



# 하이퍼파라미터 최적화(HPO) 기술

성명 김중헌

소속 고려대학교

인공지능 기술의 대중화  
(AI Democratization)를 위한  
제2회 탱고 커뮤니티 컨퍼런스





1	하이퍼파라미터 최적화	00
	1. 하이퍼파라미터 개념	
	2. 하이퍼파라미터 최적화	
	3. 하이퍼파라미터 최적화 기술	
	4. 하이퍼파라미터 최적화 툴	
2	하이퍼파라미터 최적화 활용	08
	1. 객체 검출 신경망 (YOLO)	
	2. HPO Sequence	
	3. HPO 적용 코드	
	4. 실험 결과	

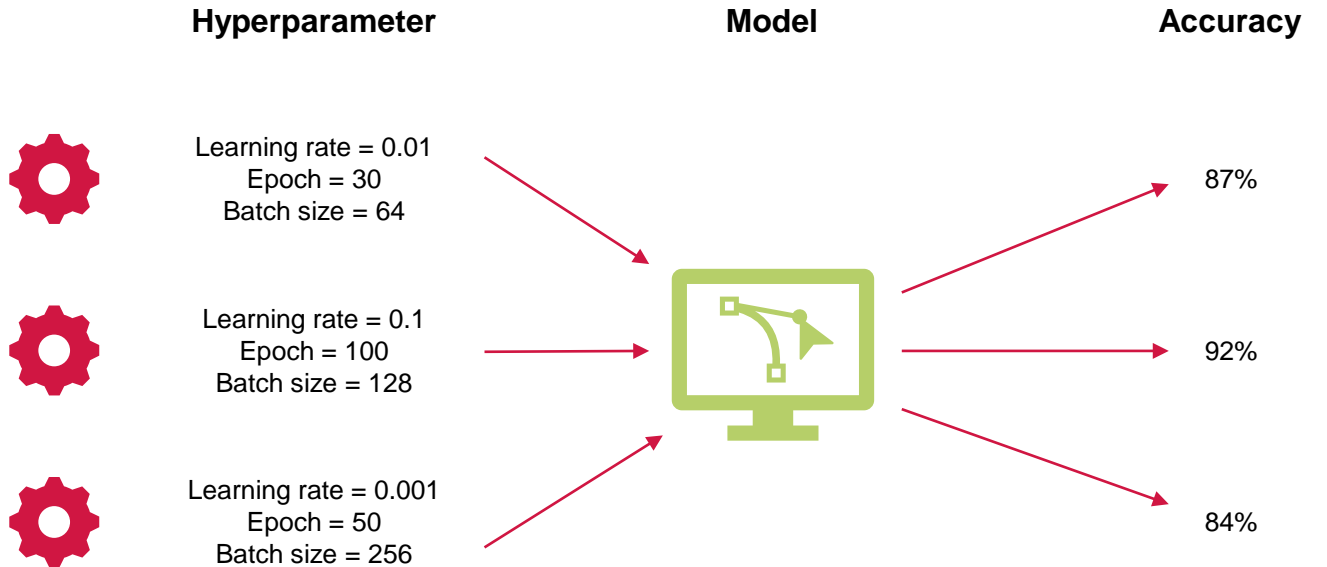


## 1. 하이퍼파라미터 개념

- 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

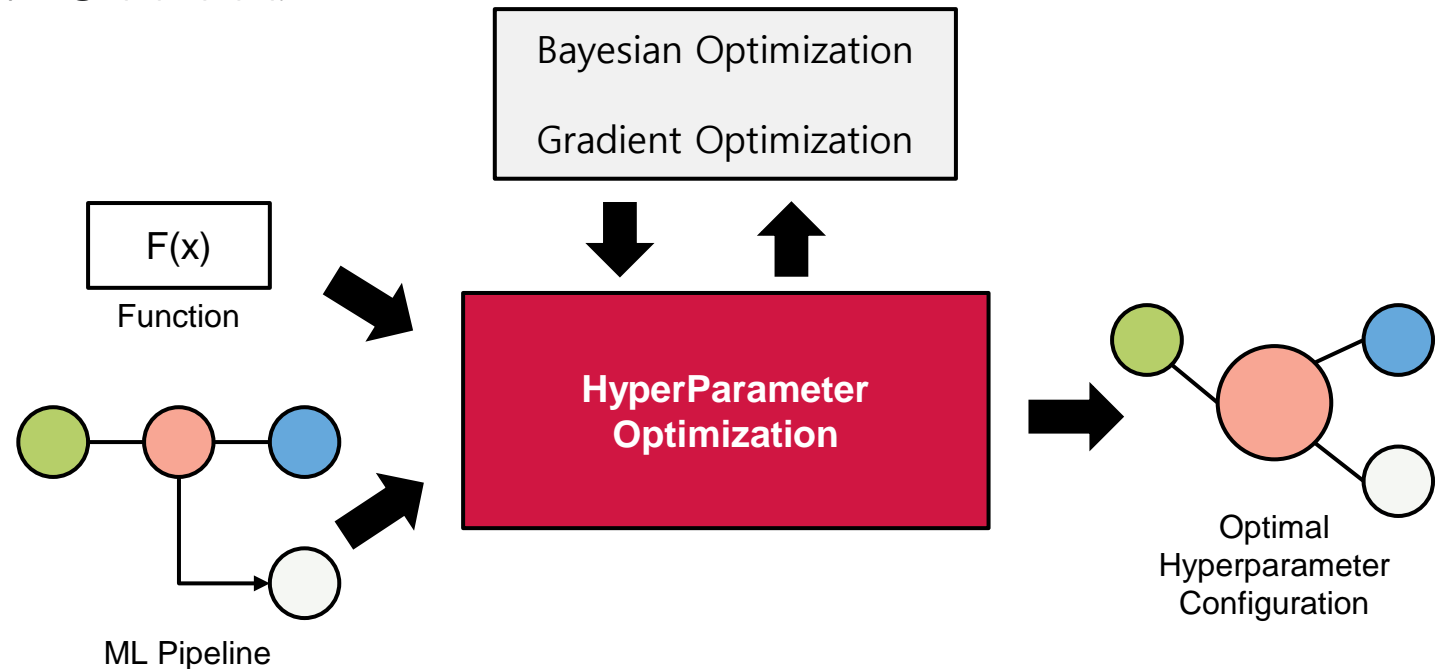
- 학습률 (Learning rate)
- 파라미터 업데이트 변화율 (Momentum)
- 훈련 반복 횟수 (epoch, training epochs)
- 배치 사이즈 (batch size)





## 2. 하이퍼파라미터 최적화

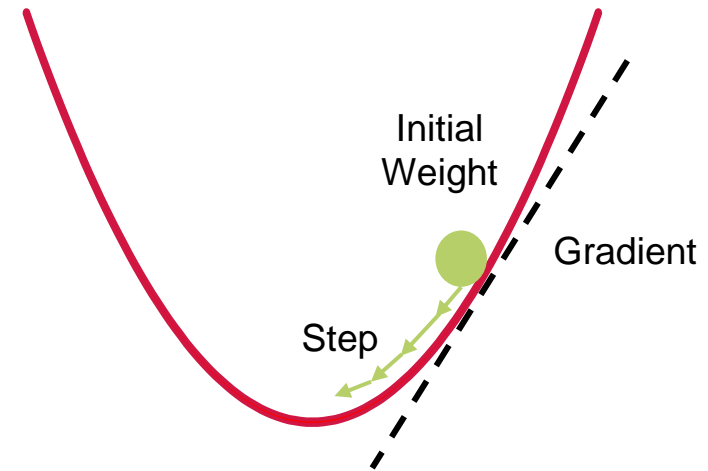
- 하이퍼파라미터 최적화 (Hyperparameter Optimization; HPO)
  - HPO통해 최적 결과를 위해 모델 성능을 조정할 수 있음
  - HPO는 학습이 완료되었을 때, 높은 성능의 모델이 도출되도록 결정하는 모델 학습 파라미터를 역으로 찾는 과정
  - 학습에 영향을 주는 하이퍼파라미터를 기존 수동적 조정에서 나아가, 학습을 통한 최적의 하이퍼파라미터 추정
- 하이퍼파라미터 최적화 기술
  - Gradient Descent Optimization
  - Bayesian Optimization





## 3. 하이퍼파라미터 최적화 기술

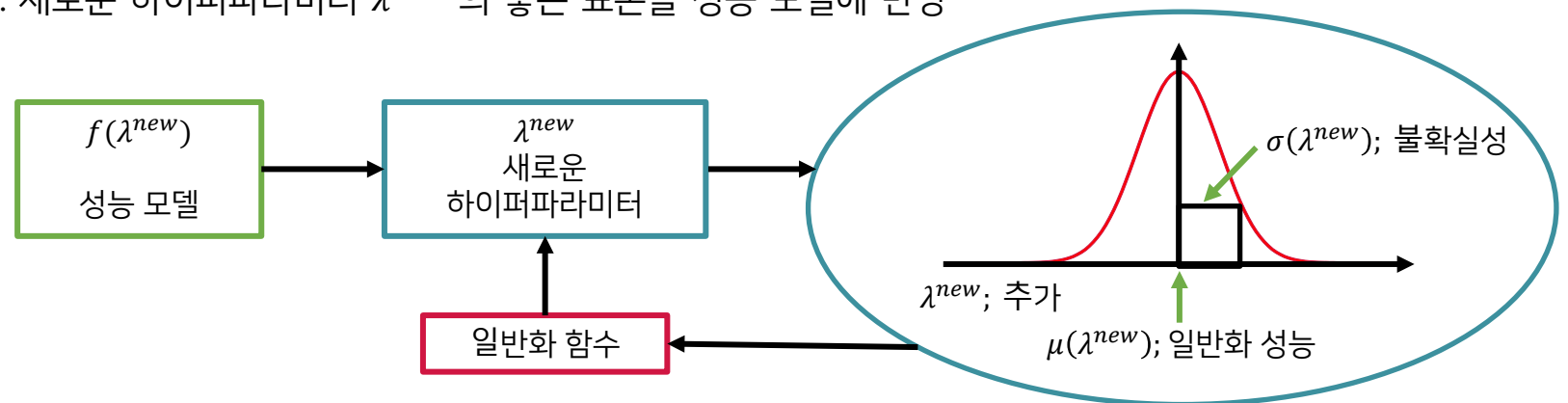
- Gradient Descent Optimization
  - 현재의 미분값(gradient)을 기반으로 하이퍼파라미터가 업데이트 해야 할 방향과 크기를 설정
  - 하이퍼파라미터에 대한 기울기를 얻기 위해, 자동 미분을 사용하여 반복 최적화 알고리즘의 단계에서 미분 진행
  - 기존 Blackbox 기법과 다르게 근사 gradient 정보 활용
  - 직접적인 gradient 계산은 시간 메모리 측면에서 높은 비용 발생





## 3. 하이퍼파라미터 최적화 기술





- Bayesian Optimization
  - 목적함수  $f$  에 대해 함수값  $f(x)$ 를 최대로 만드는 최적해  $x$ 를 탐색하는 방법
  - 이 때의 목적 함수  $f(x)$ 는 black-box function으로 미지의 함수  $\rightarrow$  지금까지의 데이터로 추정해야 하는 함수
  - 순차적으로 하이퍼파라미터를 업데이트해 가면서 평가를 통해 최적의 하이퍼파라미터 조합을 탐색
  - 가능 한 적은 수의  $x$  후보에 대해서만 함수 값을 연산하며  $f$ 를 최대로 만드는 최적해  $x$ 를 빠르고 효과적으로 탐색
  - 성능모델 (Surrogate model): 하이퍼파라미터 집합과 성능의 관계를 모델링
  - 일반화 함수 (Acquisition function): 새로운 하이퍼파라미터  $\lambda^{new}$ 의 좋은 표본을 성능 모델에 반영





## 4. 하이퍼파라미터 최적화 툴

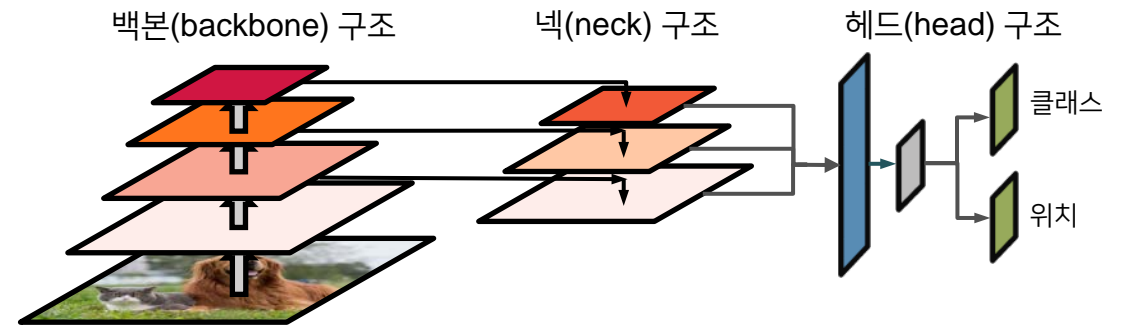
- 하이퍼파라미터 최적화 툴

	 Neural Network Intelligence Microsoft NNI	 Ray Tune	 Optuna	 Hyperopt
오픈 소스	✓	✓	✓	✓
지원 알고리즘	완전 탐색 - RS, GS Heuristic 탐색 - Anneal, EA, HB, ... BO - TPE, GP, BOHB,...	Ax/Botorch, Hyperopt, BO	AxSearch, DragonflySearch, HyperOptSearch, OptunaSearch, BayesOptSearch	RS, TPE, Adaptive TPE
지원 프레임워크	<b>Pytorch</b> , TF, Keras, Theano, Caffe2	<b>Pytorch</b> , TF, sklearn, Keras, XGBoost, LightGBM	<b>Pytorch</b> , TF, sklearn, Keras, MXNet, LightGBM	<b>Pytorch</b> , TF, sklearn, XGBoost
라이선스	MIT	Apache-2.0	MIT	BSD 3-clause



## 1. 객체검출신경망 (YOLO)

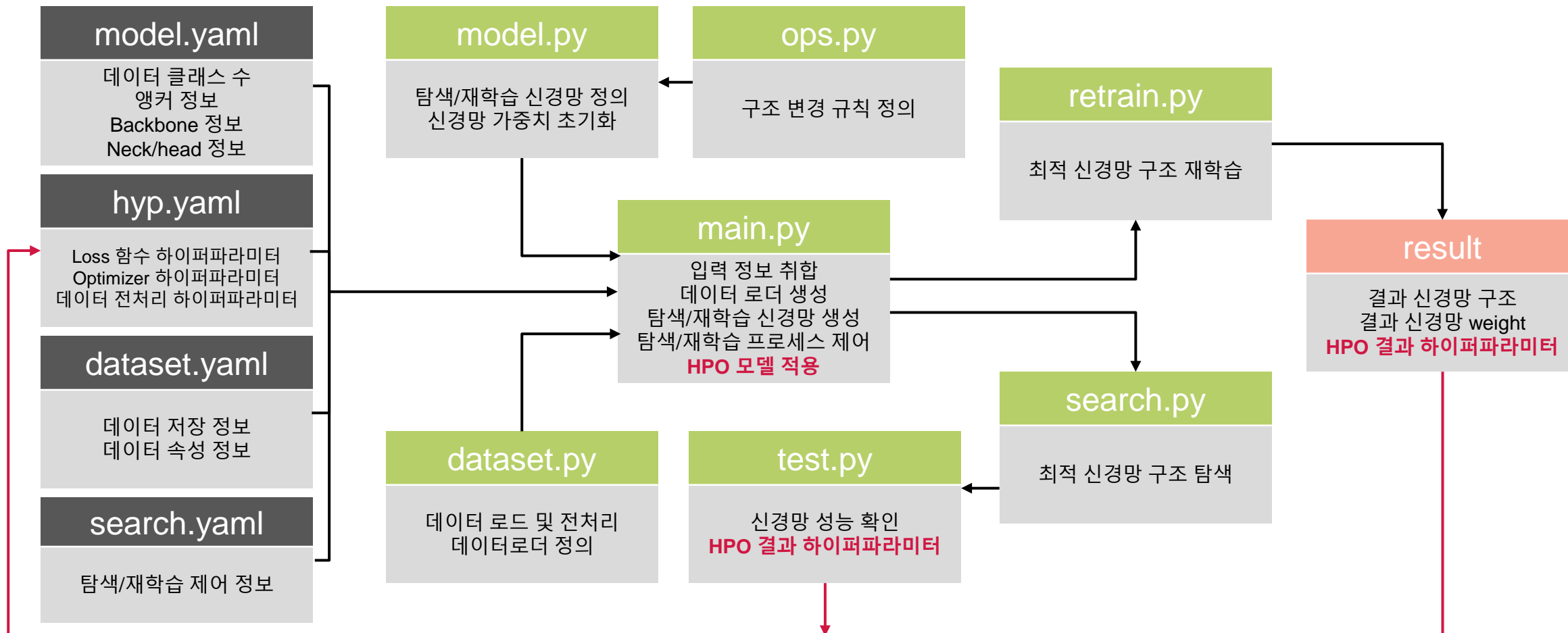
- You Only Look Once (YOLO)
  - 2016년 CVPR, Joseph Redmond 공개 → YOLOv3 이후 연구 커뮤니티 탈퇴
  - 대표적인 One-stage 검출기
    - 단일 신경망으로 클래스 예측과 객체 위치 추론
    - 실시간(Real-time) 영상 처리 가능
  - 신경망 구조
    - 백본 구조: 피쳐 추출, 상대적으로 크고 무거움
    - 넥 구조: 다양한 수준의 피쳐 가공
    - 헤드 구조: 클래스 분류, bounding box 추측
- YOLOv7 모델을 채택하여 사용





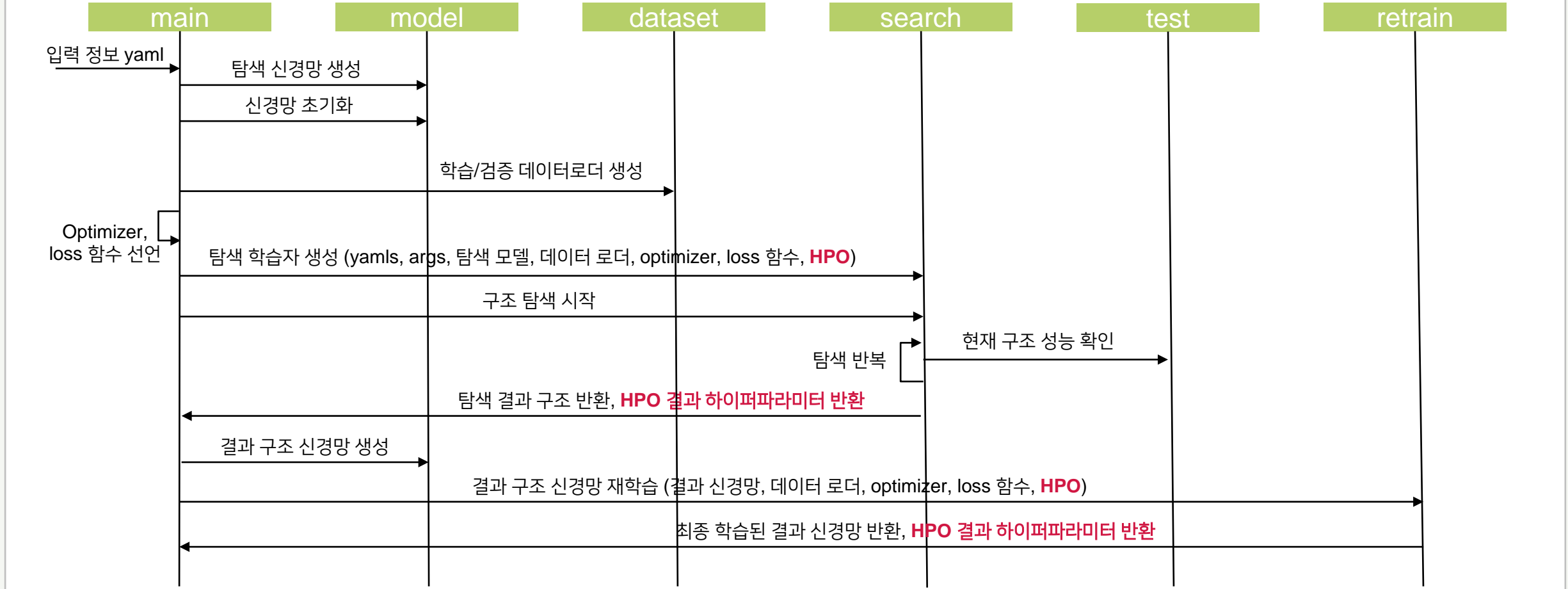
## 2. HPO Sequence

- HPO를 적용한 객체 검출 신경망 구조 탐색 프레임워크 시퀀스



## 2. HPO Sequence

- HPO를 적용한 객체 검출 신경망 구조 탐색 프레임워크 시퀀스



## 3. HPO 적용 코드

- YOLOv7 모델에 HPO 적용
  - Microsoft NNI 하이퍼파라미터 최적화 툴 사용
  - Bayesian Optimization 방법 사용

lr: 0.01, lr: 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": 8, "parameter_source": "algorithm"  
"parameters": {"lr": 0.0022791217338123154,  
"momentum": 0.025144362542212972},  
"parameter_index": 0}
```

최적화된 결과 하이퍼파라미터 출력



## 4. 실험 결과

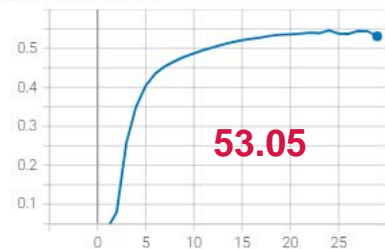
- 최적화된 하이퍼파라미터 적용 전

하이퍼파라미터	값
Learning rate	0.01
Momentum	0.937
Epoch	29
Batch size	16
Optimizer	SGD

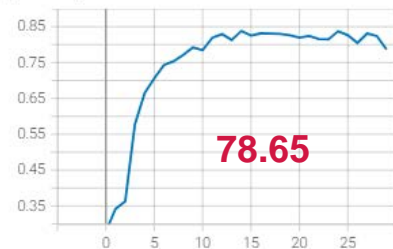
metrics/mAP\_0.5  
tag: metrics/mAP\_0.5



metrics/mAP\_0.5\_0.95  
tag: metrics/mAP\_0.5\_0.95



metrics/precision  
tag: metrics/precision



metrics/recall  
tag: metrics/recall



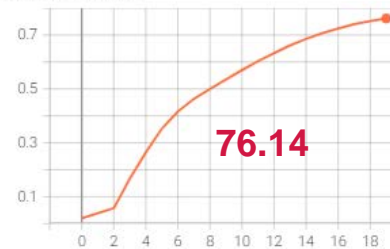
약 4%  
성능 향상



- 최적화된 하이퍼파라미터 적용 후

하이퍼파라미터	값
Learning rate	0.0005306488004008417
Momentum	0.325774080243569
Epoch	19
Batch size	16
Optimizer	SGD

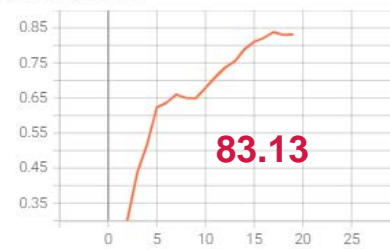
metrics/mAP\_0.5  
tag: metrics/mAP\_0.5



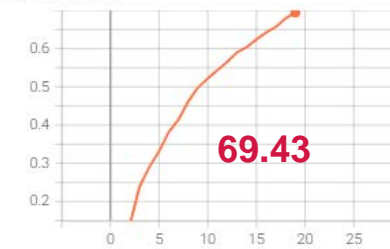
metrics/mAP\_0.5\_0.95  
tag: metrics/mAP\_0.5\_0.95



metrics/precision  
tag: metrics/precision



metrics/recall  
tag: metrics/recall



# 감사합니다.

T

A

N

감사합니다.

