# A General Reinforcement learning algorithm that masters chess, shogi, and Go through self-play

* Chess became a grand challenge task for early artificial intelligence researchers, culminating in high-performance computer chess programs that play like a chess professional player.
* These systems are highly tuned to their domain and cannot be generalize to other games.
* A long-standing ambition of artificial intelligence has been to create programs that can instead learn for themselves from the first principles.
* In recent years the artificial intelligence’s knowledge and performance increase significantly, AlphaGo is an artificial intelligence system which achieve superhuman performance in the game of Go, it uses deep convolutional neural networks which are then trained solely by reinforcement learning from games of self-play.
* In this paper the researchers introduce a more generic version of the AlphaGo Zero algorithm which accommodates, without special casing, a broader class of game rules.
* The researchers apply the AlphaZero to the game chess and shogi and Go, using same algorithm and network architecture for all three games.
* Their result shows that a general-purpose reinforcement learning can learn, tabula rasa – without domain-specific human knowledge or data.
* Also prove that the same algorithm succeeds in multiple domains.
* In 1997 a landmark is achieved in the field of artificial intelligence when a AI system (Deep Blue) defeated the world chess champion in the game of chess.
* These programs evaluate positions by using handcrafted features and carefully tuned weights, constructed by strong human players and programmers, combined with a high-performance alpha-beta search that expands a vast search tree by using a large number of clever heuristics and domain-specific adaptations.
* Shogi is much complex game compare to chess and the strongest shogi programs, like 2017 Computer Shogi Association (CSA) defeated the world champions.
* These programs are also based on same algorithm of chess and optimized on alpha-beta search engine with many domain-specific adaptations.
* AlphaZero replaces the handcrafted knowledge and domain-specific augmentations used in traditional game-playing programs with deep neural networks, a general-purpose reinforcement learning algorithm, and a general-purpose tree search algorithm.
* AlphaZero uses a deep neural network (**p**, *v*) = with parameters . In this neural network takes the board position *s* as an input and outputs a vector of move probabilities **p** with the components for each action *a* and a scalar value *v* estimating the expected outcome *z* of the game from position *s, v* .
* AlphaZero uses a general-purpose Monte Carlo tree search (MCTS) algorithm instead of alpha-beta search with domain-specific enhancements.
* In this method each search consists of a series of simulated games of self-play that traverse a tree from root state until a leaf state is reached.
* Each simulation proceeds by selecting in each state *s* a move *a* with low visit count, high move probability, and high value according to the current neural network . The search returns a vector representations a probability distribution over moves, .
* The parameters of the deep neural network in AlphaZero are trained by reinforcement learning from self-play games, starting from randomly initialized parameters .
* Each game is played by running an MCTS from the current positions at turn *t* and then selecting a move, either proportionally with respect to the visit counts at the root state.
* The end of the game outcome z: -1 for a loss, 0 for a draw, and +1 for a win.
* The neural network parameters are updated to minimize the error outcome z and to maximize the error between the predicted outcome and the game outcome z and to maximize the similarity of the policy vector **p**t to the search probabilities .
* The parameters are adjusted by gradient descent on a loss function *l* that sums over mean-squared error and cross-entropy losses



* Here *c* is a parameter controlling the level of *l2* weight regularization. The updated parameters are used in subsequent games of self-play.
* This paper describes alpha zero is different from the original AlphaGo zero algorithm.
* AlphaGo Zero estimated and optimized the probability of wining, exploiting the fact that Go games have binary win or loss outcome.
* The rules of Go are invariant to rotation and reflection. This fact was exploited I AlphaGo and AlphaGo Zero in two ways.
  + First, the training data were augmented by generating eight symmetries for each position.
  + Second, during MCTS, board position was transformed by using a randomly selected rotation or reflection before being evaluated by the neural networks, so that the Monte Carlo evaluation was averaged over different biases.
* To accommodate a broader class of games AlphaZero does not assume symmetry; the rules of chess and shogi are asymmetric.
* In AlphaGo Zero, self-play games were generated by the best player from all previous iterations. After each iteration of training, the performance of the new player was matched against the best player.
* If the new player won and the margin is above than 55% then it replace the best player.
* By contrast, AlphaZero simply maintains a single neural network that is updated continually rather than waiting for an iteration to complete.
* AlphaGo Zero used a convolutional neural network architecture that is particularly well suited to Go: the rules of the game are translationally invariant and are defined in terms of libraries corresponding to the adjacencies between points on the board.
* On the other hand, the rules of chess and shogi are position dependent and include long-range interactions.
* Even with these differences the AlphaZero uses the same convolutional network architecture as AlphaGo Zero for chess, shogi, and Go.
* The hyperparameters of AlphaGo Zero were tuned by Bayesian optimization. In AlphaZero the researchers use the same hyperparameters, algorithm settings, and network architecture for all games without game-specific tuning.
* The differences are the exploration noise and the learning rate schedule.
* They trained separate instances of AlphaZero for chess, shogi, and Go.
* Training proceeded for 700,000 steps starting from randomly initialized parameters. During the training only 5,000 first-generation tensor processing units were used to generate self-play games, and 16 second-generation TPUs were used to train the neural networks.
* In chess, AlphaZero first outperformed Stockfish after just 4 hours (300,000 steps); in shogi, AlphaZero first outperformed Elmo after 2 hours (110,000 steps); and in Go, AlphaZero first outperformed AlphaGo Lee after 30 hours (74,000 steps).
* The researchers evaluated the fully trained instances of AlphaZero against Stockfish, Elmo, and the previous version of AlphaGo Zero in chess, shogi, and Go, respectively.
* In Go, AlphaZero defeated AlphaGo Zero, wining 61% of games. It demonstrates that a general approach can recover the performance of an algorithm that exploited board symmetries to generate eight times as much data.
* In chess the AlphaZero defeated the Stockfish, wining 155 games and losing just 6 games out of 1000. To cross check its robustness the researchers played additional matches that started from common human openings.
* The AlphaZero defeated the Stockfish every opening of this, it suggests that AlphaZero has mastered a wide spectrum of chess play.
* The researchers also played a match that started from the set of opening positions used in the 2016 TCEC world championship; AlphaZero won convincingly in this match also.
* In several games, AlphaZero sacrificed pieces for long-term strategic advantages, suggesting that it has a more fluid, context-dependent positional evaluation than the rule-based evaluation used by previous chess programs.
* In Shogi, AlphaZero defeated Elmo, wining 98.2% of games when playing black and 91.2% overall.
* The researchers also play a match under the faster time controls used in the 2017 CSA world championship and against another state-of-the-art shogi program; AlphaZero again won both matches by a wide margin.
* AphaZero searchers just 60,000 positions per second in class and shogi, compared with 60 million for Stockfish and 25 million for Elmo.
* The researchers think that AlphaZero may compensate for the lower number of evaluations by using its deep neural network to focus much more selectively on the most promising variations arguably a more humanlike approach to searching, as originally proposed by Shannon.
* The high performance of AlphaZero with the use of MCTS calls into question the widely held belief that alpha-beta search is inherently superior in these domains.
* AlphaZero is a generic reinforcement learning and search algorithm – originally devised for the game of Go – that achieved superior results within a few hours, searching 1/1000 as many positions, given no domain knowledge except the rules of chess.