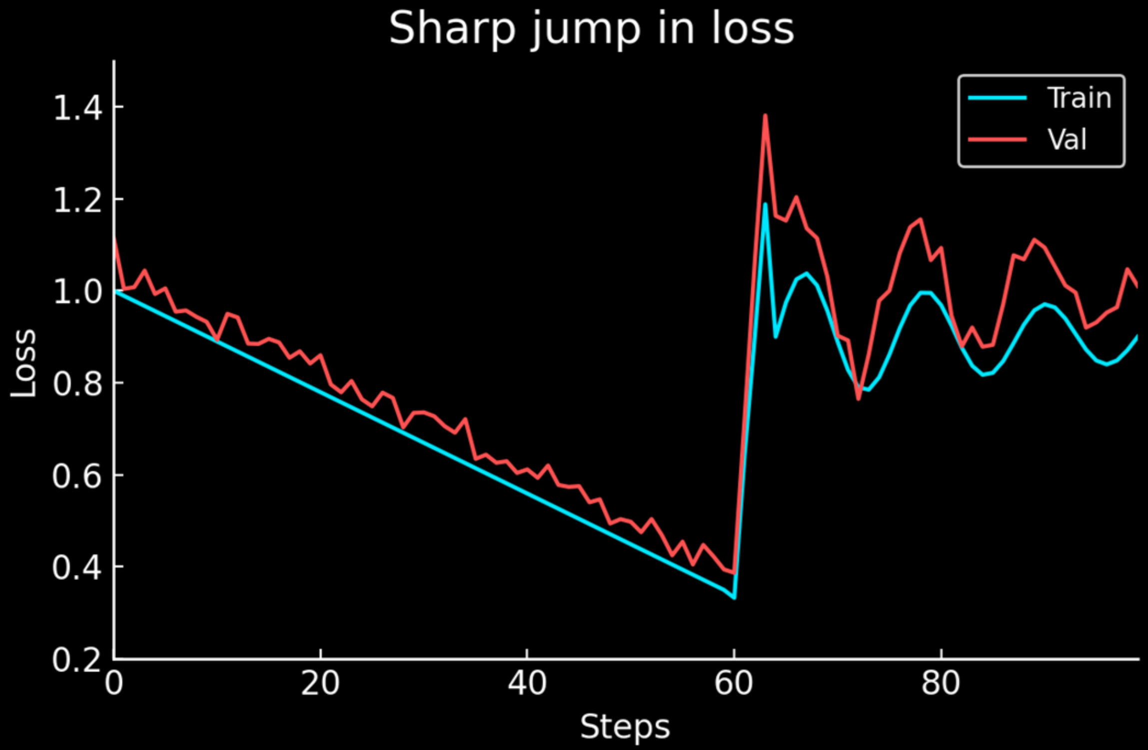
Why did this happen?

- Grad explode
- Log/Div 0



Case 7: Sharp jump

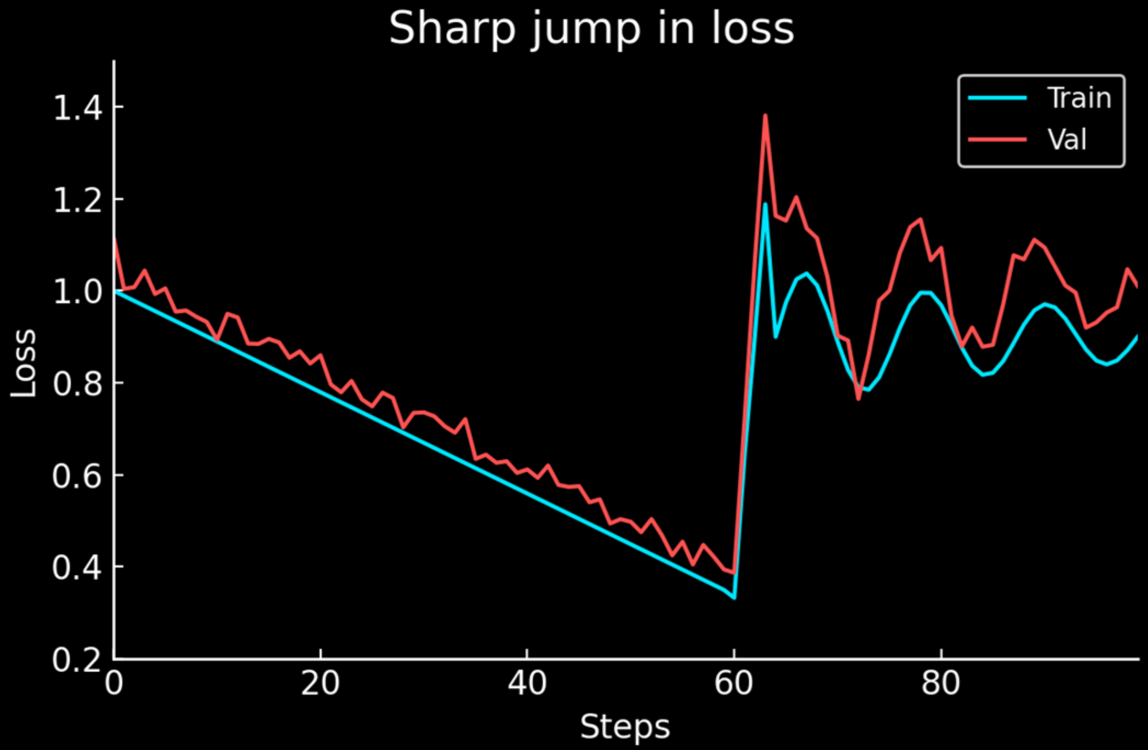
Why did this happen?



Case 7: Sharp jump

Why did this happen?

- NaNs/Inf
- Outliers
- Poor shuffling



Case 8: Val rises, train falls

Why did this happen?

Case 8: Val rises, train falls

Why did this happen?

- Overfitting (Big Model, many epochs, few data, weak aug,...)

Memory & Speed Optimisation

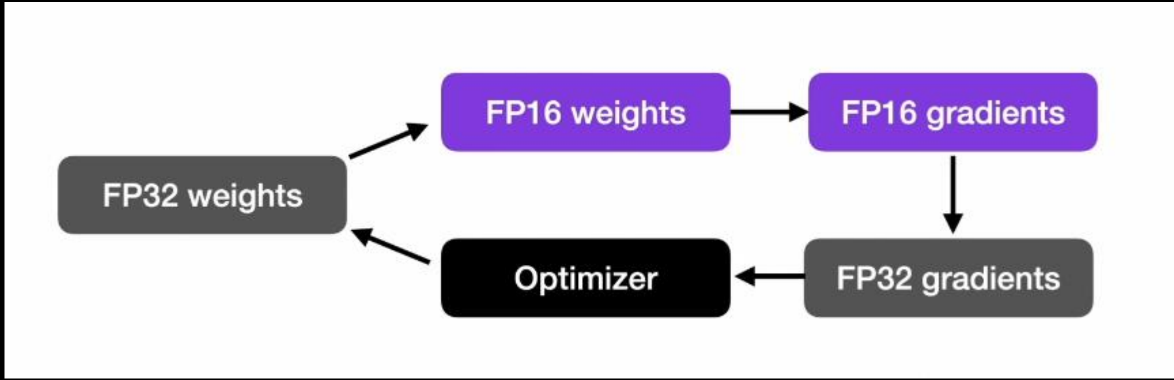
Train Faster, Fit Bigger.

Mixed Precision

- **Mixed precision** combines the use of both FP32 and lower bit floating points (FP16) to reduce memory footprint during model training.
- It halves GPU memory use and often boosts training throughput by 1.5–2×.
- Implementation.

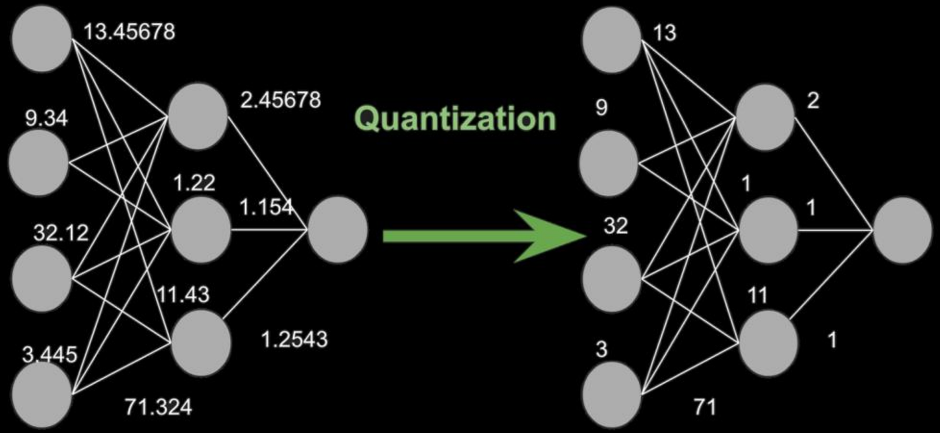
Mixed Precision

- **How it works?**
  1. FP32 original weights are kept for full-precision updates.
  2. Cast to FP16 for forward/backward.
  3. Compute FP16 gradients, then cast back to FP32.
  4. Optimizer updates the FP32 "original" copy.



Quantization

- Convert the weights of a trained model from FP32 ⟶ INT8/4.
- **Why?**
  - Model size ↓ 4–8×
  - Inference speed ↑ 2–4×
  - Minimal accuracy drop ↓



جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

أكاديمية كاوست
KAUST ACADEMY

Mixed Precision vs Quantization

	Mixed Precision	Quantization
Data type	FP32 ↔ FP16 (floats)	FP32 → INT8/4 (integers)
When to use	Training	Inference
Benefit	½ memory & 1.5–2× training speed	4–8× smaller model & 2–4× faster inference
Drawback	Needs GPU support (AMP)	Possible small accuracy drop

Why not training on INT8/4 to make training even faster/lighter? 👀

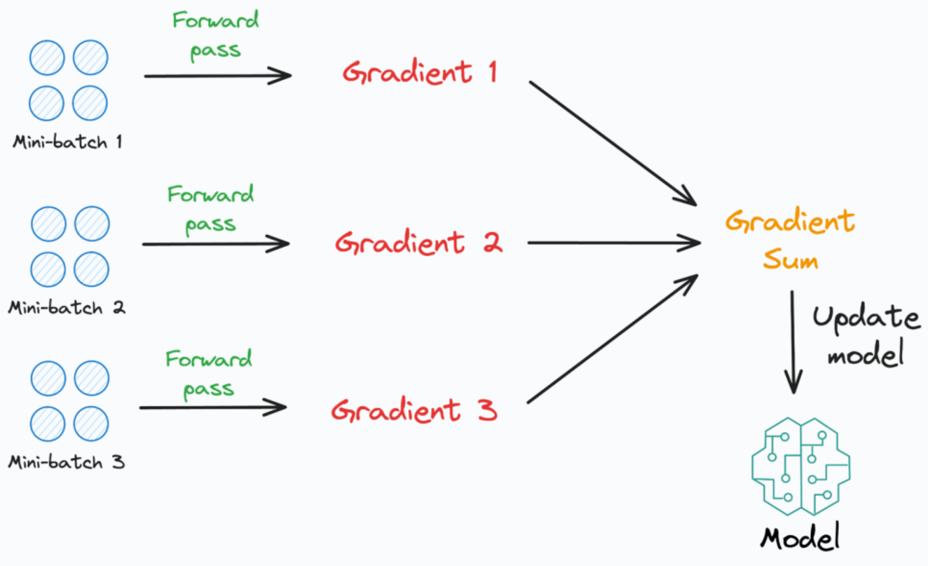
Mixed Precision vs Quantization

	Mixed Precision	**Quantization**
Data type	FP32 ↔ FP16 (floats)	FP32 → INT8/4 (integers)
When to use	Training	Inference
Benefit	½ memory & 1.5–2× training speed	4–8× smaller model & 2–4× faster inference
Drawback	Needs GPU support (AMP)	Possible small accuracy drop

Why not training on INT8/4 to make training even faster/lighter? unstable or no learning.

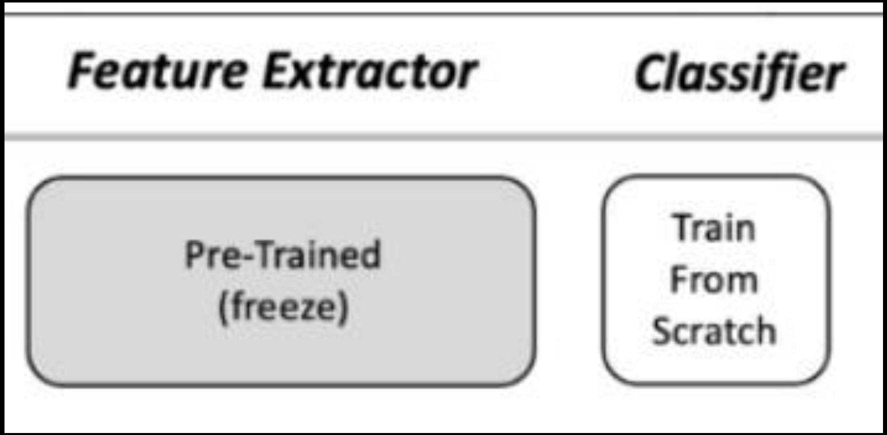
Gradient Accumulation

- Simulate large-batch training on limited GPU memory.
- How: accumulate gradients over k mini-batches before optimizer step.
- Result:
  - Stable updates
  - Bigger effective batch size without OOM.



Freezing Backbone

- Freeze pretrained backbone and train only head or adapter layers.
- Benefits:
  - Faster training
  - Lower memory
  - Less overfitting on small data



جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

أكاديمية كاوست
KAUST ACADEMY

Knowledge Distillation (Teacher-Student Models)

- Knowledge distillation is a machine learning technique that aims to transfer the learnings of a large pre-trained model (teacher) to a smaller model (student).
- Why?
  - ≈ 95 % accuracy with < ½ parameters.
  - Memory & latency ↓ → deploy on edge/phone.
  - Decrease inference cost significantly.

Knowledge Distillation (Teacher-Student Models)

- Knowledge distillation is a machine learning technique that aims to transfer the learnings of a large pre-trained model (teacher) to a smaller model (student).
- Why?
  - ≈ 95 % accuracy with < ½ parameters.
  - Memory & latency ↓ → deploy on edge/phone.
  - Decrease inference cost significantly.

Example: GPT-o3 → GPT-o3-mini

Knowledge Distillation (Teacher-Student Models)

There are many ways of doing distillation, but simplest way is:

1. Inference teacher on large unlabeled pool ⟶ soft logits.
2. Consider these logits as labels.
3. Train student on these labels with loss:

```
KLDiv(Student logits ‖ Teacher logits)
```

Thanks for Attending!