Tipos de redes neuronales y cuál usar: chuleta con código

> Todas las redes neuronales usan la misma pieza: **neuronas que suman sus entradas por unos pesos y deciden, en capas, con los pesos ajustados por retropropagación**. Lo que cambia es cómo se conectan: **densas** para datos sueltos, **convolucionales** para imágenes, **recurrentes** para leer en orden y **Transformers** para texto. Apila Transformers y tienes GPT.

Los tipos de redes neuronales, de 1958 a GPT

Red	Qué hace	Cuándo usarla
Perceptrón Rosenblatt, 1958. Una sola neurona.	Suma las entradas por sus pesos y dice sí o no. Solo sabe separar con una línea recta.	Para entender cómo aprende una red. Si tu problema se separa con una recta, no necesitas más.
Densa (perceptrón multicapa) Retropropagación: Rumelhart, Hinton y Williams, 1986.	Capas ocultas en las que cada neurona mira todas las salidas de la capa anterior. Ya separa con curvas.	Como capa final de las demás redes. Con datos en tabla, solo si un modelo clásico se queda corto.
Convolucional LeCun y otros, 1989. AlexNet la hizo despegar en 2012.	Filtros pequeños (5×5 en el ejemplo) que recorren la imagen. El mismo filtro sirve en cualquier posición.	Imágenes: clasificar, detectar, segmentar. Parte de una ya entrenada y ajústala.
Recurrente (LSTM) Hochreiter y Schmidhuber, 1997.	Lee en orden, una pieza tras otra, arrastrando una memoria de lo que ya ha leído.	Señales que llegan en directo a un aparato pequeño, como un sensor: es ligera y lee sobre la marcha.
Transformer Google, 2017: «Attention Is All You Need».	Con la atención, cada posición mira a la vez todas las anteriores y elige cuáles le importan. Se entrena en paralelo.	Texto, y casi todo lo que hoy se hace a gran escala. Ajusta uno que ya exista, o úsalo directamente.
GPT OpenAI: GPT-2 en 2019, GPT-3 en 2020.	Muchos bloques de Transformer apilados, entrenados para predecir el siguiente trozo de texto.	Cuando necesitas generar o entender texto: no lo entrenas tú, lo usas o lo ajustas.

Las pruebas reales: cada red, entrenada en un portátil

Todas en un MacBook Pro M2 Max. Los números salen de los scripts que tienes más abajo.

Prueba	Resultado	Qué te enseña
Perceptrón con AND y con XOR (o exclusivo), 20 vueltas	AND: aprende en la 5.ª vuelta. XOR: 2 de 4 aciertos en las 20.	Una neurona sola no puede con lo que no se separa con una recta.
Capa oculta de 4 neuronas con XOR	4 de 4 con 17 pesos (salidas 0,000 · 0,997 · 0,997 · 0,004).	Las capas ocultas y la retropropagación son lo que desatascó las redes en 1986.
Dígitos escritos a mano (MNIST), 3 épocas: densa contra convolucional	Densa: 101.770 pesos, 96,27 %. Convolucional: 11.274 pesos, 98,27 %.	Con imágenes, la convolucional acierta más con 9 veces menos pesos.
Los mismos dígitos movidos 4 píxeles a la derecha	Densa: 30,51 %. Convolucional: 67,08 %.	Una red que va bien en el test puede hundirse con un cambio que tú ni ves.
Recurrente (LSTM, 9,3 M de pesos) contra mini GPT (10,8 M), 10 minutos cada una con 7,7 millones de caracteres de un podcast	Error de validación: 1,112 la LSTM, 1,111 la mini GPT. Empate.	En pequeño, el Transformer no gana. Solo se adelanta, por poco, al final de cada texto (posiciones 129-256: 1,093 frente a 1,085).

El código de las cuatro redes, línea a línea

Tres scripts listos para ejecutar, con `uv`:

- perceptron.py: perceptrón y capa oculta con XOR. `uv run --with numpy python perceptron.py`
- digitos.py: densa contra convolucional con MNIST. `uv run --with torch --with torchvision python digitos.py`
- texto.py: recurrente contra mini GPT con tus propios textos. `uv run --with torch --with numpy python texto.py mis_textos/ 10`

Qué escribes	Qué hace	Cuándo usarlo
`err = t - int(w @ x + b > 0)` `w, b = w + err * x, b + err`	La regla del perceptrón: si falla, mueve los pesos hacia la respuesta correcta; si acierta, no toca nada.	Para ver aprender a una neurona en dos líneas. No sirve para redes con capas ocultas.

Qué escribes	Qué hace	Cuándo usarlo
`loss.backward()` `opt.step()`	Retropropagación: calcula cuánto culpa tiene cada peso del error, y el optimizador los ajusta.	En cada paso de entrenamiento de cualquier red con capas ocultas. Antes, `opt.zero_grad()`.
`nn.Linear(784, 128)`	Capa densa: 128 neuronas, cada una conectada a los 784 píxeles (más un sesgo).	Al final de casi cualquier red, para decidir. Como única capa con imágenes, aprende posiciones y se hunde si mueves el dibujo.
`nn.Conv2d(1, 8, 5, padding=2)`	8 filtros de 5×5 que recorren la imagen; `padding=2` mantiene el tamaño de 28×28.	La primera capa con imágenes. Sube el número de filtros en las capas siguientes (8 → 16 en el ejemplo).
`F.max_pool2d(x, 2)`	Se queda con el valor más alto de cada cuadro de 2×2: la imagen pasa a la mitad.	Tras cada convolución, para resumir y dar margen a que el trazo se mueva un poco.
`nn.LSTM(d, oculto, capas, batch_first=True)`	Recurrente: lee la secuencia en orden y devuelve una salida por posición más su memoria.	Para texto o señales en orden con pocos recursos; la memoria se puede ir pasando pieza a pieza.
`F.scaled_dot_product_attention( q, k, v, is_causal=True)`	La atención: cada posición pondera a las demás; `is_causal=True` le prohíbe mirar las que vienen después.	Dentro de cada bloque de Transformer que genera texto: sin la máscara causal, haría trampa viendo la respuesta.
`p = F.softmax(logits / temp, -1)` `torch.multinomial(p, 1)`	Elige la letra siguiente al azar según sus probabilidades. La temperatura (0,8 en el ejemplo) las aplana o las afila.	Al generar texto: temperatura baja, más repetitivo y seguro; alta, más variado y más disparates.

Qué red usar para cada cosa: 5 reglas

Qué tienes	Qué usar	Cuándo cambiar de idea
1. Datos en tabla, como una hoja de cálculo	Primero un modelo clásico, como un bosque aleatorio. Muchas veces gana a la red.	Si con muchos datos el modelo clásico se estanca, prueba una red densa.
2. Imágenes	Convolucional, y casi nunca desde cero: parte de una ya entrenada.	Entrena desde cero solo si tus imágenes no se parecen a nada que exista (un sensor propio, por ejemplo) y tienes muchas.

Qué tienes	Qué usar	Cuándo cambiar de idea
3. Texto	Transformer. Tampoco desde cero: ajusta un modelo que ya existe, o úsalo directamente.	Entrénalo tú (como la mini GPT) para aprender cómo funciona, no para producción.
4. Señales en directo en un aparato pequeño, como un sensor	Una recurrente sigue teniendo sentido: es ligera y lee sobre la marcha.	Si el aparato tiene potencia de sobra y necesitas recordar mucho atrás, compara con un Transformer pequeño.
5. Cualquier red	Pruébala con casos como los de verdad, no solo con el test limpio.	Siempre. La red de dígitos pasó del 96 % al 30 % por mover el número 4 píxeles.

Errores típicos

- **Usar una red para datos en tabla sin probar antes un modelo clásico.** Un bosque aleatorio es rápido de entrenar y muchas veces gana. Empieza por él.
- **Entrenar desde cero lo que ya existe entrenado.** Para imágenes y texto hay redes preentrenadas mucho mejores que lo que sacarás en tu ordenador. Ajústalas.
- **Fiarte del porcentaje del test.** El 96 % de la red densa se convirtió en un 30 % con los dígitos movidos 4 píxeles. Prueba con datos como los que llegarán de verdad.
- **Usar una red densa para imágenes.** Aprende posiciones, no formas: necesita 9 veces más pesos que la convolucional y aun así falla más.
- **Sacar conclusiones de escala de una prueba pequeña.** A 10 millones de pesos, la LSTM empata con la mini GPT. El Transformer gana cuando crece y cuando el texto es largo.

Preguntas frecuentes

¿Qué tipos de redes neuronales hay?

Las grandes familias son cuatro: las densas (perceptrón multicapa), donde cada neurona mira todas las entradas; las convolucionales, que recorren la imagen con filtros pequeños; las recurrentes (como la LSTM), que leen en orden arrastrando una memoria; y los Transformers, que con la atención miran todo el texto a la vez. Todas usan la misma pieza, la neurona que suma y decide, y aprenden con retropropagación.

¿Por qué un perceptrón no puede aprender XOR?

Porque un perceptrón solo separa con una línea recta, y los casos de XOR (o exclusivo) no se pueden separar con una sola recta. En la prueba de esta chuleta se queda en 2 de 4 aciertos durante 20 vueltas. Con una capa oculta de 4 neuronas, 17 pesos en total, acierta los 4.

¿Qué diferencia hay entre una red convolucional y una red densa con imágenes?

La densa conecta cada neurona con todos los píxeles, así que aprende posiciones concretas. La convolucional usa los mismos filtros pequeños en toda la imagen, así que reconoce un trazo esté donde esté. En la prueba con dígitos, la convolucional acertó el 98,27 % con 9 veces menos pesos, y con el dígito movido 4 píxeles aguantó un 67 % frente al 30,5 % de la densa.

¿Qué es un Transformer y qué tiene que ver con GPT?

Un Transformer es una red que usa la atención: cada posición del texto mira a la vez a todas las anteriores y decide cuáles le importan. Lo publicó un equipo de Google en 2017. GPT es eso mismo apilado muchas veces y entrenado para predecir lo que viene después: la mini GPT de esta chuleta tiene 6 bloques y unos 10 millones de pesos; GPT-3, 175.000 millones.

¿Las redes recurrentes (LSTM) están obsoletas?

Para texto a gran escala, sí las ha superado el Transformer. Pero en pequeño no pierden: en la prueba de esta chuleta, una LSTM y una mini GPT de tamaño parecido empataron tras 10 minutos de entrenamiento (1,112 frente a 1,111). Siguen teniendo sentido para señales que llegan en directo a un aparato pequeño, porque son ligeras y leen sobre la marcha.