Feature set analysis for chess ЭUИИ networks

Tesis de Licenciatura

Martín Emiliano Lombardo

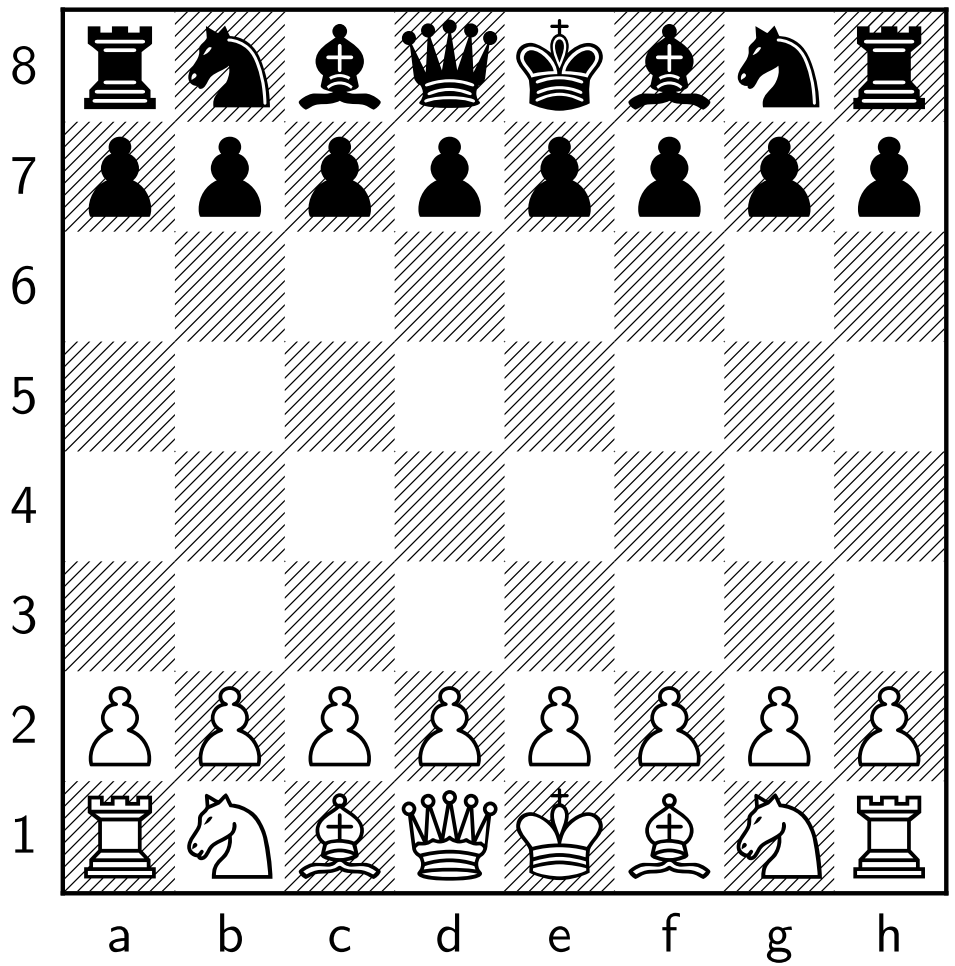
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

2024

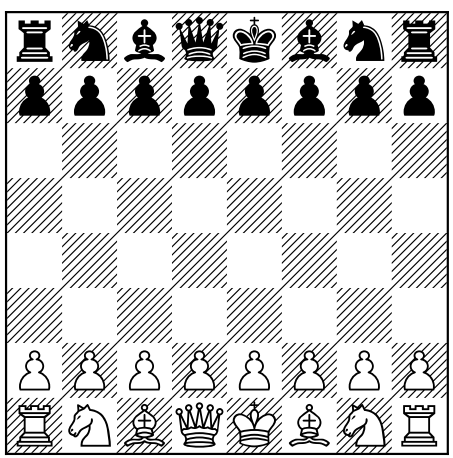
Introducción

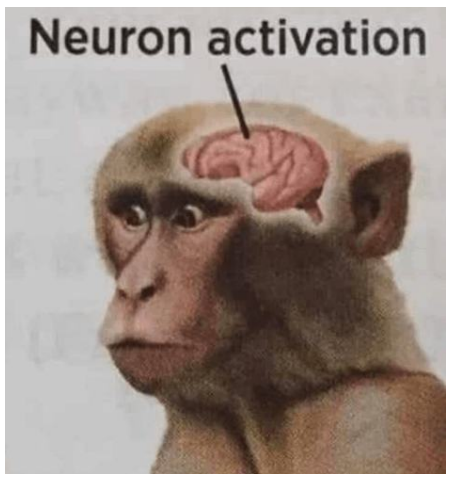
Ajedrez

- Dos jugadores
- Suma cero



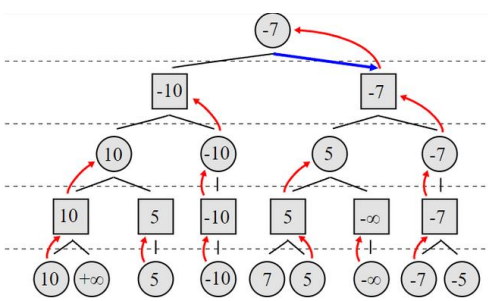
Humano vs. Computadora





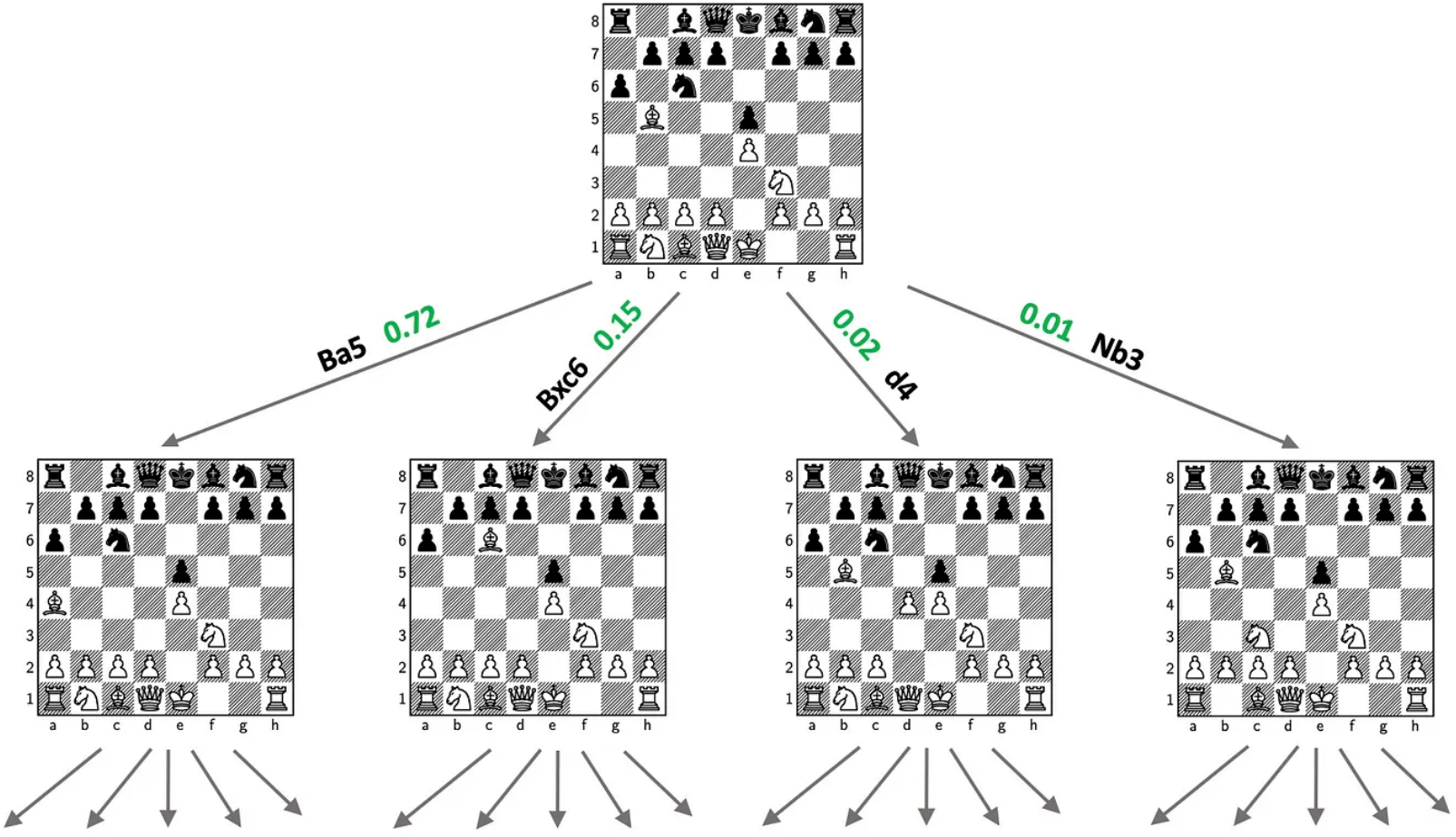
e2e4

Chess Engine



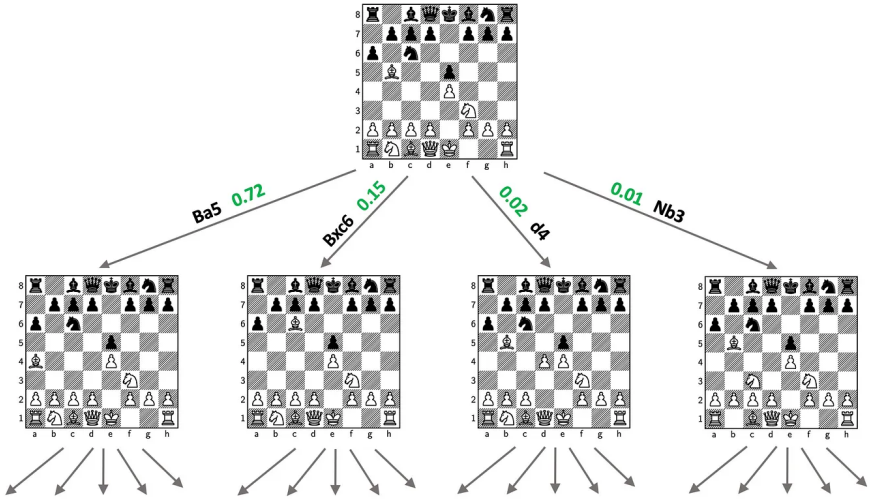
e2e4

Ajedrez como árbol



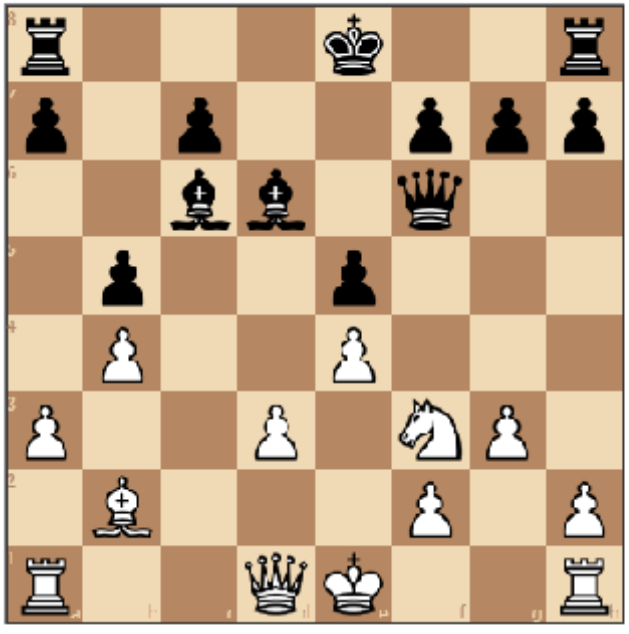
Motores de ajedrez (Chess Engines)

- Exploran el árbol de juego (Minimax, MCTS, etc.)
- Utilizan funciones de evaluación en las hojas
- La evaluación se propaga hacia arriba, según el algoritmo

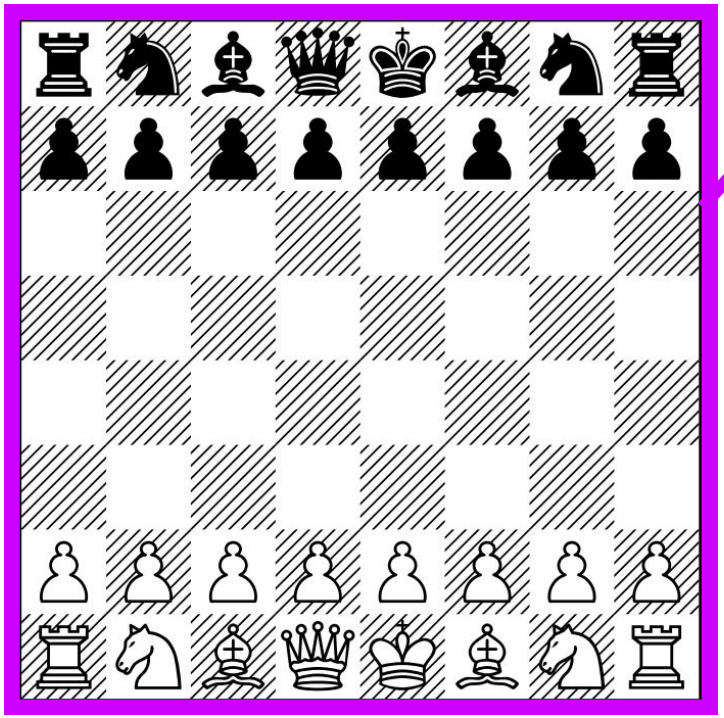


Función de evaluación

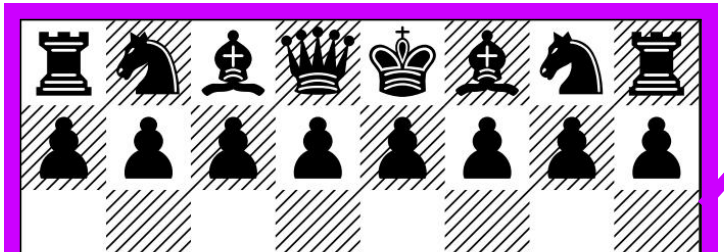
$$f\left(\quad\right) = 5$$



(adelanto) Feature set: ¿Cómo transformar la posición a un vector?



$$f(?) = ?$$



feature set!

Motores de ajedrez (breve historia)

Plan

asdasd

- Text visible on slide 1
- Text visible on slide 2
- Text visible on slide 3
- Text visible on slide 4

asdasd

Contenido

1 Introducción

2 Engine

3 Feature set
- Motivación
- Definición
- Operadores
- Feature sets conocidos
- Resumen

4 ЭUИИ (NNUE)

5 Training

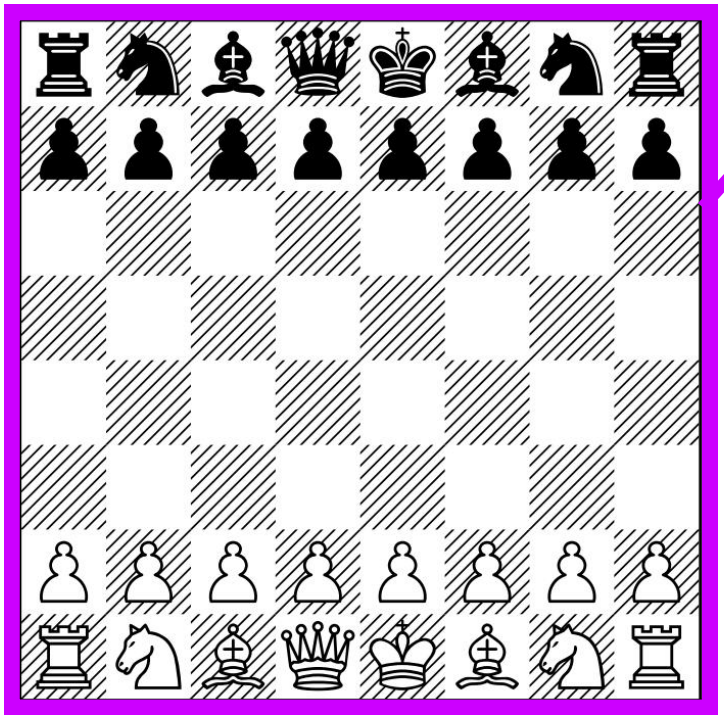
6 Experimentos

7 Conclusión

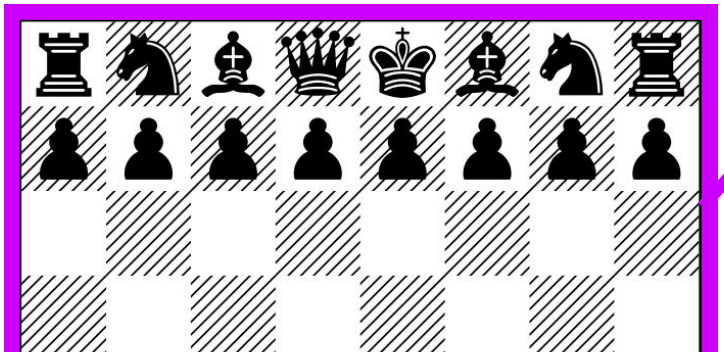
Engine

Feature set

¿Cómo transformar la posición a un vector?



$$f(?) = ?$$



feature set!

$$f(?) = ?$$

Definición

Un **feature set** S_P se define con un conjunto S y un predicado asociado $P(e)$, donde:

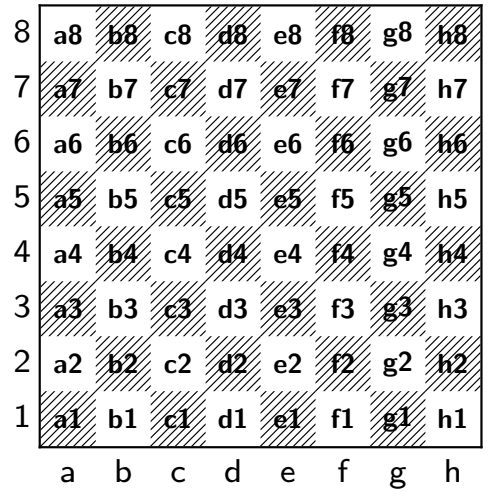
- S es un conjunto de conceptos (rol, color, celda, número, etc.).
- $P(e)$ es un predicado que determina si e está presente (o *activo*) en la posición (implícita).

- Cada elemento en S_P es un *feature*.
- Cada *feature* es un valor en el vector de entrada, valiendo 1 si está *activo* y 0 si no.

Ejemplos de S

Información posicional:

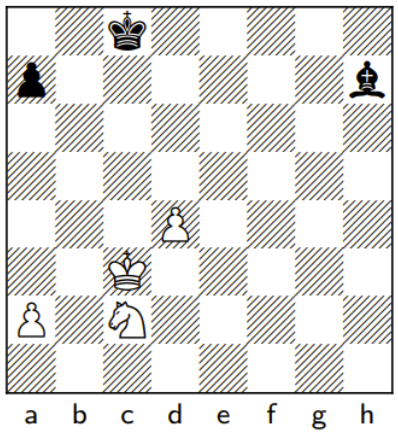
$$\textsc{Files} = \{a, b, ..., h\}$$
$$\textsc{Ranks} = \{1, 2, ..., 8\}$$
$$\textsc{Squares} = \{a1, a2, ..., h8\}$$



Información sobre las piezas:

$$\textsc{Roles} = \{ \text{♙ Pawn}, \text{♘ Knight}, \text{♗ Bishop}, \text{♖ Rook}, \text{♕ Queen}, \text{♔ King}$$
$$\textsc{Colors} = \{\text{○ White}, \text{● Black}\}$$

Ejemplo completo



	Feature set	
	$(\textsc{Files} \times \textsc{Colors})_P$	$(\textsc{Files} \times \textsc{Roles})_Q$
Active features	$\langle \text{a}, \bigcirc \rangle$, $\langle \text{a}, \bullet \rangle$, $\langle \text{c}, \bullet \rangle$, $\langle \text{c}, \bigcirc \rangle$, $\langle \text{d}, \bigcirc \rangle$, $\langle \text{h}, \bullet \rangle$	$\langle \text{a}, \♙ \rangle$, $\langle \text{c}, \♔ \rangle$, $\langle \text{c}, \♘ \rangle$, $\langle \text{d}, \♙ \rangle$, $\langle \text{h}, \♗ \rangle$

$P(\langle f, c \rangle)$: there is a piece in file f with color c.
$Q(\langle f, r \rangle)$: there is a piece in file f with role r.

Operación: Suma $\oplus$ (concatenación)

Hay veces que es útil combinar información de dos *feature sets*

$$S_P, T_Q : \text{feature sets}$$

$$S_P \oplus T_Q = (S \cup T)_R$$

$$\text{where } R(e) = \begin{cases} P(e) & \text{if } e \in S \\ Q(e) & \text{if } e \in T \end{cases}$$

Operación: Producto $\times$ (and)

$$S_P \times T_Q = (S \times T)_R$$

$$\text{where } R(\langle e_0, e_1 \rangle) = P(e_0) \ \wedge \ Q(e_1)$$

Feature set: All

La codificación más natural de una posición de ajedrez

$$\textsc{All} : (\textsc{Squares} \times \textsc{Roles} \times \textsc{Colors})_P$$

$P(\langle s, r, c \rangle)$: there is a piece in square s with role r and color c

- Es pequeño: $64 \times 6 \times 2 = 768$ *features*
- Es completo: contiene toda la información de la posición
- Es muy rápido computar cuáles *features* están activas

Feature set: King-All ó "KA"

Los engines modernos usan variaciones del siguiente feature set. Permite entender la posición en relación a la posición del rey:

$$\text{King-All} = \text{Square}_K \times \text{All}$$

$K(s)$: s is the square of the king of the side to move

- Es grande: $64 \times 768 = 49152$ *features*
- Es muy rápido como All
- Entrenarlo require un dataset más grande y lleva más tiempo (no me meto acá)

Feature sets: resumen

- $\boldsymbol{S}$: set of concepts (roles, colors, squares, files, ranks, etc.).
- $\boldsymbol{P}(\boldsymbol{e})$: predicate that defines when the feature e is present in the (implicit) position.
- $\boldsymbol{S_P}$: a feature set. Every element in S_P is a feature. Features that satisfy P are *active*.
- $S_P \times T_Q = (S \times T)_R$ where $R(\langle e_0, e_1 \rangle) = P(e_0) \ \wedge \ Q(e_1)$
- $S_P \oplus T_Q = (S \cup T)_R$ where $R(e) = \begin{cases} P(e) & \text{if } e \in S \\ Q(e) & \text{if } e \in T \end{cases}$

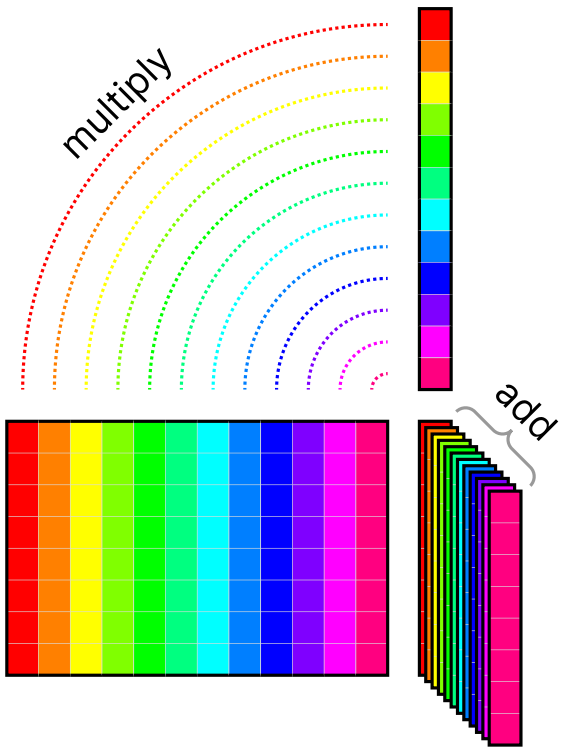
ƎUИИ (NNUE)

ЭUИИ: **E**fficiently **U**pdatable **N**eural **N**etworks

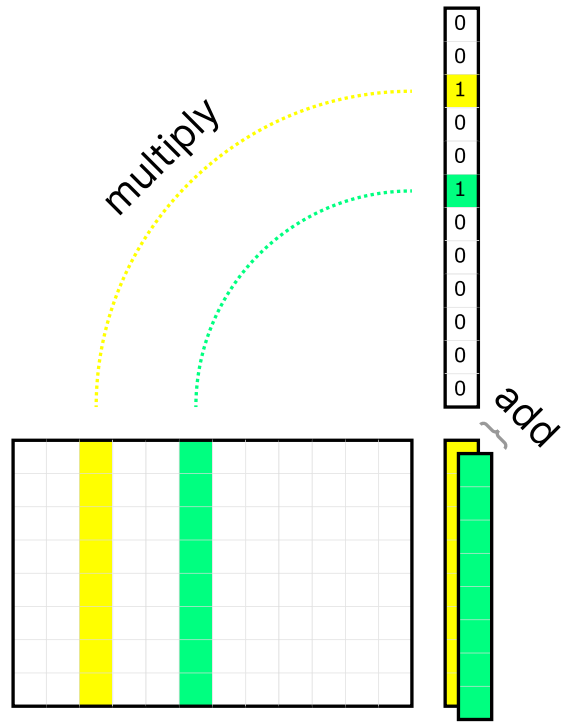
ƎUИИ: **N**eural **N**etworks

- El input es un vector one-hot generado por el *feature set*.
  - Debe tener pocos *features* activos (rala): introduce una cota superior.
- La red es una *feedforward* clásica con dos capas ocultas.

Linear layer



(c) Linear layer



(d) Linear layer with sparse inputs

Figure: Linear layer operation comparison. Figures from [18].

ƎUИИ: **E**fficient **U**pdates

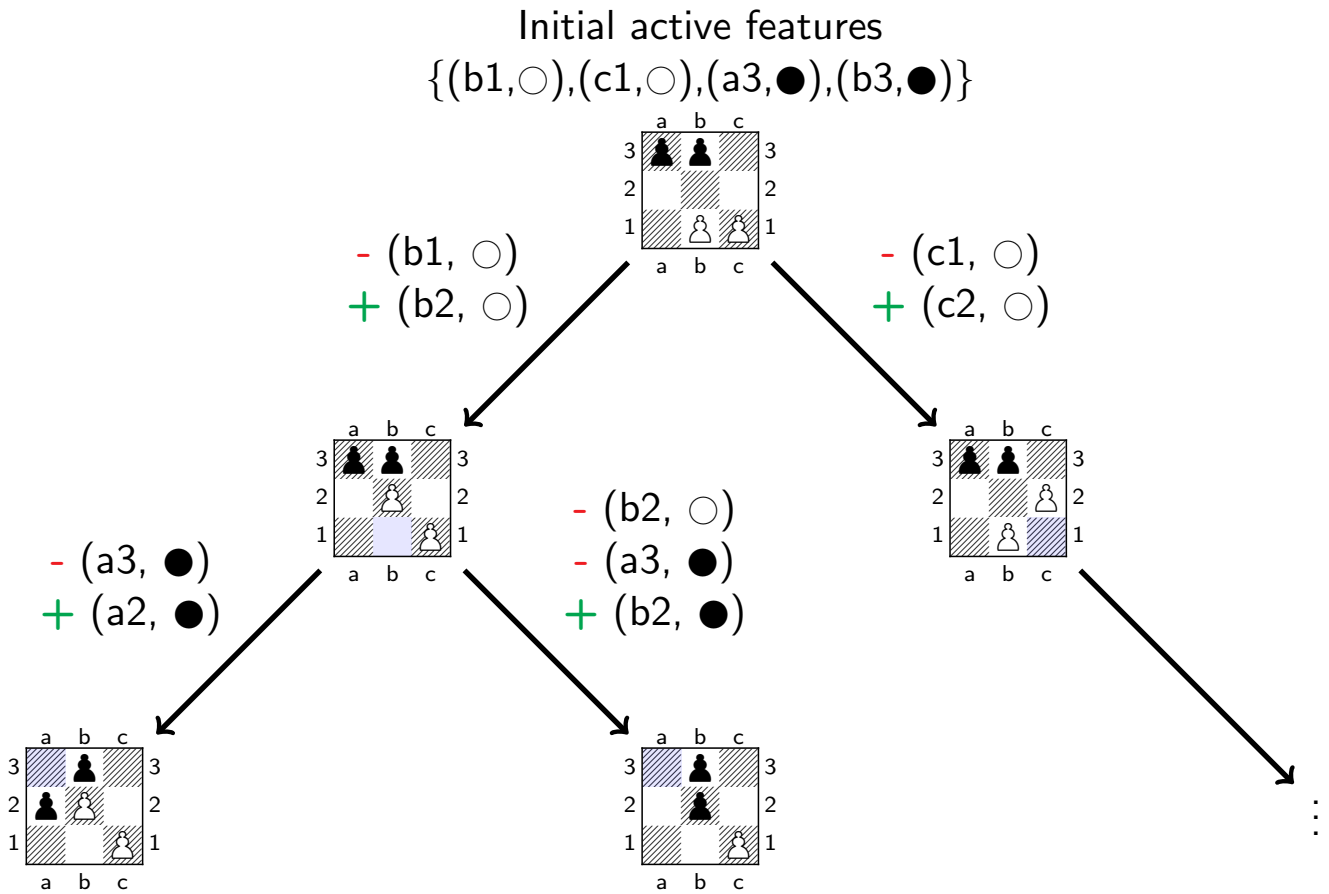


Figure: Partial tree of feature updates (removals and additions) for (Squares × Colors) (white's point of view) in a simplified 3x3

ƎUИИ: Tradeoff

motivacion comparacion de burns

Training

Experimentos

Setup de training

Recapitulando... ¿Qué hay que definir para entrenar una red?

- **Feature set**: determina la codificación y los patrones que se pueden aprender
- **Dataset**: datos de entrenamiento, visto anteriormente
- **Arquitectura de la red**: el tamaño de cada capa; L_1 y L_2
- **Método de entrenamiento**: PQR/target scores; determina el formato de las muestras y la loss function
- **Hiperparámetros**: learning rate, batch size, epochs, etc.

Setup de evaluación

¿Cómo evalúo el performance de una red entrenada?

- **Loss** (train y val.): indica la calidad de las predicciones.
  - Permite detectar overfitting y otros problemas
- **Puzzle accuracy**: porcentaje de movimientos acertados en puzzles de Lichess.
  - Sólo hay un movimiento correcto
  - Proxy (muy malo) de la fuerza de la red
- **Elo relativo**: la medida más común para comparar engines.
  - Se realizan torneos de 100ms por movimiento
  - El elo es calculado a partir de Ordo

Baseline: motivación

Busco fijar el setup de entrenamiento con valores razonables

- El feature set va a cambiar cada experimento
- El dataset está fijo
- El método de entrenamiento principal es *target scores*

Entonces queda por determinar...

- La arquitectura de la red (L_1 y L_2)
- Los hiperparámetros

Baseline: hiperparámetros

Los hiperparámetros fueron seleccionados en base al trainer oficial de Stockfish:

- **Learning rate**: 0.0005
- **Exponential decay**: 0.99
- **Batch size**: 16384
- **Epoch size**: 100 million
  - cada epoch realiza 6104 batches
- **Epochs**: 256
  - cada run observa *25.6 billion* samples

Baseline: experimento

Sólo queda buscar parámetros L_1 y L_2 razonables. Realizo una búsqueda en grilla con:

- L1 $\in \{256, 512, 1024, 2048\}$
- L2 $\in \{32, 64, 128, 256\}$

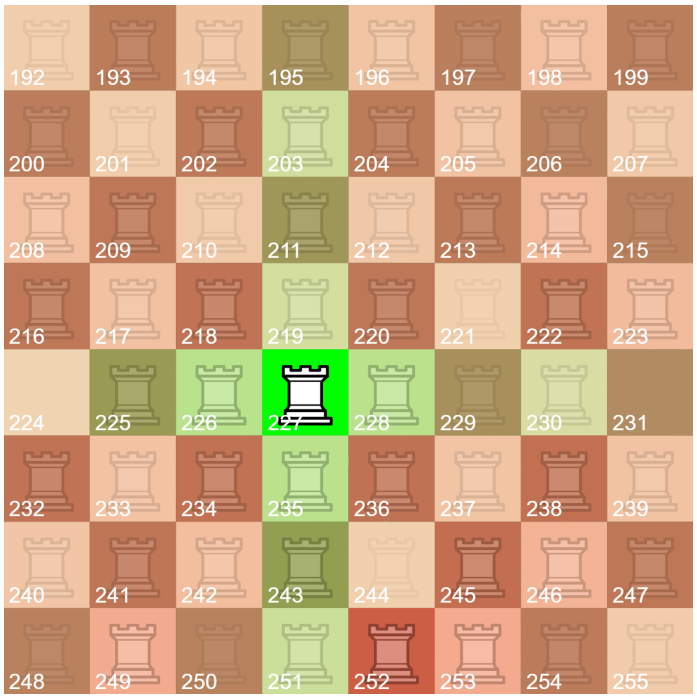
El feature set a utilizar es $\textsc{All}[768]$.

Baseline: resultados

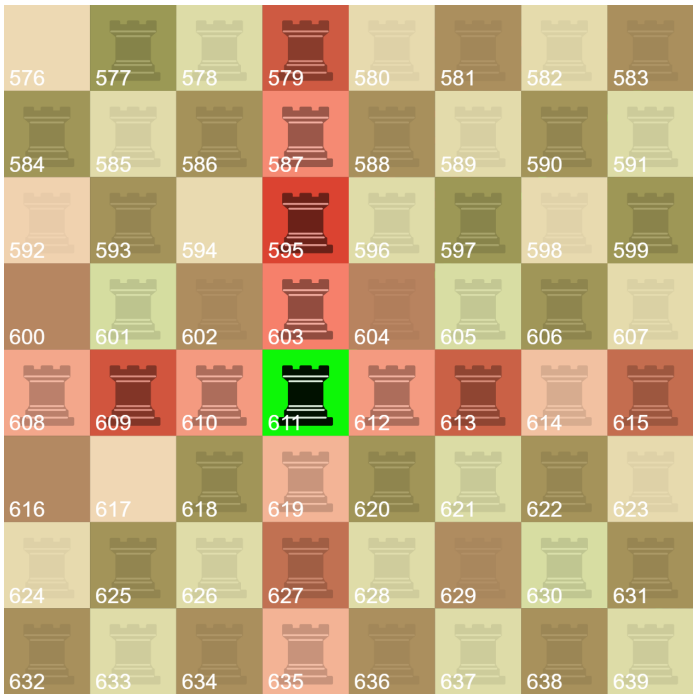
Baseline: conclusión

- **L2=32**. El performance cae dramáticamente si L2 aumenta, utilizo el más bajo.
  - Sería buena idea probar valores más chicos de L2.
- **L1=512**. Es el mejor valor para L2=64 y L2=128, y en margen de error para L2=32.
  - Además es el más rápido de entrenar.

Axis encoding: motivación



(a) ○ White



(b) ● Black

Figure: Weights of **a neuron** in the L1 layer, which are connected to features in ALL where the role is ♖ Rook. The intensity represents the weight value, and the color represents the sign (although not relevant).

Axis encoding: motivación

La red detecta patrones parecidos a los movimientos de las piezas. Para hacerle la vida más fácil a la red, propongo agregar features como:

"*there is a ○ White ♖ Rook in the 4th rank*"

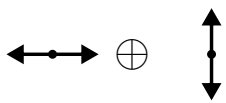
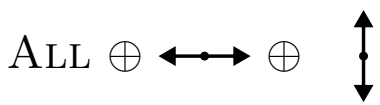
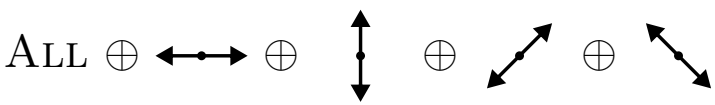
Axis encoding: experimento

Horizontal (across files) | Vertical (across ranks) | Diagonal 1 | Diagonal 2

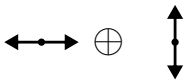
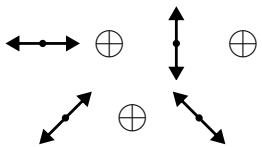
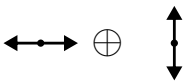
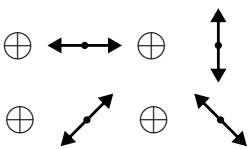
Depiction	Block name	Definition	Number of features
↔	H	$(\textsc{Files} \times \textsc{Roles} \times \textsc{Colors})_P$	96
↕	V	$(\textsc{Ranks} \times \textsc{Roles} \times \textsc{Colors})_P$	96
⤢	D1	$(\textsc{Diags1} \times \textsc{Roles} \times \textsc{Colors})_P$	180
⤡	D2	$(\textsc{Diags2} \times \textsc{Roles} \times \textsc{Colors})_P$	180

$\mathrm{P}(\langle x, r, c \rangle)$: there is a piece in x with role r and color c

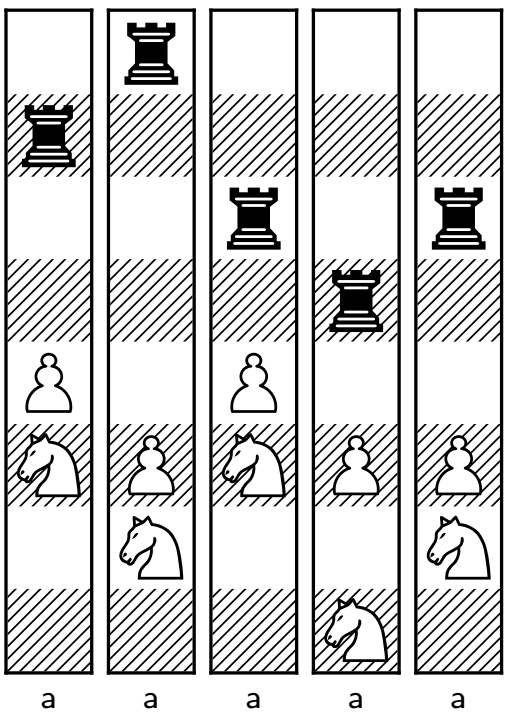
Axis encoding: experimento

Depiction	Feature set	Number of features
	H ⊕ V	192
	D1 ⊕ D2	360
	H ⊕ V ⊕ D1 ⊕ D2	552
	All ⊕ H ⊕ V	960
	All ⊕ D1 ⊕ D2	1128
	All ⊕ H ⊕ V ⊕ D1 ⊕ D2	1320

Axis encoding: resultados

Feature set	Number of features	Val. loss *min*	Rating *elo (rel. to* ALL*)*	Puzzles *move acc.*
	192	0.005810	-384.3 ± 5.1	0.8618
	360	0.006707	-444.1 ± 5.1	0.8517
	552	0.003907	-183.5 ± 4.1	0.8748
ALL (reference)	768	0.003134	**0.0**	0.8865
ALL ⊕ 	960	0.003082	-27.1 ± 4.1	0.8851
ALL ⊕ 	1128	0.003087	-26.1 ± 3.8	0.8814
ALL ⊕ 	1320	**0.003067**	-58.7 ± 3.7	0.8766

Pairwise axes: motivación



Configuraciones distintas, situaciones similares

Las mismas dos features (par rojo y par azul)

Pairwise axes: motivación

Comparando con el experimento anterior, es más específico en vez de más general:

"*there is a ○ White ♖ Rook in the 4th rank*"

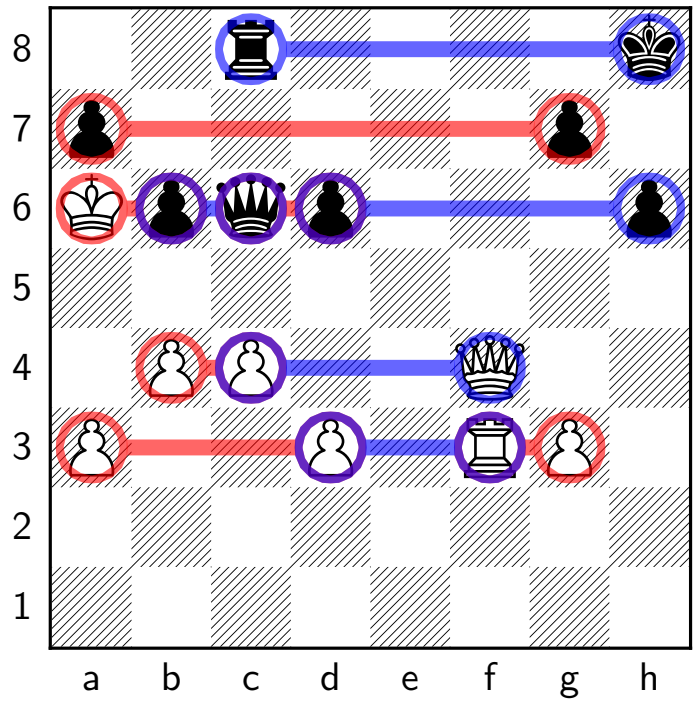
vs.

"*there is a ● Black ♖ Rook next to a ○ White ♙ Pawn in the 'a' file*"

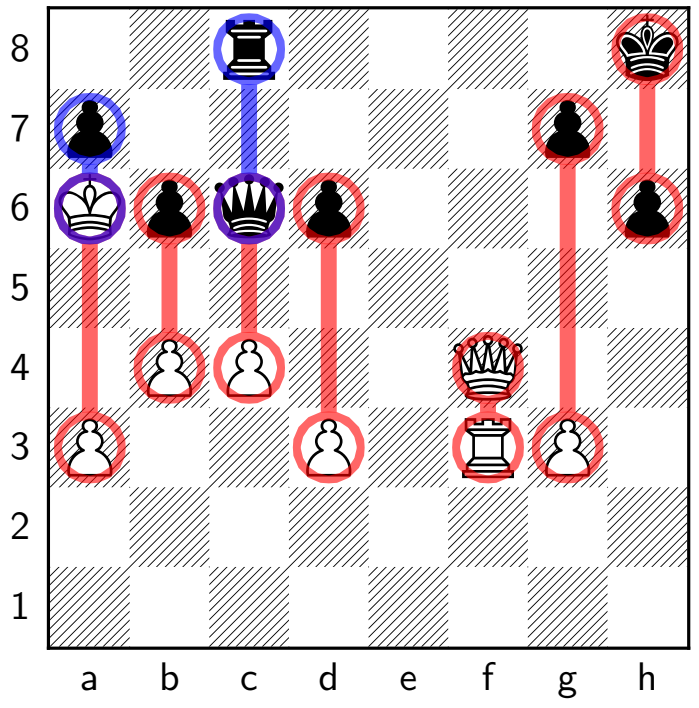
Pairwise axes: experimento

D.	Block name	Definition	Num. of features
	PH	$(\textsc{Ranks} \times (\textsc{Roles} \times \textsc{Colors}) \times (\textsc{Roles} \times \textsc{Colors}))_P$ $\mathrm{P}(\langle r, r_1, c_1, r_2, c_2 \rangle)$: there is a piece in rank r with role r_1 and color c_1 to the left of a piece with role r_2 and color c_2	1152
	PV	$(\textsc{Files} \times (\textsc{Roles} \times \textsc{Colors}) \times (\textsc{Roles} \times \textsc{Colors}))_Q$ $\mathrm{Q}(\langle f, r_1, c_1, r_2, c_2 \rangle)$: there is a piece in file f with role r_1 and color c_1 below a piece with role r_2 and color c_2	1152

Pairwise axes: experimento



Pairwise horizontal (PH)



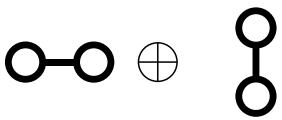
Pairwise vertical (PV)

Pairwise axes: experimento

Los feature sets a entrenar son:

- $\textsc{All} \oplus \mathrm{PH}$ (1920 features)
- $\textsc{All} \oplus \mathrm{PV}$ (1920 features)
- $\textsc{All} \oplus \mathrm{PH} \oplus \mathrm{PV}$ (3072 features)

Pairwise axes: resultados

Feature set	**Number of features**	**Val. loss** *min*	**Rating** *elo (rel. to* ALL*)*
ALL (reference)	768	0.003134	**0.0**
ALL ⊕ 	1920	0.003033	-38.2 ± 4.8
ALL ⊕ 	1920	0.002946	-8.4 ± 5.0
ALL ⊕ 	3072	**0.002868**	-37.6 ± 4.9

- Reducir el número de pairs puede llevar a una mejora por sobre ALL (ej. ♙)

Mobility: motivación

- La *mobilidad* en ajedrez es una medida de la cantidad de movimientos que puede hacer un jugador en una posición.
- Un paper de Eliot Slater (1950) mostró que hay una correlación entre la mobilidad de un jugador y la cantidad de partidas ganadas.
- Se usa en funciones de evaluación hechas a mano.
- Propongo agregar mobilidad como features en la red.

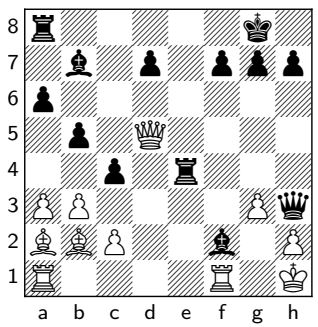
Mobility: experimento

Hay dos maneras de codificar la mobilidad:

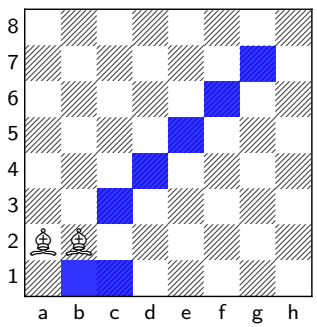
- Bitsets (por rol/color)
- Cantidades (por rol/color)

Mobility: experimento (bitsets)

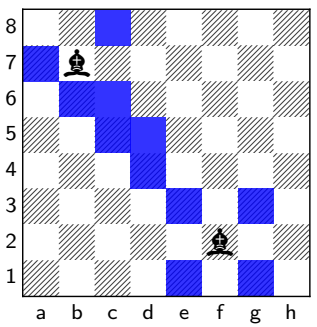
Los features proveen **las celdas** a las que una pieza de determinado rol/color puede moverse.



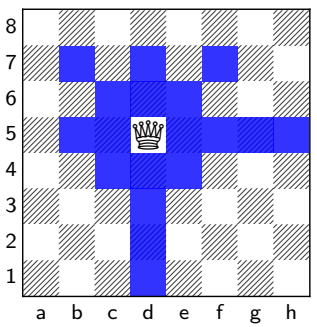
Board



○White
♗ Bishop



●Black
♗ Bishop



○White
♕ Queen

...

La cantidad de features es $64 \times 6 \times 2 = 768$, la misma que ALL.

Mobility: experimento (counts)

Los features proveen **la cantidad de celdas** a las que una pieza de determinado rol/color puede moverse. Esto reduce la cantidad de features significativamente.

Piece role	Min	Max
♙ Pawn	0	8+
♘ Knight	0	15+
♗ Bishop	0	16+
♖ Rook	0	25+
♕ Queen	0	25+
♔ King	0	8

Mobility: experimento (counts)

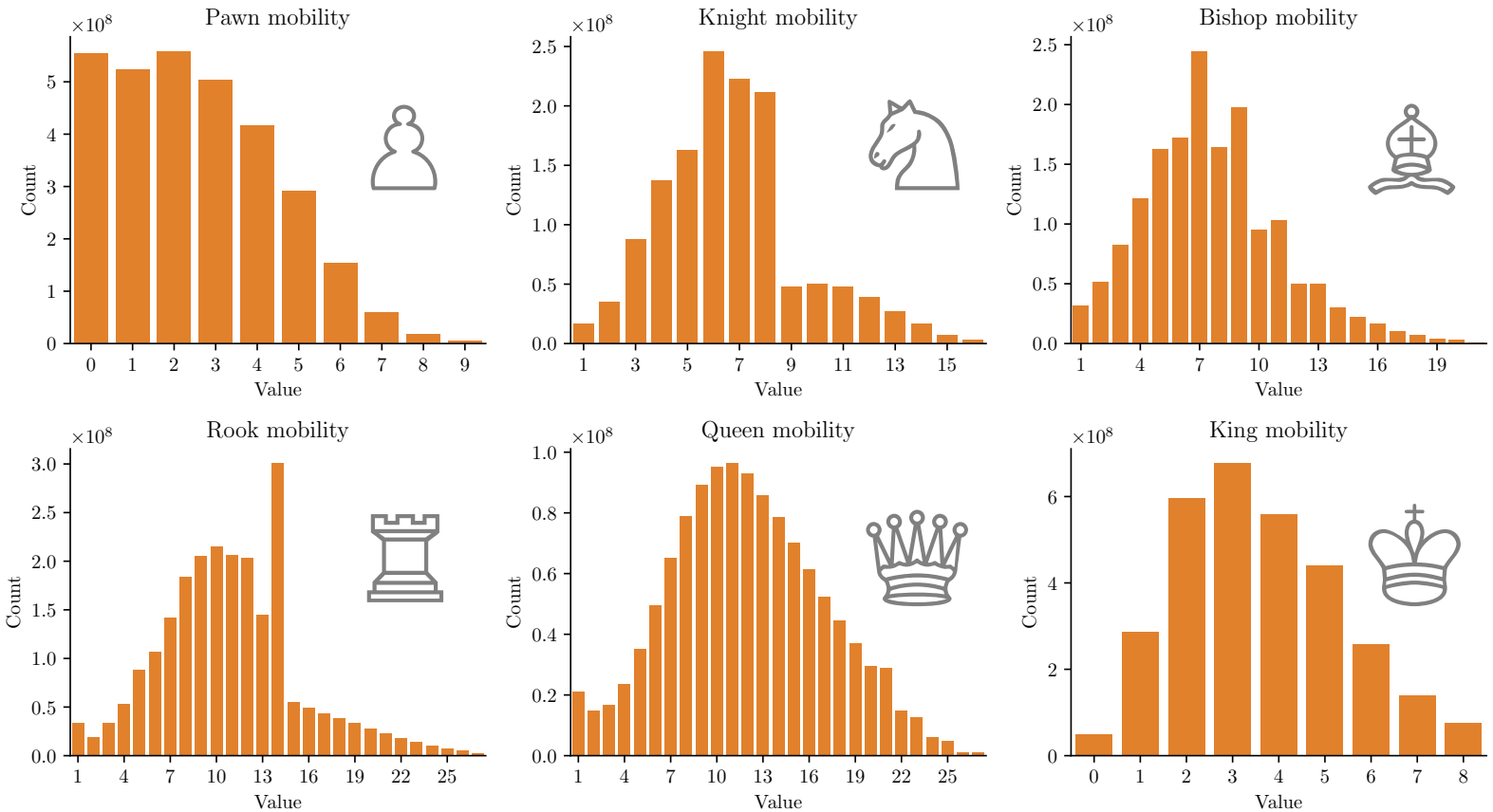


Figure: Total mobility values for each piece on the board. Computed using 2 billion boards. The value 0 for the ♘ Knight, ♗ Bishop, ♖ Rook,

Mobility: experimento

Block name	Definition	Number of features
MB	$(\textsc{Squares} \times \textsc{Roles} \times \textsc{Colors})_P$ P($\langle s, r, c \rangle$): there is a piece of role r and color c that **can move to** square s	768
MC	$(\{0, 1, \ldots\} \times \textsc{Roles} \times \textsc{Colors})_P$ P($\langle m, r, c \rangle$): the value of mobility for a piece of role r and color c is m	206

Los feature sets a entrenar son: $\textsc{All} \oplus \mathrm{MB}$ (1536 features) y $\textsc{All} \oplus \mathrm{MC}$ (974 features).

Mobility: resultados

Table: Mobility encodings results

Feature set	**Number of features**	**Val. loss** *min*	**Rating** *elo (rel. to* ALL*)*
ALL (reference)	768	0.003134	**0.0**
ALL $\oplus$ MB	1536	0.002824	-260.9 $\pm$ 5.4
ALL $\oplus$ MC	974	0.003032	-280.9 $\pm$ 5.6

- Las predicciones mejoran muy poco (el loss no se reduce tanto).
- Por ende, el costo de las actualizar los features es más alto al beneficio que aportan.
- MB tiene más updates que MC, pero menor loss que

Feature set statistics

Depiction	Feature block	Number of features	Average features...		
			active per position	added per move	removed per move
	All	768	14.68	0.98	0.60
	H	96	14.68	0.60	0.43
	V	96	14.68	0.61	0.43
	D1	180	14.68	0.77	0.52
	D2	180	14.68	0.77	0.52
	PH	1152	8.23	0.92	0.57
	PV	1152	8.30	0.83	0.53
MB	MB	768	48.93	5.68	4.35
MC	MC	206	12.00	2.34	1.48

PQR

:)

Conclusión

Ajedrez

- asdasd