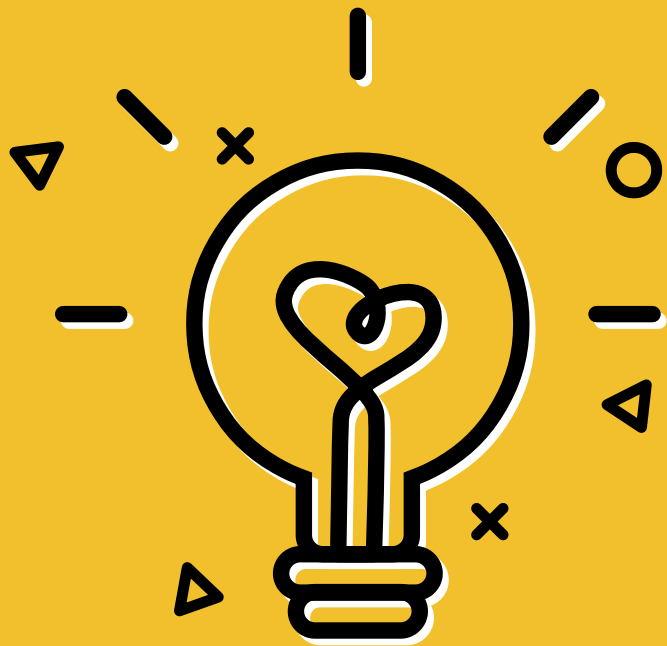


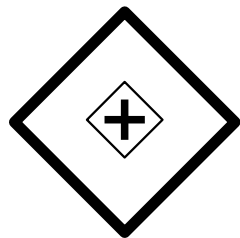


REDES PROFUNDAS

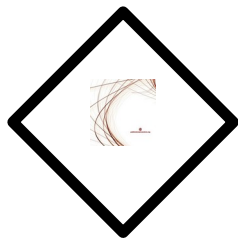
O sea, grandes



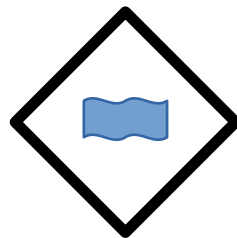
¿Qué encontramos?



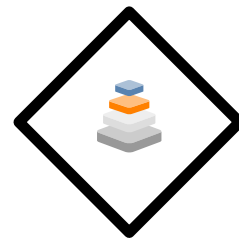
Aunque ya tengamos buen ajuste, puede ser beneficioso ampliar la red para mejorar el margen de precisión y la generalización



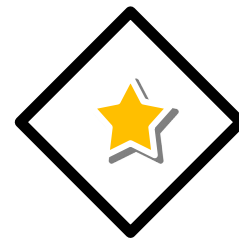
Una interpolación suave en D dimensiones, requiere D veces más parámetros que una interpolación a secas, que ya son muchos para muestras grandes.



En errores cuadráticos, los mínimos locales presentan errores más cercanos a los globales si la red se hace mayor.



Las redes profundas parecen la solución mejor para datos en los que haya una estructura compleja



Con datos complejos, es difícil encontrar modelos con resultados máximos que tengan muchos menos parámetros que muestras



**Veamos sus
características y
dificultades**

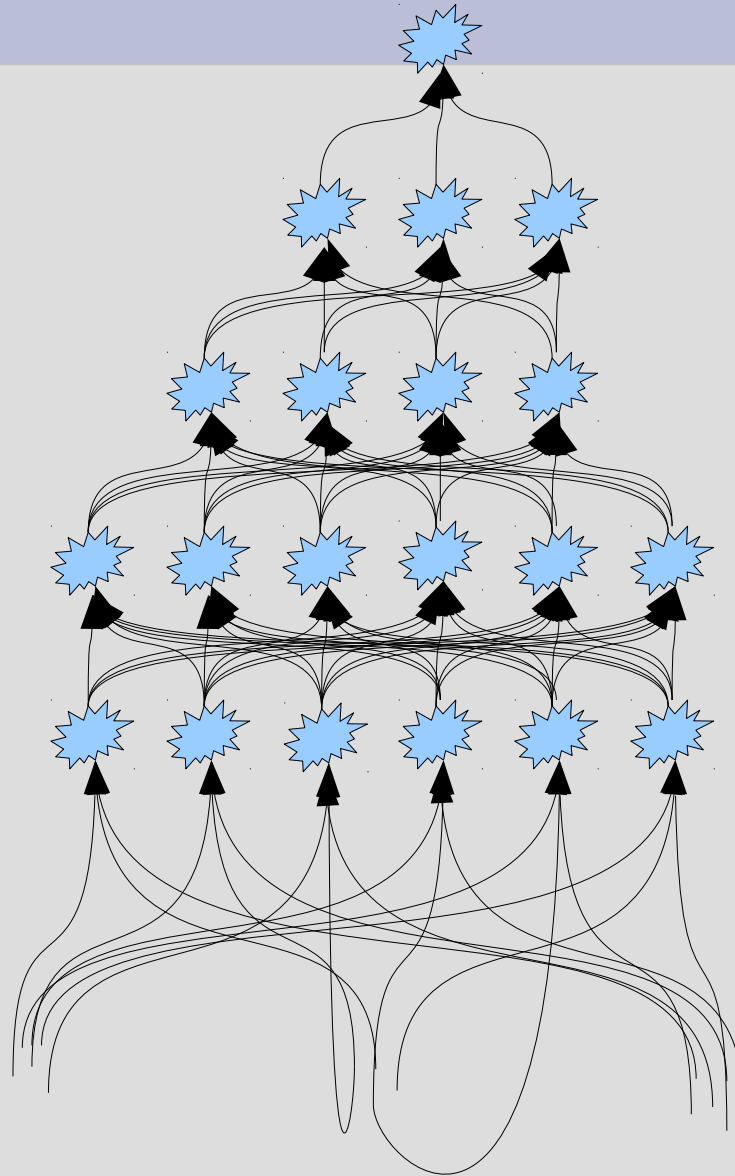
Ajuste en profundidad

Perceptrones grandes: varias capas

Ajuste en profundidad

- ¿Qué has oído de Deep Learning?

Arquitectura profunda



Problemas de ajuste

- Convergencia de los métodos basados en gradiente
 - Gradiente puede hacerse demasiado pequeño o demasiado grande: limitar la magnitud máxima y mínima
 - Cuidado con la inicialización: valores pequeños
- El algoritmo de ajuste tiene que moverse en una dimensión altísima
 - Espacio de búsqueda enorme (iteraciones necesarias, influencia de punto inicial)
 - Función de error con geometría enrevesada, mínimos locales

Problemas de ajuste

- Gradientes. Extiende a 5 capas el caso de 2. Da cotas razonables

$$\frac{\partial E}{\partial p_{ij}} = \left(\sum_k \frac{\partial E}{\partial y_{s_k}} \frac{\partial y_{s_k}}{\partial e_{s_k}} \frac{\partial e_{s_k}}{\partial y_i} \right) \frac{\partial y_i}{\partial e_i} \frac{\partial e_i}{\partial p_{ij}}$$

$$\begin{aligned} \frac{\partial E}{\partial y_{s_k}} &= 2(y_{s_k} - c_k) \\ \frac{\partial y_{s_k}}{\partial e_{s_k}} &= 1 - y_{s_k}^2 & \frac{\partial y_i}{\partial e_i} &= 1 - y_i^2 \\ \frac{\partial e_{s_k}}{\partial y_i} &= p_{s_k i} \\ \frac{\partial e_i}{\partial p_{ij}} &= y_j \end{aligned}$$

- Dimensión: 1000 entradas y capas con 300, 100, 50, 20, 5 y 1 procesador (5 ocultas) ¿Cuántos pesos tiene? No olvides los términos independientes

Normalización

- Cada capa puede considerarse un pequeño perceptrón que se ajusta sobre las anteriores
- Pero como las anteriores se están ajustando también, su entrada se mueve
- La podemos llevar cada ciertas iteraciones a media cero y varianza unidad
- La transformación se compensa con unos parámetros de afinidad que son ajustables

Normalización

- ¿Dónde habías visto esta estrategia antes de ahora? ¿Con qué explicación?

Problemas de generalización

- Demasiados grados de libertad
 - Peligro de que el modelo se ajuste a peculiaridades de la muestra (varianza del modelo, influencia del ruido)
- Procedimientos generales para garantizar extrapolación:
 - Penalizar/forzar que la respuesta sea a veces no nula en los ocultos
 - Reducir el número de pesos libres, porque se usan los mismos en muchos procesadores

Problemas de generalización

- ¿Cuál es la relación en forzar un procesador a 0 y los grados de libertad?
- Vuelve a calcular el número de pesos en la red de 5 capas ocultas si la mitad de los procesadores de cada capa comparten los mismos pesos

¿Por qué lo de que haya muchos nulos?

- Se parece más a lo biológico: hay pocas neuronas activas en cada momento
- Implica dimensión alta:
 - Mejor separabilidad (como SVM)
 - Permite representaciones de tamaño "variable", que son más útiles en muchos problemas
- ~ clasificación: mejor representación
- La función de error tiene mínimos más claros
- Se limitan los grados de libertad locales, pero se permite una flexibilidad global

¿Por qué lo de que haya muchos nulos?

- Si queremos que haya al menos N procesadores activos, pero queremos que sólo sean el $P\%$ del total ¿cuántos hay en total?
Prueba N y P en decenas

Autocodificadores

- Cada capa se inicializa como un codificador de las entradas
 - Entrada: capa anterior (o entrada externa), quizá perturbada
 - Salida: lo mismo, sin perturbación

Autocodificadores

- Plantea esquema para 6 entradas y capa de 4 procesadores

Autocodificadores-ajuste

- Se preajustan los pesos de las capas inferiores hacia arriba
- En cada caso se intenta hacer un autocodificador: que la salida coincida con la entrada
 - Capa intermedia de tamaño menor
 - Penalizar capa intermedia nula o robusta frente a ruido
 - Obligar a pocos activos evita soluciones triviales: la salida copia la entrada
- Las capas anteriores ya no se ajustan más, excepto al aprender la salida: descenso gradiente estocástico

Autocodificadores-ajuste

- Plantea el proceso para la red de 5 capas ocultas

Máquinas de Boltzmann/creencia profundas

- Cada capa se entrena como si fuera la oculta de una máquina de Boltzmann restringida
- La primera capa se contruye como MBR
- Las capas sucesivas también, pero fijando los pesos de las capas anteriores

Máquinas de Boltzman/creencia profundas

- Busca qué es una máquina de Boltzman restringida

Redes recursivas

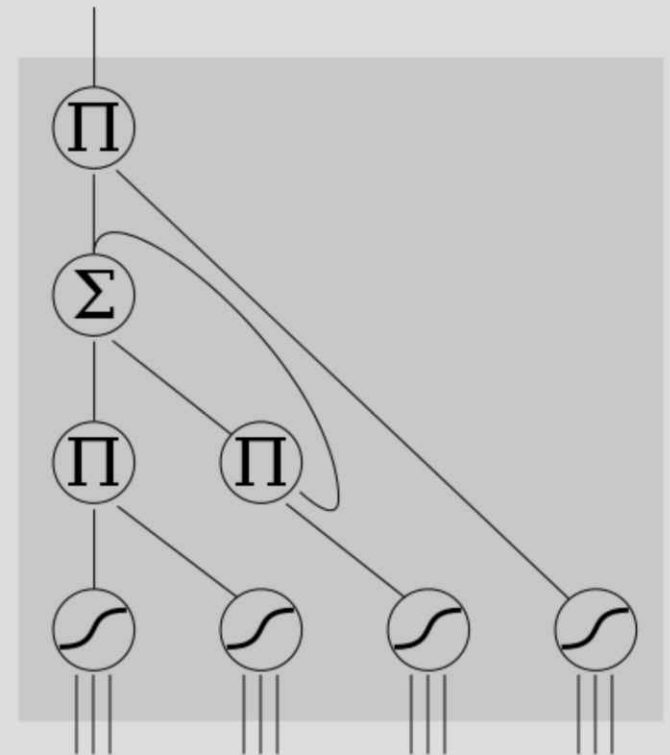
- Pequeña red que se aplica para obtener una representación intermedia, sobre la que se vuelve a aplicar, etc. etc.
- Al final es una red profunda pero con la restricción de que los pesos son los mismos en cada copia de la pequeña red

Redes recursivas

- ¿Cuál es la relación con el caso de que los procesadores compartan pesos?

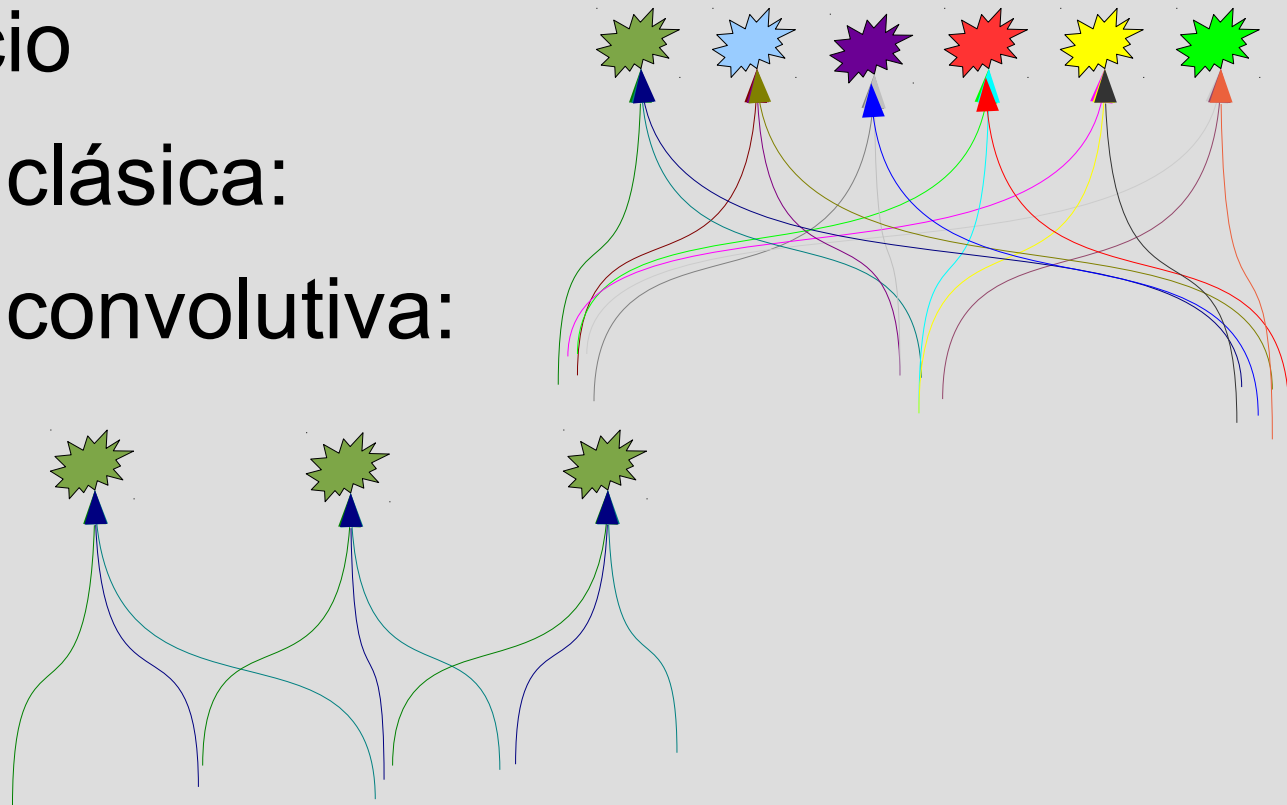
Red recursiva: memoria de largo y corto plazo

- Red modular construida con pequeños bloques que son miniredes realimentadas
- Cada bloque tiene cuatro lotes de entradas
- Se usa recursivamente



Redes convolutivas - Idea

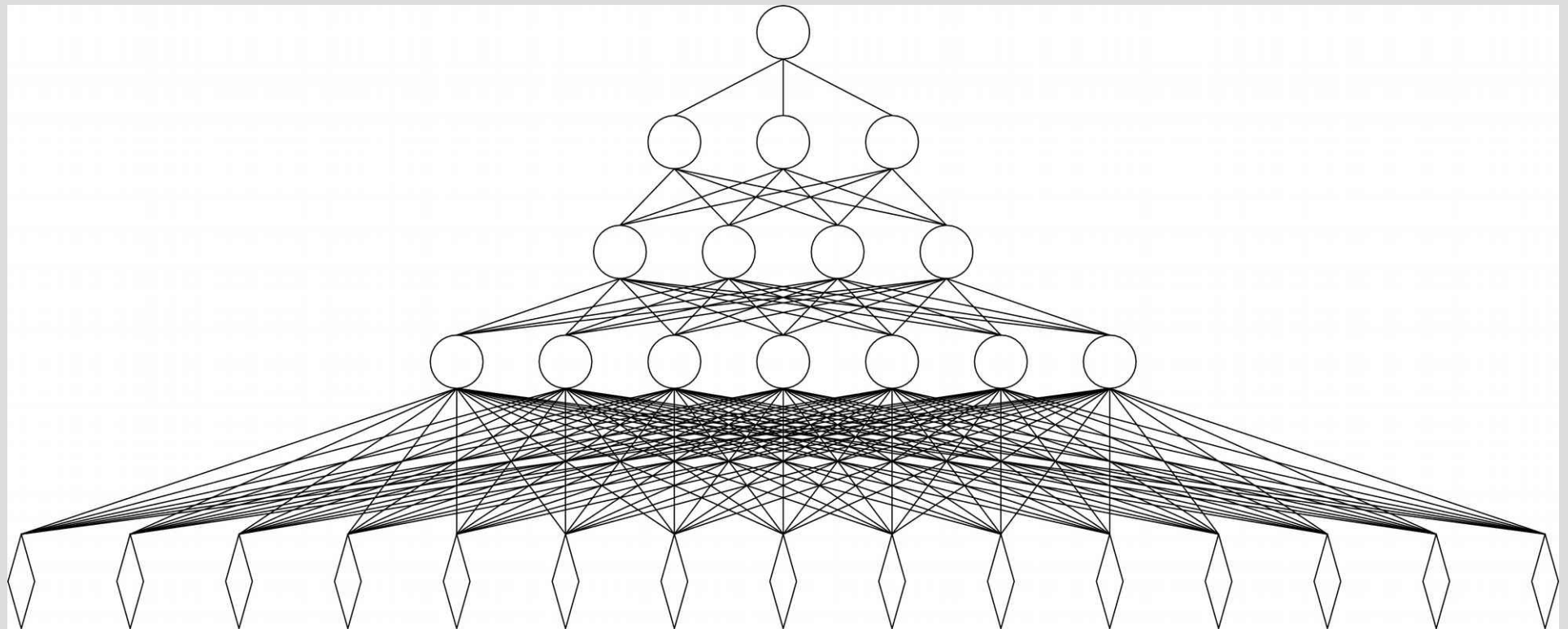
- Entradas: secuencias, series históricas, señal en el tiempo, imágenes, señales en el espacio
- Capa clásica:
- Capa convolutiva:



Redes convolutivas - idea

- Comenta un caso de convolución que conozcas

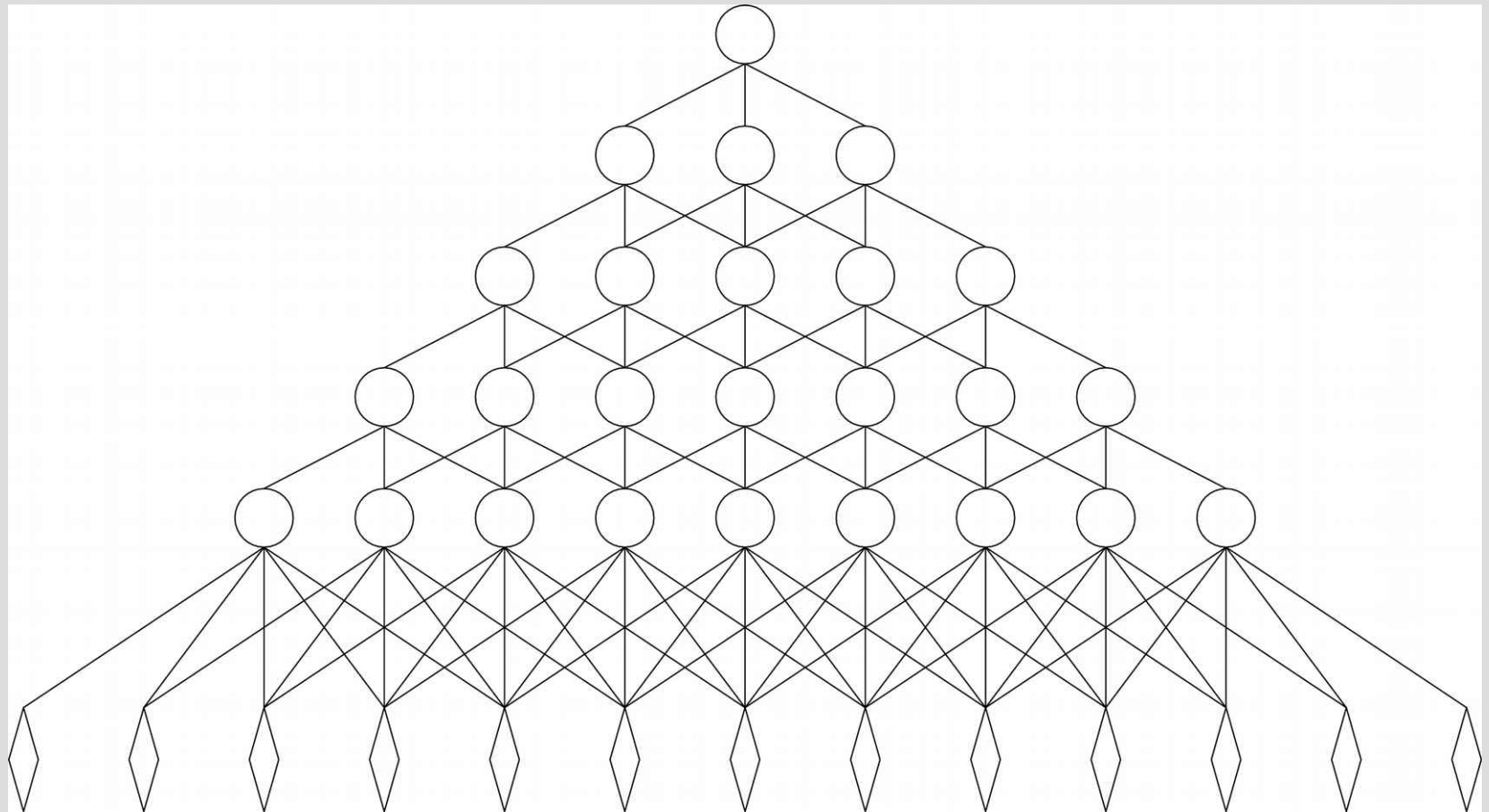
Red profunda no convolutiva



Red profunda no convolutiva

- ¿Cuántos pesos tiene?

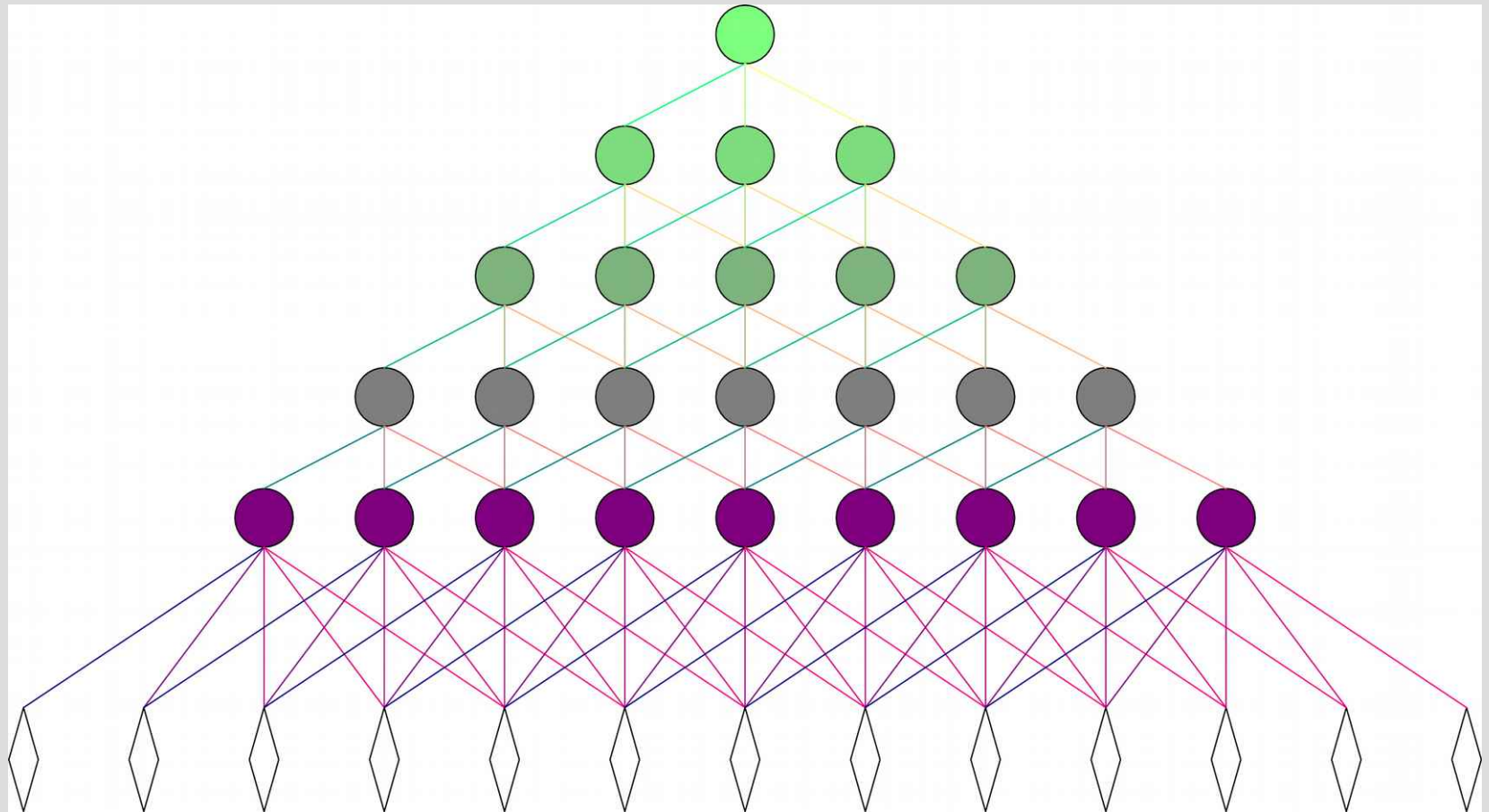
Red profunda con conexión convolutiva



Red profunda con conexión convolutiva

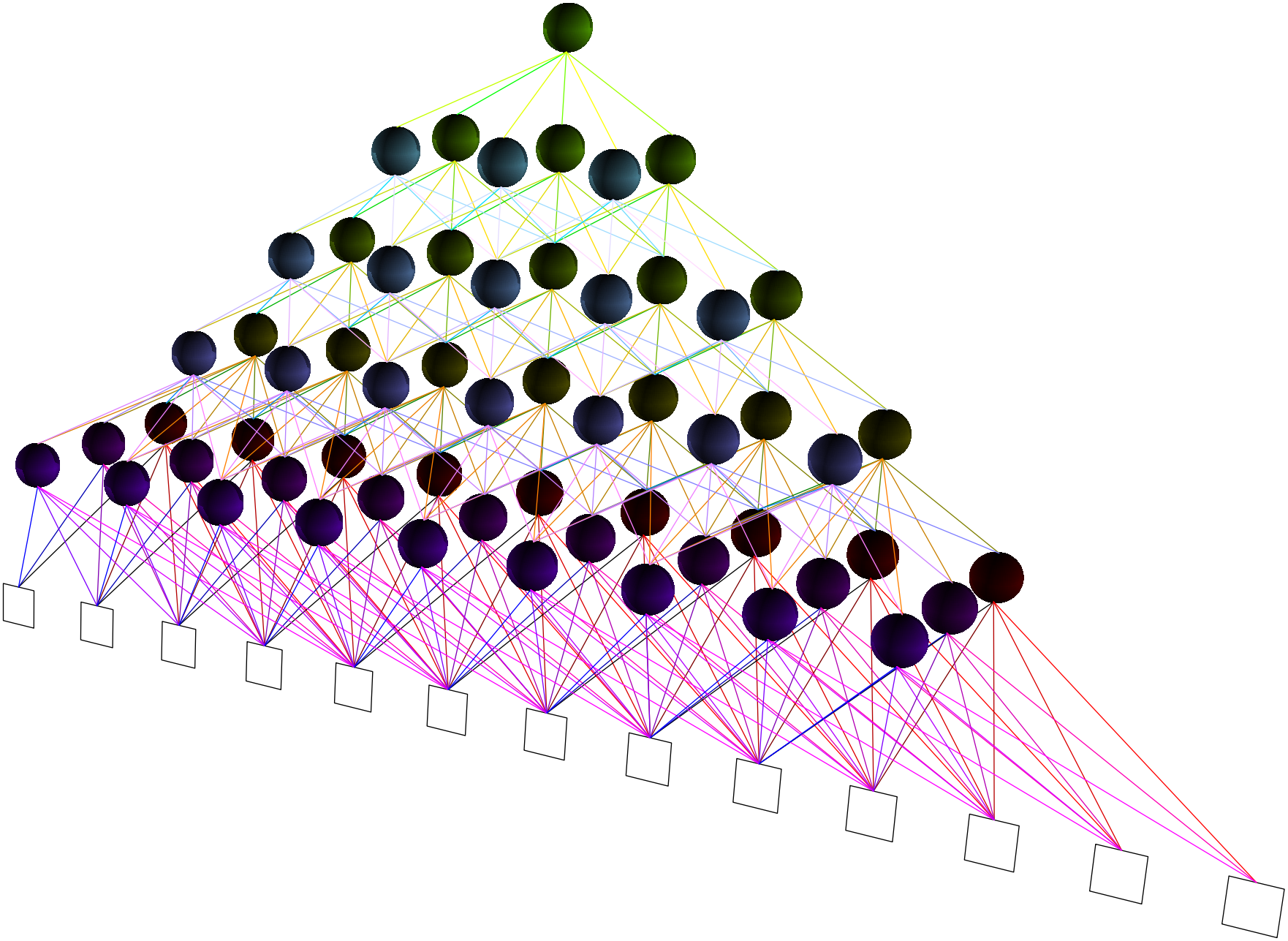
- ¿Cuántos pesos tiene?

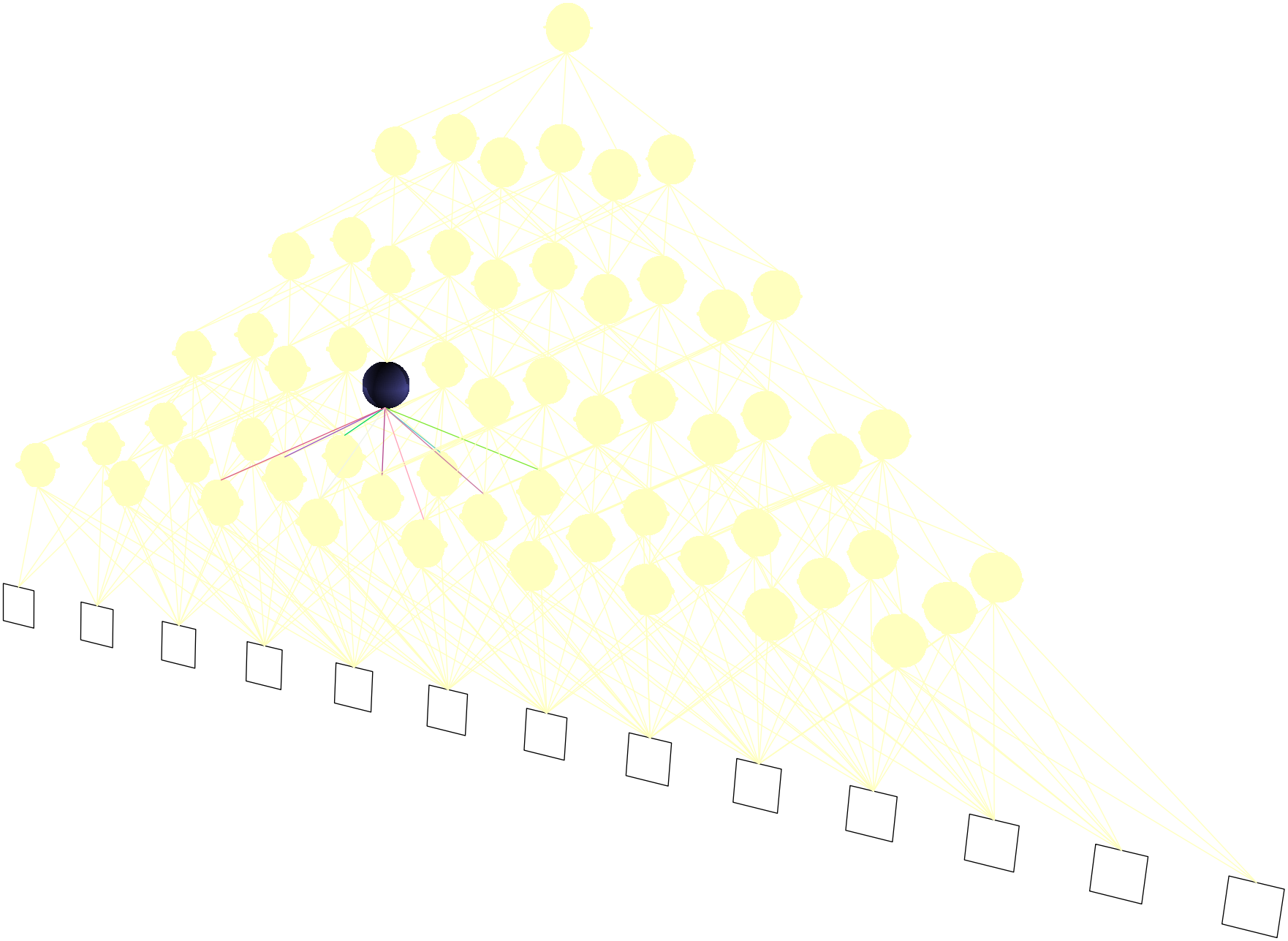
Red profunda convolutiva: hay 1 procesador libre por capa

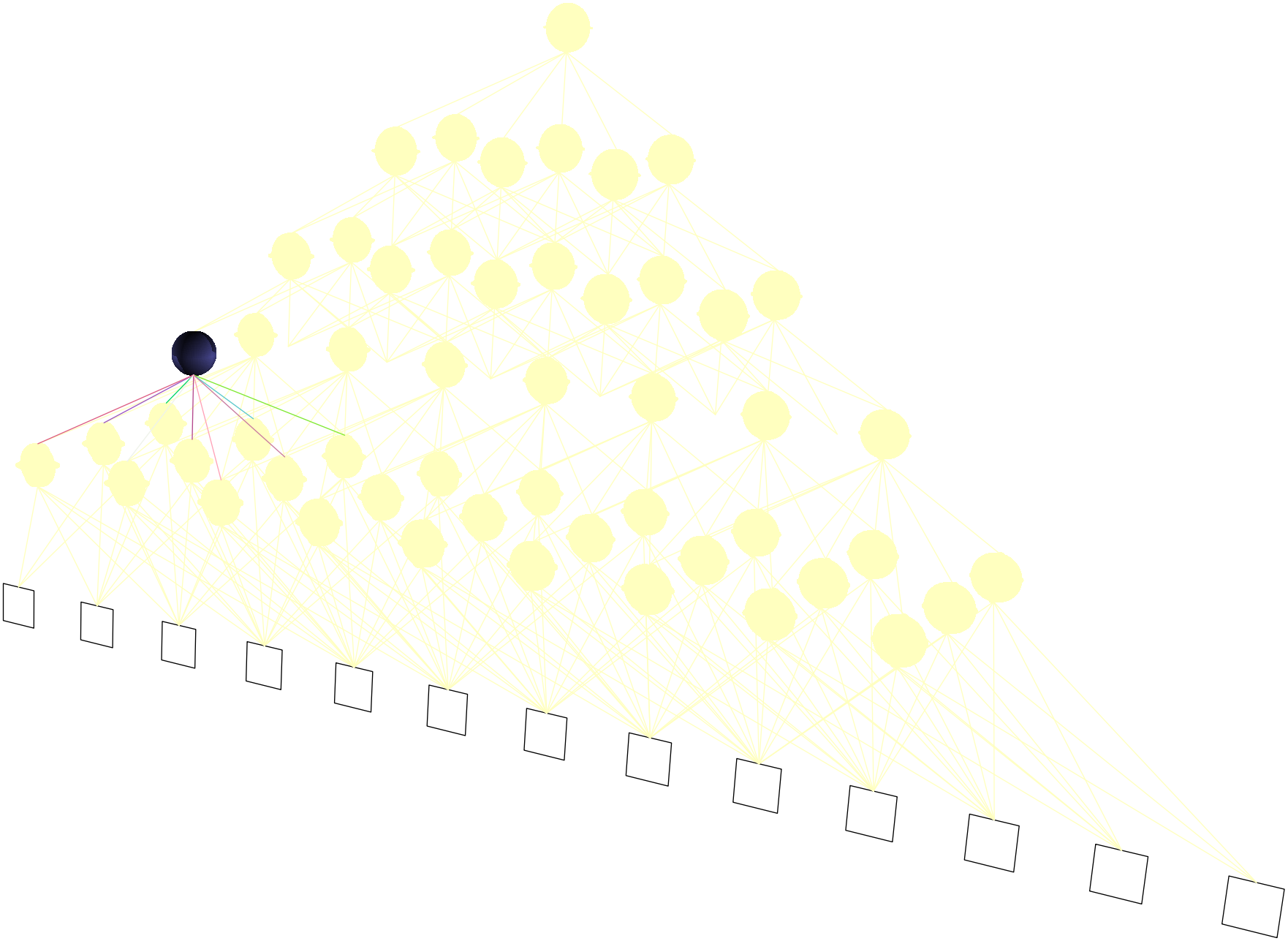


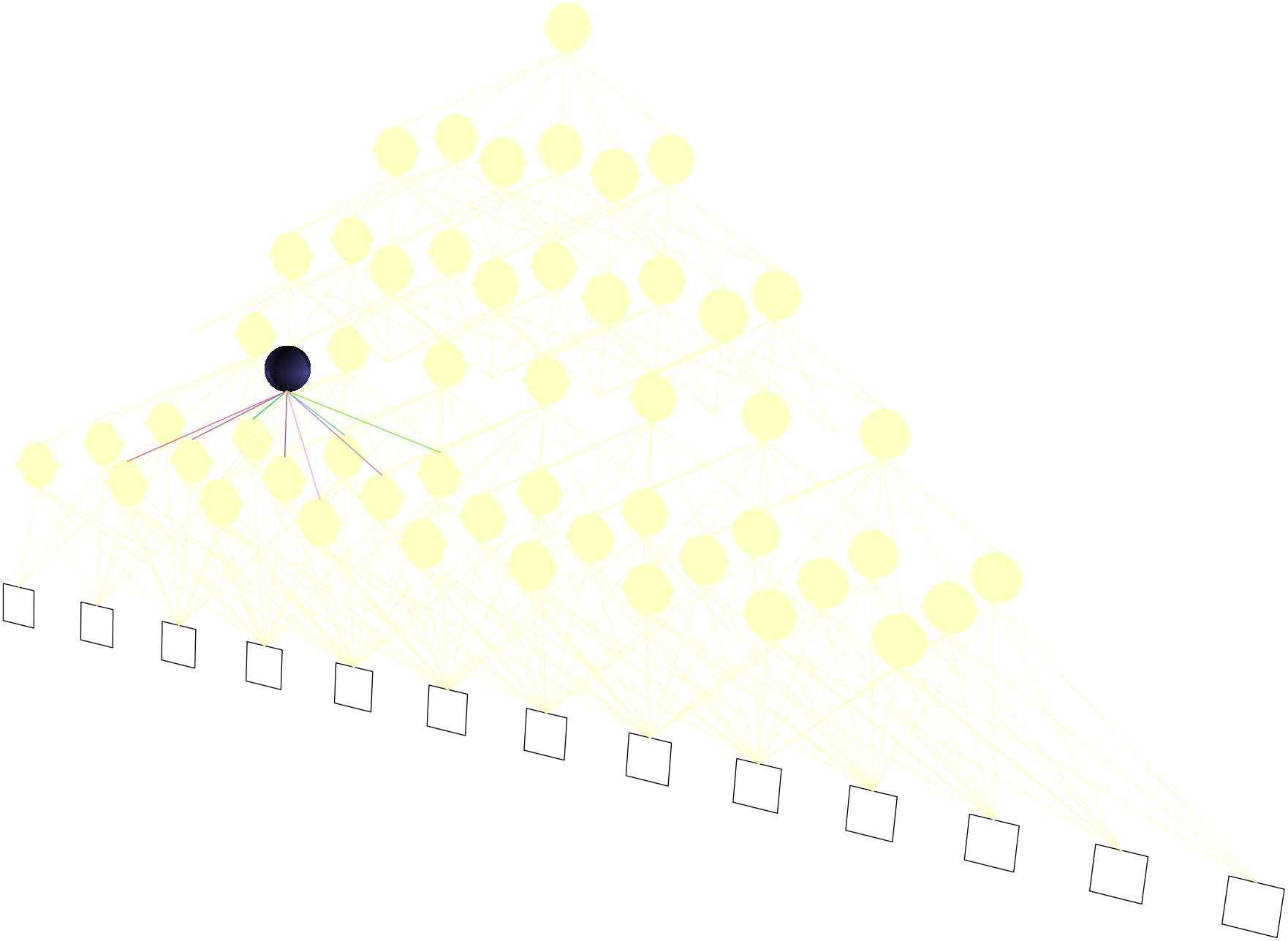
Red profunda convolutiva: 1 procesador libre por capa

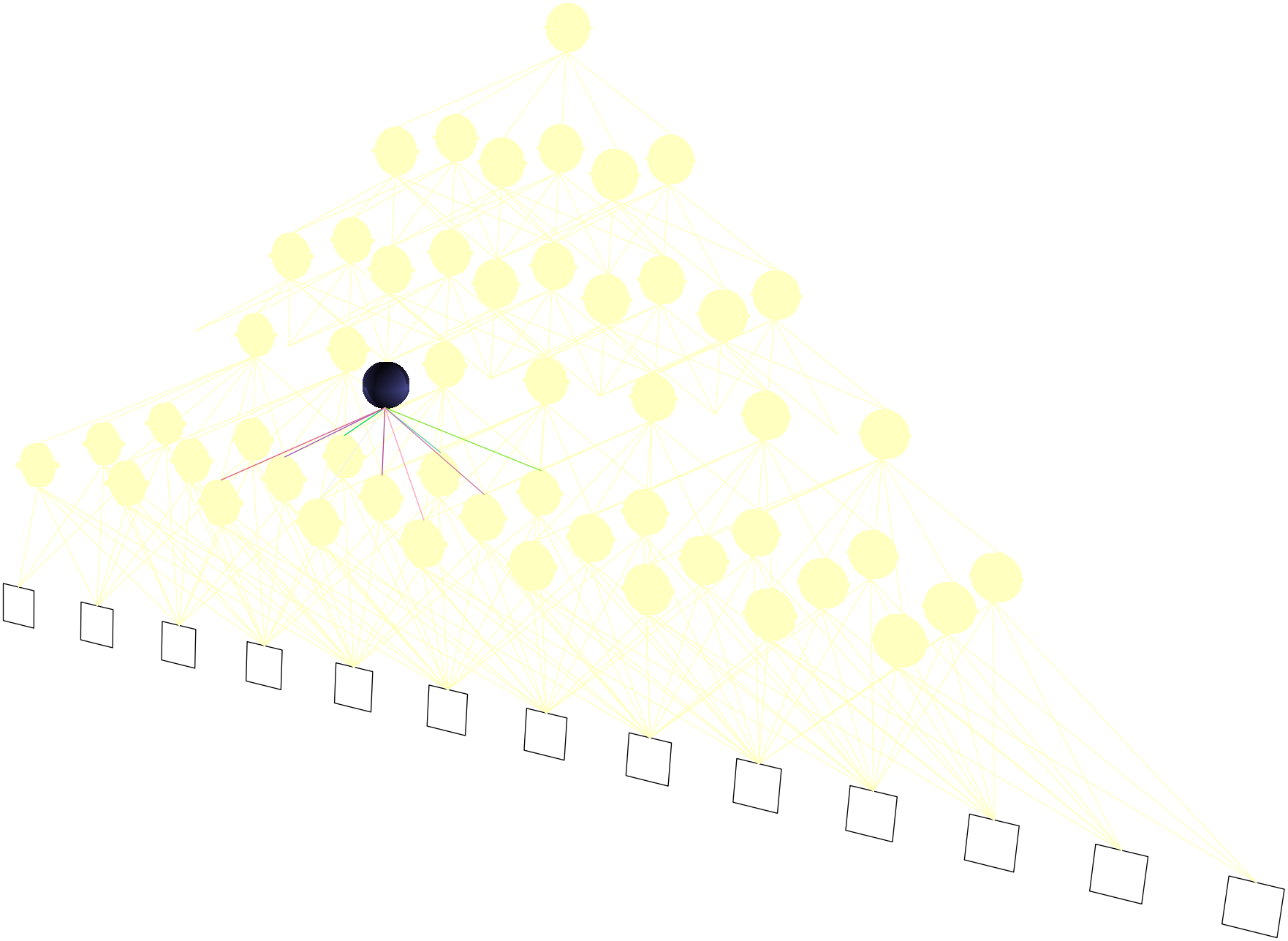
- ¿Cuántos pesos tiene?

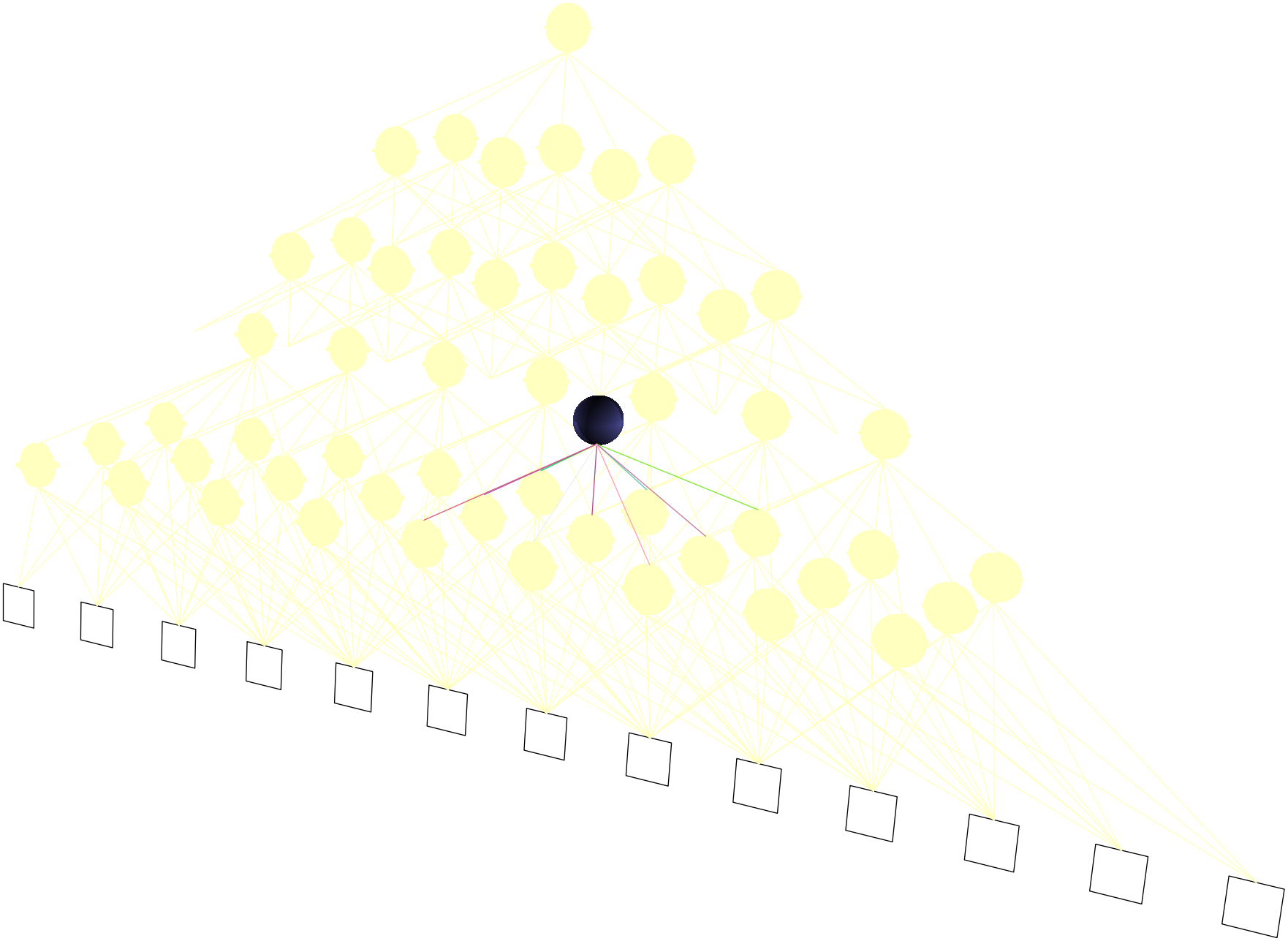


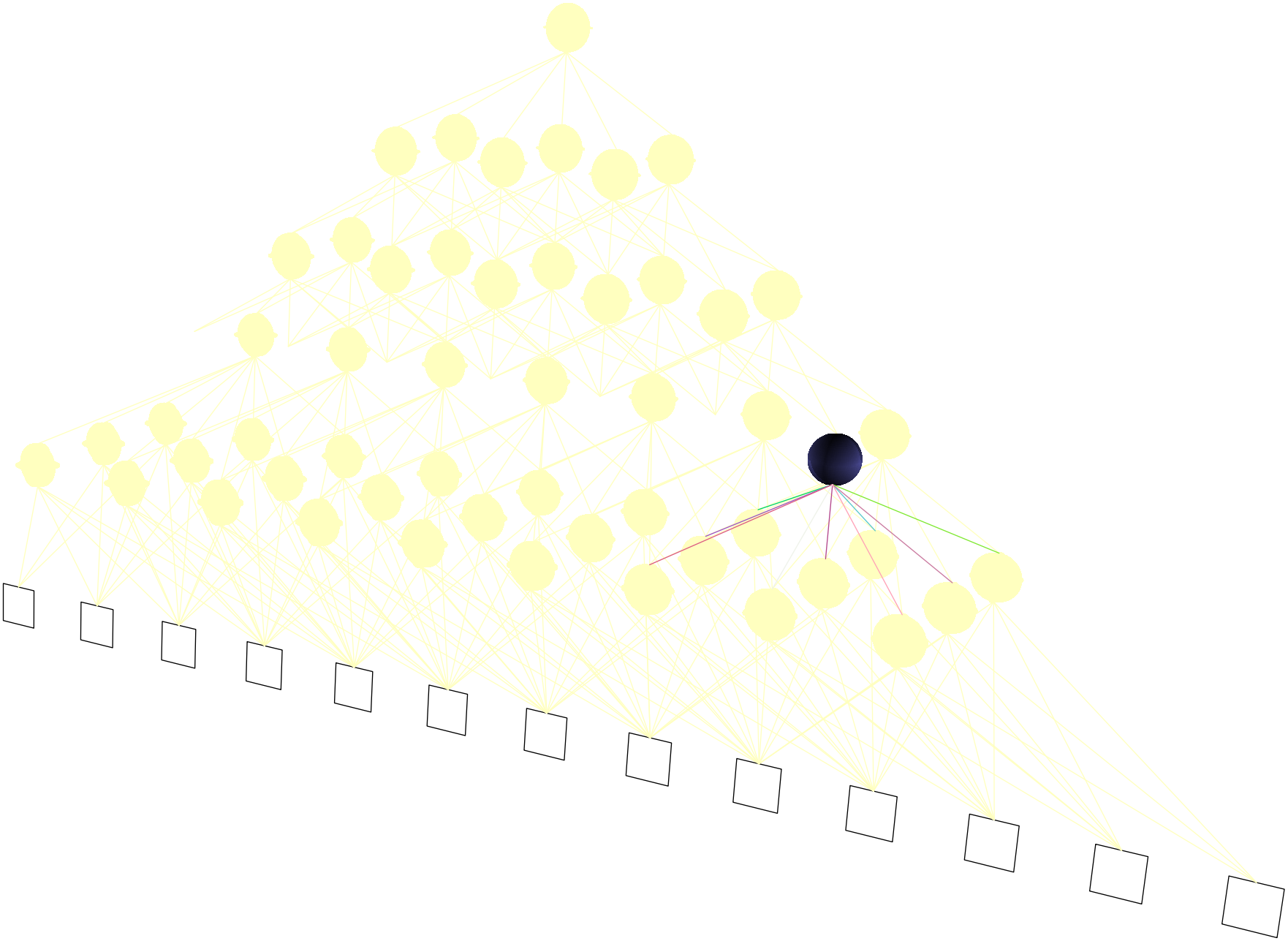


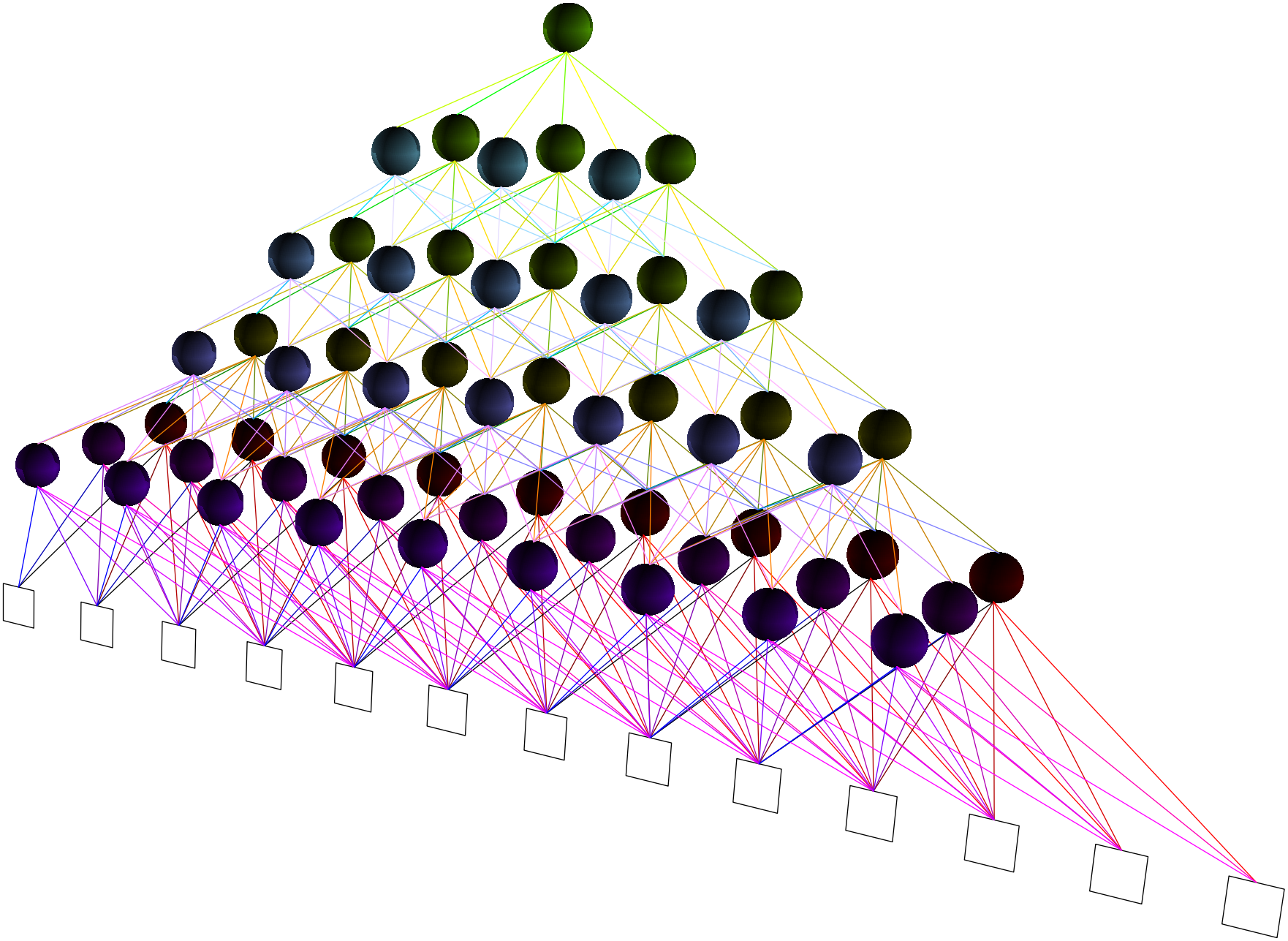












Red profunda convolutiva: varios procesadores libres por capa

- ¿Cuántos pesos tiene?

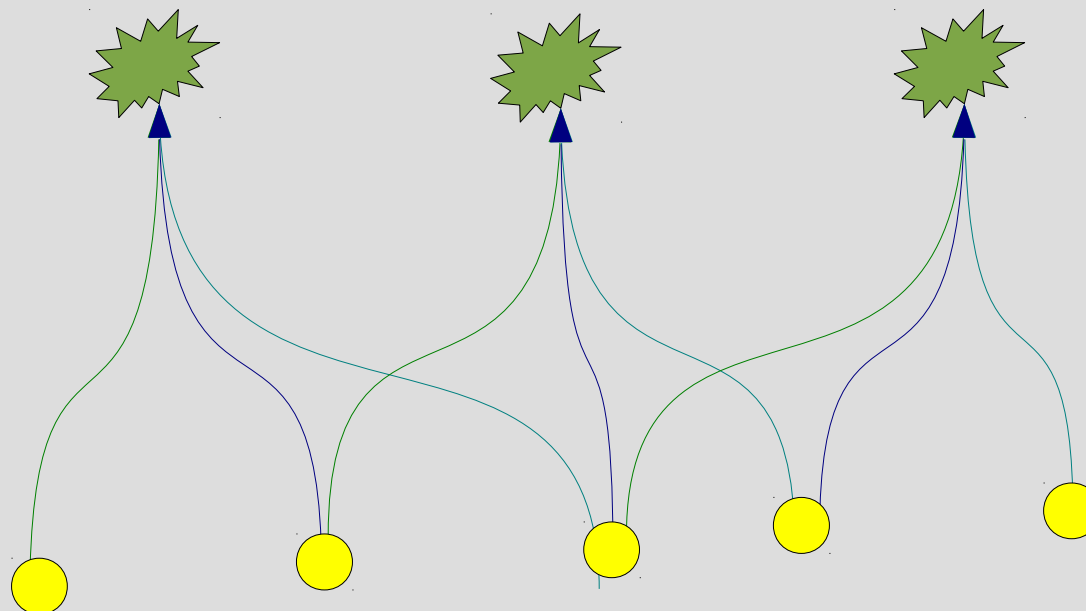
Paso, salto, ancho de convolución

- La capa hace un tipo de filtro de su entrada
- Ancho del filtro (kernel size)
- Paso del filtro (dilation)
- Salto entre filtros (stride)

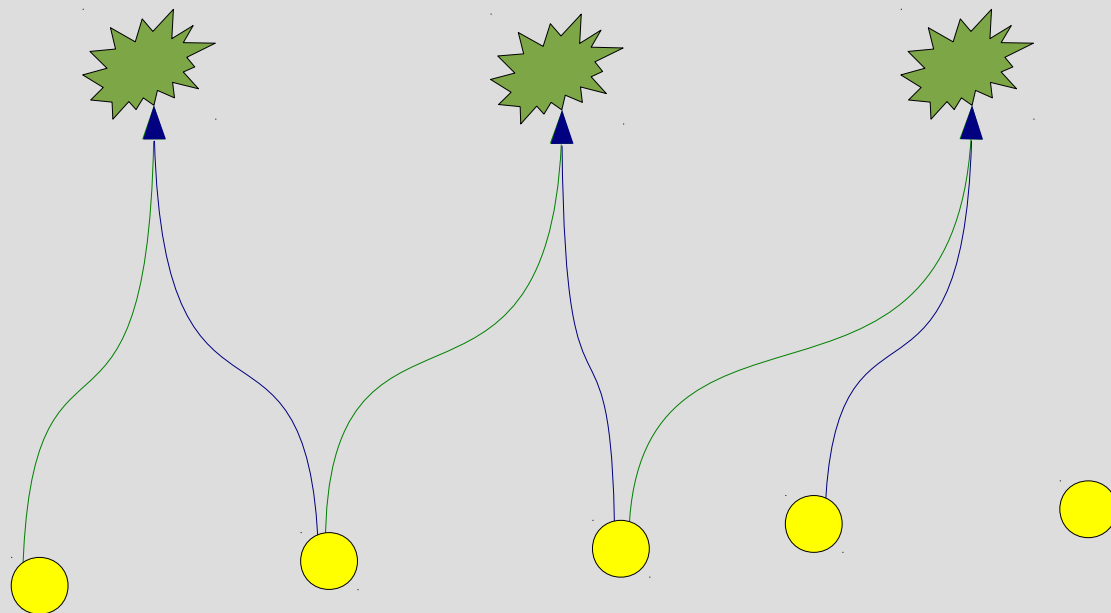
Paso, salto, ancho de convolución

- Plantea un caso de paso 2, salto 2 y ancho 4
- Si son 100 entradas, ése es un procesador libre de una capa y hay 4 en esa capa ¿Cuántos pesos tiene la capa?

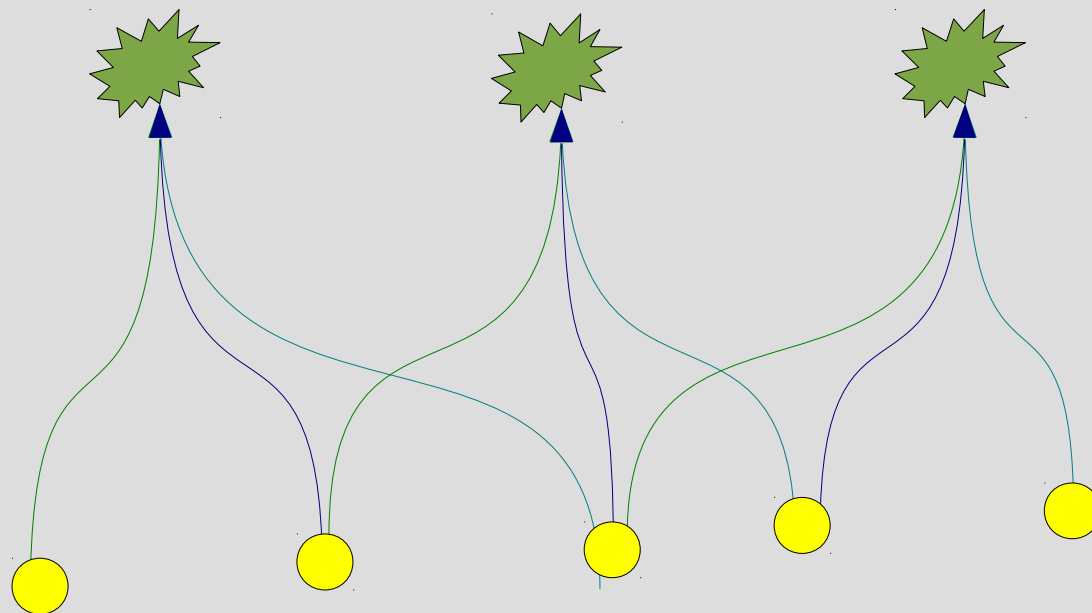
salto=paso=1, ancho=3



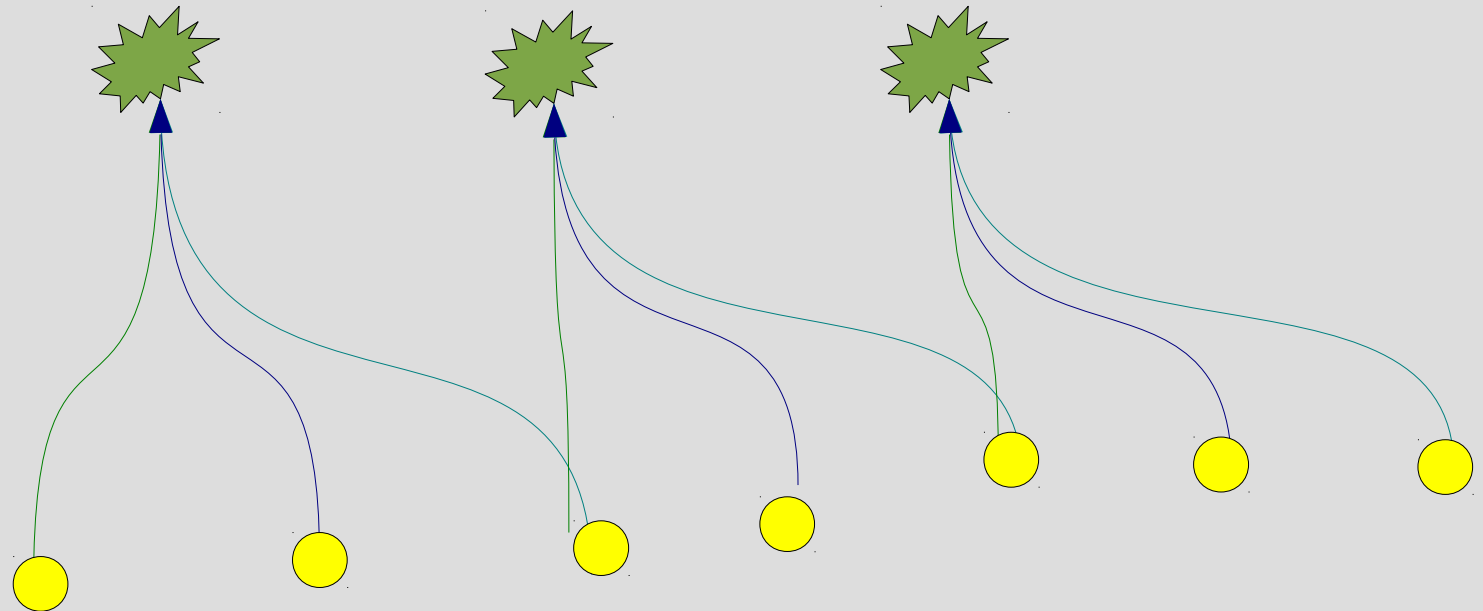
salto=paso=1, ancho=2



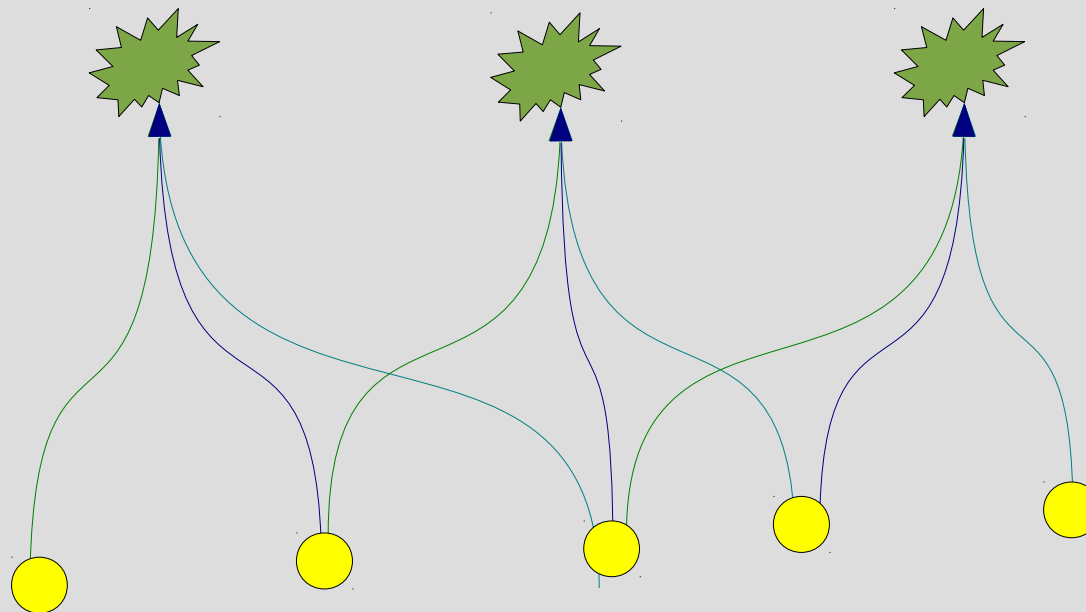
salto=paso=1, ancho=3



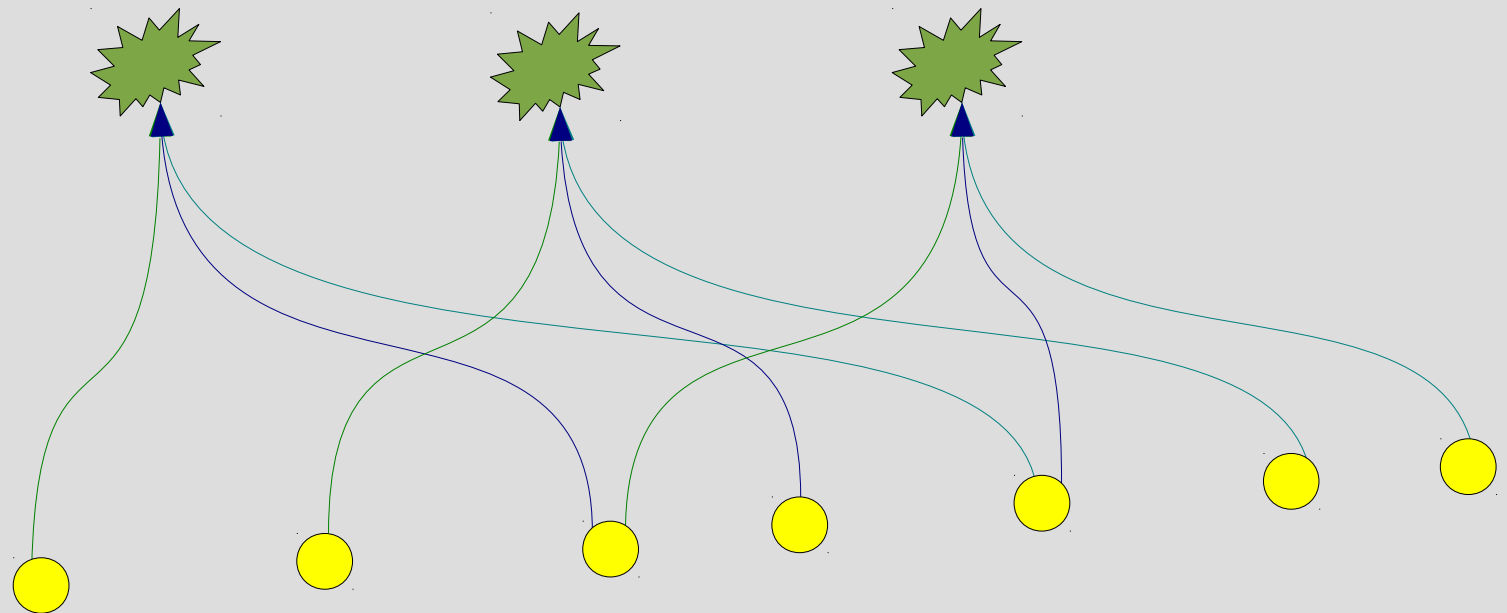
paso=1, ancho=3, salto=2



salto=paso=1, ancho=3



paso=2,salto=1,ancho=3



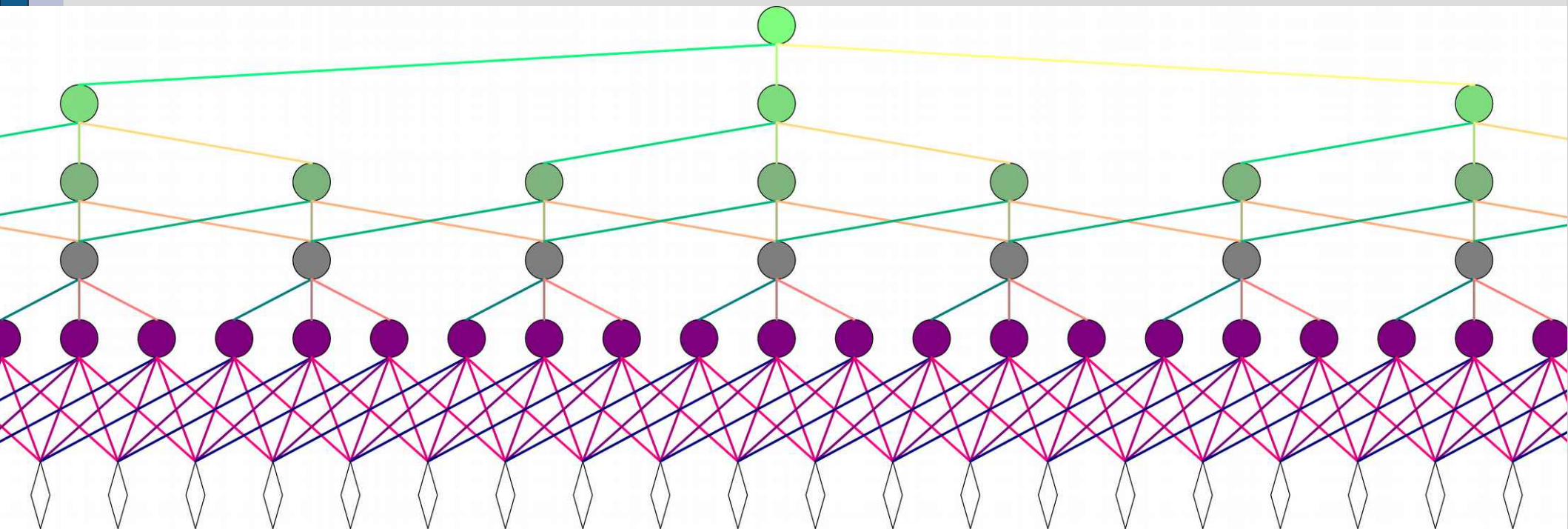
Salto de convolución

- Longitud de salida: $L_{sal} = \left\lfloor \frac{L_{ent} - \text{paso}(\text{ancho} - 1) - 1}{\text{salto}} + 1 \right\rfloor$
- salto=paso=1 $L_{sal} = L_{ent} - (\text{ancho} - 1)$
 - Disminución aritmética
- paso=1, salto=ancho $L_{sal} = \left\lfloor \frac{L_{ent}}{\text{ancho}} \right\rfloor$
 - Disminución geométrica
 - Común: capas **reductoras** alternando normales
 - Sin pesos; máximo, promedio de entradas

Salto de convolución

- Comprueba la salida de capa para el caso anterior
- Esquematiza los dos casos comentados, para ancho 4

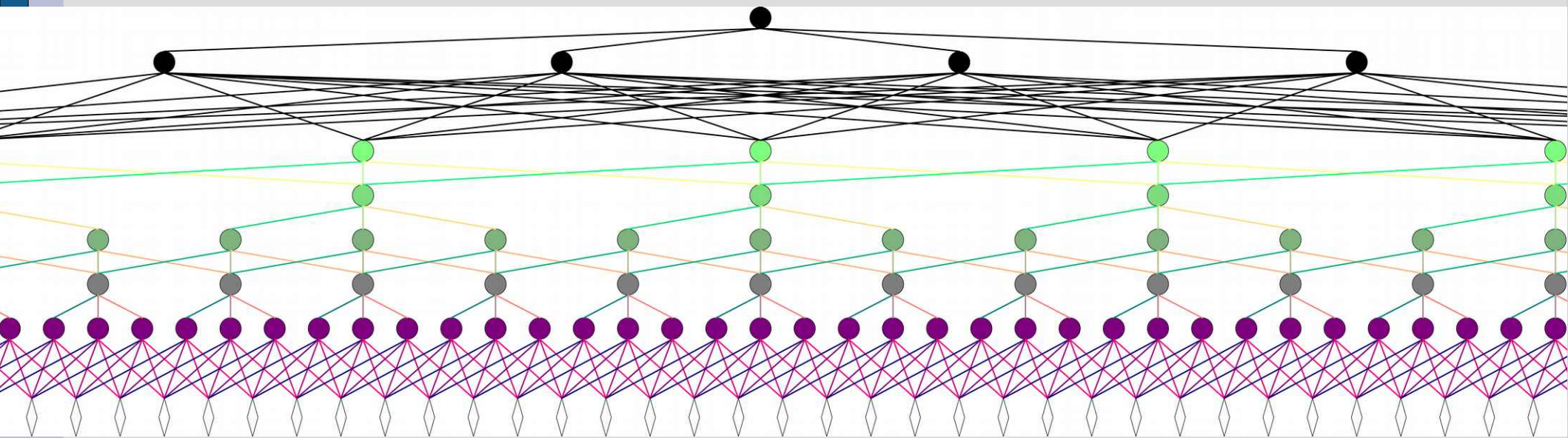
Red profunda alternada



Red profunda alternada

- Con los dos casos de capa previos y 100 entradas ¿en cuántas capas se nos queda en una salida?

Red profunda con final convencional

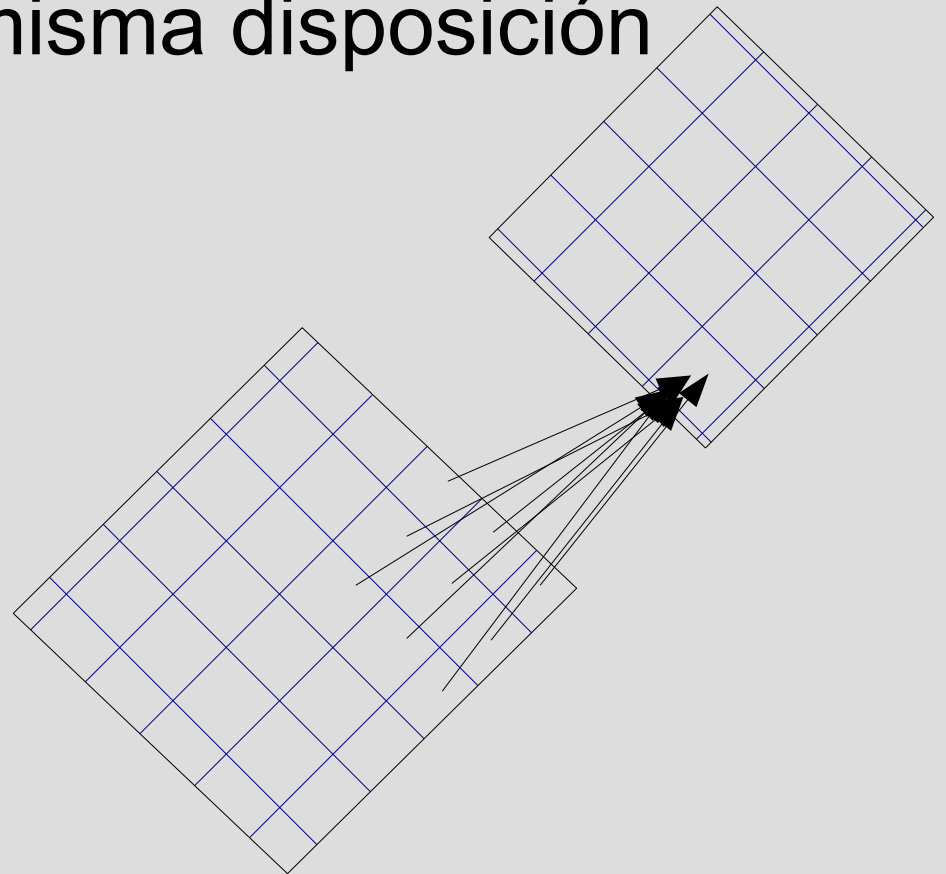


Red profunda con final convencional

- Mismo caso anterior, si pasamos a convencional cuando tengamos 10 variables ¿cuántas capas convolutivas habrá?

Otra forma de graficarlas: una capa, un procesador

- Caso imagen: entradas 2D, cada aplicación del procesador convolutivo de la primera capa se coloca en la misma disposición

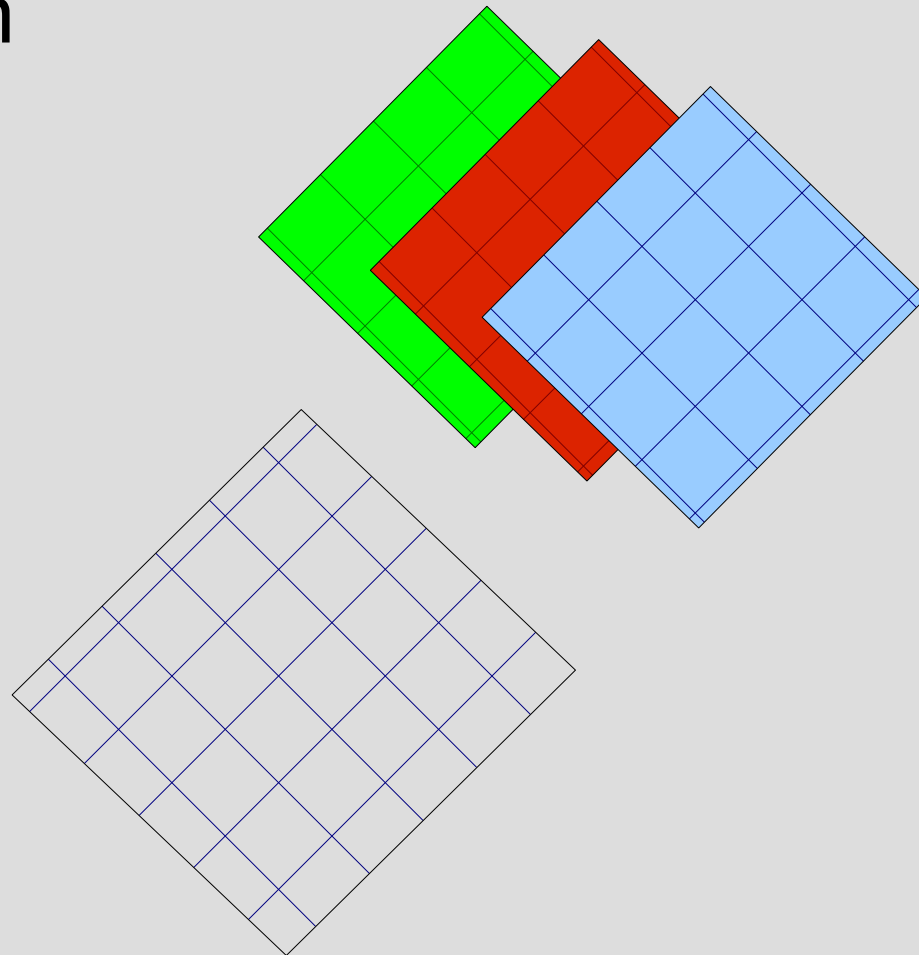


Otra forma de graficarlas: una capa, un procesador

- Si la entrada es 100×100 , salto=paso=1, ancho=4 ¿Qué tamaño tiene la salida?

Otra forma de graficarlas: una capa, varios procesadores

- Cada procesador (filtro) da una salida-imagen

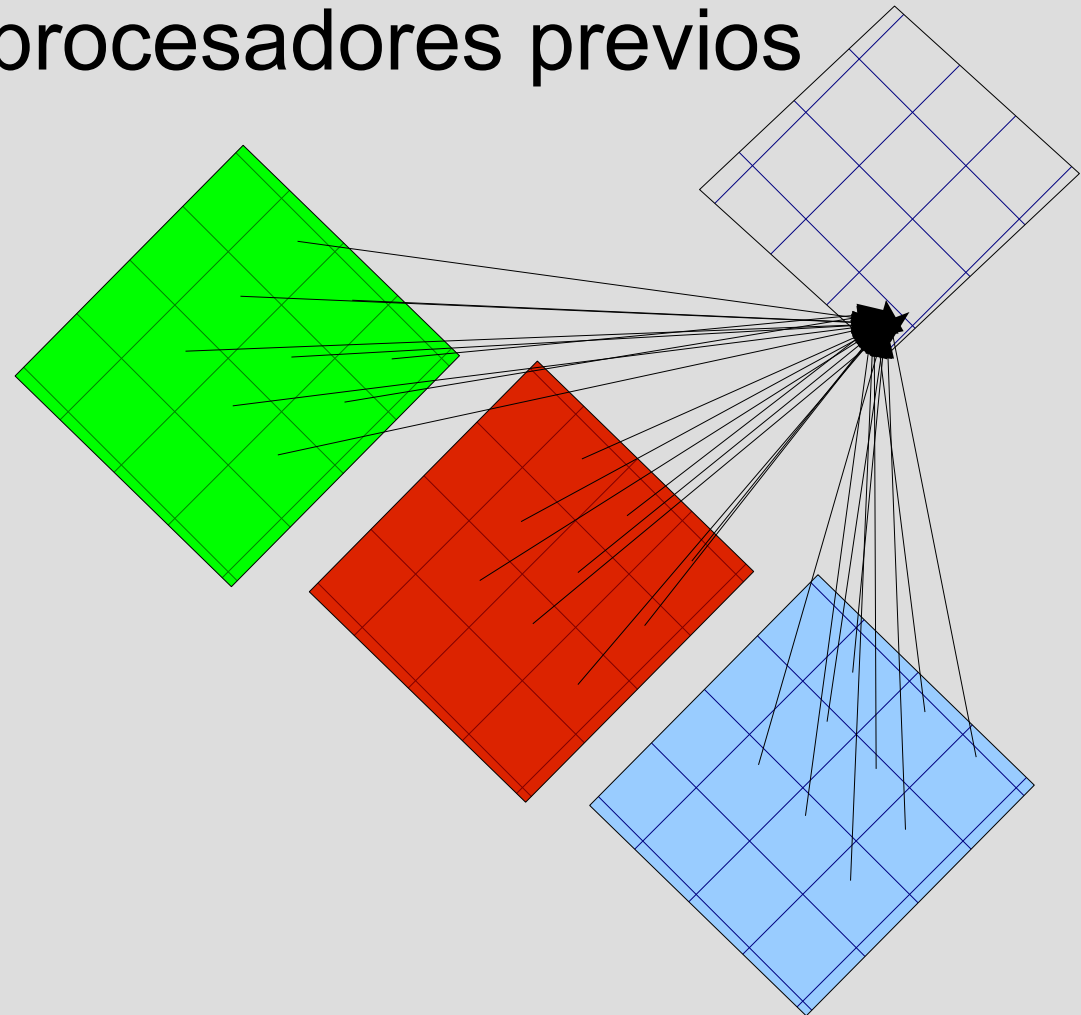


Otra forma de graficarlas: una capa, varios procesadores

- Si hay 5 procesadores en el caso anterior
¿cuántos pesos tiene esa capa?

Otra forma de graficarlas: perspectiva de la segunda capa, un procesador

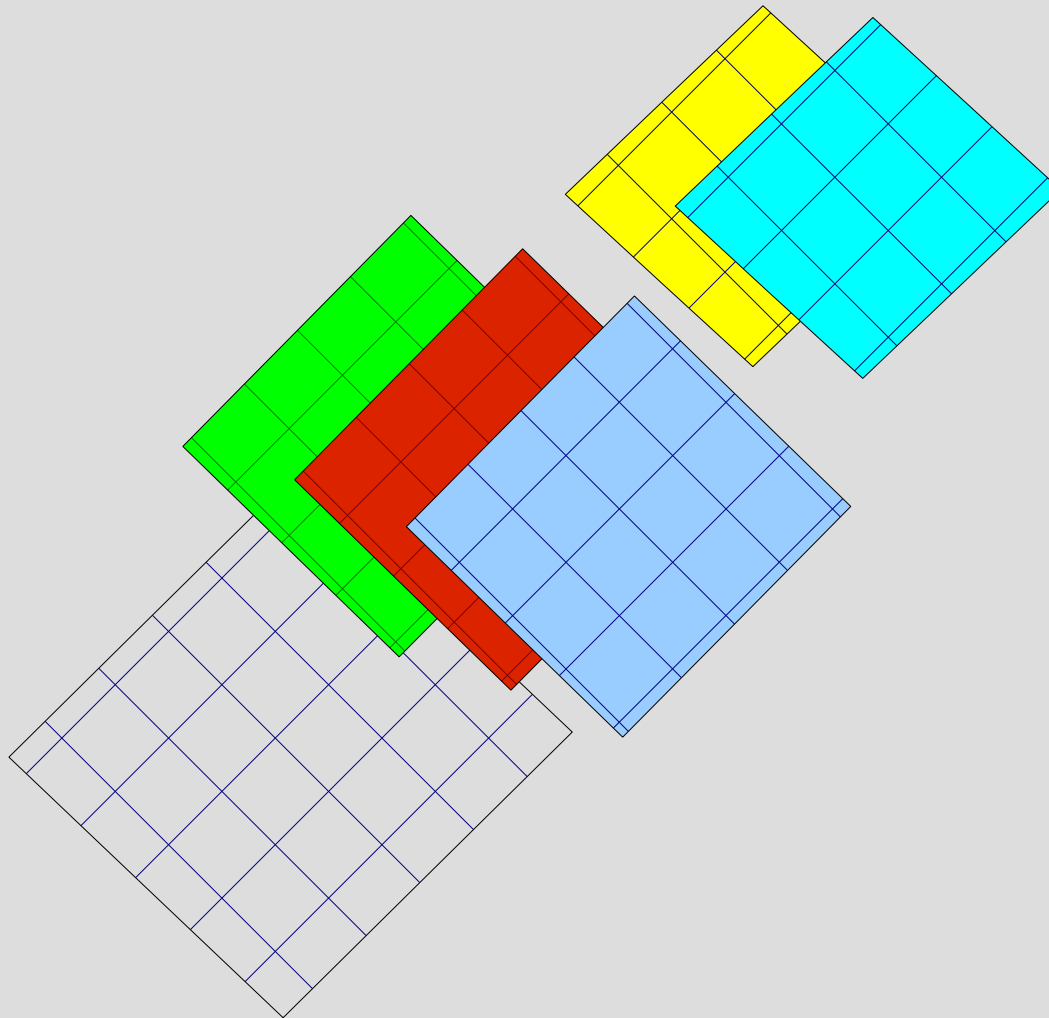
- El procesador hace la convolución sobre todos los filtros/procesadores previos



Otra forma de graficarlas: perspectiva de la segunda capa, un procesador

- Si repetimos en la segunda capa, salto=paso=1, ancho=4 ¿cuántos pesos tiene ese procesador? ¿qué tamaño tiene la salida?

Otra forma de graficarlas: varias capas, varios procesadores

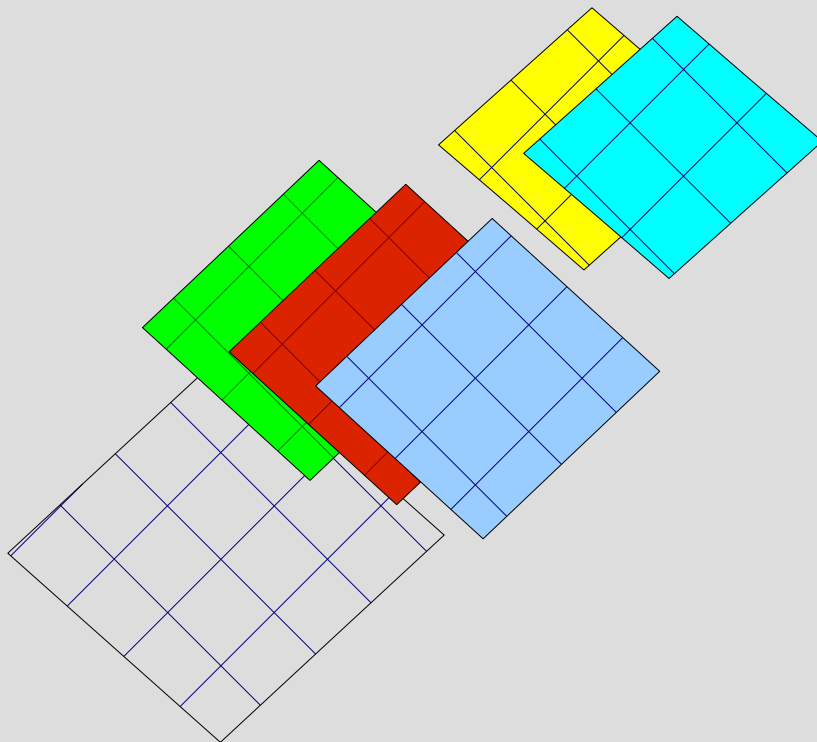


Otra forma de graficarlas, varias capas, varios procesadores

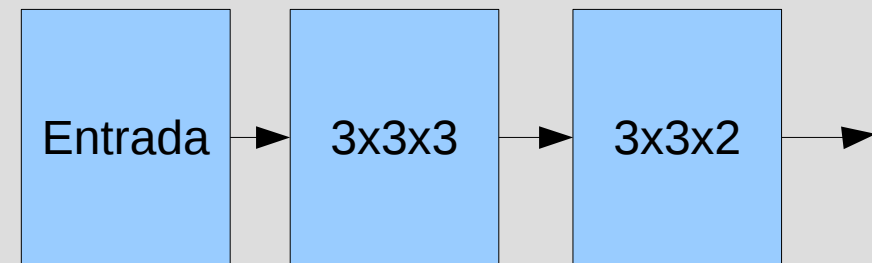
- Pongamos el caso anterior extendido a 5 capas, la primera con 5 procesadores y las siguientes bajando de uno en uno ¿Qué tamaño tiene la salida? ¿Cuántos pesos hay?

Notación compacta de tamaños

- En cada capa, indicar el tamaño de convolución y cuántas convoluciones (procesadores, filtros) se aplican

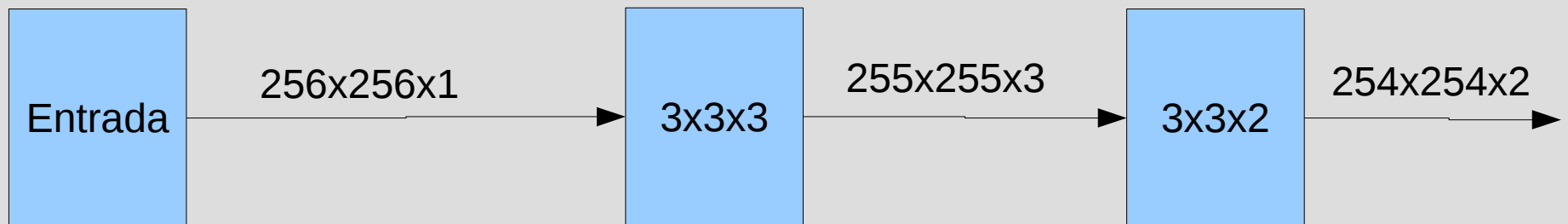


=



Notación compacta de tamaños

Podemos indicar los tamaños de salida

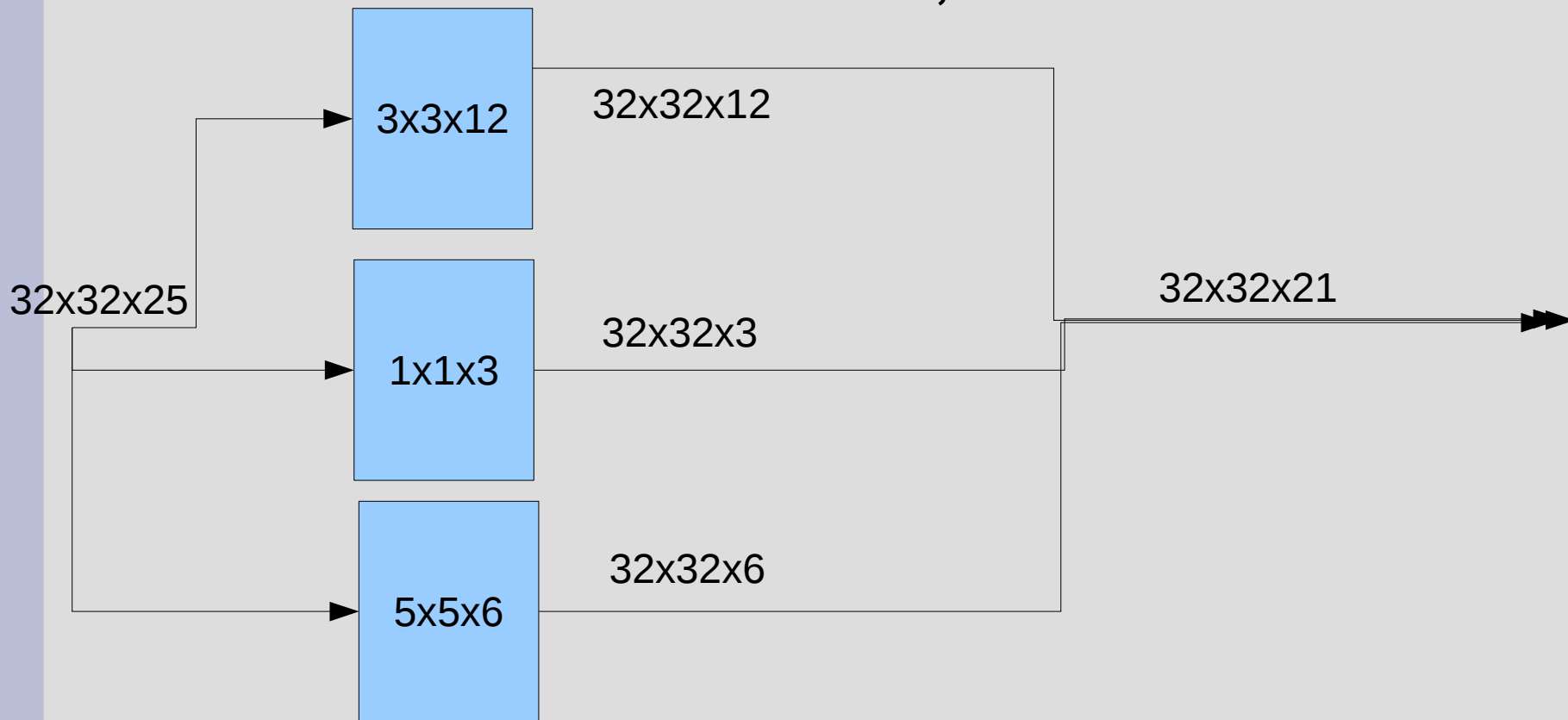


Notación compacta de tamaños

- Presenta el caso anterior
- Pon también los tamaños de salida

Variaciones

- Podemos aplicar en paralelo convoluciones de distintos tamaños (Inception) Para poder concatenar sus salidas, las rellenamos

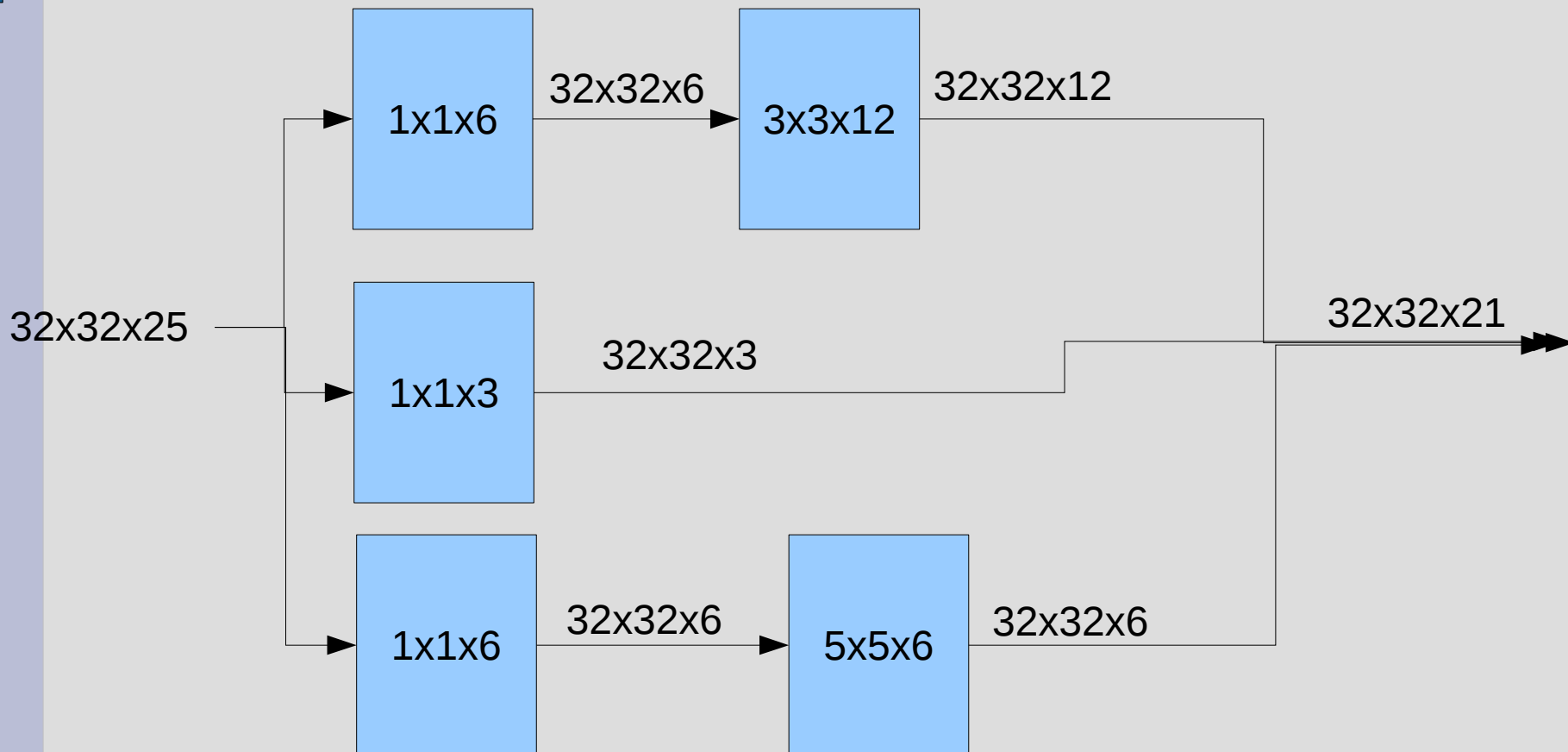


Variaciones

- ¿Cómo se pueden rellenar para que el tamaño de salida coincida?
- Volviendo al caso 100x100 Si la primera capa incluye un bloque de 6 procesadores de ancho 4 y 5 procesadores de ancho 3 ¿Qué tamaño tenemos en la salida?

Variaciones

- Podemos bajar las operaciones de algunos bloques poniéndolos antes una reducción

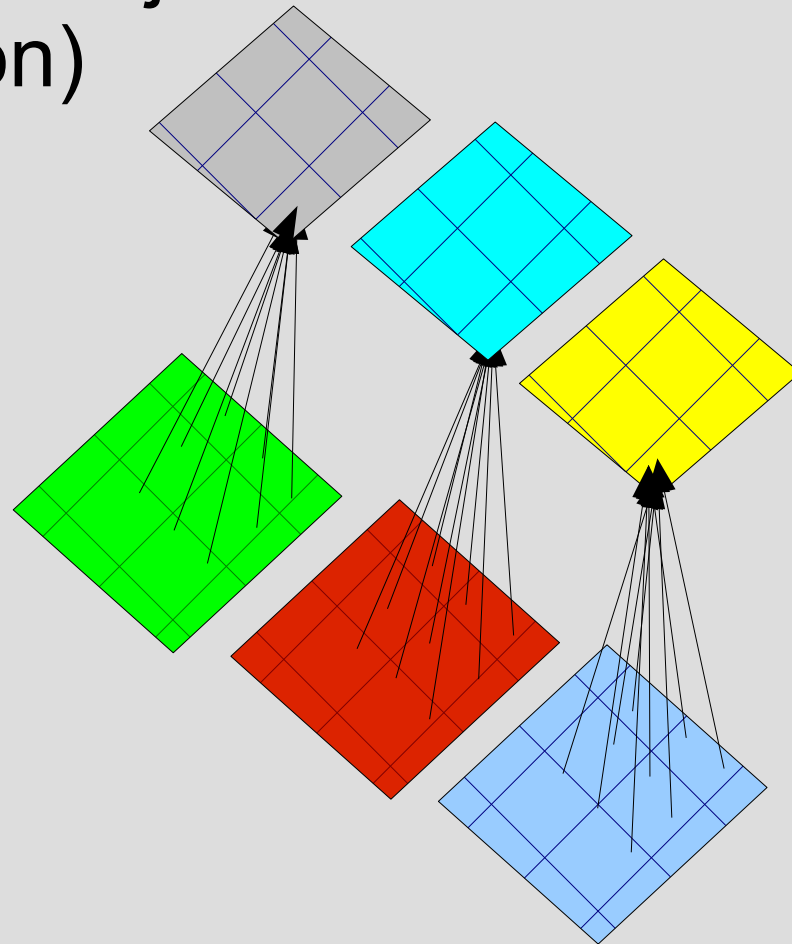


Variaciones

- Si en el caso anterior la segunda capa tiene la misma estructura que la primera, ¿cuántos pesos tiene?
- Esquematiza las conexiones de un bloque en segunda capa que fuese $1 \times 1 \times 3$
- Si anteponemos a ambos bloques unos de esos, ¿cuántos pesos va a tener la segunda capa en total?

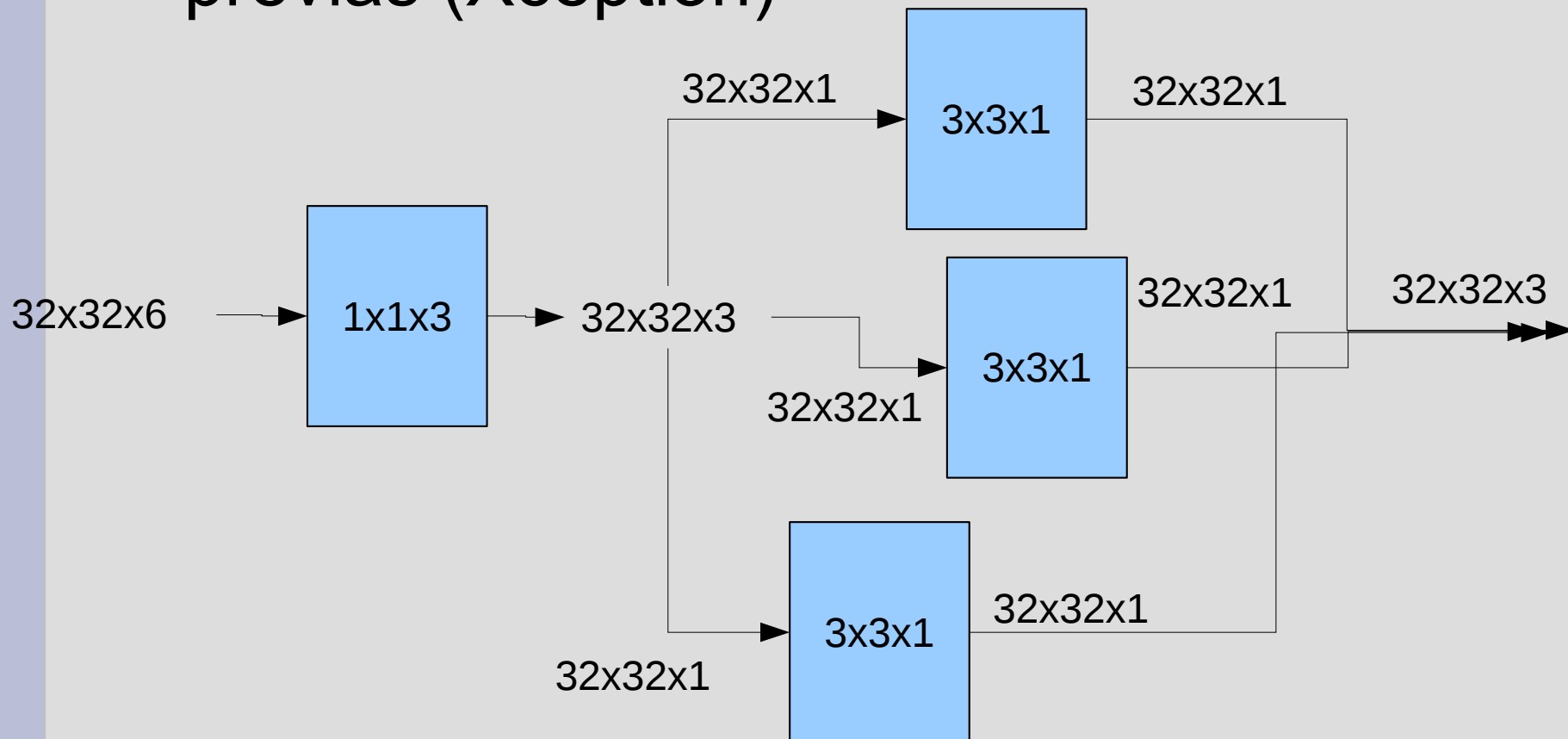
Variaciones

- Podemos hacer que cada convolución de una capa sólo trabaje con una de las salidas previas (Xception)



Variaciones

- Podemos hacer que cada convolución de una capa sólo trabaje con una de las salidas previas (Xception)



Variaciones

- Volvemos a rediseñar la segunda capa de nuestra red: ahora ponemos un bloque de $1 \times 1 \times 6$ y luego ancho 3 con cada salida previa. ¿Tamaño y cantidad de pesos?