The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 17, No. 5, pp.119-124, Oct. 31, 2017. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2017.17.5.119
JIIBC 2017-5-16

딥퍼플 : 딥러닝을 이용한 체스 엔진

DeepPurple : Chess Engine using Deep Learning

김성환*, 김영웅**

Sung-Hwan Yun*, Young-Ung Kim**

요 약 1997년 IBM의 딥블루가 세계 체스 챔피언인 카스파로프를 이기고, 최근 구글의 알파고가 중국의 커제에게 완승을 거두면서 딥러닝에 대한 관심이 급증하였다. 본 논문은 딥러닝에 기반을 둔 인고지능 체스엔진인 딥퍼플(DeepPurple) 개발에 대해 기술한다. 딥퍼플 체스엔진은 크게 몬테카를로 트리탐색과 컨볼루션 신경망으로 구현된 정책망 및 가치망으로 구성되어 있다. 딥러닝을 통해 구축된 정책망을 통해 다음 수를 예측하고, 가치망을 통해 주어진 상황에서의 판세를 계산한 후, 몬테카를로 트리탐색을 통해 가장 유리한 수를 선택하는 것이 기본 원리이다. 학습 결과, 정책망의 경우 정확도 43%, 손실함수 비용 1,9로 나타났으며, 가치망의 경우 정확도 50%, 손실함수 비용 1점대에서 진동하는 것으로 나타났다.

Abstract In 1997, IBM's DeepBlue won the world chess championship, Garry Kasparov, and recently, Google's AlphaGo won all three games against Ke Jie, who was ranked 1st among all human Baduk players worldwide, interest in deep running has increased rapidly. DeepPurple, proposed in this paper, is a AI chess engine based on deep learning. DeepPurple Chess Engine consists largely of Monte Carlo Tree Search and policy network and value network, which are implemented by convolution neural networks. Through the policy network, the next move is predicted and the given situation is calculated through the value network. To select the most beneficial next move Monte Carlo Tree Search is used. The results show that the accuracy and the loss function cost of the policy network is 43% and 1.9. In the case of the value network, the accuracy is 50% and the loss function cost is 1, respectively.

Key Words : Deep Learning, Chess Engine, Policy Network, Value Network, Monte Carlo Tree Search

Ⅰ. 서 론

1997년, IBM에서 체스의 모든 경우의 수를 컴퓨터로 계산한 체스 프로그램인 딥블루가 러시아 체스 챔피언이 카스파로프에게 승리했다. 2016년 구글 딥마인드가 딥러닝을 이용한 바둑 인공지능 프로그램 알파고로 세계 바둑 챔피언인 이세돌 9단과 중국 바둑기사 커제와의 대국을 승리함으써 딥러닝에 대한 관심을 집중시켰다.

본 논문은 구글 딥마인드의 알파고의 원리를 바탕으로 딥러닝을 이용한 인공지능 체스 엔진인 딥퍼플에 대해 기술한다. 딥퍼플 체스 엔진은 크게 몬테카를로 트리탐색과 컨볼루션 신경망Convolutional Neural Network: CNN)[1] 구조를 갖는 정책망, 가치망으로 구성되어 있다.

정책망(Policy Network)은 주어진 체스판에서 체스판

*준회원, 한성대학교 컴퓨터공학과
**정회원, 한성대학교 컴퓨터공학부(교신저자)
접수일자: 2017년 8월 16일, 수정완료: 2017년 9월 15일
게재확정일자: 2017년 10월 13일

Received: 16 August, 2017 / Revised: 15 September, 2017 / Accepted: 13 October, 2017
**Corresponding Author: yukim@hansung.ac.kr
Dept. of Computer Engineering, Hansung University, Korea

에 배치된 말들의 상태가 압력으로 주어지면, 가능한 수들과 함께 그 수를 선택할 확률을 계산하여 가장 확률이 높은 수를 예측하는 모델이다.

가치망(Value Network)은 체스판의 상황이 입력으로 주어지면 결과 값으로 백승과 흑승 그리고 무승부가 될 확률을 반환하여 누가 얼마나 유리한 상황인지를 예측하는 모델이다. 가치망은 정책망과 달리 승과 패에 대한 결과를 학습에 반영함으로써 정책망으로부터 예측 받은 수가 실제로 유리한 수인지를 판단하는 기준이 된다.

정책망과 가치망을 완성하면, 이 두 개의 모델만으로도 어느 정도의 패턴을 파악한 결과를 알 수 있지만, 딥퍼플의 경우 충분한 성능을 확보하기 위해 몬테카를로 트리 탐색(Montecarlo tree search: MCTS)알고리즘[2]을 사용한다. MCTS의 기본 원리는 동일한 상황에서 수많은 시뮬레이션을 반복함으로써 결과적으로 가장 좋은 선택을 찾는 것이다. 이 기본 원리를 바탕으로 딥퍼플의 시뮬레이션은 트리 탐색 방법으로서 선택(selection), 확장(expansion), 시뮬레이션, 역전달(back propagation) 총 4단계를 통해 이루어진다. 통계적으로 유의한 횟수만큼의 시뮬레이션을 통해 루트노드에서 선택 가능한 수들 중에서 가장 가치가 높은 자식노드를 최종 선택한다.

본 논문의 구성은 다음과 같다. 제 2장에서는 관련 연구에 대해 기술하고, 제 3장에서는 본 논문의 중심이 되는 딥퍼플 체스 엔진의 시스템 및 소프트웨어 구조에 대해 기술한다. 제 4장에서는 딥퍼플의 구현 환경과 정책망과 가치망의 학습 결과에 대해 기술하고, 끝으로 제 5장에서 결론 및 향후 연구과제에 대해 기술한다.

II. 관련 연구

기존에 딥러닝에 사용한 대표적인 체스 엔진으로는 CNN을 사용한 ConvChess[3]와 완전 연결 신경망을 사용한 Giraffe[4]가 있다.

ConvChess에서는 어떤 종류의 말이 움직이는 것이 좋은지 선택하는 CNN과, 앞에서 선택된 말이 어디로 움직이는지 좋은지에 대한 점수를 측정하는 CNN으로 구성되며, 이 두 CNN을 이용해서 체스게임에서 최적의 수를 예측한다.

ConvChess에서 사용한 데이터로는 Elo rating[5]이 2000점 이상인 상대들과 2만 게임을 하면서 245,000개의 체스말 이동을 통해 CNN을 학습시켰다. ConvChess의 최적의 말을 선택하는 CNN 모델의 가장 높은 정확도는 38.3%이며, 선택된 말의 최적의 이동할 위치를 선택하는 CNN 모델의 정확도는 말의 종류에 따라 27% ~ 56% 범위의 정확도를 가진다.

Giraffe 엔진의 특징은 수정된 TD-Leaf(λ)[6] 알고리즘과 확률에 기반을 둔 탐색(probability-based search)을 이용하여 체스 엔진의 성능을 높이는데 있다. 매 학습 반복 중 학습 데이터에서 무작위로 256의 체스 상황을 추출하여, 스스로 12번 순서대로 다음 수를 두고, 12번 모두의 움직임에 대한 체스 상황에 대한 점수 변화를 저장하여, 모든 점수 변화를 축적하면서 탐색 시작점에 추가함으로써, 평가함수를 학습시킬 때, 장기적인 결과에 대한 패턴을 모델화하였다.

또한, Giraffe는 minimax를 수행할 때 확률한계탐색(probability-limited search)이 깊이한계탐색(depth-limited search) 보다 더 정확하다고 판단하여, 확률한계탐색을 이용한 트리탐색을 제안하였으며, 주어진 체스 게임 상황에서, 미리 수를 내다보지 않고, 최적의 수가 되는 말의 움직임에 대한 각각의 개연성에 대해서 점수를 측정하였다. Giraffe는 대략 FIDE 세계 랭킹[7] 국제 마스터 레벨의 체스 실력을 갖추고 있다.

III. 딥퍼플 시스템 구조

1. 시스템 구조

그림 1은 딥퍼플 시스템 구조도를 보여준다. 딥퍼플의 전체 시스템 구조는 GUI 부분으로 이루어진 언리얼 엔진(Unreal Engine)[8] 부분과 파이썬 체스엔진 부분으로 구성되어 있다. 언리얼 엔진은 그래픽과 블루프린트 언어로 체스의 기본 로직을 구성한다. 블루프린트 체스 로직과 파이썬으로 구성된 딥퍼플 체스엔진과 통신은 8자리 정수로 이루어지며 앞의 4자리는 명령어를 의미하고, 뒤의 4자리 정수는 말의 시작좌표와 도착좌표를 의미한다. 게임 모드는 인간 vs 체스엔진, 체스엔진 vs 체스엔진의 2가지 모드가 제공되며, 언리얼 엔진과 파이썬 체스 엔진 각각 현재 체스판을 가지고 있다. 게임의 실질적인 로직은 파이썬 체스 엔진을 통해 진행되고, 그래픽으로 보여주는 현재 체스판의 상황은 블루프린트 체스 프로그램 안에 있다.

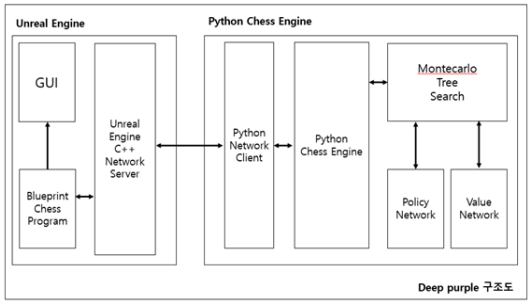

그림 1. 딥퍼플 시스템 구조도
Fig. 1. Deep Purple System Architecture

파이썬 체스 엔진에서 시시각각 변화하는 현재의 체스판을 몬테카를로 트리 탐색 모듈로 넘겨주면, 현재의 체스판을 0과 1로 구성된 특징들로 변환하여 입력 데이터를 만들고, 이전에 먼저 학습된 가중치 값들을 텐서플로의 saver를 통해 불러온다. 입력 데이터와 불러온 가중치 값을 이용해 비동기적으로 정책망과 가치망으로부터 값을 전달 받아 몬테카를로 트리 탐색을 진행하고, 기존에 미리 정해 놓은 트리 탐색의 깊이, 시뮬레이션 횟수에 따라 시뮬레이션을 진행한다. 모든 시뮬레이션 완료되면 루트 노드의 자식 노드 중 가장 높은 방문횟수를 가진 노드를 가장 좋은 다음 수로 판단해 해당 노드가 가진 체스 명령어를 다음 수로 선택한다. 선택된 다음 수는 체스 엔진과에 블루프린트 체스 프로그램에 전달되어 현재 체스 상황에 적용시킨다. 이 후 위와 같은 과정을 반복하여 게임을 진행한다.

정책망은 현재 주어진 체스판에서 이동 가능한 모든 경우의 수인 4096개의 실수 배열로 결과를 반환한다. 이때 4096개의 실수 배열은 각각의 경우의 수를 softmax를 이용해 0~1사이의 확률 값을 가지며, 각 요소를 모두 합치면 1이 된다.

가치망은 현재 주어진 체스판에 대해서 체스판의 가치를 판단한다. 이때의 가치는 주어진 체스판에서 승리할 확률을 의미한다. 가치망은 주어진 데이터들의 결과로 승, 무, 패를 학습하고, 주어진 상황에 대해서 정책망과 마찬가지로 softmax 함수를 이용해 승, 무, 패의 확률을 예측한다.

2. Py-ChessGame 구조

그림 2는 Py_ChessGame의 클래스 구조도를 나타낸다.

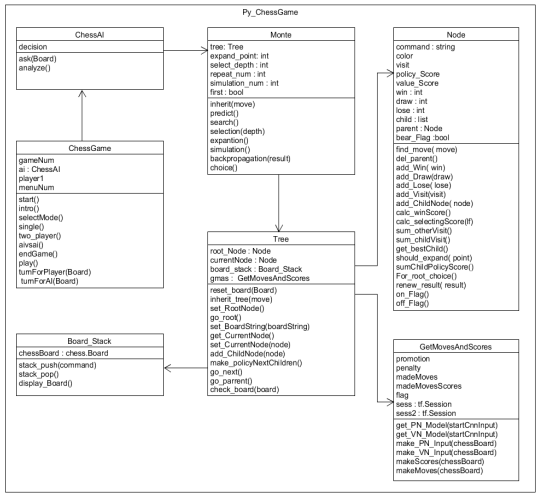

그림 2. Py_ChessGame 클래스 구조도
Fig. 2. Py_ChessGame Class Architecture

Py_ChessGame은 기존에 학습된 정책망과 가치망을 바탕으로 게임을 진행하는 부분으로 ChessGame클래스를 통해 게임이 진행된다. ChessAI의 ask()함수를 호출하면 Monte 클래스에서 탐색 깊이와 횟수인 select_depth, simulation_num을 통해 predict() 메서드를 호출하고, MCTS의 과정인 탐색, 선택, 확장, 역전달의 메서드가 호출되어 탐색을 진행한다. 마지막에 choice() 메서드를 통해 Tree 클래스 root_Node의 자식 노드 중에서 가장 높은 방문횟수를 가진 노드를 선택한다.

Tree 클래스는 MCTS를 진행하면서 만들어지는 트리를 관리하는 클래스이다. 트리는 Node 클래스로 구성되어 있으며, Node 클래스는 부모 노드, 자식노드, 승/무/패 횟수, 색상, 점수들의 정보를 가진다.

GetMovesAndScores 클래스는 정책망과 가치망의 점수를 비동기적으로 얻기 위해 사용하는 클래스로 get_PN_Model(), get_VN_model() 메서드를 통해 각각의 텐서플로의 그래프를 생성하고 makeScores()와 makeMoves() 메서드를 통해 정책망, 가치망의 결과 값을 얻는다.

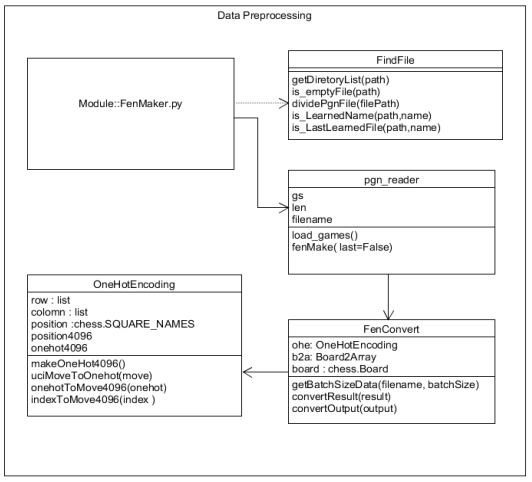

그림 3. Data Processing 클래스 구조도
Fig. 3. Data Processing Class Architecture

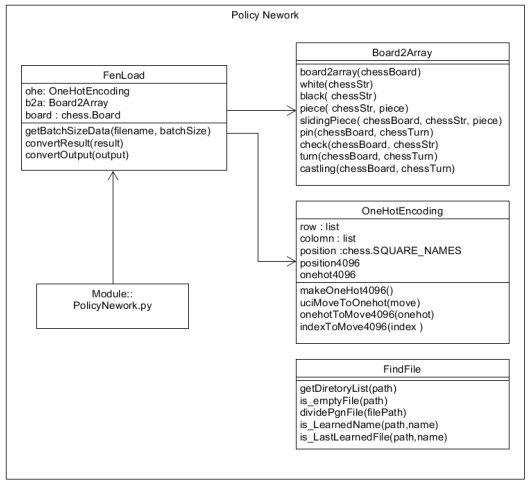

그림 4. 정책망 클래스 구조도
Fig. 4. Policy Network Class Architecture

3. Data Processing 구조

그림 3은 데이터를 전처리하기 위한 Data Processing 클래스 구조도를 나타낸다. 체스 기보 데이터는 King Base 사이트[9]에서 pgn 기보 형식으로 되어있다. 데이터를 학습시키는 과정에서 pgn 기보 데이터를 일일이 입력 데이터로 변환하여 학습시킨다. 이때 데이터를 불러오는 시간을 절약하기 위해, 한 줄의 정보로 모든 체스 대국의 정보를 가질 수 있는 Forsyth - Edwards Notation(FEN) 기보 방식[10]을 이용해 저장한다.

FenMaker.py 모듈에서 pgn_reader 클래스 이용해 pgn 기보 파일을 읽는다. 전처리 과정에서 FEN기보, 다음 수, 게임결과를 콜론으로 구분하여 데이터를 [FEN기보] : [다음 수] : [게임결과]의 형식으로 저장한다. fenMake()메서드에서 변수를 조정함으로써 원하는 개수의 기보 데이터를 추출할 수 있다.

전처리 과정에서 파이썬 pickle API를 이용해 객체를 파일로 저장하고 불러올 수 있는 기능을 사용하기 위해 입력 데이터를 변환하는 Board2Array 클래스와 One-Hot 인코딩으로 생성된 결과 값을 얻을 수 있는 OneHotEncoding 클래스를 이용한다.

4. 정책망 클래스 구조

그림 4는 정책망 클래스 구조도를 나타낸다.

정책망은 다음과 같은 방법으로 학습시킨다. FenLoad 클래스의 getBatchSizeData() 메서드를 통해 지정한 경로에서 원하는 배치 사이즈의 데이터를 불러온다. Board2Array 클래스와 OneHotEncoding 클래스를 이용해 입력 데이터와 결과 데이터를 받아낸다.

Board2Array 클래스는 넘겨받은 체스판을 board2array() 메서드를 이용해 입력 데이터로 변환한다. 각각의 체스판은 8x8의 크기에 0과 1로 구성되어 있다. 학습 모델에 필요한 입력 데이터는 다음과 같다.

- white(), black(): 각 색상을 판단
- piece(): 체스판 위에 말의 존재 유무 판단
- slidingPiece(): 주어진 체스판의 각 좌표에서 말이 이동할 수 있는 좌표 유무 판단
- pin(): 체스 상황에서 pin에 해당하는 말을 판단
- check(): 주어진 체스판에서 킹의 위치 판단
- turn(): 체스판에서 흑/백 턴 판단
- castling(): 현재 체스판에서 캐슬링 권한을 가지고 있는 경우 해당 말을 판단

가치망의 경우 정책망과 전체 구조는 같으며, 결과 값에 대해서 One-Hot 인코딩의 개수가 3개로 사용되고, 같은 입력 데이터에 대해 출력 데이터를 승, 무 ,패 세 가지 경우에 대해서 One-Hot인코딩으로 변환해 학습을 진행한다.

IV. 딥퍼플 구현 환경 및 학습 결과

딥퍼플은 Windows10 운영체제에서 딥러닝 구현에 필요한 라이브러리로 텐서플로우를 이용하였다. Elo 점수 2000대 이상의 체스 선수들의 기보를 이용하여 체스 기보 데이터를 전처리하여 텐서플로우를 통해서 데이터를 학습시켜 유효한 수를 내놓는 모델과 각각의 체스판에서 이길 확률을 측정하는 모델을 구현하였다. 체스 로직은 python-chess api를 이용하였으며, 텐서플로우와 python-chess api의 개발 언어는 파이썬을 사용하였으며, 체스 엔진에서 사용하는 사용자 인터페이스는 언리얼 엔진4를 사용하여 구현하였다.

그림 5는 CNN 모델을 통해 정책망과 가치망을 학습시킨 결과를 보여준다. 정책망의 경우 1천만 개의 기보를 epoch(반복학습 횟수)=5, batch size=10000으로 학습을 진행한 결과, 43%의 정확도를 보이며, 손실함수 cost는 1.9를 보였다. 출력 값이 19x19개인, CNN을 이용한 알파고의 정확도가 50~60% 사이의 정확도를 보이고 본 연구에서의 정책망은 4096개의 출력 값에 대하여 43%의 정확도를 보이므로 결과를 제대로 학습이 제대로 진행되었음을 알 수 있다.

초기화에 사용한 방법은 Xavier initializer를 사용하였고 초기 cost가 8.3 정도로 형성되어, 초기화가 잘 되었는지 확인하는 공식 $-\ln(1/4096) = 8.31776616672$와 비교해 보았을 때 적절하게 되었음을 확인할 수 있다.

모델에 대한 실험은 layer 개수를 3, 5, 13로 진행하였고, 1천만 개의 데이터에 대해서는 정확도와 손실함수 cost가 비슷하게 나타난다. 더 오래 학습을 진행한 13개의 layer가 cost 값이 가장 낮으므로 선택하여야 하나, 몬테카를로 트리탐색을 보다 빠르게 진행하기 위해서는 layer의 개수가 작은 모델을 사용하여 학습하였다.

가치망의 경우 일반적으로 주어진 체스판에 대해서 승, 무, 패의 3가지 경우의 수가 고르게 분포되어 있어 손실함수 cost가 진동하면서 줄어들지만 안정화되지 않은 모습을 보여준다. 그림에서 보는바와 같이 조금씩 감소하는 양상을 보이지만 1점대에서 계속 진동한다.

기존의 CNN을 이용한 체스엔진의 경우 1억 만개의 데이터를 사용하고, 알파고 또한 3천만 개의 데이터를 가지고 3주간 학습을 진행한 것과 비교해 볼 때, 본 연구에서는 1천만 개의 데이터를 사용하였으므로, 데이터의 개수를 증가시키고 더 오랜 시간 학습을 진행한다면 개선의 여지가 남아있다.

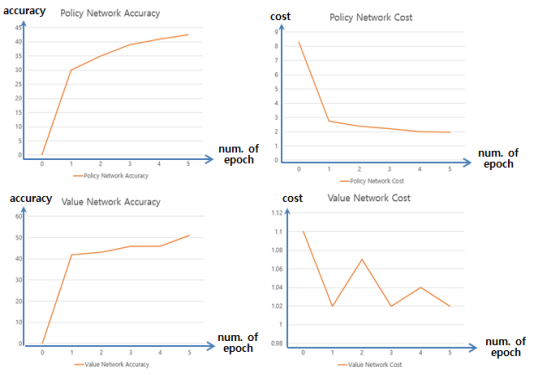

그림 5. 정책망 및 가치망 정확도 및 비용 그래프
Fig. 5. Accuracy and Cost Graph of Policy/Value Network

V. 결 론

본 논문은 딥러닝을 이용한 인공지능 체스 엔진인 딥퍼플에 대해 기술하였다. 딥퍼플 체스 엔진은 크게 몬테카를로 트리탐색과 CNN구조로 구성된 정책망, 가치망으로 구성되어 있다. 정책망은 주어진 체스판 상황에 대하여 모든 말들의 위치를 파악함으로써 최적의 다음 수를 예측하는 모델이며, 가치망은 주어진 체스상황이 누구에게 얼마나 유리한 상황인지를 예측하여 주는 모델이다.

딥러닝을 통해 구축된 모델인 정책망과 가치망으로부터 산출된 값을 이용하여 몬테카를로 트리 탐색 방식의 시뮬레이션을 진행한 후 결과를 분석하여 가장 좋은 수를 결정하도록 구현하였다.

CNN 모델을 통해 정책망과 가치망을 학습 시킨 결과, 정책망의 경우 43%의 정확도와 1.9의 손실함수 Cost를 보였으며, 가치망의 경우 승, 무, 패의 3가지 경우의 수가 고르게 분포되어 있어 손실함수 cost가 진동하면서 줄어들지만 1점대에서 계속 진동하는 결과를 보였다.

향후 연구과제로는 딥퍼플의 더 많은 데이터를 사용하여 학습을 진행하는 일과, 성능 개선을 위해 정책망과 가치망의 알고리즘을 개선시키는 연구가 진행될 예정이다.

References

[1] Clark, Christopher and Storkey, Amos. “Teaching deep convolutional neural networks to play Go”,

arXiv preprint arXiv:1412.3409, 2014.

[2] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., Rohlfshagen, P.,Tavener, S., Perez, D., Samothrakis, S., & Colton, S. “A survey of Monte Carlo tree search methods”. IEEE Transactions on Computational Intelligence and AI in Games, Vol. 4 No. 1, pp.1-43, 2012. DOI: https://doi.org/10.1109/TCIAIG.2012.2186810

[3] Barak Oshri and Nishith Khandwala, “Predicting Moves in Chess using Convolution Neural Networks”, http://github.com/BarakOshiri/ConvChess

[4] Matthew Lai., “Giraffe: Using Deep Reinforcement Learning to Play Chess”, arXiv:1509.01549v2, 2015.

[5] https://en.wikipedia.org/wiki/Elo_rating_system

[6] Jonathan Baxter, Andrew Tridgell, and Lex Weaver “TDLeaf(λ) Combining Temporal Difference Learning with Game-Tree Search”Australian Journal of Intelligent Information Processing Systems, 1998.

[7] https://en.wikipedia.org/wiki/FIDE_World_Rankings.

[8] https://www.unrealengine.com/ko/what-is-unreal-engine-4.

[9] http://www.kingbase-chess.net/

[10] https://en.wikipedia.org/wiki/Forsyth_Edwards_Notation

저자 소개

김 성 환(준회원)

- 2014년 ~ 현재 한성대학교 컴퓨터공학과

<주관심분야 :모바일시스템, 딥러닝>

김 영 웅(정회원)

- 1993년 KAIST 전산학과 박사
- 1984 ~ 1997년 KT 통신망 연구소
- 1997년 ~ 현재 한성대학교 컴퓨터공학부 교수

<주관심분야 :데이터모델링, 소프트웨어 신뢰도, 소프트웨어 설계>

※ 본 연구는 한성대학교 교내연구비 지원 과제임.