

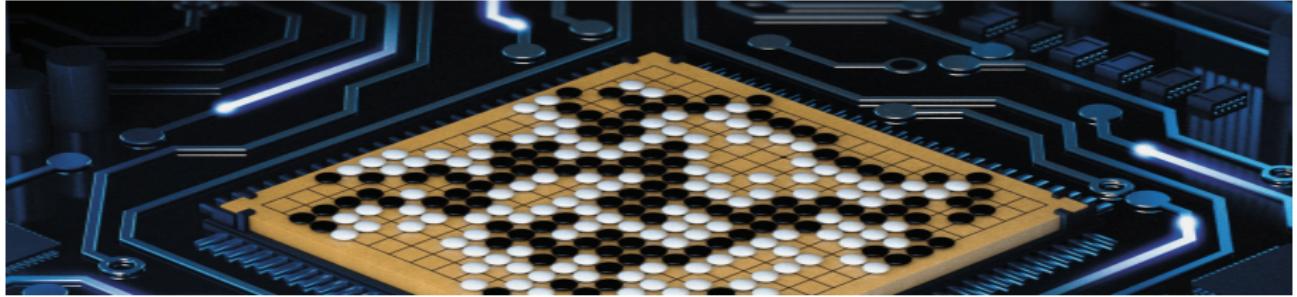
Alpha Zero

¿Cómo y porqué funciona?

Julio Waissman

Semana Nacional de Investigación y Docencia en Matemáticas
Universidad de Sonora

9 de marzo de 2018





Plan de la presentación

① ¿Qué es *Alpha Zero*?

② ¿Cómo funciona?

- Aprendizaje por refuerzo
- Árbol de búsqueda Monte Carlo

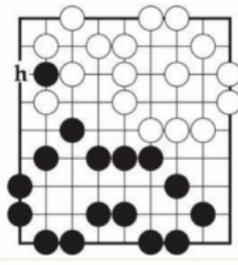
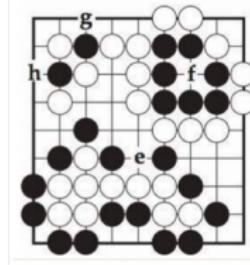
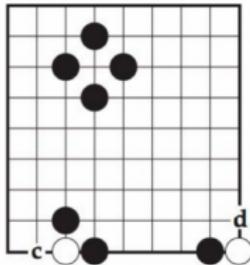
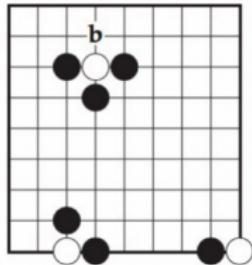
③ ¿Porqué funciona?

- Redes neuronales convolucionales
- *Tensorflow* y TPUs

④ Algunas consideraciones

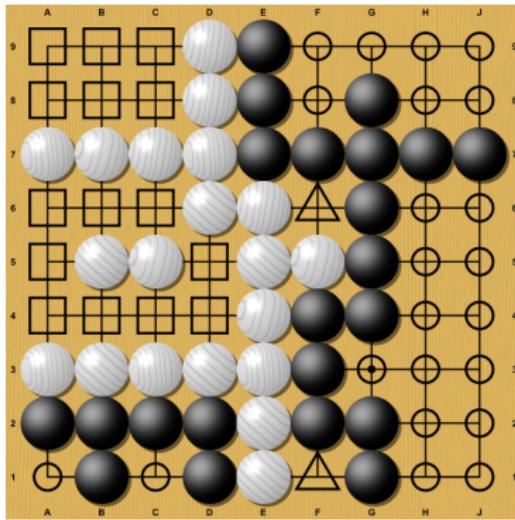
El juego de Go

- Comienza en un tablero vacío de 19×19
- Jugadores colocan *piedras* negras y blancas por turnos
- Si una parte conectada es rodeada por las *piedras* del oponente, se eliminan del tablero
- No se puede repetir posición en el tablero (regla de Ko)



El juego de Go

- La partida se termina cuando ambos jugadores pasan turno
- Se cuentan los espacios dominados y piezas capturadas
- Al jugador de las piedras blancas se le otorgan 7.5 puntos (Komi)

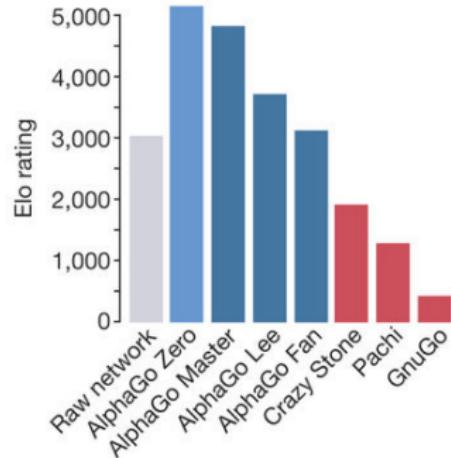
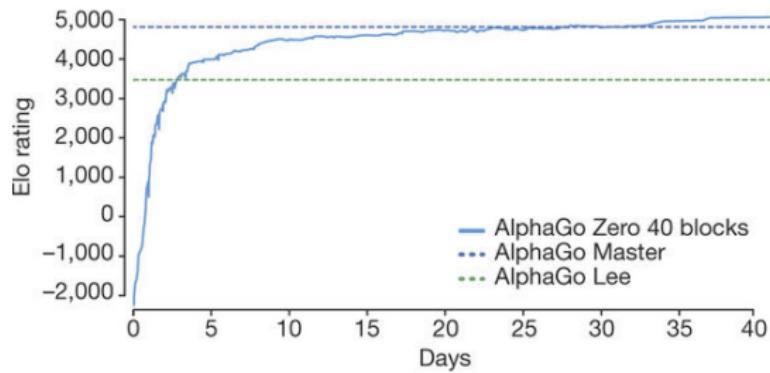


Go y jugadores computacionales

- 1997 John Tromp apuesta 1000 dólares que en 10 años no le ganará una computadora
- 2007 Tromp gana su apuesta
- 2015 *AlphaGo Fan* es el primer programa computacional en derrotar a un jugador profesional (Fan Hui)
- 2016 *AlphaGo Lee* es el primer programa computacional en derrotar al campeón mundial (Lee Sedol)
- 2017 *AlphaGo Master* gana 60 partidas contra profesionales, incluyendo 3 a el campeón mundial (Ke Jie)

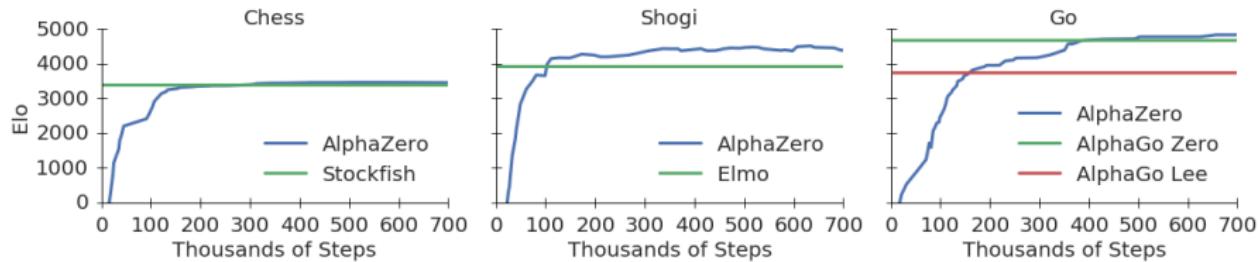
AlphaGo Zero

- Algoritmo que no usa conocimiento experto
- Octubre de 2017



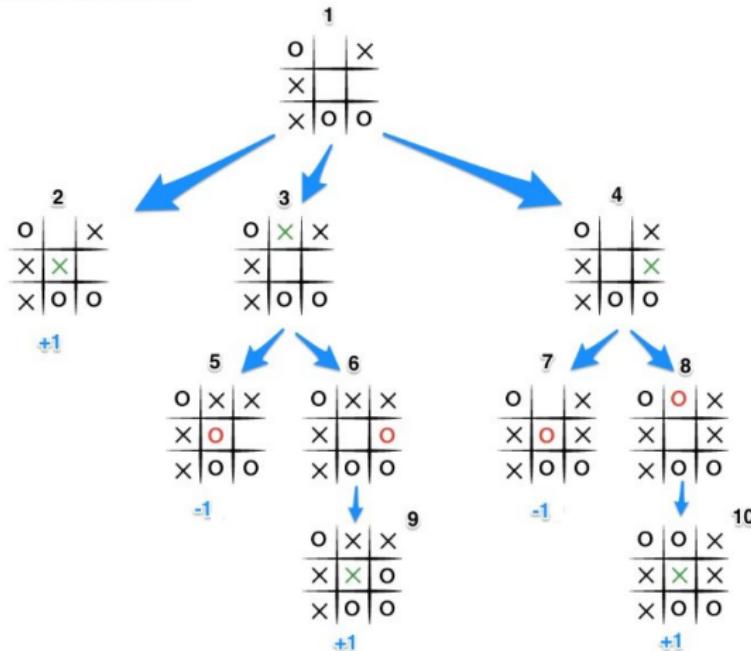
Alpha Zero

- Algoritmo con arquitectura genérica
- Diciembre de 2017



Minimax: La estrategia ganadora (hasta ahora...)

partial credits for this image to neverstopbuilding.com/minimax



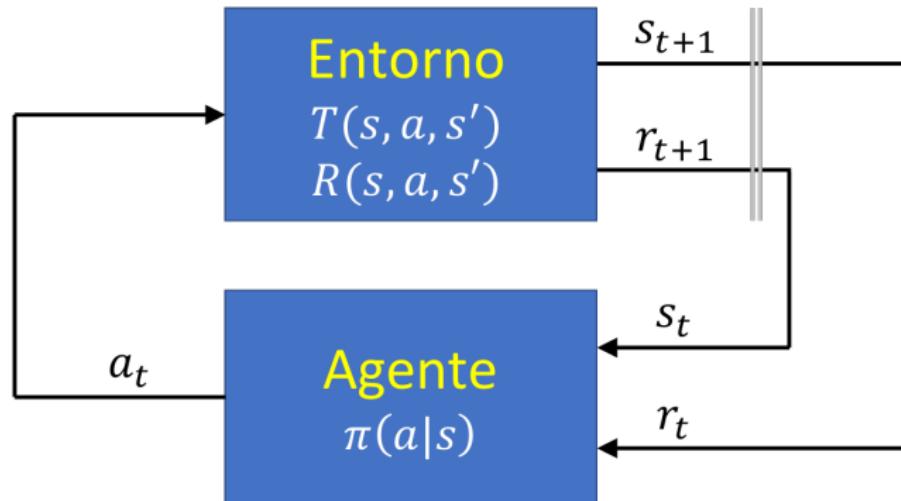
$$\max(1, \min(-1, \dots), \min(-1, \dots))$$

Características del minimax

- La poda $\alpha - \beta$ requiere conocimiento experto para ordenar las jugadas
- La evaluación de la utilidad requiere conocimiento experto en forma de características ($\phi(s)$)
- El conocimiento experto puede obtenerse a partir de bases de datos (por aprendizaje supervisado)
- Para cada problema diferente es necesario realizar muchas operaciones de optimización (i.e. estructuras de datos para la representación del estado).

¿Como funciona Alpha Zero?

Aprendizaje por refuerzo



Objetivo: Encontrar π que maximice la esperanza de $R_t = \sum_{i=0}^T \gamma^i r_{t+1+i}$

¿Cómo funciona Alpha Zero?

Iteración de política

Ajusta los parámetros θ de la función

$$(\mathbf{p}, v) = f_{\theta}(s),$$

- \mathbf{p} es el vector de probabilidad de seleccionar una acción a en el estado s , $p_a = \Pr(a|s)$,
- v es la probabilidad que el jugador actual gane el juego.
- Los parámetros θ se ajustan entre una política *a priori* (\mathbf{p}) y otra más fuerte (π), aproximada por MCTS.

¿Cómo funciona Alpha Zero?

Iteración de política

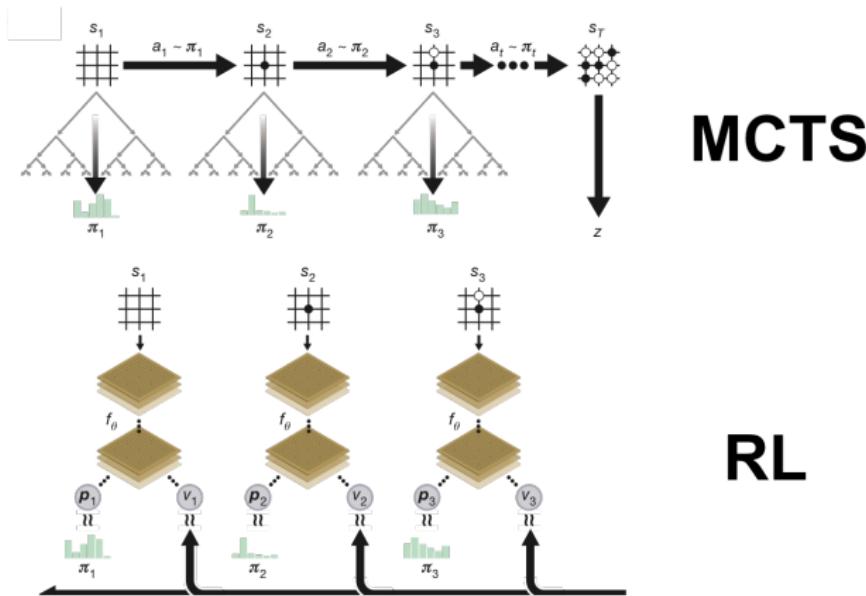
① Para cada iteración:

- ① Genera muchos juegos.
- ② En cada juego:
 - ① Para cada paso, encuentra la jugada a partir de una política generada por MCTS y f_θ .
 - ② Al terminar el juego se obtiene $r_T \in \{-1, 1\}$.
 - ③ Para cada paso se almacena $(s_k, \pi(s_k), z_k)$, donde $z_k = \pm r_T$, en una BdD de *replay*.
- ④ Toma una muestra amplia de la BdD de *replay*
- ⑤ Optimiza θ utilizando un método de descenso de gradiente donde:

$$LOSS(\theta) = (z - v)^2 - \pi^T \log \mathbf{p} + c\|\theta\|^2$$

¿Cómo funciona Alpha Zero?

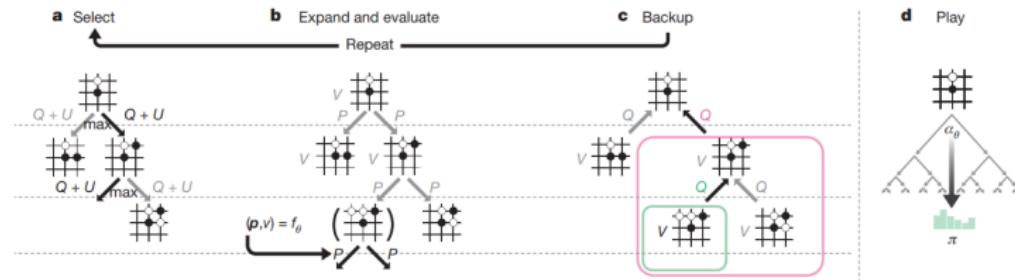
Iteración de política



Montecarlo tree search (MCTS)

Algoritmo general

- Se inicializa con un nodo s_0 posición inicial
- En cada arista (s, a) se almacena:
 - $Q(s, a)$, una función de valor estado/acción
 - $W(s, a)$, un peso
 - $N(s, a)$, un contador de veces que se a pasado por el nodo
 - $P(s, a)$, la política *a priori* (a partir de $f_\theta(s)$)



Montecarlo tree search (MCTS)

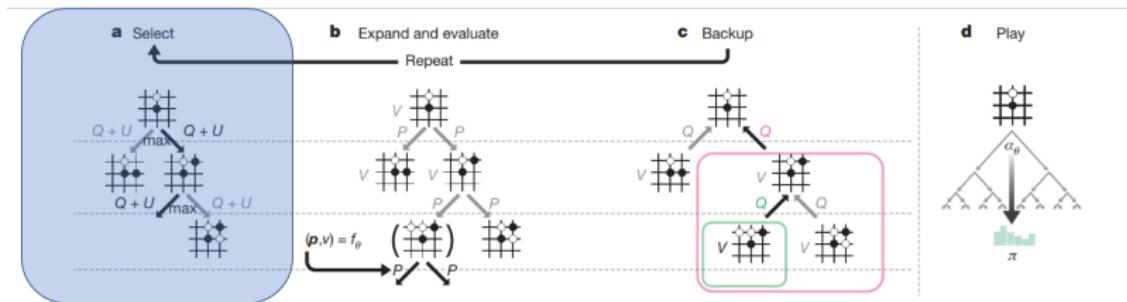
Selección

En cada nodo s_t se escoge la jugada tal que

$$a_t = \arg \max_a (Q(s_t, a) + U(s_t, a)),$$

$$U(s, a) = c_{puct} P(s, a) \frac{\sqrt{\sum_b N(s, b)}}{1 + N(s, a)}$$

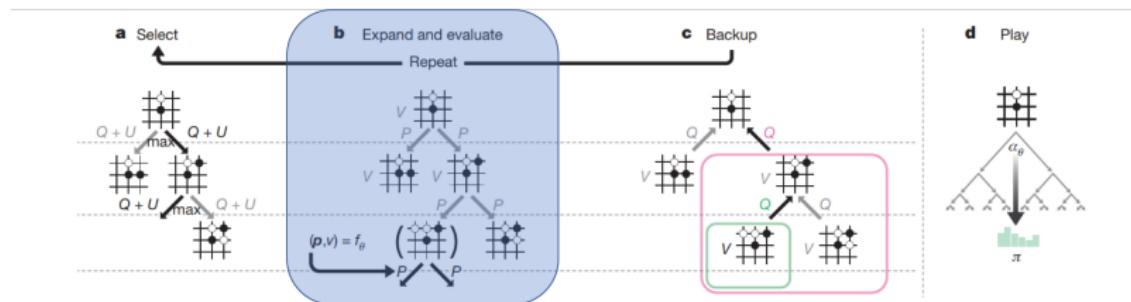
hasta encontrar un nodo sin expandir.



Montecarlo tree search (MCTS)

Expansión

- Evaluar $(p, v) = f_\theta(s_L)$
- Asignar a cada arista (jugada legal):
 - $N(s_L, a) = 0$,
 - $W(s_L, a) = 0$,
 - $Q(s_L, a) = 0$,
 - $P(s_L, a) = p_a$

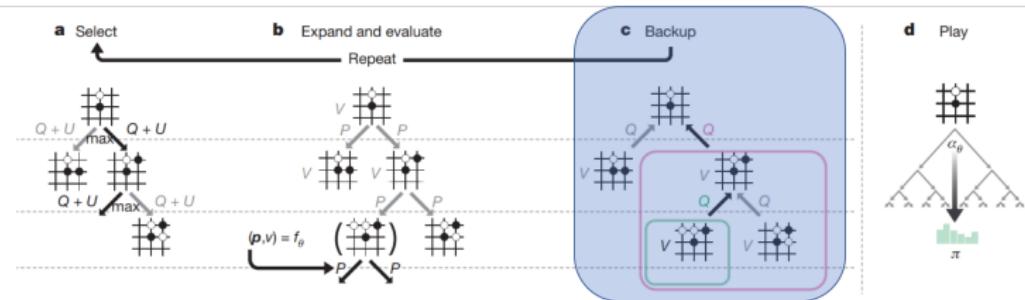


Montecarlo tree search (MCTS)

Respaldo

- Para cada vértice en la trayectoria:

- $N(s, a) = N(s, a) + 1$,
- $W(s_L, a) = W(s, a) + v$,
- $Q(s_L, a) = \frac{W(s, a)}{N(s, a)}$



Montecarlo tree search (MCTS)

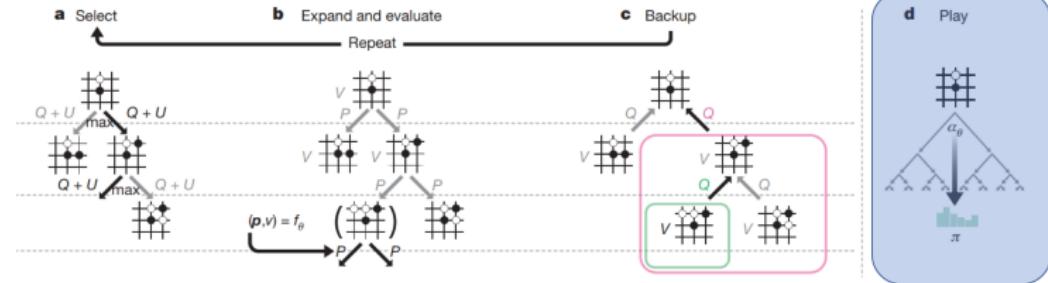
Jugar

- Política ávida:

$$a^* = \arg \max_a N(s_0, a)$$

- Política de exploración:

$$\pi(a|s_0) = \frac{N(s_0, a)}{\sum_b N(s_0, b)}$$

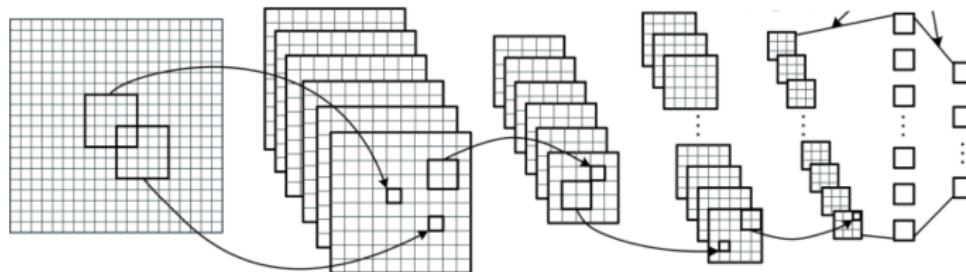


¿Pero cuántos juegos realiza?

En Alpha Zero se utilizó para todos los casos

- Por cada iteración: 25,000 juegos
- Para cada movimiento (de cada juego): 1600 simulaciones de MCTS
- Para los primeros 30 movimientos de cada juego se utiliza la política de exploración.

¿Porqué funciona Alpha Zero?



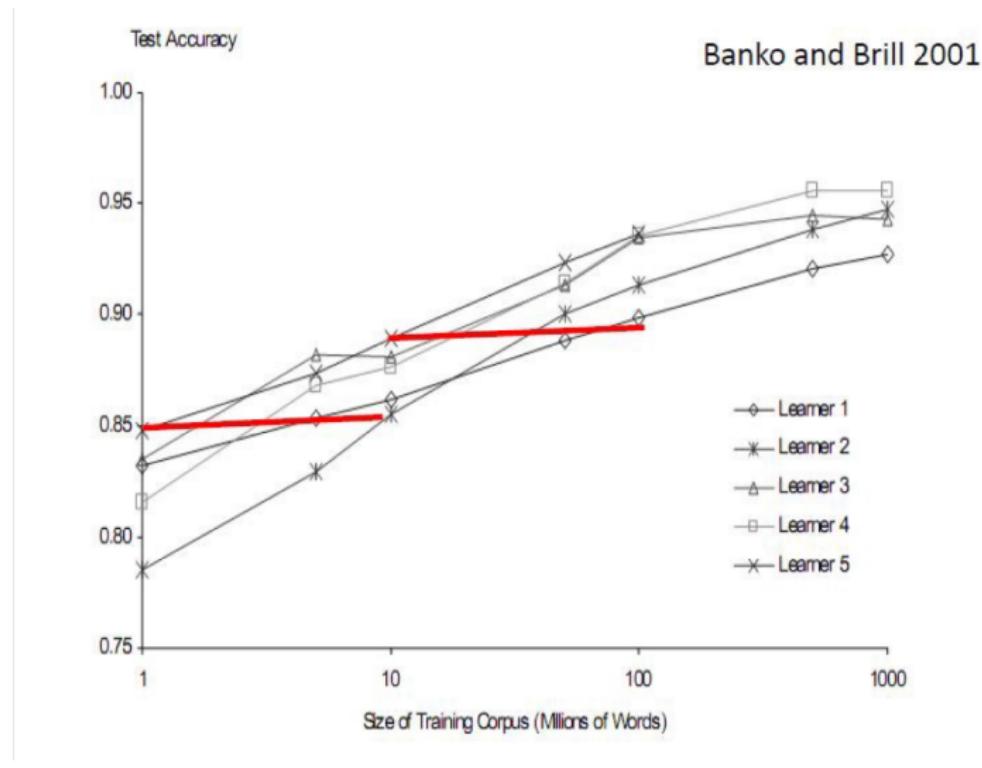
La receta secreta

$f_\theta(s)$ se modela con una red neuronal profunda

- Redes neuronales convolucionales (CNN)
- Tensorflow
- *Tensor processing units* TPUs

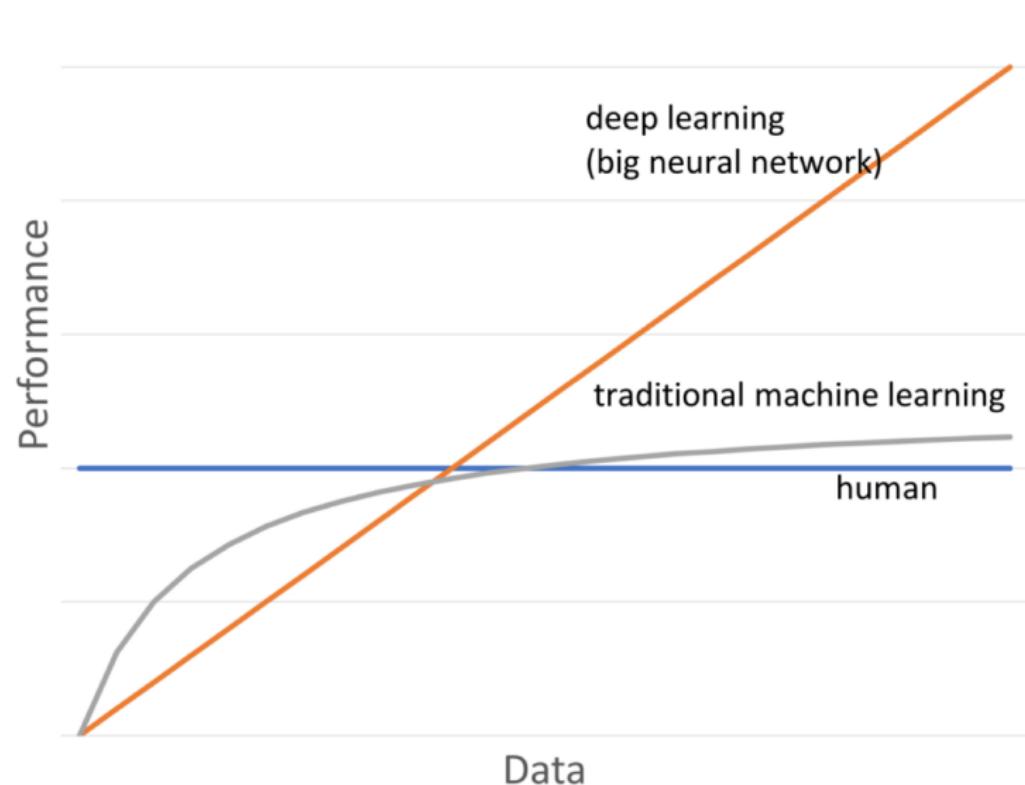
Aprendizaje profundo (DL)

Motivación



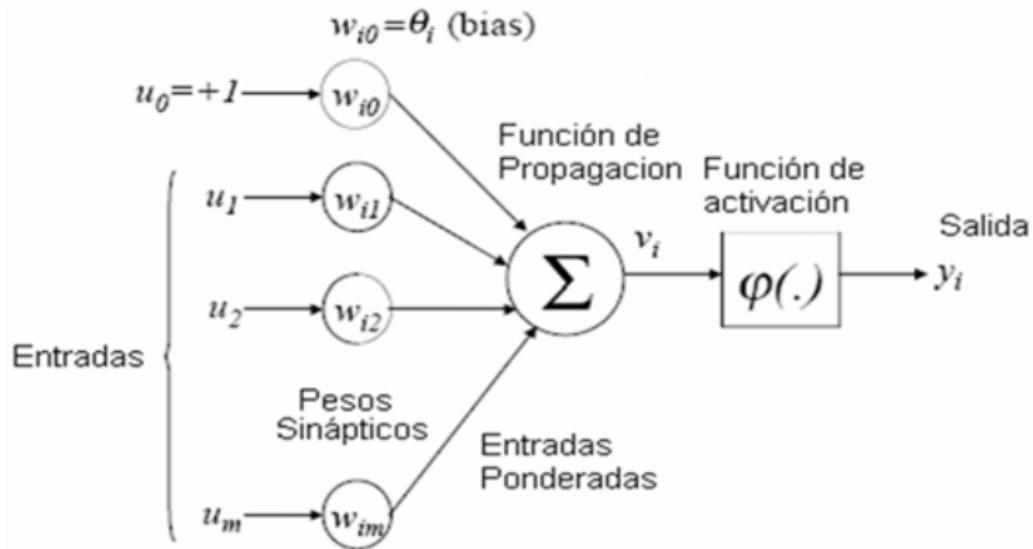
Aprendizaje profundo (DL)

Motivación



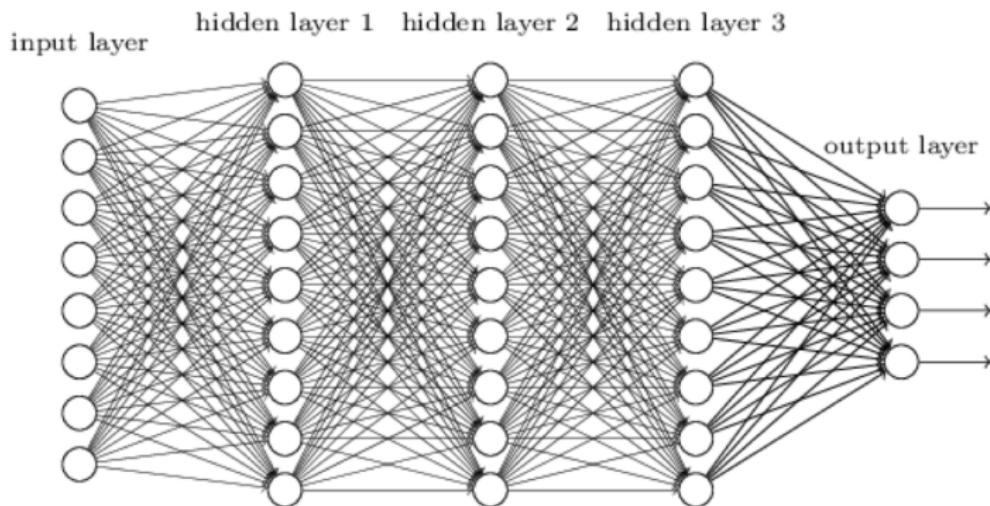
Arquitecturas hacia adelante

Unidad o Neurona



Arquitecturas hacia adelante

Estructura general



Redes convolucionales (CNN)

Información local

Capa de partida

0	1	1	1	0	0	0	0
0	0	1	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	1	1	0	0	0
0	0	1	1	0	0	0	0
0	1	1	0	0	0	0	0
1	1	0	0	0	0	0	0

Capa convolucionada

1	4	3	4	1
1	2	4	3	3
1	2	3	4	1
1	3	3	1	1
3	3	1	1	0

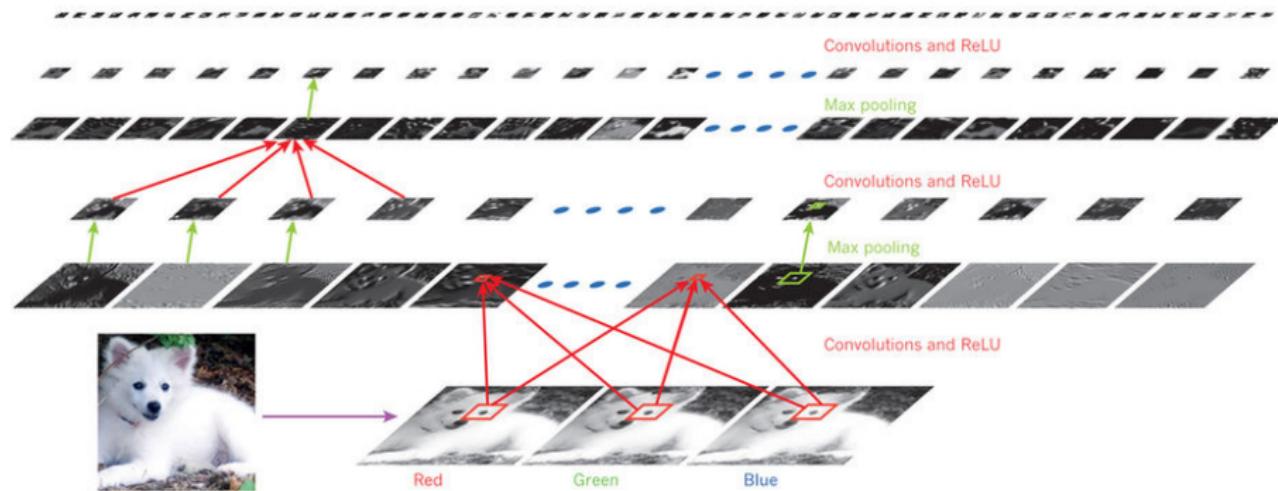
Filtro utilizado

$$\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{matrix} *$$

Redes convolucionales (CNN)

Arquitectura general

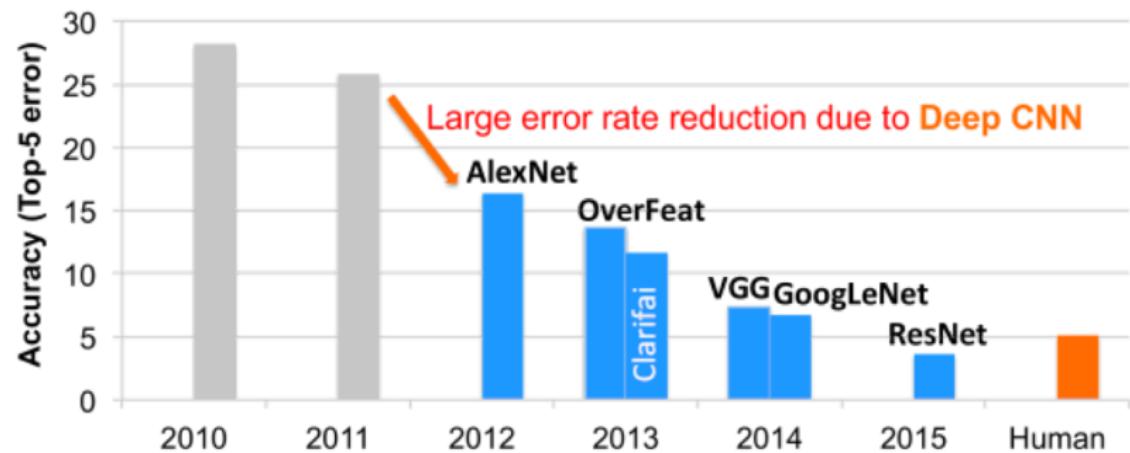
Samoyed (16); Papillon (5.7); Pomeranian (2.7); Arctic fox (1.0); Eskimo dog (0.6); white wolf (0.4); Siberian husky (0.4)



Redes convolucionales (CNN)

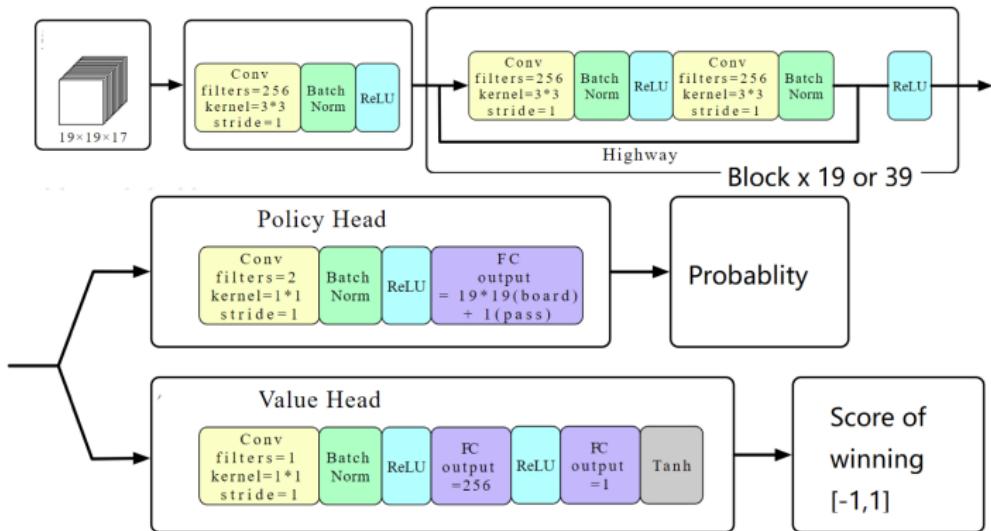
Desempeño en tratamiento de imágenes

Resultados aplicados al conjunto de datos *ImageNet*



Redes convolucionales (CNN)

Arquitectura de Alpha Zero



Redes convolucionales (CNN)

Representación

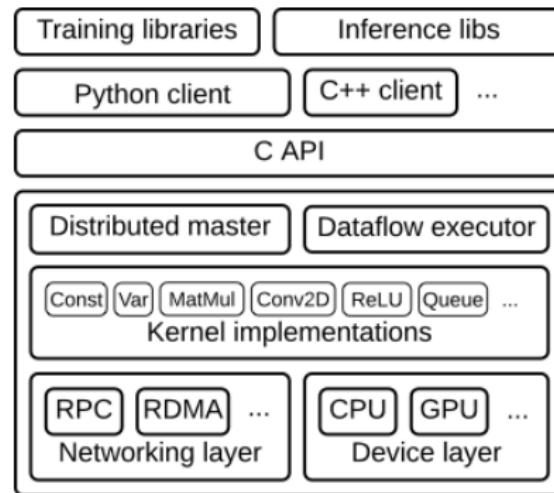
Feature	Go Planes	Chess	Shogi Planes
P1 stone	1	P1 piece	6
P2 stone	1	P2 piece	6
		Repetitions	2
Colour	1	P1 piece	14
		P2 piece	14
		Repetitions	3
		P1 prisoner count	7
		P2 prisoner count	7
Total	17	Total	119
		Total	362



- Desarrollado por *Google* para el proyecto *Google Brain*
- Biblioteca para cómputo numérico basado en graficas de flujo de datos
- Los Vértices representan operaciones y las aristas tensores
- Se adapta a diferentes plataformas

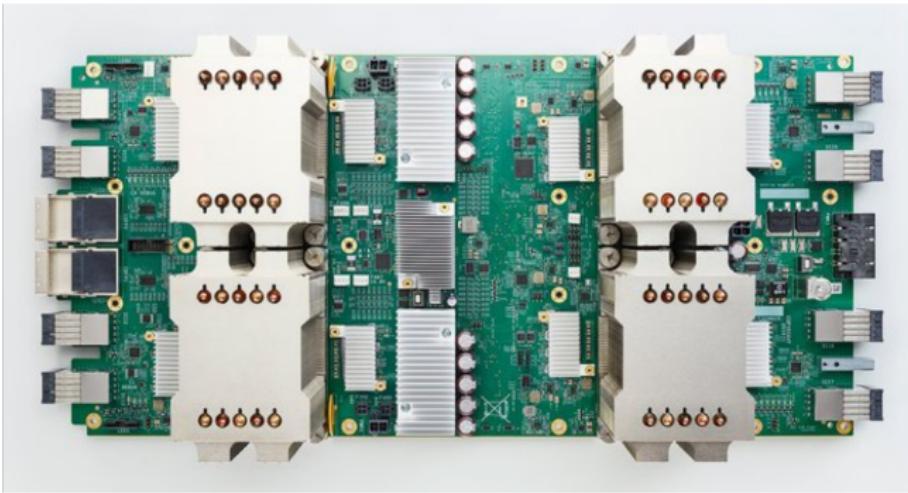
Tensorflow

Arquitectura



Tensor Processing Unit (TPU)

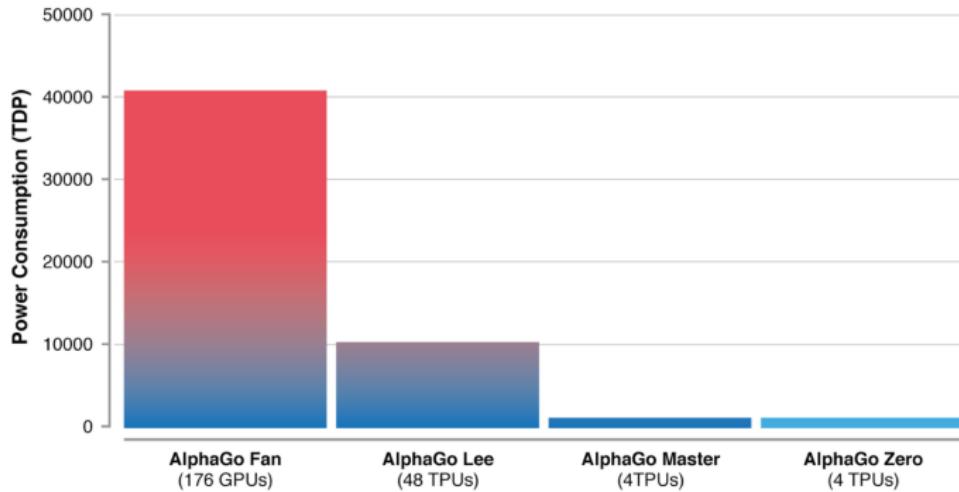
Unidad de procesamiento específica para TensorFlow



Tensor Processing Unit (TPU)

- TPU primera generación (febrero 2016):
 - Operaciones en 8 bits (enteros)
 - Arreglos de multiplicadores de 256×256
 - Set de instrucciones con multiplicación matricial, convoluciones y ReLU
- TPU segunda generación (mayo 2017):
 - Mayor ancho de banda
 - Operaciones en números flotantes
 - Módulos de 4 circuitos (180 TFLOPS)
 - 15 a 30 veces más rápidos que la primer generación

Costo energético



¿Es Alpha Zero un hito en la IA?

"Los juegos es a la computación lo que la formula 1 al diseño automotriz"

- Definitivamente es un logro excepcional
 - Sencillo y elegante
 - Para entornos con información perfecta (*i.e.* plegado de proteínas, reacciones químicas)
 - Combina técnicas muy atractivas
-
- Cuidadosamente promocionado por Google
 - La comunidad de ajedrez cuestiona los resultados respecto a *Stockfish*
 - ¿Depende únicamente de la capacidad de computo de Google?

Regresando al XX aniversario de la LCC

La computación actual requiere (requerirá):

- Modelos teóricos de computación
- Algoritmos y estructuras de datos eficientes
- Trabajo en grupos interdisciplinarios
- Álgebra lineal, optimización
- Probabilidad, estadística
- Inteligencia artificial

Muchas gracias por su atención