TANGO

하이퍼파라미터 최적화(HPO) 기술

성명 김중헌 교수 소속 고려대학교

SUBJECT

인공지능 기술의 대중화 (AI Democratization)를 위한
TANGO 커뮤니티 3회 컨퍼런스

주관 ETRI 한국전자통신연구원 (TANGO) 주최 과학기술정보통신부 IITP 정보통신기획평가원

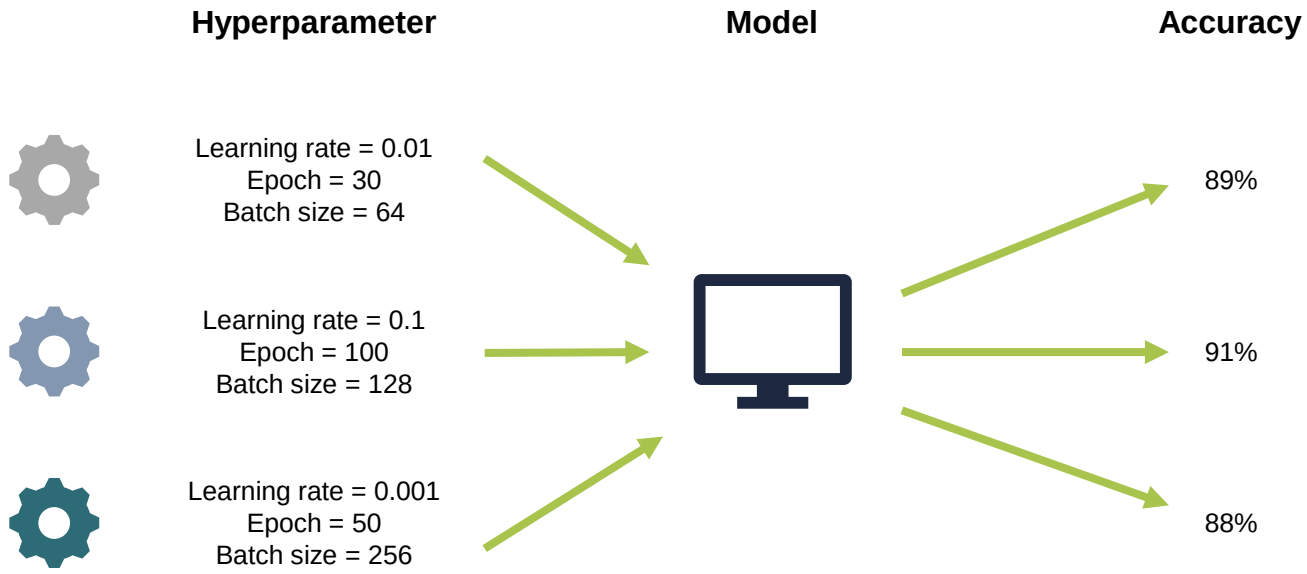
후원 Lablup weda tesla System (사)한국인공지능협회 SNUH 서울대학교병원 고려대학교 KOREA UNIVERSITY 경희대학교 KYUNG HEE UNIVERSITY 중앙대학교

목 차

Hyperparameter

- 하이퍼파라미터 (Hyperparameter)
  - 모델이 학습하면서 최적의 값을 자동으로 찾는 것이 아니라 사람이 직접 지정해 주어야 하는 변수
  - 모델링할 때 사용자가 직접 세팅해주는 값
  - 하이퍼파라미터는 모델 구조, 기능 및 성능을 직접 제어함

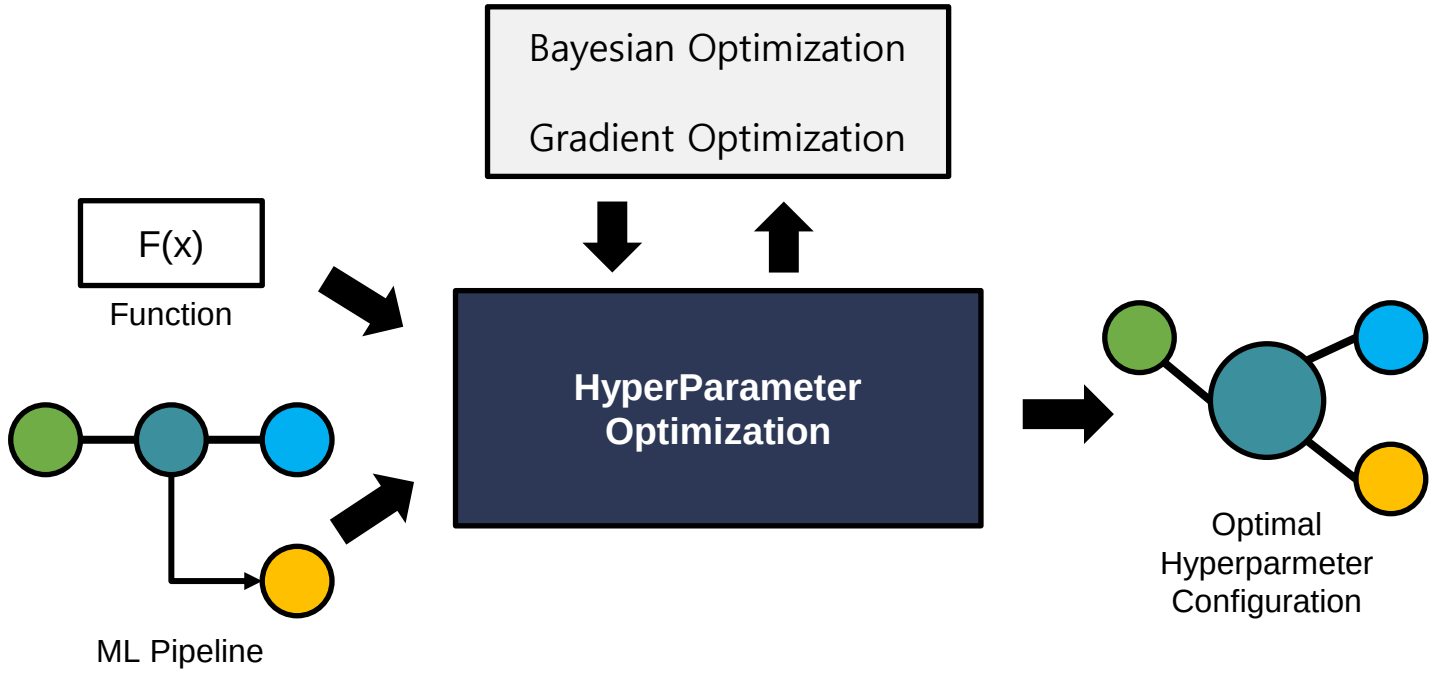
- 하이퍼파라미터 종류
  - 학습률 (Learning rate)
  - 파라미터 업데이트 변화율 (Momentum)
  - 훈련 반복 횟수 (epoch, training epochs)
  - 배치 사이즈 (batch size)



Hyperparameter Optimization (HPO)

- 하이퍼파라미터 최적화 (Hyperparameter Optimization; HPO)
  - HPO통해 최적 결과를 위해 모델 성능을 조정할 수 있음
  - HPO는 학습이 완료되었을 때, 높은 성능의 모델이 도출되도록 결정하는 모델 학습 파라미터를 역으로 찾는 과정
  - 학습에 영향을 주는 하이퍼파라미터를 기존 수동적 조정에서 나아가,
    학습을 통한 최적의 하이퍼파라미터 추정

- 하이퍼파라미터 최적화 기술
  - Gradient Descent Optimization
  - Bayesian Optimization

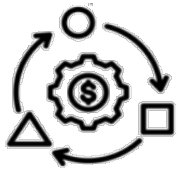


Ⅰ. Hyperparameter Optimization (HPO)

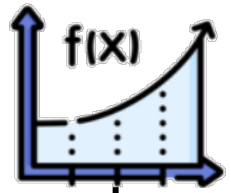
Bayesian Optimization

- 블랙박스 함수를 최적화할 때 베이지안 최적화 방법을 주로 사용
- 사전 데이터를 바탕으로 목적함수를 추정하는 확률 모델 사용

- 1) 미지의 목적 함수 (블랙박스 함수)에 하이퍼파라미터를 랜덤하게 넣어서 성능 측정하고 이를 기반으로 최적 함수 추정
  - Learning rate 0.7 → 성능 95%
  - Learning rate 0.5 → 성능 93%
  - Learning rate 0.3 → 성능 92%



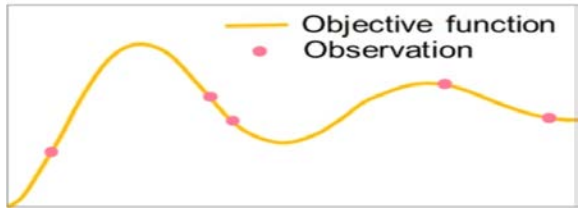
- 2) 추정된 최적 함수를 기반으로 다음으로 관측할 하이퍼파라미터 계산



- 3) 더 큰 최적 관측값을 갖을 가능성이 높은 하이퍼파라미터를 전달
- 4) 전달받은 하이퍼파라미터를 통해 다시 최적 함수 추정
- 5) 반복을 통해 최적 함수의 불확실성 개선 및 추정 가능성 높임

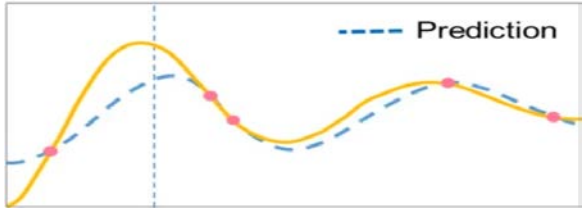
Bayesian Optimization

- Surrogate model
  - 현재까지 조사된 입력값, 결과값(평균적인 예상되는 모델의 정확도)를 바탕으로 목적함수의 대략적인 확률 추정
- Acquisition function
  - 현재까지 관측되었던 가장 좋은 결과값과 평균 값의 차이 (확률분포)를 기반으로, 다음 입력값 후보인 하이퍼파라미터 추천



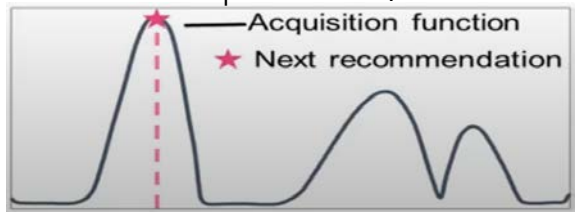
처음에는 랜덤하게 하이퍼파라미터들을
샘플링하여 결과를 관측함

Surrogate 모델 생성



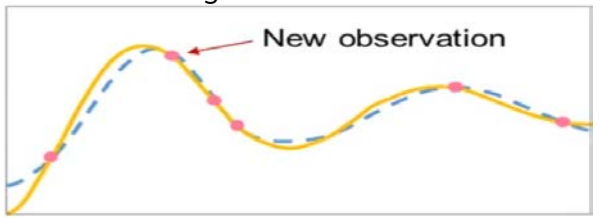
관측된 결과를 바탕으로 Surrogate 모델은
최적 함수를 추출함. 여기서 파란색 선을 의미함

Acquisition 함수



다음으로 관측할 하이퍼파라미터
조합을 추천함

Surrogate 모델 업데이트

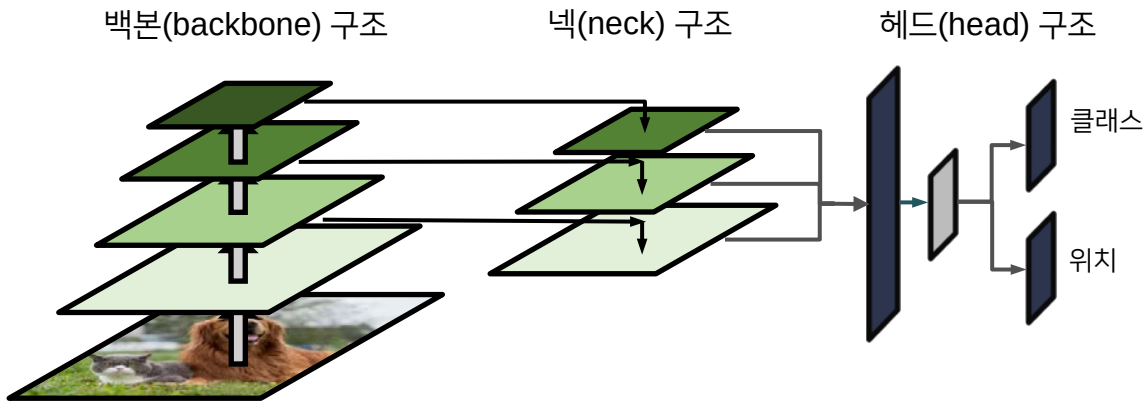


새로운 관측치를 바탕으로
Surrogate 모델은 최적 함수를 추정하게 됨

반복

You Only Look Once (YOLO)

- 2016년 CVPR, Joseph Redmond 공개 → YOLOv3 이후 연구 커뮤니티 탈퇴
- 대표적인 One-stage 검출기
  - 단일 신경망으로 클래스 예측과 객체 위치 추론
  - 실시간(Real-time) 영상 처리 가능
- 신경망 구조
  - 백본 구조: 피처 추출, 상대적으로 크고 무거움
  - 넥 구조: 다양한 수준의 피처 가공
  - 헤드 구조: 클래스 분류, bounding box 추측



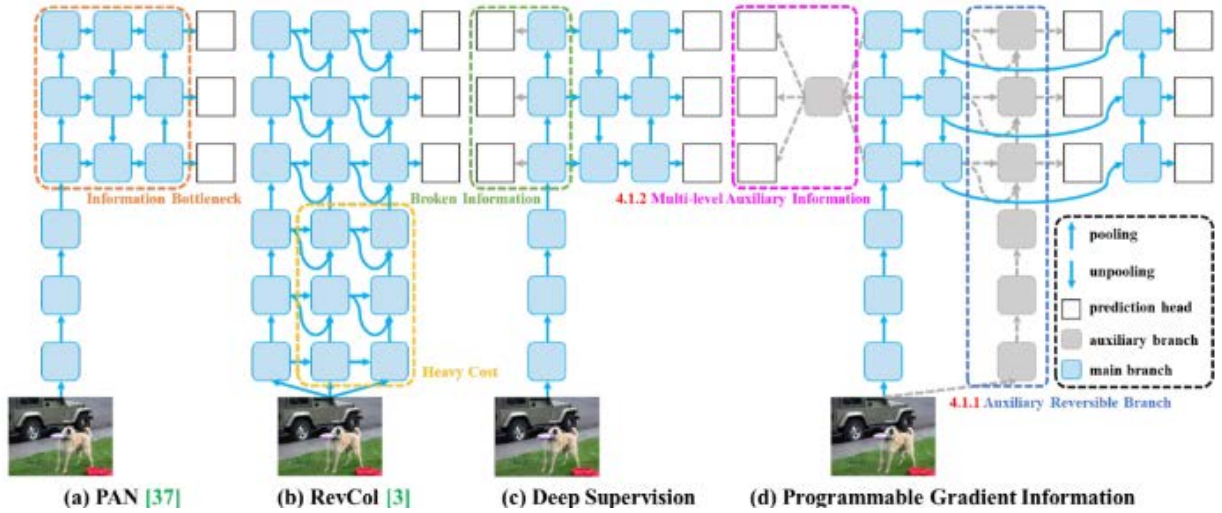
YOLOv9

- Layer 통과할수록 정보가 손실된다는 문제점 해결 위해 등장

- PGI(Programmable gradient information)
  - Auxiliary branch
    - 학습할 때 사용하고, inference 시 사용하지 않음
      → 모델 크기 줄이고 속도 개선
    - 완전한 원본 정보를 얻지는 않지만, 중요한 정보를 추출 할 수 있게 해 줌

  - Multi-level Auxiliary Information
    - Main branch와 prediction head 사이에 auxiliary branch 삽입
      → 다양한 prediction heads에서 반환된 gradients를 결합
      → 정보 손실 줄일 수 있음

- Generalized ELAN (GELAN)
  - CSPNet과 ELAN 2개의 네트워크 구조 결합
  - 빠른 속도와 높은 정확도 보장



YOLOv9-NAS

- 탐색 공간이 넓어질수록 최적 신경망 구조를 포함할 가능성 높아짐
- 그러나 탐색 공간의 확장은 컴퓨팅 리소스 및 시간 측면에서 높인 비용 초래

- → 성능이 검증된 참조 신경망 구조, 변경 규칙을 활용하여
  상대적으로 좁은 탐색 공간에서도 좋은 성능의 신경망 구조를 찾음

- 변경 규칙
  - Convolutional 커널 크기 변화 : 3×3 또는 5× 5
  - 활성화 함수 (activation) 변화 : ReLU, LeakyReLu, Mish

```
OPS = {
    '3x3_relu_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='relu', kernel_size=3),

    '3x3_leaky_BNCSP': lambda c1, c2, c3, c4, c5 :
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='leaky', kernel_size=3),

    '3x3_mish_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='mish', kernel_size=3),

    '3x3_silu_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='silu', kernel_size=3),

    '5x5_relu_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='relu', kernel_size=5),

    '5x5_leaky_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='leaky', kernel_size=5),

    '5x5_mish_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='mish', kernel_size=5),

    '5x5_silu_BNCSP': lambda c1, c2, c3, c4, c5:
    MRepNCSPELAN4Layer(c1, c2, c3, c4, c5, act='silu', kernel_size=5),

}
```

YOLOv9-NAS

- Microsoft NNI 하이퍼파리미터 최적화 툴 사용
- Bayesian Optimization 방법 사용

lr0: 0.01, lrf: 0.1,
momentum: 0.937,
batch size: 16,
epoch: 30

초기 설정
하이퍼파라미터

```
hparams = {
    'lr': 0.001,
    'momentum': 0.937
}

search_space = {
    'lr': {'_type': 'loguniform', '_value': [0.0001, 0.1]},
    'momentum': {'_type': 'uniform', '_value': [0, 1]},
}
from nni.experiment import Experiment
experiment = Experiment('local')
experiment.config.trial_command = 'search.py'
experiment.config.trial_code_directory = '.'
experiment.config.search_space = search_space
experiment.config.tuner.name = 'SMAC'
experiment.config.tuner.class_args['optimize_mode'] = 'maximize'
experiment.config.max_trial_number = 100
experiment.config.trial_concurrency = 40
optimized_params = nni.get_next_parameter()
hparams.update(optimized_params)
print(hparams)

experiment.run(9876)

experiment.stop()
```

초기 설정 하이퍼파라미터

하이퍼파라미터 탐색 범위

Bayesian optimization 진행

하이퍼파라미터 툴 적용

{"parameter_id": 0, "parameter_source": "algorithm", "parameters": {"lr": 0.0004808482805459193, "momentum": 0.31675833970975287}, "parameter_index": 0}

{"parameter_id": 1, "parameter_source": "algorithm", "parameters": {"lr": 0.024665866237339353, "momentum": 0.6762546707509746}, "parameter_index": 0}

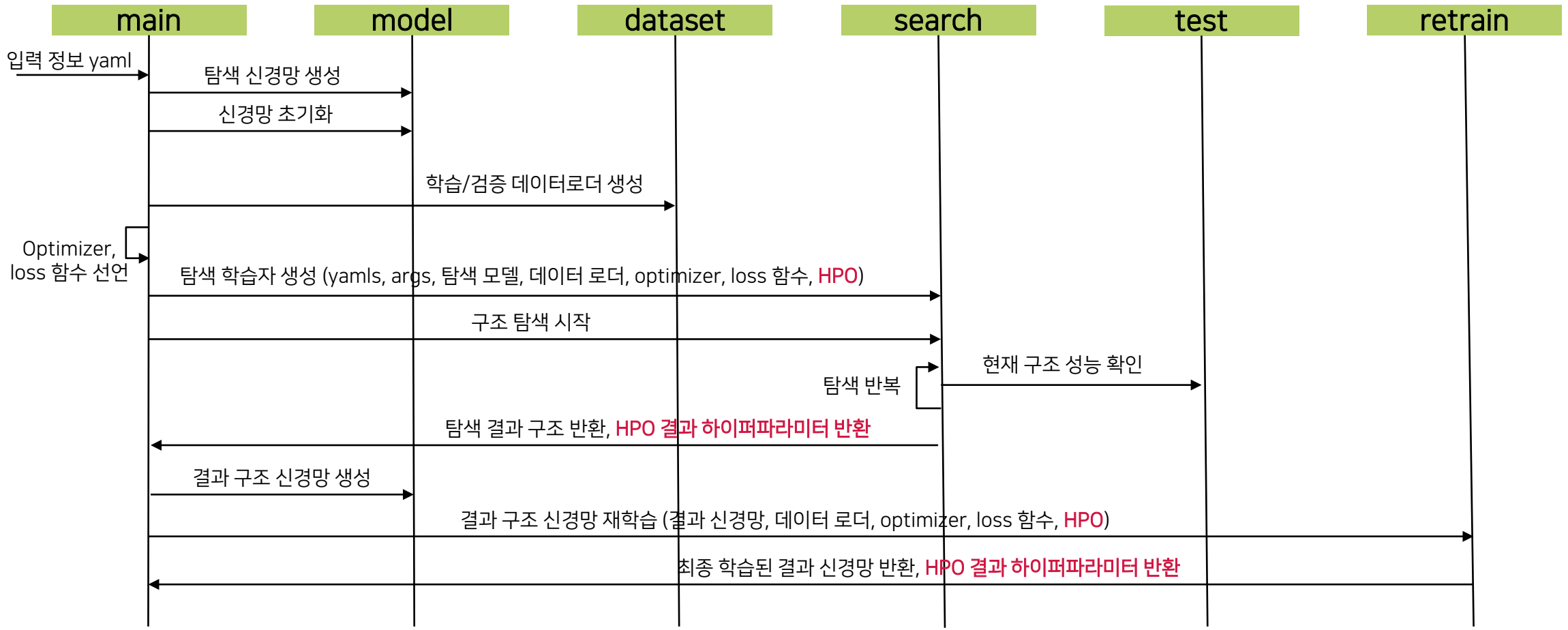
{"parameter_id": 99, "parameter_source": "algorithm", "parameters": {"lr": 0.011532316925081854, "momentum": 0.3661002387352602}, "parameter_index": 0}

하이퍼파라미터 탐색 중

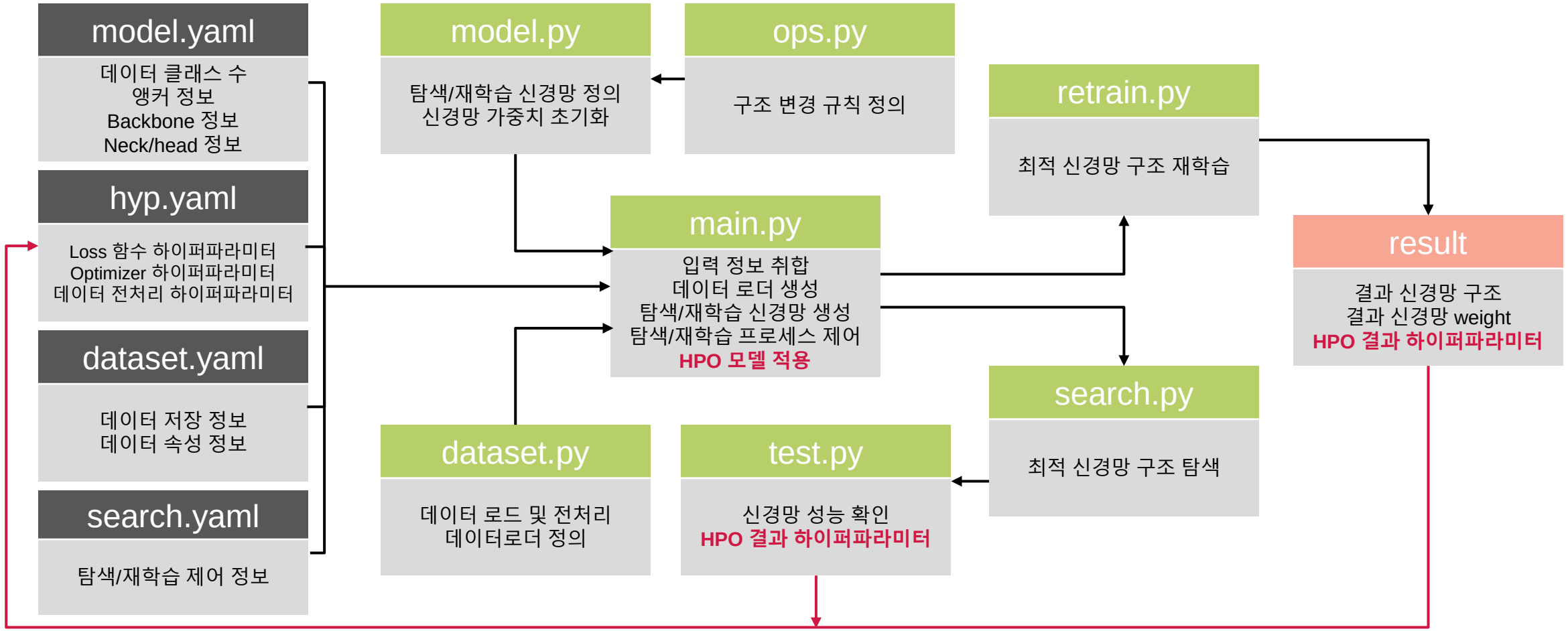
{"parameter_id": 8, "parameter_source": "algorithm"
"parameters": {"lr": 0.0022791217338123154,
"momentum": 0.025144362542212972},
"parameter_index": 0}

최적화된 결과 하이퍼파라미터 출력

하이퍼파라미터 최적화 과정



하이퍼파라미터 최적화 과정

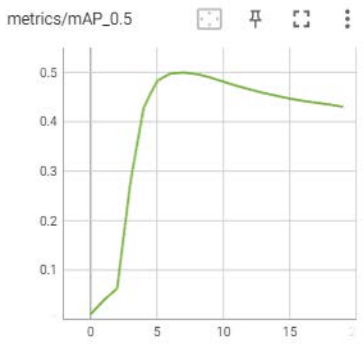


Ⅱ. YOLONAS +HPO

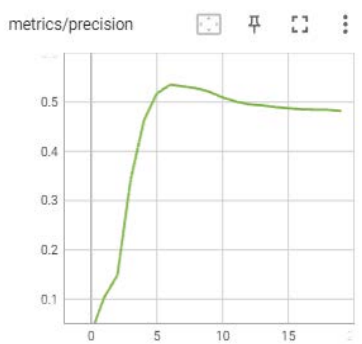
YOLOv9-NAS + HPO

- 실험 결과

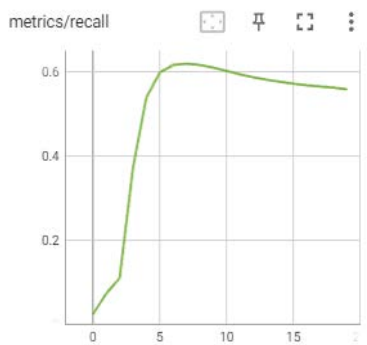
최적화된 하이퍼파라미터 적용 전



42.96%

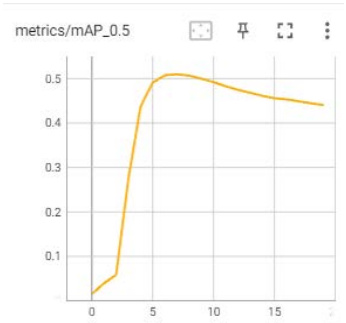


48.13%

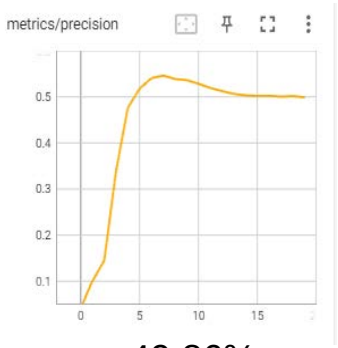


55.8%

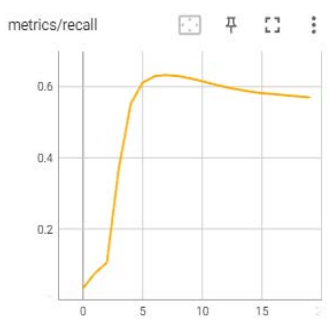
최적화된 하이퍼파라미터 적용 후



44%



49.86%



56.95%

약 1%
성능 향상

Final architecture: {'m_13': 6, 'm_16': 2, 'm_19': 1, 'm_22': 2, 'm_28': 0, 'm_31': 2, 'm_34': 2, 'm_37': 2}

하이퍼파라미터	값
Learning rate	0.001
Momentum	0.9
Epoch	20
Batch size	16
Optimizer	SGD

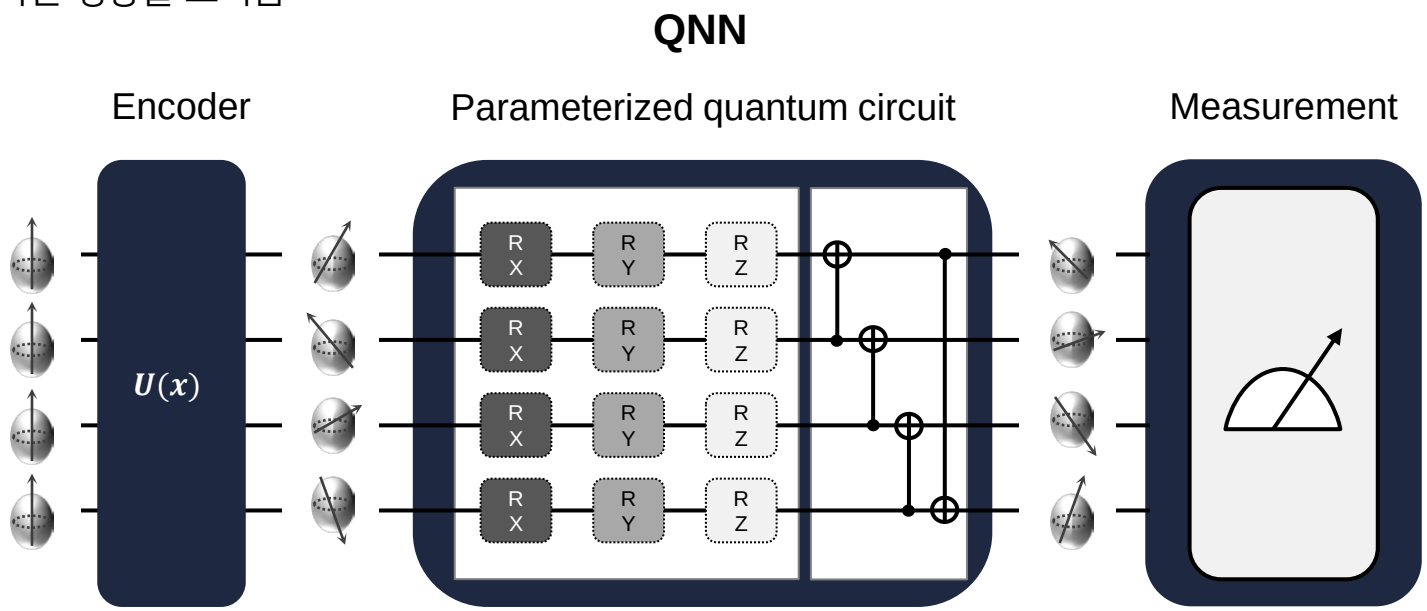
Final_architecture : {"m_13": 6, "m_16": 6, "m_19": 5, "m_22": 6, "m_28": 4, "m_31": 2, "m_34": 2, "m_37": 5}

하이퍼파라미터	값
Learning rate	0.0005306488004008417
Momentum	0.325774080243569
Epoch	20
Batch size	16
Optimizer	SGD

Quantum Neural Network (QNN)

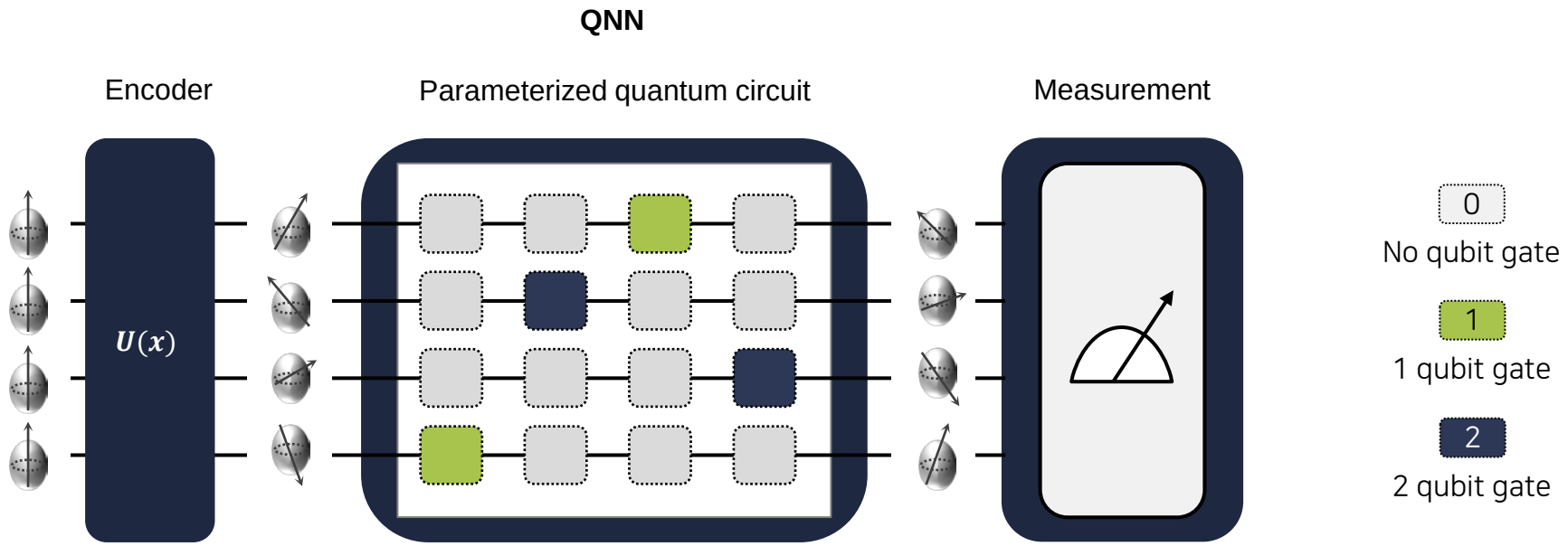
- Quantum Neural Network (QNN)
  - 양자 신경망(QNN)은 대규모 데이터 세트를 효율적으로 처리하기 위한 유망한 접근법임
  - 중첩과 얽힘의 양자 특성을 활용하여 기존 신경망(NN)으로는 실현 불가능한 계산을 가능하게 함
  - 훨씬 적은 수의 파라미터를 사용하면서도 NN에 필적하는 성능을 보여줌

- QNN의 구성
  - Encoder
    - 기존 데이터를 양자 호환 형식으로 변환
  - Parameterized quantum circuit (PQC)
    - NN의 hidden layer과 유사한 형식으로 작동
  - Measurement
    - PQC에서 생성된 양자 상태가 고전 출력으로 변환



Quantum Neural Architecture Search

- PQC 내의 게이트 배치는 얽힘의 정도와 중첩의 정도를 결정하는 데 중요한 역할을 하며, QNN의 성능에 영향을 미침
- PQC 내의 게이트를 많이 배치할수록 노이즈가 증가하고 학습이 제대로 되지 않는 문제가 있음
- Quantum Neural Architecture Search는 효율적인 양자 회로를 자동으로 설계하는 것을 목표로 하고 있음



TANGO

감사합니다.

주관 ETRI 한국전자통신연구원 (TANGO) 주최 과학기술정보통신부 IITP 정보통신기획평가원

후원 Line lablup weda tesla system (사)한국인공지능협회 SNUH 서울대학교병원 고려대학교 KOREA UNIVERSITY 홍익대학교 HONGIK UNIVERSITY CAU 중앙대학교