Alpha Zero.

Edwin Alexander Enciso, Luis David Restrepo Cadavid,
Sebastián Lopera Osorio
david95rc@utp.edu.co, edwinenciso@utp.edu.co
sebastian.lopera@utp.edu.co

Resumen- El futuro ha llegado de golpe. Hace veinte años muchos dudaban que una máquina fuese capaz de derrotar a los humanos en una partida de ajedrez. La victoria de Deep Blue, el ordenador creado por IBM, en 1997 sobre el campeón del mundo Gary Kasparov supuso un cambio de paradigma. Desde ese momento, la historia del deporte-ciencia cambió para siempre. Sin embargo, hace una semana llegó el holocausto absoluto. Un programa de inteligencia artificial destrozó de forma humillante al campeón de los ordenadores ajedrecistas, cambiando para siempre la disciplina y abriendo una nueva era a nivel científico.

Palabras clave- *ajedrez, Deep Blue, inteligencia artificial.*

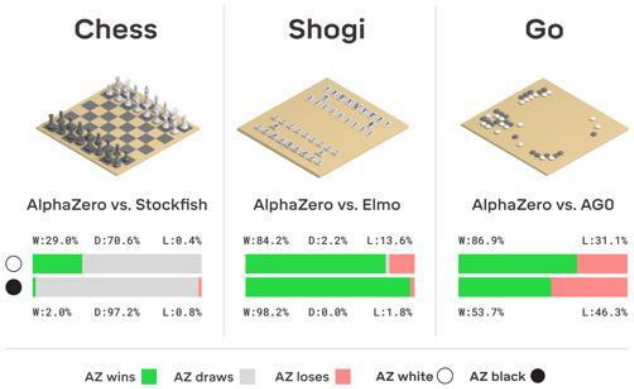
Abstract— The future has come suddenly. Twenty years ago many doubted that a machine was capable of defeating humans in a game of chess. The victory of Deep Blue, the computer created by IBM, in 1997 over world champion Gary Kasparov was a paradigm shift. From that moment on, the history of sports science changed forever. However, a week ago the absolute holocaust arrived. A program of artificial intelligence humiliatingly destroyed the champion of chess computers, forever changing discipline and opening a new era at the scientific level.

Keywords- *Chess, Deep Blue, Artificial intelligence.*

I. INTRODUCCION

Han pasado veinte años desde aquello. Hoy, uno de los campeones más implacables de la IA es AlphaZero, un programa de ordenador desarrollado por la compañía DeepMind y que nace de AlphaGo, que fue capaz de derrotar al mejor jugador de go del mundo. AlphaZero es capaz de alcanzar un nivel de maestría sobrehumana en el ajedrez, el go y el shogi, y además lo hace aprendiendo. Puede aprender desde cero, tan solo conociendo las reglas, en solo 24 horas, sencillamente por el hecho de jugar contra sí mismo. Ya no necesita que nadie programe reglas que debe seguir, o hacer infinidad de cálculos antes de cada jugada.

Hace un año, DeepMind publicó un borrador en el que demostraba que AlphaZero, era capaz de derrotar a los programas más avanzados para jugar a los tres juegos comentados: Stockfish, Elmo y el propio AlphaGo Zero. Lo hizo usando circuitos diseñados para permitir el aprendizaje maquinal y basados en redes neurales. Todos ellos le permitían a esta IA no solo alcanzar un rendimiento extraordinario, sino también acomodarse a una gama más amplia de reglas. Estos logros acaban de publicarse en la revista Science.



Resultados de AlphaZero jugando contra Stockfish, Elmo y AlphaGo Zero – DeepMind.

Nuestros resultados demuestran que un algoritmo de aprendizaje por refuerzo y de propósito general puede aprender, partiendo de cero –sin necesidad de añadir conocimientos o datos previos, proporcionados por humanos, solo conociendo las reglas– y alcanzar un rendimiento sobrehumano en varios juegos de gran complejidad», han escrito los autores de la investigación, dirigida por David Silver. Por tanto, AlphaZero representa un importante paso adelante en la tarea de crear una avanzada inteligencia artificial capaz de dominar juegos más complejos por su cuenta.

El objetivo de DeepMind es construir sistemas que puedan solucionar algunos de los problemas más complejos del mundo y crear un programa que puede enseñarse a sí mismo cómo dominar el ajedrez, el shogi y el go desde cero es un importante primer paso en ese camino», ha dicho en un comunicado Demis Hassabis, director y cofundador de DeepMind.

II. CONTENIDO

En la teoría de juegos, más que razonar sobre juegos específicos, a los matemáticos les gusta razonar sobre una clase especial de juegos: juegos de dos jugadores por turnos con información perfecta. En estos juegos, ambos jugadores saben todo lo relevante sobre el estado del juego en un momento dado. Además, no hay aleatoriedad o incertidumbre sobre cómo los movimientos afectan el juego; hacer un movimiento dado siempre dará como resultado el mismo estado final del juego, uno que ambos jugadores conocen con total certeza. Llamamos juegos como estos juegos clásicos.

Estos son ejemplos de juegos clásicos:

Tic-Tac-Toe
Ajedrez
Ir
Gomoku (un juego de 5 en fila en un tablero de go 19 por 19)
Mancala
Mientras que estos juegos no son:

Fecha de Recepción: (Letra Times New Roman de 8 puntos)
Fecha de Aceptación: Dejar en blanco

Póker (y la mayoría de los otros juegos de cartas)
Piedra Papel tijeras
El dilema del prisionero
Videojuegos como Starcraft
Cuando los juegos tienen elementos aleatorios y estados ocultos, es mucho más difícil diseñar sistemas de IA para jugarlos, aunque se han desarrollado potentes póquer y Starcraft AI. Por lo tanto, el algoritmo AlphaZero se limita a resolver solo juegos clásicos.[4]

Sin embargo con unas pequeñas mejoras es capaz de adaptarse a cualquier tipo de juego. En marzo de 2016, AlphaGo de Deepmind venció 18 veces al campeón mundial de Go player Lee Sedol 4–1 en una serie vista por más de 200 millones de personas. Una máquina había aprendido una estrategia sobrehumana para jugar Go, una hazaña que antes se creía imposible, o al menos, al menos a una década de lograrse.[5]

Imagen 2[5]

Esto en sí mismo, fue un logro notable. Sin embargo, el 18 de octubre de 2017, DeepMind dio un salto gigante más allá. El artículo ' Dominando el juego de Go sin conocimiento humano ' reveló una nueva variante del algoritmo, AlphaGo Zero, que había derrotado a AlphaGo 100-0. Increíblemente, lo hizo aprendiendo únicamente a través del juego propio, comenzando la 'tabula rasa' (estado en blanco) y encontrando gradualmente estrategias que superaran las encarnaciones anteriores de sí mismo. Ya no se requería una base de datos de juegos expertos humanos para construir una IA superhumana.[5]

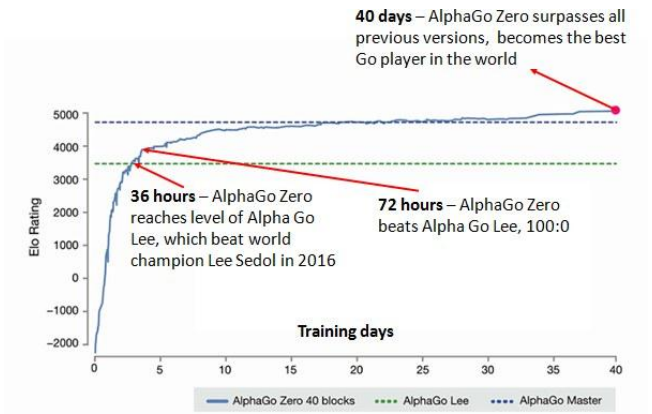


Imagen 3 [5]

Apenas 48 días después, el 5 de diciembre de 2017, DeepMind publicó otro artículo `Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm' que mostraba cómo AlphaGo Zero podía adaptarse para vencer a los programas campeones del mundo StockFish y Elmo en ajedrez y shogi. Todo el proceso de aprendizaje, desde mostrar los juegos por primera vez, hasta convertirse en el mejor programa de computadora del mundo, había llevado menos de 24 horas.
Con esto nació AlphaZero - el algoritmo general para ser bueno en algo, rápidamente, sin ningún conocimiento previo de la estrategia de expertos humanos.[5]

Cómo construir tu propio AlphaZero
En primer lugar, consulte la hoja de trucos de AlphaGo Zero para obtener un alto nivel de comprensión de cómo funciona AlphaGo Zero. Vale la pena referirnos a eso a medida que recorremos cada parte del código. También hay un gran artículo aquí que explica cómo funciona AlphaZero con más detalle.[5]

El código
Clona este repositorio Git, que contiene el código que voy a referenciar.
Para iniciar el proceso de aprendizaje, ejecute los dos paneles superiores del cuaderno.ipynb Jupyter. Una vez que haya construido suficientes posiciones de juego para llenar su memoria, la red neuronal comenzará a entrenar. A través del auto-juego y entrenamiento adicional, gradualmente mejorará en la predicción del valor del juego y los siguientes movimientos desde cualquier posición, resultando en una mejor toma de decisiones y un juego más inteligente en general. [5]
Ahora veremos el código con más detalle, y mostraremos algunos resultados que demuestran que la IA se hace más fuerte con el tiempo.
N.B - Este es mi propio entendimiento de cómo funciona AlphaZero basado en la información disponible en los documentos mencionados anteriormente. Si alguno de los siguientes puntos es incorrecto, ¡disculpe y me esforzaré por corregirlo! [5]

Conectar4
El juego que nuestro algoritmo aprenderá a jugar es Connect4 (o Four In A Row). No tan complejo como el Go... pero todavía hay 4.531.985.219.092 posiciones de juego en total.

Imagen 4 [5]

Las reglas del juego son sencillas. Los jugadores se turnan para introducir una pieza de su color en la parte superior de cualquier columna disponible. El primer jugador que consiga cuatro de su color seguidos - cada uno vertical, horizontal o diagonalmente - gana. Si toda la cuadrícula se llena sin que se cree una cuadrícula de cuatro en una fila, el juego se empata.
He aquí un resumen de los archivos clave que componen la base de código:

game.py
Este archivo contiene las reglas del juego para Connect4.
A cada cuadrado se le asigna un número del 0 al 41, como sigue:

0	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31	32	33	34
35	36	37	38	39	40	41

Imagen 5 [5]

El archivo game.py da la lógica para pasar de un estado de juego a otro, dada una acción elegida. Por ejemplo, dado el tablero vacío y la acción 38, el método takeAction devuelve un nuevo estado de juego, con la ficha del jugador titular en la parte inferior de la columna central.
Puede reemplazar el archivo game.py por cualquier archivo de juego que se ajuste a la misma API y el algoritmo, en principio, aprenderá la estrategia a través del auto-juego, basado en las reglas que usted le ha dado.[5]

run.ipynb
Contiene el código que inicia el proceso de aprendizaje. Carga las reglas del juego y luego itera a través del bucle principal del algoritmo, que consta de tres etapas:

1. **Auto-juego**
2. **Reentrenamiento de la red neuronal**
3. **Evaluación de la red neuronal**

Hay dos agentes implicados en este bucle, el mejor_jugador y el current_player.
El best_player contiene la red neuronal de mejor rendimiento y se utiliza para generar las memorias de auto-juego. El current_player entonces vuelve a entrenar su red neuronal en estas memorias y es entonces lanzado contra el best_player. Si gana, la red neuronal dentro del best_player es cambiada por la red neuronal dentro del current_player, y el bucle comienza de nuevo.[5]
agent.py
Contiene la clase Agent (un jugador en el juego). Cada jugador se inicia con su propia red neuronal y el árbol de búsqueda de Monte Carlo.
El método de simulación ejecuta el proceso de búsqueda en el árbol de Monte Carlo. Específicamente, el agente se mueve a un nodo de hoja del árbol, evalúa el nodo con su red neural y luego rellena el valor del nodo a través del árbol.
El método de acción repite la simulación varias veces para comprender cuál es el movimiento más favorable desde la posición actual. A continuación, devuelve la acción elegida al juego, para realizar la jugada.
El método de repetición vuelve a entrenar la red neuronal, utilizando memorias de juegos anteriores.[5]

Model.py

Imagen 6 [5]

Este archivo contiene la clase Residual_CNNN, que define cómo construir una instancia de la red neuronal.

Utiliza una versión condensada de la arquitectura de la red neuronal en el documento AlphaGoZero, es decir, una capa convolucional, seguida de muchas capas residuales, que luego se dividen en un valor y un jefe de política.

La profundidad y el número de filtros convolucionales se pueden especificar en el archivo de configuración.

La biblioteca Keras se utiliza para construir la red, con un backend de Tensorflow.

Para ver filtros convolucionales individuales y capas densamente conectadas en la red neural, haga lo siguiente dentro del cuaderno de run.ipynb:

current_player.model.viewLayers()

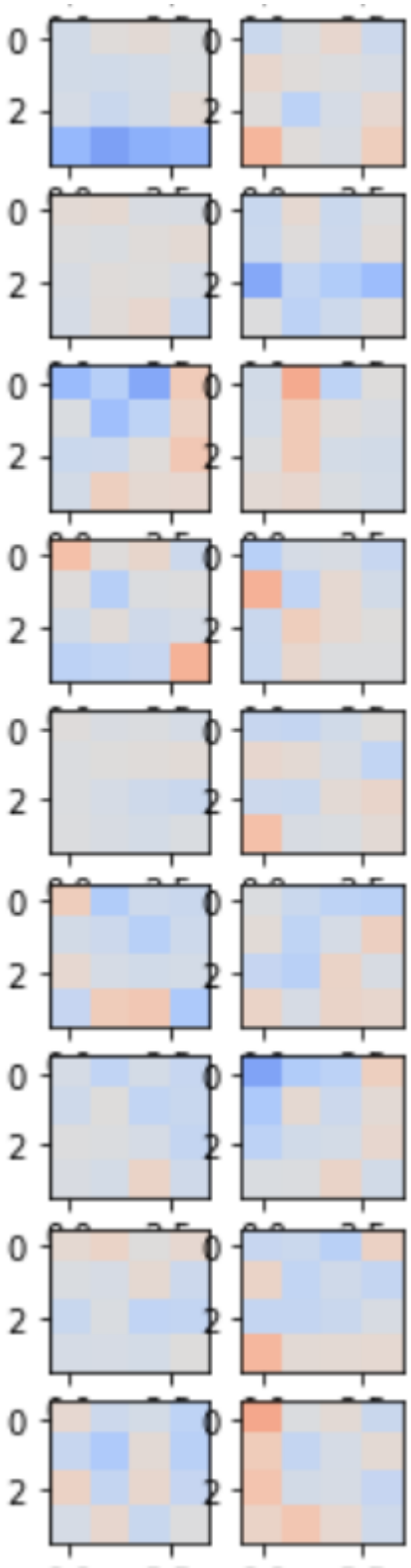


Imagen 7 [5]

MCTS.py

Contiene las clases Node, Edge y MCTS, que constituyen un árbol de búsqueda de Monte Carlo.

La clase MCTS contiene los métodos moveToLeaf y backFill mencionados anteriormente, e instancias de la clase Edge almacenan las estadísticas sobre cada movimiento potencial.

config.py

Aquí es donde se establecen los parámetros clave que influyen en el algoritmo.[5]

```
#### CONFIG FOR Connect 4

#### SELF PLAY
EPISODES = 75
MCTS_SIMS = 100
MEMORY_SIZE = 90000
TURNS_UNTIL_TAU0 = 10 # turn on which it starts playing deterministically
CPUCT = 1
EPSILON = 0.2
ALPHA = 0.8


#### RETRAINING
BATCH_SIZE = 256
EPOCHS = 1
REG_CONST = 0.0001
LEARNING_RATE = 0.1
MOMENTUM = 0.9
TRAINING_LOOPS = 10

HIDDEN_CNN_LAYERS = [
    {'filters':75, 'kernel_size': (4,4)}
    , {'filters':75, 'kernel_size': (4,4)}
    , {'filters':75, 'kernel_size': (4,4)}
    , {'filters':75, 'kernel_size': (4,4)}
    , {'filters':75, 'kernel_size': (4,4)}
    , {'filters':75, 'kernel_size': (4,4)}
    ]

#### EVALUATION
EVAL_EPISODES = 20
SCORING_THRESHOLD = 1.3
```

Imagen 8 [5]

El ajuste de estas variables afectará el tiempo de ejecución, la precisión de la red neuronal y el éxito general del algoritmo. Los parámetros anteriores producen un reproductor Connect4 de alta calidad, pero lleva mucho tiempo hacerlo. Para acelerar el algoritmo, pruebe con los siguientes parámetros.

```
EPISODES = 25
MCTS_SIMS = 50
MEMORY_SIZE = 30000
```

Imagen 9[5]

funcs.py

Contiene las funciones playMatches y playMatchesBetweenVersions que reproducen partidos entre dos agentes.

Para jugar contra tu creación, ejecuta el siguiente código (también está en el cuaderno run.ipynb)

```
from game import Game
from funcs import playMatchesBetweenVersions
import loggers as lgenv = Game()
playMatchesBetweenVersions(
env
, 1  # the run version number where the computer player is located
, -1 # the version number of the first player (-1 for human)
, 12 # the version number of the second player (-1 for human)
, 10 # how many games to play
, lg.logger_tourney # where to log the game to
, 0  # which player to go first - 0 for random
)
```

inicializar.py

Al ejecutar el algoritmo, todos los archivos de modelo y de memoria se guardan en la carpeta de ejecución, en el directorio raíz.

Para reiniciar el algoritmo desde este punto de control más tarde, transfiera la carpeta de ejecución a la carpeta run_archive, adjuntando un número de ejecución al nombre de la carpeta. A continuación, introduzca el número de ejecución, el número de versión del modelo y el número de versión de la memoria en el archivo initialise.py, correspondiente a la ubicación de los archivos relevantes en la carpeta run_archive. La ejecución del algoritmo como de costumbre comenzará entonces desde este punto de control.

memory.py

Una instancia de la clase Memory almacena las memorias de juegos anteriores, que el algoritmo utiliza para volver a entrenar la red neuronal del current_player.

loss.py

Este archivo contiene una función de pérdida personalizada, que enmascara las predicciones de movimientos ilegales antes de pasar a la función de pérdida de entropía cruzada.

settings.py

Las ubicaciones de las carpetas run y run_archive.

loggers.py

Los archivos de registro se guardan en la carpeta de registro dentro de la carpeta de ejecución.

Para activar el registro, establezca los valores de las variables logger_disabled en False dentro de este archivo.

Ver los archivos de registro le ayudará a entender cómo funciona el algoritmo y ver dentro de su "mente". Por ejemplo, aquí hay una muestra del archivo logger.mcts. [5]

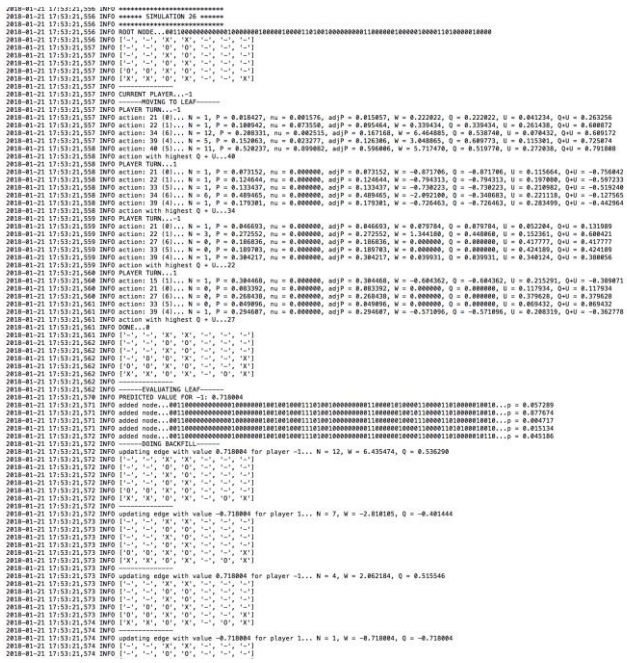


Imagen 10 [5]

Del mismo modo, desde el archivo logger.tourney, puede ver las probabilidades adjuntas a cada movimiento, durante la fase de evaluación:

```
2018-01-21 22:37:27,826 INFO ====================
2018-01-21 22:37:27,827 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-21 22:37:27,827 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-21 22:37:27,827 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-21 22:37:27,827 INFO ['-', '-', '-', 'X', 'O', '-', '-']
2018-01-21 22:37:27,827 INFO ['-', '-', '-', 'X', 'O', 'X', '-']
2018-01-21 22:37:27,827 INFO ['O', 'X', 'X', 'O', 'X', 'O', '-']
2018-01-21 22:37:27,827 INFO --------------
2018-01-21 22:37:30,619 INFO action: 18
2018-01-21 22:37:30,619 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-21 22:37:30,619 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-21 22:37:30,619 INFO ['----', '----', '----', '0.11', '0.44', '----', '----']
2018-01-21 22:37:30,620 INFO ['----', '----', '----', '----', '----', '0.10', '----']
2018-01-21 22:37:30,620 INFO ['0.08', '0.10', '0.09', '----', '----', '----', '----']
2018-01-21 22:37:30,620 INFO ['----', '----', '----', '----', '----', '----', '0.09']
2018-01-21 22:37:30,620 INFO MCTS perceived value for O: 0.130000
2018-01-21 22:37:30,621 INFO NN perceived value for O: 0.040000
```

Imagen 11 [5]

Resultados

El entrenamiento durante un par de días produce la siguiente tabla de pérdidas contra el número de iteración de mini-lotes:

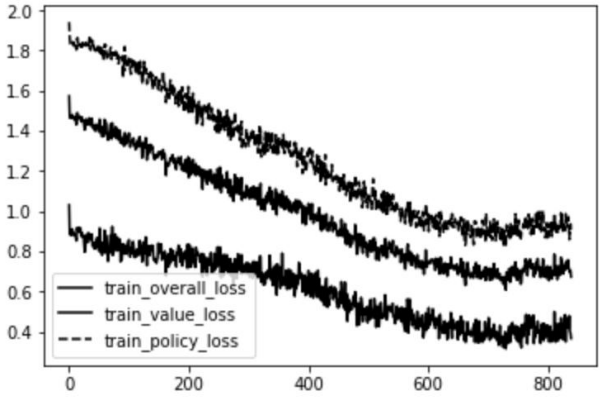


Imagen 12 [5]

La línea superior es el error en la cabeza de la política (la entropía cruzada de las probabilidades de movimiento del MCTS, contra la salida de la red neuronal). El resultado final es el error en la cabeza del valor (el error promedio al cuadrado entre el valor real del juego y la predicción de la red neuronal del valor). La línea media es una media de las dos.

Claramente, la red neural está mejorando en la predicción del valor de cada estado de juego y los próximos movimientos probables. Para mostrar cómo esto resulta en un juego cada vez más fuerte, corrí una liga entre 17 jugadores, desde la primera iteración de la red neuronal, hasta la 49ª. Cada pareja jugó dos veces, y ambos jugadores tuvieron la oportunidad de jugar primero.

Aquí están las posiciones finales:

	player2																
player 1	1	4	7	10	13	16	19	22	25	28	31	34	37	40	43	46	49	total
1		-2	-2	-2	-2	0	-1	-2	-2	-2	-2	-2	-2	-2	-2	-2	-1	-28
4	2		0	0	0	-2	-2	-1	-2	0	-2	-2	-2	-2	0	-2	-2	-17
7	2	0		-2	-2	-1	0	-1	-2	-2	-2	1	-2	-2	0	-2	-2	-17
10	2	0	2		0	-1	-1	-2	-1	-2	-2	-2	-2	-2	-2	0	-2	-15
13	2	0	2	0		-1	-2	-1	-1	-2	-1	-2	-1	-2	-2	-2	-2	-15
16	0	2	1	1	1		-1	-1	-1	-1	-1	-1	-1	-1	-2	-2	-2	-9
19	1	2	0	1	2	1		0	-2	-2	1	-1	-1	2	-2	-2	-2	-2
22	2	1	1	2	1	1	0		-2	-2	-1	-1	-1	-2	-2	-2	-1	-6
25	2	2	2	1	1	1	2	2		1	1	-1	0	0	-1	-2	-2	9
28	2	0	2	2	2	1	2	2	-1		0	2	1	0	0	0	-2	13
31	2	2	2	2	1	1	-1	1	-1	0		1	0	0	0	0	-2	8
34	2	2	-1	2	2	1	1	1	1	-2	-1		0	-1	-2	-2	-2	1
37	2	2	2	2	1	1	1	1	0	-1	0	0		-1	2	1	0	13
40	2	2	2	2	2	1	-2	2	0	0	0	1	1		-2	-2	-1	8
43	2	0	0	2	2	2	2	2	1	0	0	2	2	2		1	2	22
46	2	2	2	0	2	2	2	2	2	0	0	2	-1	2	-1		0	18
49	1	2	2	2	2	2	2	1	2	2	2	2	0	1	-2	0		21

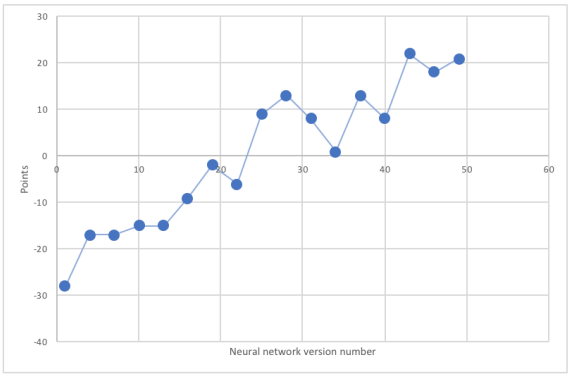


Imagen 13 [5]

Claramente, las versiones posteriores de la red neuronal son superiores a las versiones anteriores, ganando la mayoría de sus juegos. También parece que el aprendizaje no se ha saturado todavía - con más tiempo de entrenamiento, los jugadores continuarían fortaleciéndose, aprendiendo estrategias cada vez más intrincadas.

Como ejemplo, una estrategia clara que la red neuronal ha favorecido con el tiempo es agarrar la columna central temprano. Observe la diferencia entre la primera versión del algoritmo y, por ejemplo, la trigésima versión:

```
2018-01-26 12:20:26,734 INFO EPISODE 2 OF 10
2018-01-26 12:20:26,734 INFO ====================
2018-01-26 12:20:26,735 INFO player1 plays as X
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:26,736 INFO --------------
2018-01-26 12:20:28,083 INFO action: 39
2018-01-26 12:20:28,083 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:28,083 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:28,083 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:28,083 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:28,083 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:28,084 INFO ['0.15', '0.13', '0.14', '0.14', '0.16', '0.12', '0.15']
2018-01-26 12:20:28,084 INFO MCTS perceived value for X: 0.010000
2018-01-26 12:20:28,084 INFO NN perceived value for X: 0.060000
2018-01-26 12:20:28,084 INFO ====================
2018-01-26 12:20:28,084 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:28,084 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:28,084 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:28,084 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:28,084 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:28,085 INFO ['-', '-', '-', '-', 'X', '-', '-']
2018-01-26 12:20:28,085 INFO --------------
2018-01-26 12:20:30,339 INFO action: 37
2018-01-26 12:20:30,339 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:30,339 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:30,339 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:30,339 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:30,340 INFO ['----', '----', '----', '----', '0.13', '----', '----']
2018-01-26 12:20:30,340 INFO ['0.12', '0.12', '0.17', '0.17', '----', '0.14', '0.14']
2018-01-26 12:20:30,340 INFO MCTS perceived value for O: 0.010000
2018-01-26 12:20:30,340 INFO NN perceived value for O: 0.060000
2018-01-26 12:20:30,340 INFO ====================
2018-01-26 12:20:30,341 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:30,341 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:30,341 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:30,341 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:30,341 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:30,341 INFO ['-', '-', 'O', '-', 'X', '-', '-']
2018-01-26 12:20:30,341 INFO --------------
2018-01-26 12:20:31,598 INFO action: 36
2018-01-26 12:20:31,598 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:31,598 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:31,598 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:31,598 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:20:31,598 INFO ['----', '----', '0.14', '----', '0.14', '----', '----']
2018-01-26 12:20:31,598 INFO ['0.14', '0.16', '----', '0.15', '----', '0.15', '0.14']
2018-01-26 12:20:31,599 INFO MCTS perceived value for X: 0.010000
2018-01-26 12:20:31,599 INFO NN perceived value for X: 0.060000
2018-01-26 12:20:31,599 INFO ====================
2018-01-26 12:20:31,599 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:31,599 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:31,599 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:31,599 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:31,599 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:20:31,599 INFO ['-', 'X', 'O', '-', 'X', '-', '-']
```

Imagen 14 [5]

```
2018-01-26 12:18:21,529 INFO EPISODE 1 OF 10
2018-01-26 12:18:21,529 INFO ====================
2018-01-26 12:18:21,530 INFO player2 plays as X
2018-01-26 12:18:21,530 INFO --------------
2018-01-26 12:18:21,530 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:21,531 INFO --------------
2018-01-26 12:18:23,382 INFO action: 38
2018-01-26 12:18:23,382 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:23,383 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:23,383 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:23,383 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:23,383 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:23,383 INFO ['0.03', '0.04', '0.04', '0.04', '0.74', '0.05', '0.07', '0.03']
2018-01-26 12:18:23,383 INFO MCTS perceived value for X: 0.110000
2018-01-26 12:18:23,383 INFO NN perceived value for X: 0.160000
2018-01-26 12:18:23,383 INFO ====================
2018-01-26 12:18:23,384 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:23,384 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:23,384 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:23,384 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:23,384 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:23,384 INFO ['-', '-', '-', 'X', '-', '-', '-']
2018-01-26 12:18:23,384 INFO --------------
2018-01-26 12:18:25,232 INFO action: 31
2018-01-26 12:18:25,232 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:25,232 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:25,232 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:25,232 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:25,232 INFO ['----', '----', '----', '0.81', '----', '----', '----']
2018-01-26 12:18:25,232 INFO ['0.03', '0.03', '0.04', '----', '0.04', '0.03', '0.02']
2018-01-26 12:18:25,232 INFO MCTS perceived value for O: -0.070000
2018-01-26 12:18:25,233 INFO NN perceived value for O: 0.020000
2018-01-26 12:18:25,233 INFO ====================
2018-01-26 12:18:25,233 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:25,233 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:25,233 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:25,233 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:25,233 INFO ['-', '-', '-', 'O', '-', '-', '-']
2018-01-26 12:18:25,233 INFO ['-', '-', '-', 'X', '-', '-', '-']
2018-01-26 12:18:25,233 INFO --------------
2018-01-26 12:18:26,719 INFO action: 24
2018-01-26 12:18:26,720 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:26,720 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:26,720 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:26,720 INFO ['----', '----', '----', '0.82', '----', '----', '----']
2018-01-26 12:18:26,720 INFO ['----', '----', '----', '----', '----', '----', '----']
2018-01-26 12:18:26,721 INFO ['0.01', '0.02', '0.06', '----', '0.04', '0.04', '0.01']
2018-01-26 12:18:26,721 INFO MCTS perceived value for X: 0.040000
2018-01-26 12:18:26,721 INFO NN perceived value for X: 0.060000
2018-01-26 12:18:26,721 INFO ====================
2018-01-26 12:18:26,721 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:26,721 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:26,721 INFO ['-', '-', '-', '-', '-', '-', '-']
2018-01-26 12:18:26,721 INFO ['-', '-', '-', 'X', '-', '-', '-']
2018-01-26 12:18:26,722 INFO ['-', '-', '-', 'O', '-', '-', '-']
2018-01-26 12:18:26,722 INFO ['-', '-', '-', 'X', '-', '-', '-']
2018-01-26 12:18:26,722 INFO --------------
```

Imagen 15 [5]

III. CONCLUSIONES:

Algunas personas están desmeritando los logros de Alphazero al mencionar que jugó contra una versión pobre de Stokfisch etc. Comprendo sus observaciones, también le tengo un cariño especial a StockFisch, pero lo cierto es que los módulos como Stokfisch, Ribka, Houdini, etc. están fundamentados en la fuerza bruta apoyados por una evolución interesante de algoritmos, pero AlphaGo está fundamentado en Inteligencia Artificial y sus algoritmos son mucho más eficientes, "están vivos", aprenden solos.

La inteligencia artificial no nació ayer, desde los 60 ha seguido evolucionando y ha tomado su fuerza en el mundo actual gracias a los avances en computación, hardware y masificación de las TIC (Tecnologías de la información y las comunicaciones), por lo que lo digo muy respetuosamente. Los módulos que conocemos no tienen mucho por hacer contra estos monstruos de la IA y en el futuro si se siguen enfrentando veremos cómo caen irremediablemente ante el paso aplastante de la IA. Pero esto no debe de disgustarnos al contrario debe de motivarnos pues los amantes del ajedrez de alguna forma somos los beneficiados y desde ya somos bendecidos al poder contemplar estas maravillas. Imagínense lo que verán nuestros hijos y las futuras generaciones.

A la pregunta de si los seres humanos jugaremos algún día como las máquinas, la respuesta es un "NO" rotundo, porque nuestro sistema de aprendizaje es mucho más lento a pesar de que poseemos muchísimas más conexiones neurales que las que tiene Google, pero a la pregunta de si podemos mejorar nuestro juego gracias a la IA como ALphaGo la respuesta es "Sí" y acá es donde me detengo, ya que hace un par de años vengo trabajo en un proyecto de investigación que busca aprovechar algunas metodologías de IA que pueden aplicarse en el aprendizaje humano para hacerlo más eficiente, no sólo en ajedrez sino en el aprendizaje de diversas tareas cómo dominar un idioma por ejemplo, pero bueno esa es otra historia.

IV. REFERENCIAS

1. https://www.lavanguardia.com/deportes/otros-deportes/20171214/433624379301/alpha-zero-deep-mind-gary-kasparov-ajedrez-inteligencia-artificial.html

2. https://www.abc.es/ciencia/abci-inteligencia-artificial-alphazero-juega-y-aprende-como-superhombre-201812062117_noticia.html

3. https://thezugzwangblog.com/desvelamos-los-secretos-del-juego-alphazero/

4. https://nikcheerla.github.io/deeplearningschool/2018/01/01/AlphaZero-Explained/

5. https://medium.com/applied-data-science/how-to-build-your-own-alphazero-ai-using-python-and-keras-7f664945c188