Wpływ AlphaZero na rozwój silników szachowych

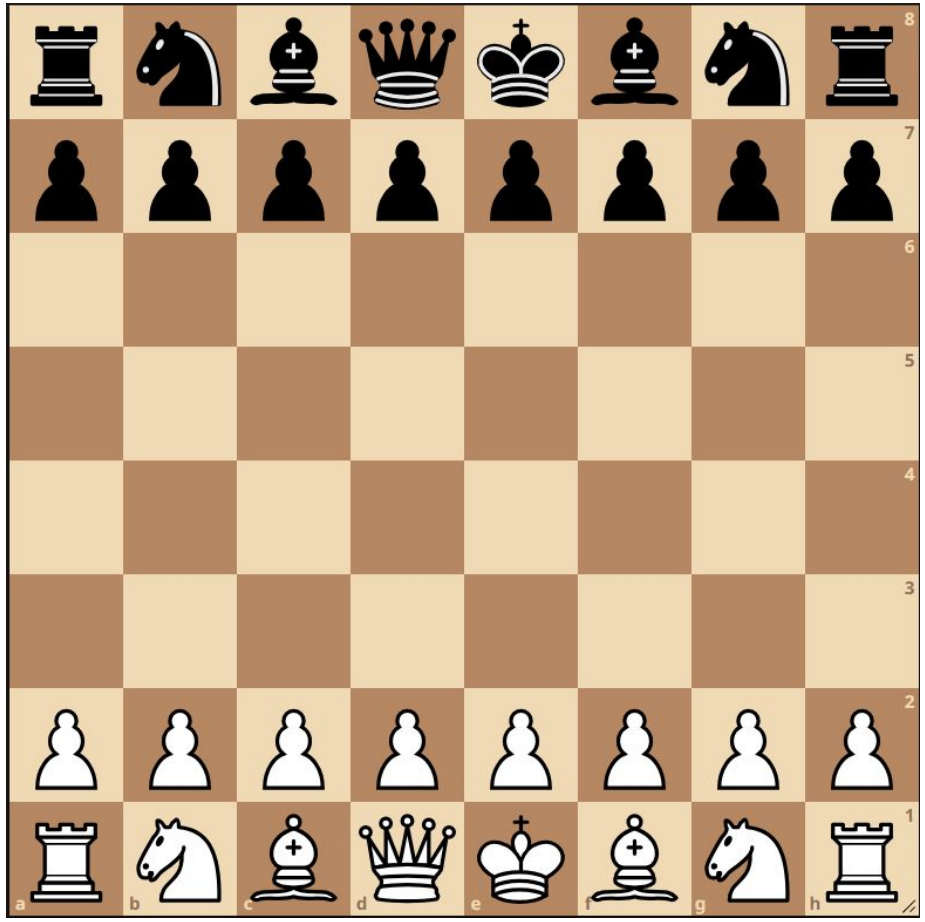
Plan prezentacji

- Szachy – krótkie przypomnienie
- Silniki szachowe przed AlphaZero
- Konstrukcja AlphaZero
- Silniki szachowe po AlphaZero

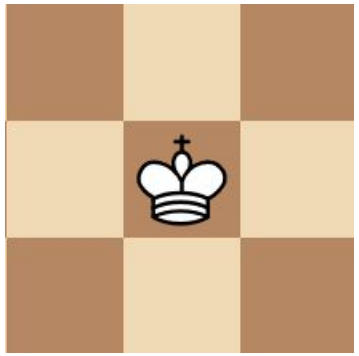
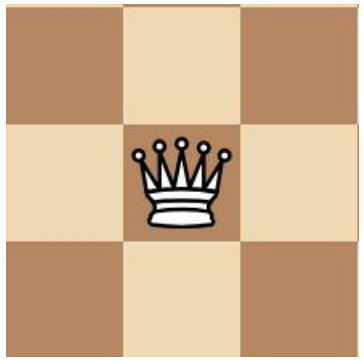
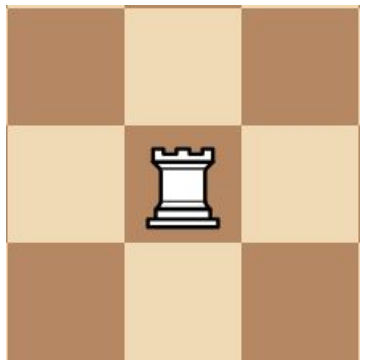
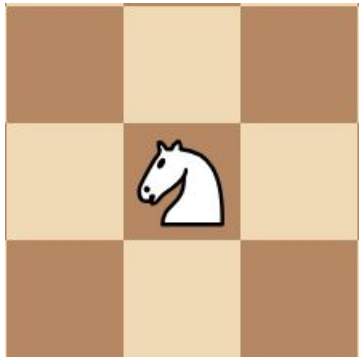
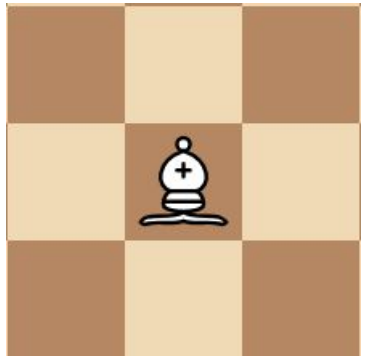
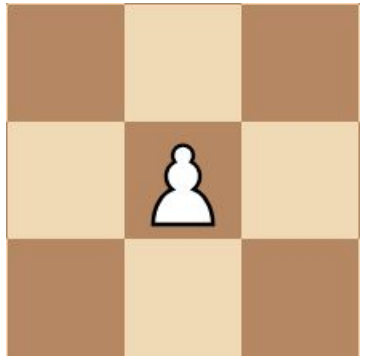
Szachy – krótkie przypomnienie

Zasady:

- 64 pola
- Gra na czas
- Zwycięstwo poprzez zamatowanie lub koniec czasu przeciwnika
- Podział teoretyczny na trzy części: debiut, gra środkowa, końcówka



Szachy – krótkie przypomnienie

Umowne punktowanie:

- Pionek – 1
- Skoczek/Goniec – 3
- Wieża – 5
- Hetman – 9
- Król – ?

Pre AlphaZero

- Statyczna ocena pozycji (HCE)
- Iteracyjne pogłębianie
- Biblioteka otwarć i tablice Nalimova
- Oparty o Alpha-Beta prunning
- Przeszukiwanie około 70 milionów pozycji na sekundę

Pre AlphaZero – ocena pozycji

- Ocena materiału
  - Efekty synergii - np. para gońców
- Tablice figura pole
- Struktura pionów
- Mobilność figur
- Bezpieczeństwo króla
- Przestrzeń (kontrolowane pola)

Tablica dla pionków

```
 0,  0,  0,  0,  0,  0,  0,  0,
50, 50, 50, 50, 50, 50, 50, 50,
10, 10, 20, 30, 30, 20, 10, 10,
 5,  5, 10, 27, 27, 10,  5,  5,
 0,  0,  0, 25, 25,  0,  0,  0,
 5, -5,-10,  0,  0,-10, -5,  5,
 5, 10, 10,-25,-25, 10, 10,  5,
 0,  0,  0,  0,  0,  0,  0,  0
```

Tablica dla skoczków

```
-50,-40,-30,-30,-30,-30,-40,-50,
-40,-20,  0,  0,  0,  0,-20,-40,
-30,  0, 10, 15, 15, 10,  0,-30,
-30,  5, 15, 20, 20, 15,  5,-30,
-30,  0, 15, 20, 20, 15,  0,-30,
-30,  5, 10, 15, 15, 10,  5,-30,
-40,-20,  0,  5,  5,  0,-20,-40,
-50,-40,-20,-30,-30,-20,-40,-50,
```

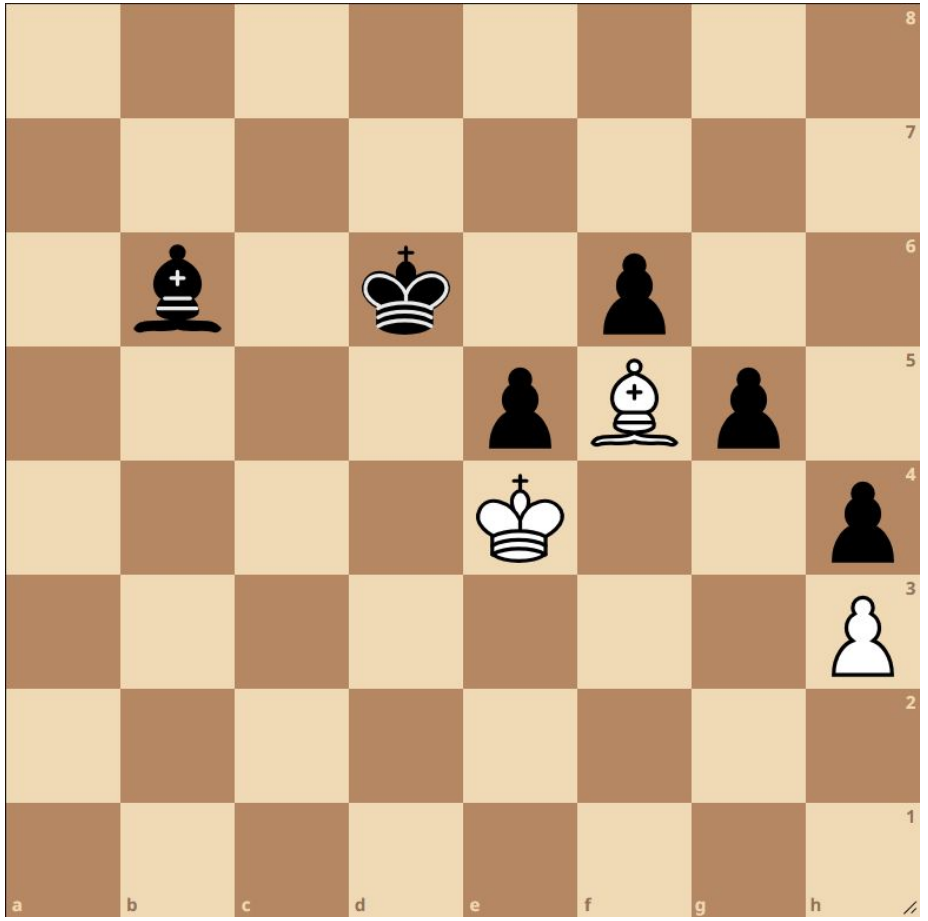
Problemy oceny pozycji (Stockfish)

Ocena pozycji przez silnik:

- Stockfish 11 (HCE): -1.4

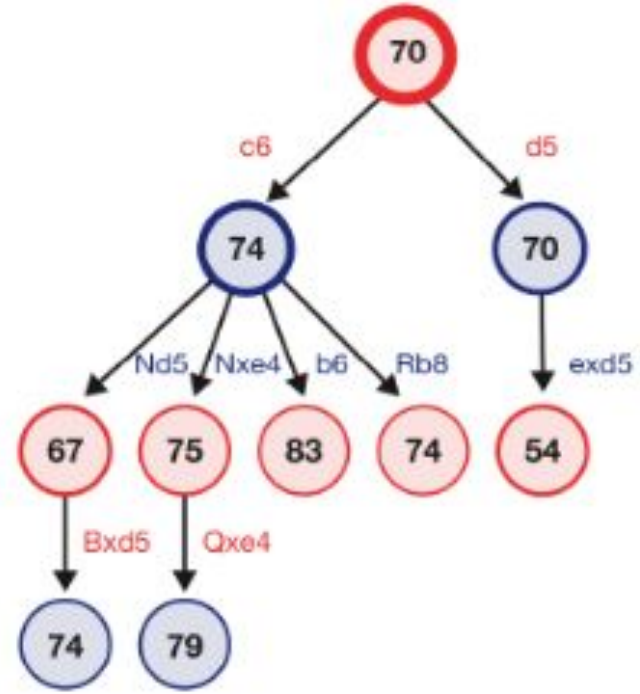
Ocena pozycji przez człowieka:

- Bardzo łatwy remis



AlphaZero

- Całkowita zmiana podejścia – oparcie o sieć neuronową
- Architektura ResNet
- Równoczesna ewaluacja pozycji i proponowanie kolejnych ruchów
- MCTS zamiast alpha-beta
- Brak biblioteki otwarć i końcówek
- Trenowany poprzez grę ze sobą
- Przeszukiwanie „zaledwie” 80 tysięcy pozycji na sekundę



AlphaZero vs. Stockfish

Stockfish:

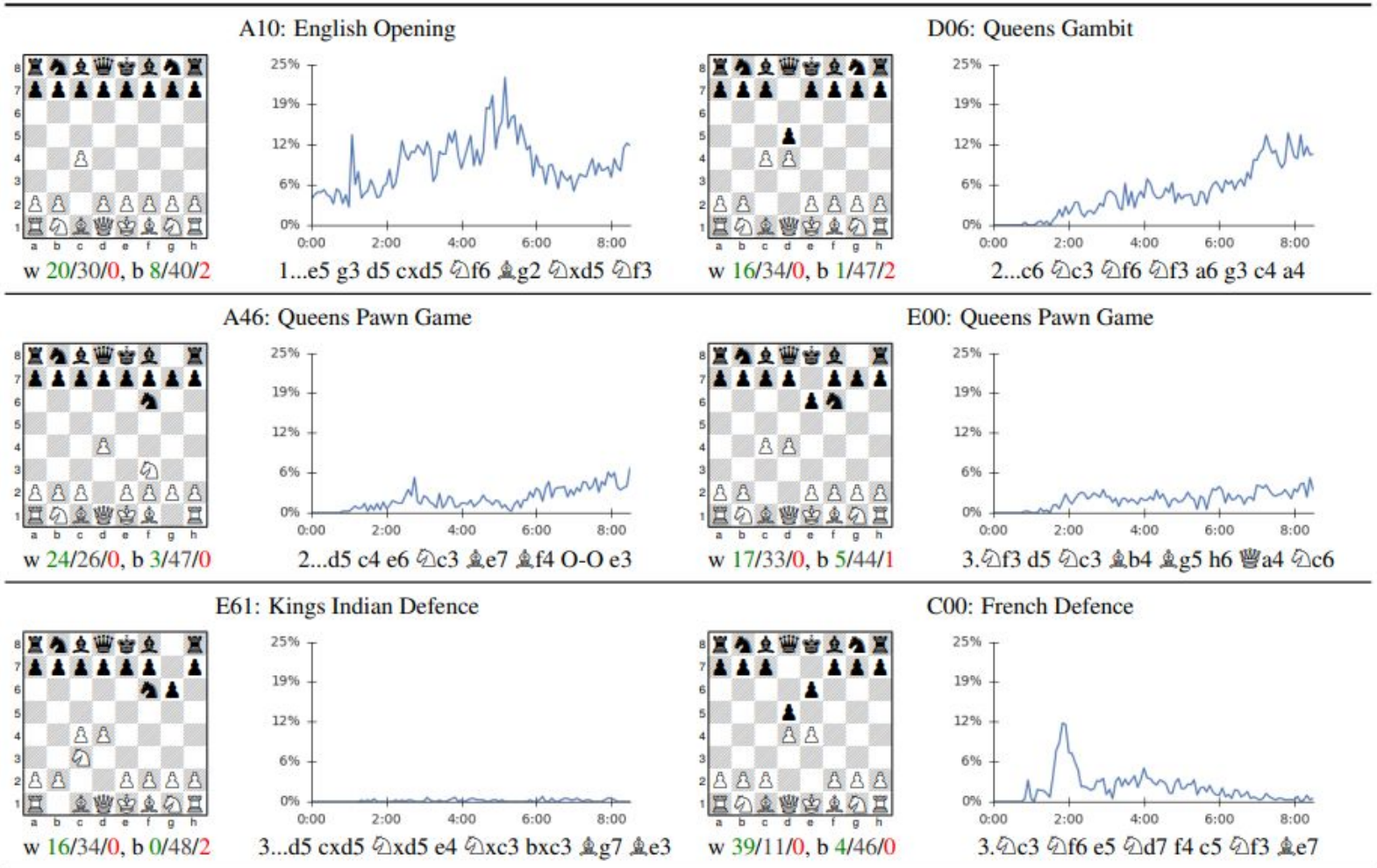
- 64 wątki
- 1 minuta na ruch

AlphaZero:

- 4TPU
- 1 minuta na ruch

Game	White	Black	Win	Draw	Loss
Chess	*AlphaZero*	*Stockfish*	25	25	0
	Stockfish	*AlphaZero*	3	47	0

AlphaZero vs. Stockfish



Lc0 – spadkobierca AlphaZero

- Kontynuowanie rozwoju według podejścia AlphaZero:
  - MCTS,
  - głęboka sieć neuronowa (oparta o ResNet),
  - trenowanie poprzez granie ze sobą

- Około 100 tysięcy pozycji na sekundę

Stockfish NNUE

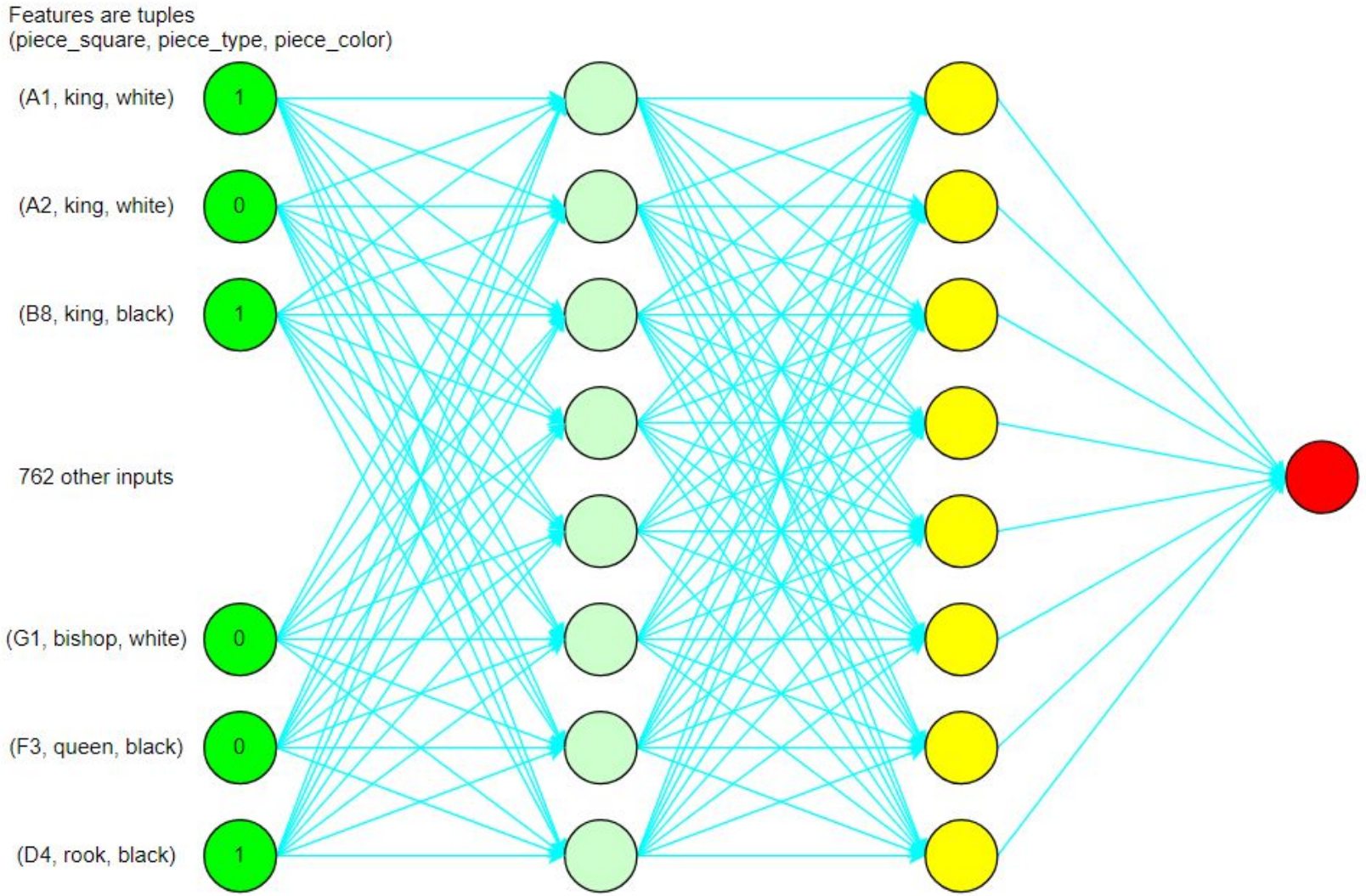
- Rozwój Stockfisha oparty o zdiagnozowanie problemu – niedokładne/niewystarczająco dobre ocenianie powstałej pozycji
- Idea zaczerpnięta z rozwoju algorytmów do Shogi – NNUE (Efficiently updatable neural network)
- Nadal wykorzystywane alfa-beta prunning
- NNUE wykorzystane do oceny pozycji (przewaga/strata wyrażona jako wartość bierek lub pp wygranej)

NNUE (Efficiently updatable neural network)

- Założenia
  - Jak najszybsze obliczanie ewaluacji (prostota modelu)
  - Możliwość działania na CPU
  - Niewielka liczba niezerowych wejść (korzystanie z rzadkich danych)
  - Jak największe podobieństwo kolejnych wejść
  - Wykorzystywanie najprostszych typów danych (kwantyzacja, obliczenia głównie na liczbach całkowitych)

NNUE – podstawowa reprezentacja wejścia

- Liczba wejść = 64 * 6 * 2 = 768
- Max liczba wejść niezerowych = 32
- Max liczba zmodyfikowanych wejść = 4



NNUE – podstawowa reprezentacja wejścia

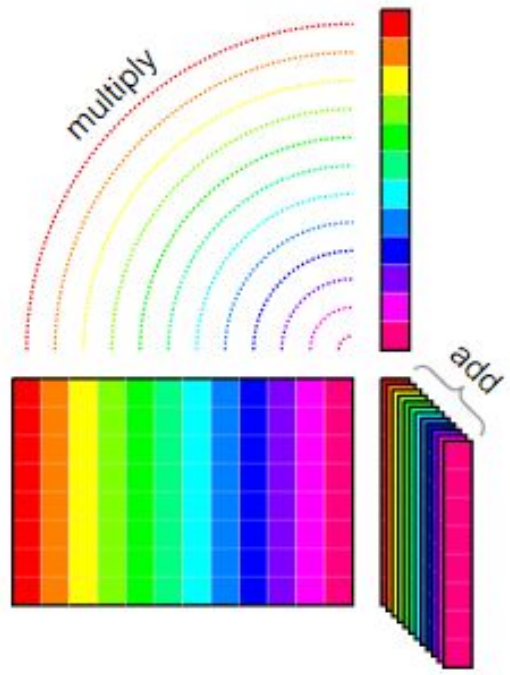
Dane wejściowe (4 niezerowe wejścia):

- (a1, król, biały)
- (c3, pionek, biały)
- (b8, król, czarny)
- (d4, wieża, czarny)

Ruch białych:

- Król – modyfikacja dwóch wejść
- Pionek – modyfikacja dwóch lub trzech wejść

NNUE - akumulator



- Stosowany do pierwszej ukrytej warstwy
- Następna warstwa: macierz wag × wektor wejścia
- Wektor wejścia – większość 0 i kilka 1
- Następna warstwa: suma kilku kolumn z macierzy wag

Modyfikacja wejścia:

- Usunięcie wejścia – pole, z którego bierka zaczynała lub bicie
- Usunięcie wejścia *i* –> odjęcie kolumny *i* od wyniku
- Dodanie wejścia – końcowe pole, na którym stanie bierka lub promocja (zamiana pionka na figurę)
- Dodanie wejścia *i* –> dodanie kolumny *i* do wyniku

NNUE - akumulator

Liczby zmiennoprzecinkowe:

- Wielokrotne dodawania i odejmowanie można spowodować błędy

Liczby całkowite:

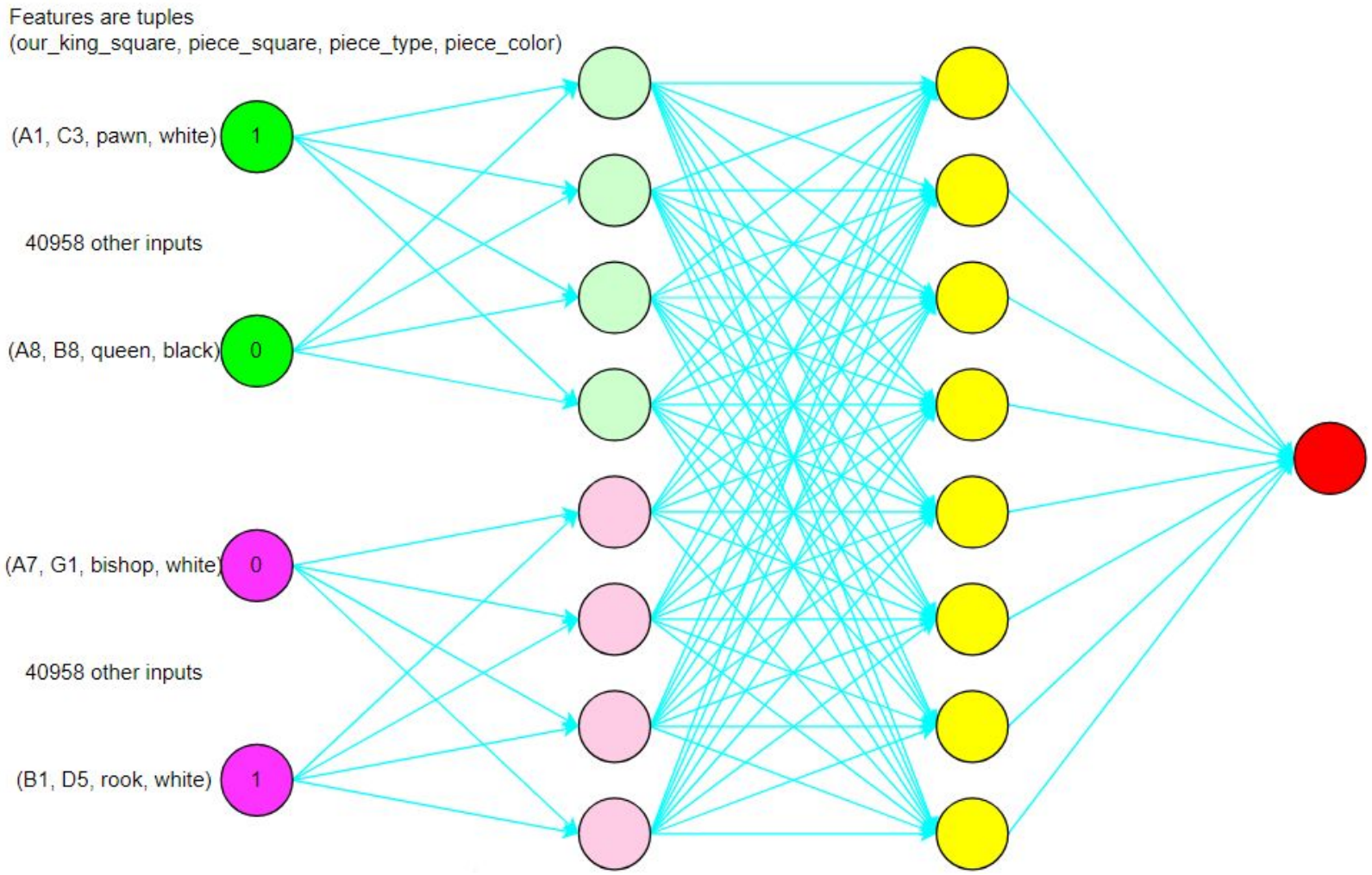
- Wymaga kwantyzacji
- Brak błędów dodawania i odejmowania

Wykorzystanie *stosu przeszukiwania*:

- wartości dla poszczególnych pozycji są zapisywane
- cofnięcie ruchu nie poprzez odejmowanie, a poprzez wybranie właściwej wartości ze stosu

NNUE - HalfKP

- Liczba wejść = 64 * 64 * 5 * 2 * 2 = 40960 * 2
- Max liczba wejść niezerowych = 60
- Max liczba zmodyfikowanych wejść = 30



NNUE – HalfKP

Dane wejściowe (4 niezerowe wejścia):

- (a1, c3, pionek, biały)
- (a1, d4, wieża, czarny)

- (b1, c6, pionek, czarny)
- (b1, d5, wieża, biały)

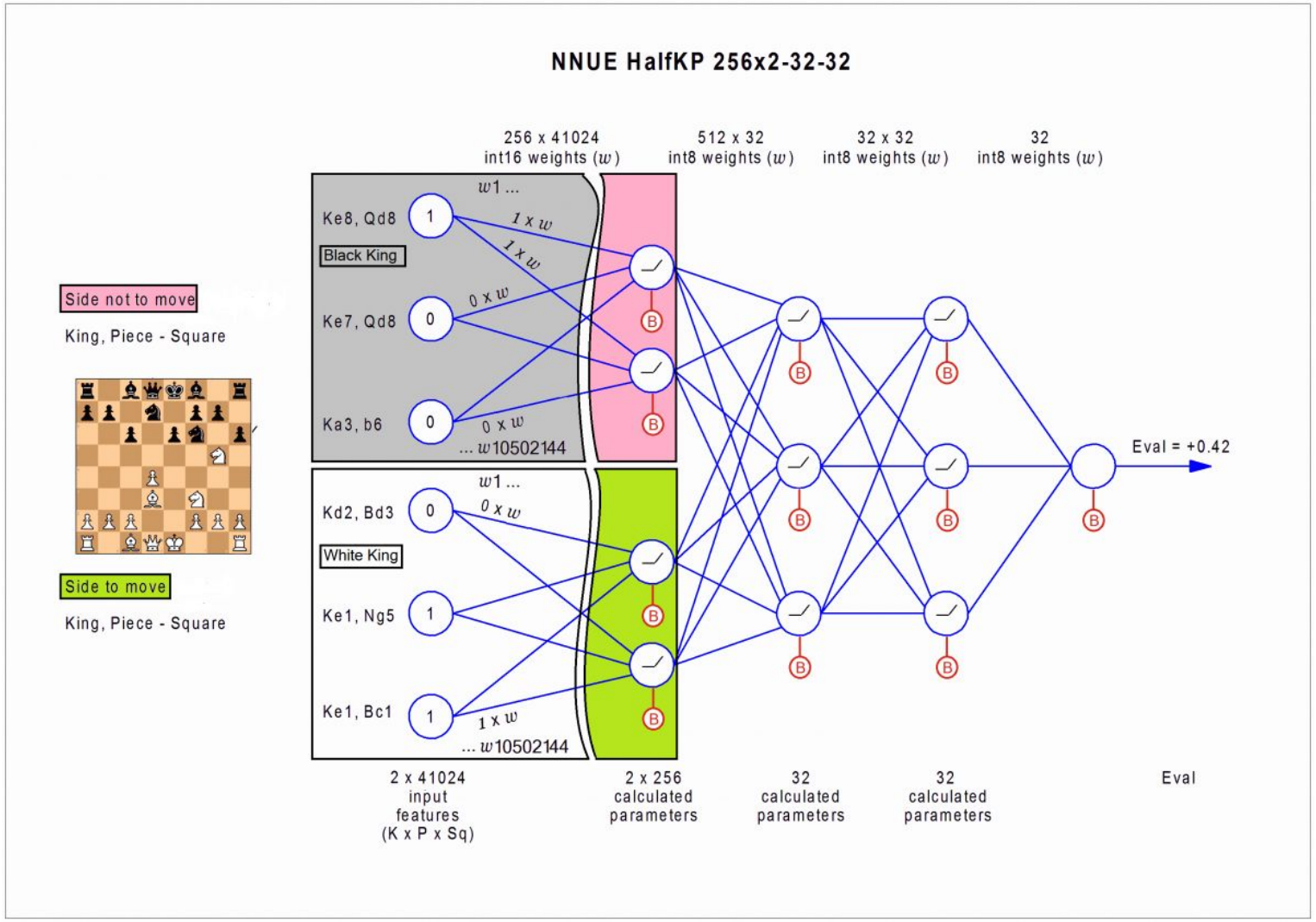
Ruch białych:

- Król – modyfikacja dwóch wejść
- Pionek – modyfikacja czterech lub sześciu wejść

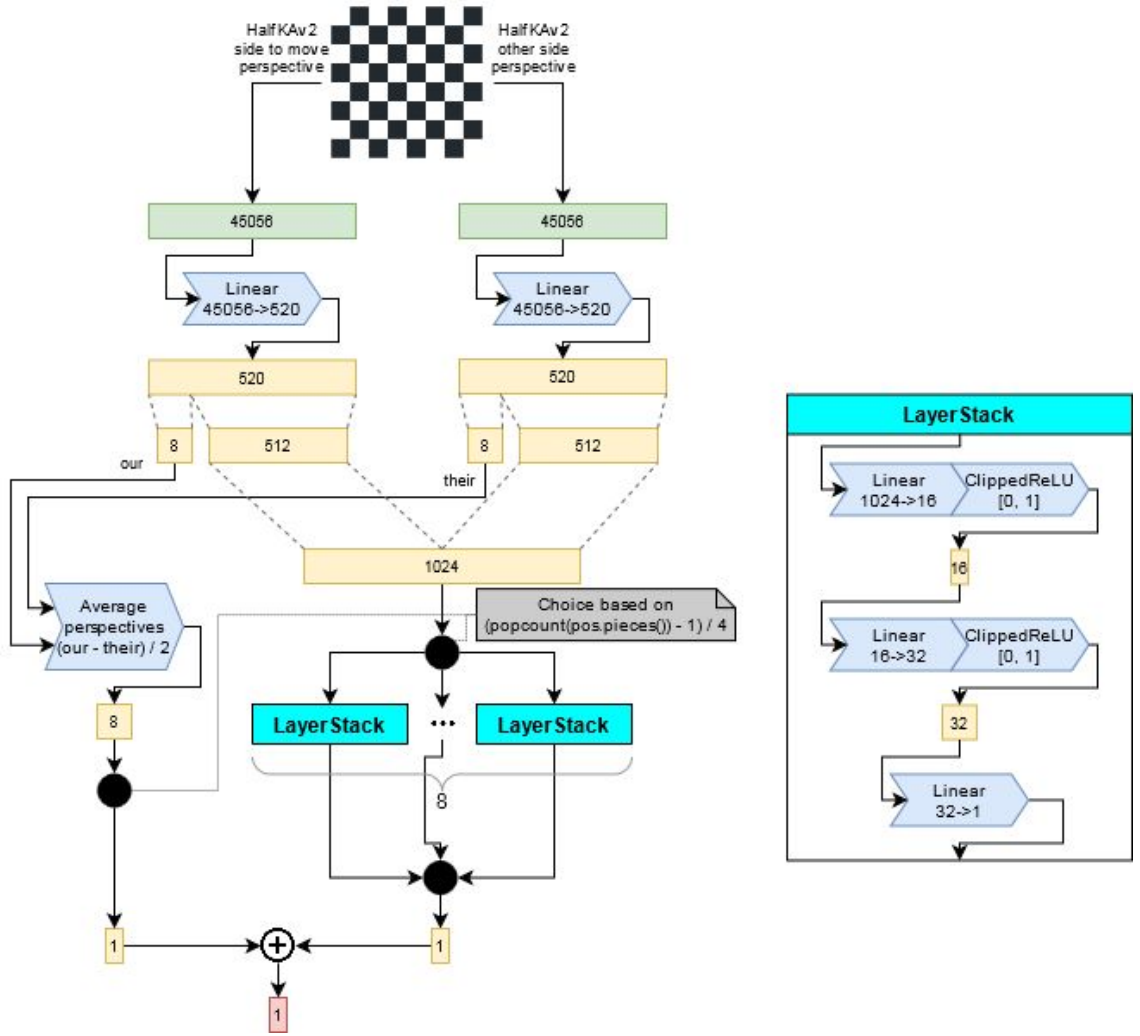
NNUE – modyfikacja akumulatora

- Dwa akumulatory – po jednym dla każdej perspektywy (brak uwspólniania wag)
- Połączenie akumulatorów przed następną warstwą:
  - Zestawienie *AwAb*
  - Zestawienie *AwAb*, jeśli ruch białych lub *AbAw*, jeśli ruch czarnych

NNUE – końcowa architektura



NNUE – końcowa architektura



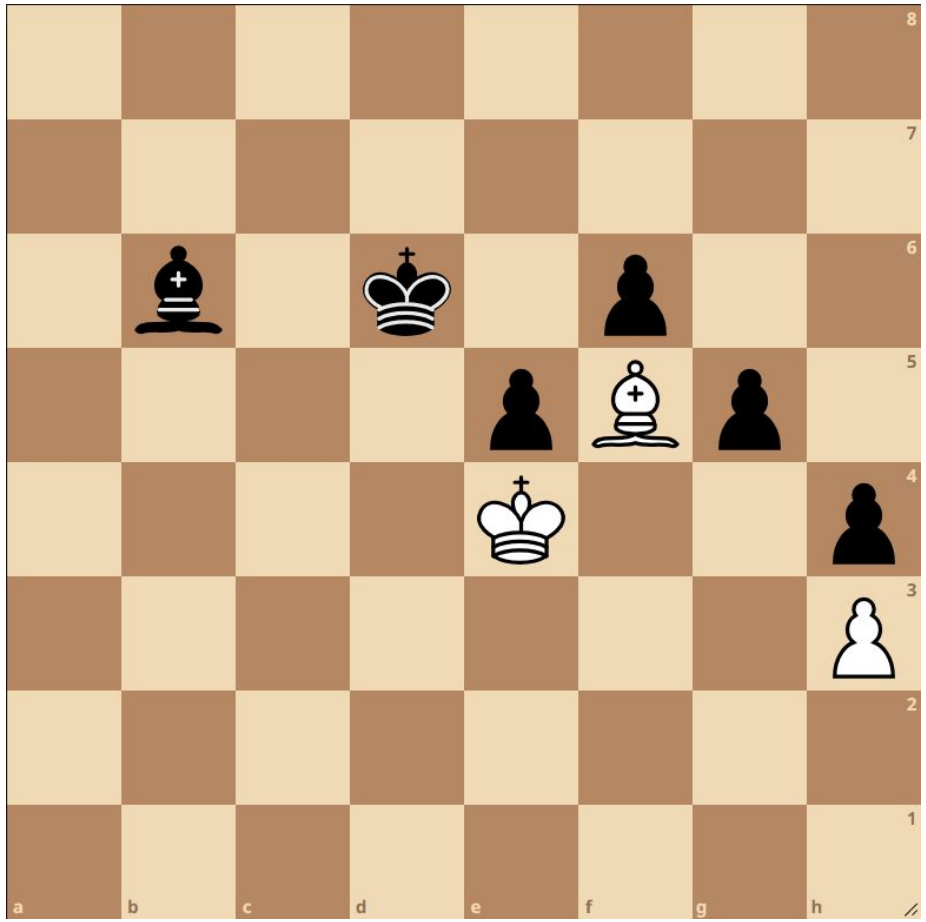
Stockfish HCE vs Stockfish NNUE

Ocena pozycji:

- Stockfish 11 (HCE): -1.4
- Stockfish 12 (NNUE): -0.2

Porównanie siły gry:

- Stockfish 11 vs Stockfish 8: +150 Elo
- Stockfish 12 (NNUE) vs Stockfish 11: +150 Elo



Stockfish NNUE vs Lc0

Pre NNUE

Season 15	2019	120+10	2nd
Season 16	2019	120+10	1st
Season 17	2020	90+5	2nd

Post NNUE

Season 18	2020	90+10	1st
Season 19	2020	120+10	1st
Season 20	2020	120+10	1st
Season 21	2021	120+10	1st
Season 22	2022	120+12	1st
Season 23	2022	120+12	1st
Season 24	2023	120+12	1st
Season 25	2023	120+12	1st

	Stockfish	Lc0
Season 15	46.5	53.5
Season 16	-	-
Season 17	47.5	52.5
Season 18	53.5	46.5
Season 19	54.5	45.5
Season 20	53	47
Season 21	56	44
Season 22	-	-
Season 23	58.5	41.5
Season 24	52	48
Season 25	52	48

Podsumowanie

- Wykorzystanie prostszego modelu może dać lepsze rezultaty niż wykorzystanie bardziej skomplikowanego modelu
- Stockfish NNUE aktualnie uważany za najlepszy (wygrywa turnieje komputerów)
- NNUE wydaje się bardzo dobrze działać przy grach podobnych do szachów (GO, Shogi)

Dziękuję za uwagę

Referencje

- Silver, David, et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm." arXiv preprint arXiv:1712.01815 (2017)
- Nasu, Yu. "Efficiently updatable neural-network-based evaluation functions for computer shogi." The 28th World Computer Shogi Championship Appeal Document 185 (2018).
- https://tcec-chess.com/
- https://github.com/official-stockfish/nnue-pytorch/blob/master/docs/nnue.md#what-is-nnue
- https://en.wikipedia.org/wiki/Stockfish_(chess)
- https://www.chessprogramming.org/Stockfish_NNUE