Clase 109 — Activaciones: ReLU, ELU, GELU, Swish, Mish

Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § Better Activation Functions. Duración estimada: 60 min.

Clase 109 — Activaciones: ReLU, ELU, GELU, Swish, Mish

> *Parte: 2 — Deep Learning · Fuente: Géron, cap. 11 § Better Activation Functions. Duración estimada: 60 min.*

Objetivo

Conocer la familia de activaciones modernas — desde ReLU (Krizhevsky et al. 2012) hasta GELU (BERT, GPT) y Swish/SiLU (EfficientNet) — entendiendo qué problema resuelve cada una y por qué los Transformers modernos usan GELU y no ReLU. Saber elegir según arquitectura.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Definir matemáticamente las 5 activaciones: ReLU, Leaky ReLU, ELU, GELU, Swish (SiLU), Mish.
- Identificar dying ReLU y aplicar Leaky ReLU / ELU como mitigación.
- Reconocer que GELU es la activación default en Transformers (BERT, GPT, ViT) y Swish/SiLU en EfficientNet, modelos modernos de visión.
- Aplicar cada activación en Keras: Dense(64, activation='relu' | 'gelu' | 'swish' | 'elu' | LeakyReLU()).
- Saber que el costo computacional de GELU/Swish es mayor (sigmoid/erf internos) pero el beneficio supera en arquitecturas profundas.

Temas

- ReLU: max(0, x). Rápida, simple, default histórico. Dying ReLU.
- Leaky ReLU: max(αx, x) con α≈0.01. Sin dying.
- ELU (Clevert et al. 2015): x si x>0, α(e^x-1) si x<0. Suave, sin dying, pero más cara.
- GELU (Hendrycks & Gimpel 2016): x · Φ(x) (Φ = CDF gaussiana). Suave, no monótona. Default en Transformers.
- Swish / SiLU (Ramachandran et al. 2017): x · sigmoid(x). Encontrada por NAS. Casi idéntica a GELU.
- Mish (Misra 2019): x · tanh(softplus(x)). Marginalmente mejor en algunos benchmarks.

Definiciones y características

- Función de activación: no linealidad aplicada elementwise tras la transformación lineal de una capa.
- Saturación: cuando la derivada se acerca a 0 → vanishing.
- Monótona: f(x) creciente en todo el dominio. ReLU/ELU/Leaky lo son; GELU/Swish/Mish no estrictamente.
- Bounded: salida acotada. Sigmoid (0,1), tanh (-1,1), softmax. La mayoría modernas no son bounded.
- Smooth: derivable continuamente. ReLU no en x=0; las demás sí.

Dataset / recursos

- Fashion-MNIST.
- Librerías: tensorflow, keras.

Ejercicios

1. Plot de funciones: graficar las 6 activaciones en x [-3, 3].
2. Comparación empírica: entrenar MLP [256, 128, 64] con cada activación, mismo init He, mismo LR. Comparar val_accuracy tras 15 épocas.
3. Dying ReLU: con LR alto (0.1), entrenar con ReLU. Contar cuántas neuronas tienen mean(activation) = 0 al final.
4. Leaky ReLU al rescate: repetir con Leaky. Verificar que el % de neuronas muertas baja.
5. GELU vs ReLU en profundidad: armar un MLP de 12 capas. Comparar GELU vs ReLU. GELU suele ganar.

Homework verificable

Sobre Fashion-MNIST, MLP [300, 200, 100, 50, 10]:

1. Entrenar 4 modelos con: ReLU, Leaky ReLU(α=0.1), ELU, GELU.
2. Reportar val_accuracy y tiempo por época.
3. Para el mejor, inspeccionar % de neuronas "muertas" por capa.

Criterio de aceptación: GELU o ELU debe ganar en accuracy; Leaky/ELU/GELU deben tener < 5 % neuronas muertas; ReLU posiblemente más.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
Cambio de ReLU a sigmoid en capa oculta pa	Vanishing immediato en redes profundas. Fi
Leaky ReLU con α=0.5	Casi lineal, pierde la no linealidad útil.
activation='swish' no reconocido	Keras antiguo lo llama 'swish', TF moderno
GELU en CPU es lento	Es ≈ 2× más caro que ReLU por la erf. En C
Mish en GPU sin implementación optimizada	TF puede no tener kernel cuda eficiente pa

Preguntas frecuentes

¿Qué activación uso por default en 2026?

- MLP/CNN clásica: ReLU o GELU.
- Transformers: GELU (es lo que usan BERT, GPT, ViT).
- EfficientNet/MobileNet: Swish (= SiLU).
- Modelos profundos sin BN: ELU o GELU (más estables).

Swish y SiLU son lo mismo?

Sí, mismo formato (x · sigmoid(x)). SiLU es el nombre PyTorch; Swish es el nombre de Google.

¿Por qué GELU en Transformers?

Históricamente: BERT (2018) la eligió porque parecía funcionar mejor empíricamente. Hoy es estándar más por inercia/compatibilidad que por una ventaja categórica. En benchmarks recientes, Swish es competitiva.

¿GELU exacta o aproximada?

tf.keras.activations.gelu(x, approximate=False) usa erf exacta. True usa la aproximación tanh, ≈10× más rápida en GPU. La diferencia numérica es < 0.001.

¿La salida del modelo también lleva GELU?

No. La activación final depende del problema (linear, sigmoid, softmax). GELU/ReLU son para capas ocultas.

Referencias

- Géron, cap. 11 — Better Activation Functions.
- Krizhevsky et al. (2012), AlexNet — ReLU al mainstream.
- Clevert et al. (2015), ELU, ICLR.
- Hendrycks & Gimpel (2016), Gaussian Error Linear Units (GELUs).
- Ramachandran et al. (2017), Searching for Activation Functions, ICLR — Swish.
- Misra (2019), Mish: A Self Regularized Non-Monotonic Activation Function.

Siguiente clase

Clase 110 — Batch Normalization, Layer Normalization

Apéndice: notebook (primer bloque)

Primera celda ejecutable del notebook de la clase.

```
# Imports y configuración inicial
```

Archivos complementarios

- notebook.ipynb