Regularización, LR schedules, dashboards

Cómo entrenar redes sin que se mientan a sí mismas.

Conceptos clave

- **Dropout**: apaga neuronas con probabilidad $p \in [0{,}1, 0{,}5]$. Solo en entrenamiento.
- **BatchNorm**: normaliza activaciones por minibatch. Permite lr más alto, acelera convergencia.
- **Weight decay** λ: penaliza pesos grandes. Valor típico $\lambda \in [10^{-5}, 10^{-3}]$.
- **Cosine LR + early stopping**: combo por defecto en 2026 con AdamW.

Ejemplo a mano

Trazado de early stopping con `patience` = 12:

Época	Train	Val	¿mejor val?	Acción
10	0,34	0,34	sí ★	guardar checkpoint
15	0,20	0,35	no, +1	seguir
22	0,10	0,48	no, +8	seguir, atención
28	0,07	0,62	no, +14	**patience superada ⇒ stop**

Receta exprés: AdamW(lr=10^{-3}, weight_decay=10^{-4}) + CosineAnnealingLR(T_max=epochs) + Dropout 0,3 + BatchNorm + EarlyStopping(patience=10).

Concepto → aplicación de la regularización

Técnica	Cuándo	Cuándo evitarla
Dropout	MLP, CNN, modelos tabulares medianos.	Modelos muy pequeños.
BatchNorm	CNN sobre imágenes, MLP densos.	Batch ≤ 4, modelos pre-entrenados.
Weight decay	Casi siempre.	Modelos sub-ajustados (empeora).
Early stop	Casi siempre, cuesta cero.	Dataset trivial sin curva.
LR cosine	Por defecto con AdamW.	Datasets pequeños sin tendencia.

Idea clave del Día 03. **Sin dashboard no hay reproducibilidad ni decisiones defendibles.** Loguea `loss/train`, `loss/val`, `metric/auc`, `optim/lr` por época en W&B o MLflow.

Para llevarte: lectura Smith, *Cyclical Learning Rates*; aplicar la receta exprés al MLP del Día 02 y comparar.