Clase 117 — Regularización moderna: Stochastic Depth, DropPath, LayerDrop

Parte: 2 — Deep Learning · Fuente: Huang et al. (2016) Stochastic Depth + Fan et al. (2020) LayerDrop + DropPath en ViT/Swin/ConvNeXt. Duración estimada: 70 min.

Clase 117 — Regularización moderna: Stochastic Depth, DropPath, LayerDrop

> *Parte: 2 — Deep Learning · Fuente: Huang et al. (2016) Stochastic Depth + Fan et al. (2020) LayerDrop + DropPath en ViT/Swin/ConvNeXt. Duración estimada: 70 min.*

Objetivo

Aplicar regularización por paths/bloques —más allá del dropout clásico— en arquitecturas profundas modernas (ResNet, ViT, ConvNeXt, Swin Transformer). Cubrir Stochastic Depth (drop bloque residual), DropPath (drop path en transformer), LayerDrop (drop layer completa). Beneficio doble: regularización + reducción de cómputo durante training.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Aplicar keras.layers.StochasticDepth(rate) o DropPath(rate) en bloques residuales.
- Diseñar rate lineal por profundidad: capa 0 → 0.0, capa N → 0.2.
- Aplicar LayerDrop durante pretraining para permitir inference con menos capas.
- Diferenciar Dropout (neurona) vs Stochastic Depth (bloque) vs LayerDrop (layer).
- Reconocer el speedup de training: 25 % más rápido en ResNet-110 (Huang 2016).

Temas

- Dropout clásico vs Stochastic Depth.
- DropPath = Stochastic Depth para Transformers.
- Rate lineal: p_i = i/N · p_max.
- LayerDrop para compresión de Transformers.
- Combinaciones: Dropout + DropPath + Label Smoothing.

Definiciones y características

- Stochastic Depth: y = x + drop(b(x)) con prob p_i. En inference, scale por 1 - p_i.
- DropPath: igual idea aplicada a attention y FFN paths.
- LayerDrop: dropear layer entera del Transformer.
- Linear rate schedule: capas tempranas seguras (p≈0), tardías regularizadas (p≈0.2).

Dataset / recursos

- CIFAR-10/100 con ResNet propia.
- ViT-Tiny preentrenado.
- Librerías: torch, timm (donde DropPath es default).

Ejercicios

1. ResNet con Stochastic Depth: implementar BasicBlock con StochasticDepth(p). Train CIFAR-10.
2. Rate lineal: aplicar p_i = i/N · 0.2 en cada bloque. Comparar contra rate constante.
3. ViT con DropPath: timm.create_model('vit_tiny_patch16_224', drop_path_rate=0.1). Comparar contra 0.0.
4. LayerDrop: simular con 12-layer BERT mini — drop 50 % layers. Verificar accuracy aún razonable.
5. Speed: medir wall-clock training con vs sin Stochastic Depth.

Homework verificable

ResNet-50 en CIFAR-100:

1. Entrenar con y sin Stochastic Depth (rate lineal a 0.2 final).
2. Reportar accuracy + tiempo total.
3. Verificar regularización: gap train-val.

Criterio de aceptación: Stochastic Depth reduce gap train-val ≥ 1 pp; speedup en wall-time visible (~10-20 %).

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Rate constante 0.5 en todas las capas	Mata bloques iniciales (críticos). Fix: li
Olvido scale en inference	Sesgo. Fix: librerías lo manejan; verifica
DropPath sin residual	Sin sentido — no hay path para droppear. F
Aplicar a 1ª y última capa	Generalmente innecesario. Fix: layers inte
Combinar Dropout + Stochastic Depth + Labe	Overregularization. Fix: ajustar individua

Preguntas frecuentes

Stochastic Depth o Dropout?

Para CNN/ViT profundas: ambos. Stochastic Depth en bloques residuales, Dropout en MLP final.

Rate final 0.1 o 0.2 o 0.5?

0.1-0.2 para ResNet/ViT base. 0.3-0.5 para modelos enormes (LayerDrop en BERT-Large).

LayerDrop sirve para inference?

Sí — train con LayerDrop=0.5, inference con k random layers. Habilita modelos compactos sin re-training.

timm vs implementación propia?

Usá timm siempre que se pueda — DropPath bien implementado, rate scheduling automático.

En LLMs modernos (Llama)?

Less común. Llama no usa DropPath. BERT/RoBERTa sí.

Referencias

- Huang et al. (2016), Deep Networks with Stochastic Depth, ECCV.
- Fan et al. (2020), Reducing Transformer Depth on Demand with Structured Dropout (LayerDrop), ICLR.

- Dosovitskiy et al. (2020), ViT — DropPath aplicado.
- timm DropPath.

Siguiente clase

Clase 118 — TensorFlow: tensores, variables, operaciones

Apéndice: notebook (primer bloque)

Huang et al. 2016: durante entrenamiento, droppea capas residuales enteras con prob p_l. En inference, escalá el output. Implementamos sin torch — todo numpy + sklearn.

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(42)

def stochastic_depth(x, p, training=True):
    """Si training: con prob p dropea (devuelve 0). Si no, escala por (1-p).
    Funciona para batch o single sample. p=0 → identidad.
    """
    if not training:
        return x * (1 - p)
    if np.random.rand() < p:
        return np.zeros_like(x)
    return x
```

Archivos complementarios

- notebook.ipynb