



Nutella

머신러닝 개발자를 위한 서비스

20170392 이해인

Date: 2021.04.14



I. 과제 개요

II. 시스템 요청 사항

III. 시스템 Image



Nutella

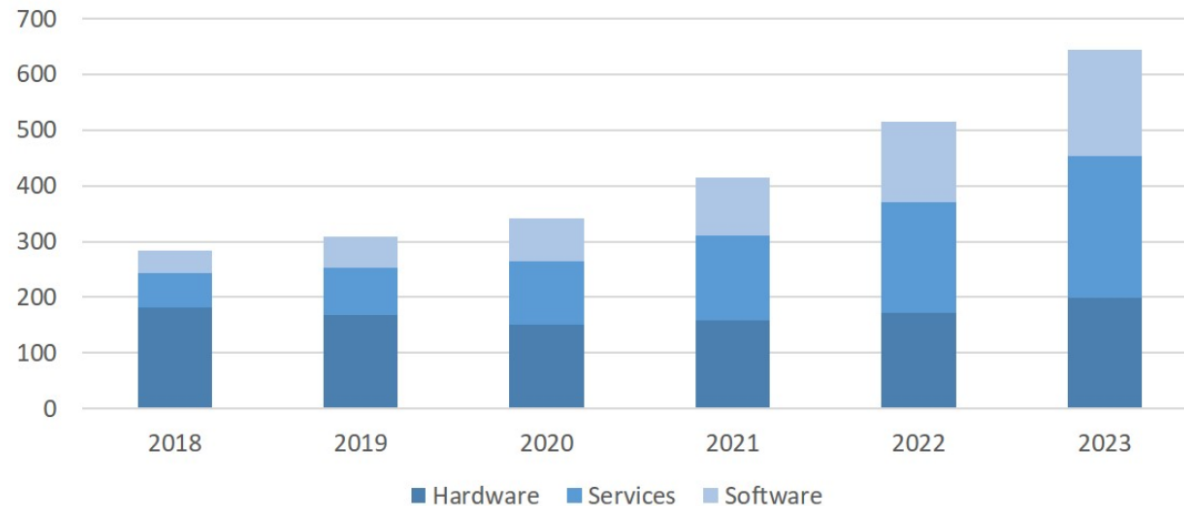
I . 과제 개요



1. 시장 전망 - 인공지능



국내 인공지능 시장 전망 2019-2023년 [단위:십억]



Source: IDC, 2019

세계 AI 시장 규모

※ 2017~2022년은 추정치

자료: IDC

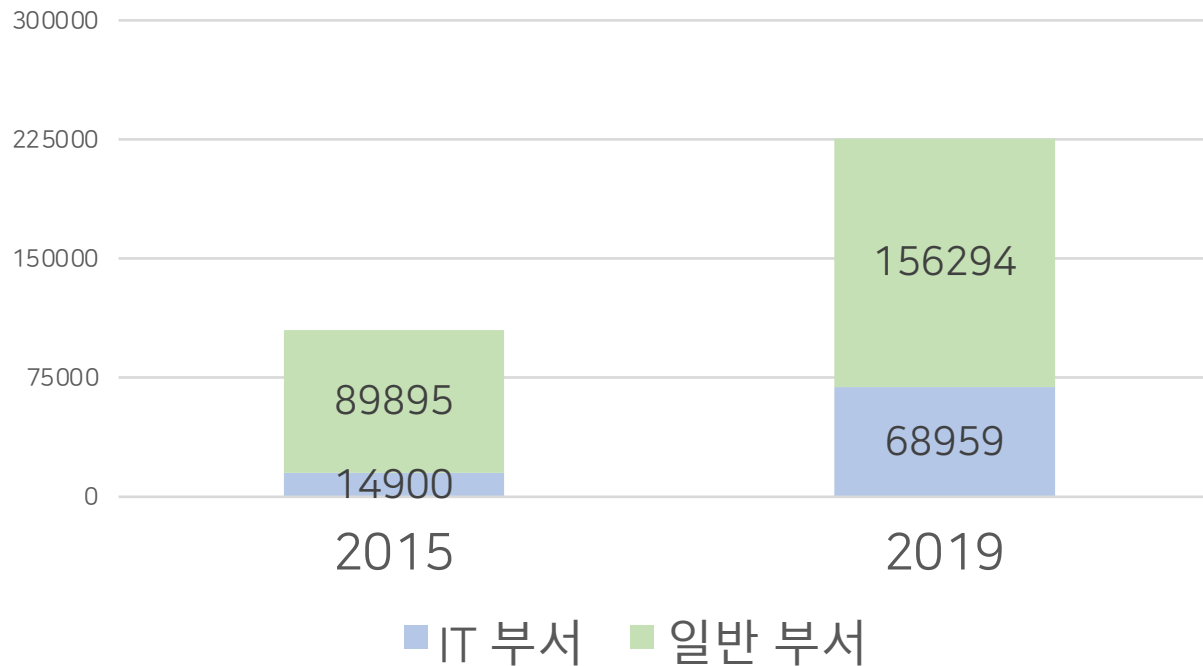


2022년 인공지능 시장 규모는 국내 510억원, 전 세계 135조 원으로 추정
시장 규모 자체와 성장속도를 보면 전 세계적으로 인공지능에 대한 수요가 급증



2. 전문 인공지능 개발자가 부족한 기업

GDP 상위 12개국에서의 총 AI 구인 공고 수

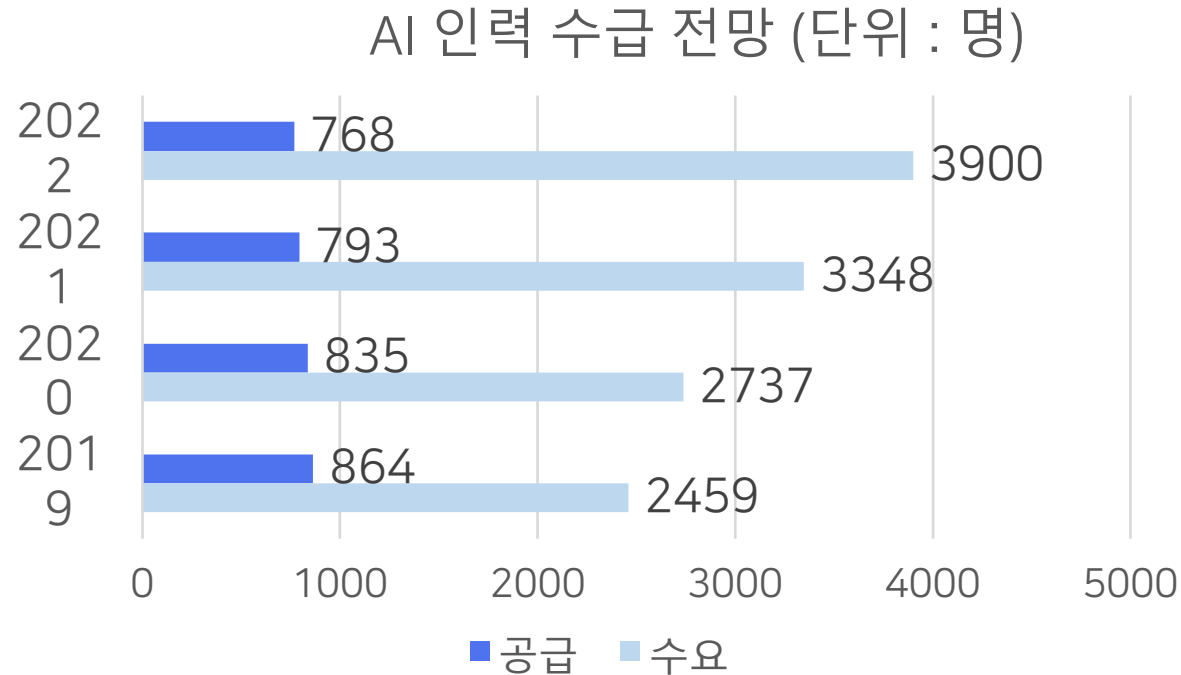


AI 개발자에 대한 수요는 증가
IT부서 외 금융 마케팅 등 일반 부서
에서도 인공지능 개발자 필요

출처 : 가트너 탠런트 뉴런 (2020년 3월)



3. 부족한 머신러닝 개발자



출처 : 소프트웨어 정책 연구소

전세계 에서 현재 AI 업체가 필요로 하는
ML Engineer수는 100만명 수준이나
활동중인 인력은 30만명수준

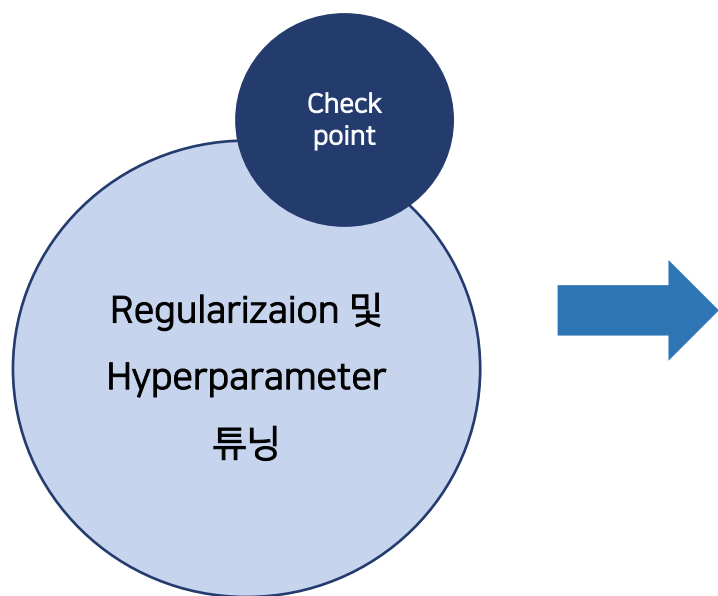
출처:정보통신기술진흥센터 17.12.14



4. 보편적인 머신 러닝 작업 흐름



5. 보편적인 머신 러닝 작업 흐름 - 머신 러닝 모델 튜닝 방법

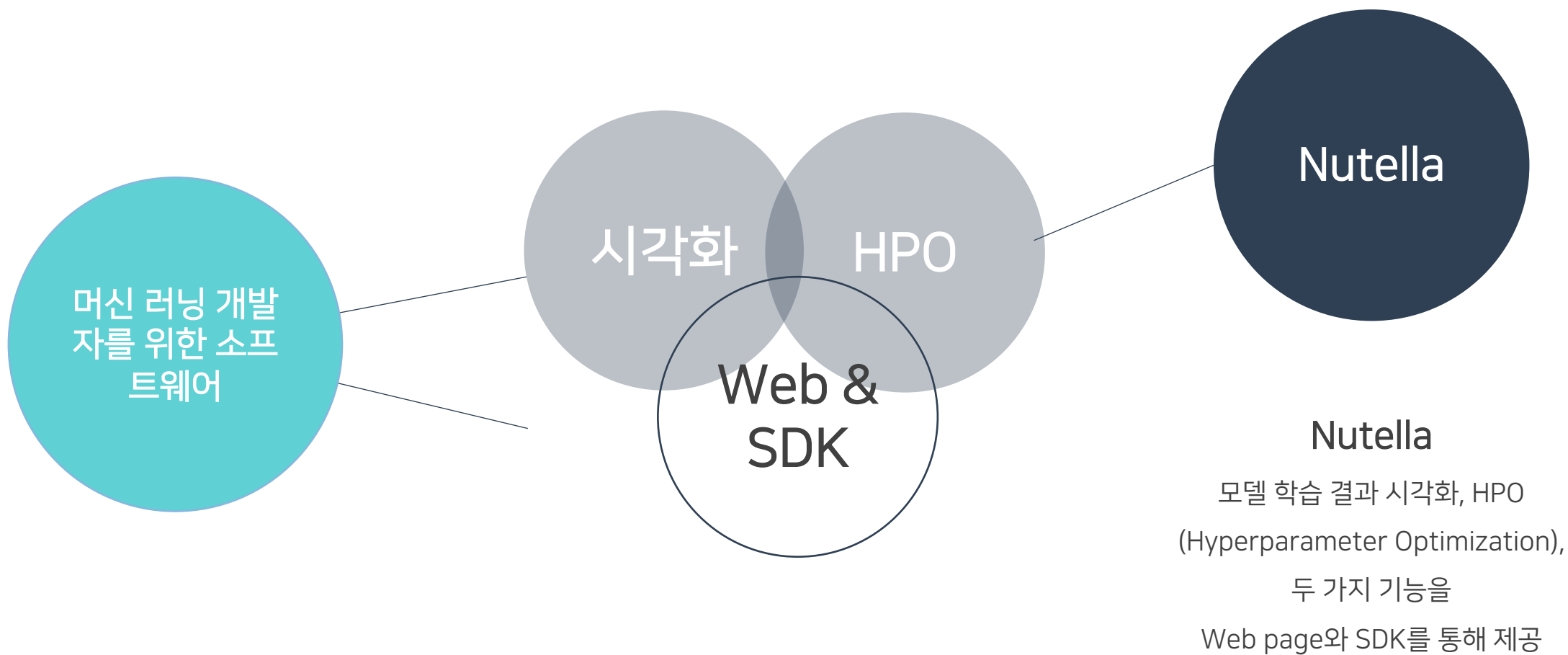


학습 알고리즘의 지표를 모니터링하고 시각화 하면서
아래의 과정을 진행해야 함

1. dropout 추가
2. layer 변경
3. hyperparameter 변경
4. 새로운 feature 추가 또는 기존 feature 제거



6. 프로젝트 개요





7. 시스템 효과 및 기능

시각화

- ML 학습 모델의 최적화 (Model Optimization)
- 데이터 셋 버전 관리 (Data Set Versioning)
- 실험 결과 트래킹 (Experiment Tracking)

HPO

- HPO (HyperParameter Optimization,
ML 엔지니어 없이 Hyperparameter를 최적
으로 선정)를 통해 최적의 결과를 도출



Nutella

II . 시스템 요청사항



1. 타겟 유저



머신러닝에 대한

지식과 경험이 부족한 개발자

- 머신러닝 실험 결과를 편하게 보고 관리하고 싶다
- SOTA 모델은 구했지만, 자신의 데이터에 맞게 튜닝하는 것이 어렵다
 - 자신의 모델을 쉽게 경량화시키고 싶다
- 위의 작업들을 하는데 초기 설정조차 어떻게 해야할지 모르겠다



Nutella

간편하고 이용하기 쉬운 서비스



2. 보편적인 지표 모니터링 방법 (1)

```
Epoch 1/20
422/422 [=====] - 1s 2ms/step - loss: 2.7317 - acc: 0.1804 - val_loss: 2.0294 - val_acc: 0.2120
Epoch 2/20
422/422 [=====] - 1s 2ms/step - loss: 2.0001 - acc: 0.2330 - val_loss: 1.8640 - val_acc: 0.2665
Epoch 3/20
422/422 [=====] - 1s 2ms/step - loss: 1.7407 - acc: 0.3181 - val_loss: 1.6251 - val_acc: 0.3378
Epoch 4/20
422/422 [=====] - 1s 2ms/step - loss: 1.5870 - acc: 0.3513 - val_loss: 1.5251 - val_acc: 0.3982
Epoch 5/20
422/422 [=====] - 1s 2ms/step - loss: 1.5120 - acc: 0.3856 - val_loss: 1.4809 - val_acc: 0.3907
Epoch 6/20
422/422 [=====] - 1s 2ms/step - loss: 1.4010 - acc: 0.4511 - val_loss: 1.2166 - val_acc: 0.5797
Epoch 7/20
422/422 [=====] - 1s 2ms/step - loss: 1.0575 - acc: 0.6449 - val_loss: 0.9362 - val_acc: 0.7223
Epoch 8/20
422/422 [=====] - 1s 2ms/step - loss: 0.6938 - acc: 0.7830 - val_loss: 0.5084 - val_acc: 0.8652
Epoch 9/20
422/422 [=====] - 1s 2ms/step - loss: 0.4687 - acc: 0.8784 - val_loss: 0.4169 - val_acc: 0.9005
Epoch 10/20
422/422 [=====] - 1s 2ms/step - loss: 0.3512 - acc: 0.9095 - val_loss: 0.3200 - val_acc: 0.9342
Epoch 11/20
422/422 [=====] - 1s 2ms/step - loss: 0.2760 - acc: 0.9287 - val_loss: 0.2745 - val_acc: 0.9442
Epoch 12/20
422/422 [=====] - 1s 2ms/step - loss: 0.2323 - acc: 0.9392 - val_loss: 0.2327 - val_acc: 0.9472
Epoch 13/20
422/422 [=====] - 1s 2ms/step - loss: 0.1939 - acc: 0.9477 - val_loss: 0.2181 - val_acc: 0.9478
Epoch 14/20
422/422 [=====] - 1s 2ms/step - loss: 0.1647 - acc: 0.9553 - val_loss: 0.1835 - val_acc: 0.9567
```

학습 결과 지표를 단순히 출력



2. 보편적인 지표 모니터링 방법 (1)

```
Epoch 1/20
422/422 [=====] - 1s 2ms/step - loss: 2.7317 - acc: 0.1804 - val_loss: 2.0294 - val_acc: 0.2120
Epoch 2/20
422/422 [=====] - 1s 2ms/step - loss: 2.0001 - acc: 0.2330 - val_loss: 1.8640 - val_acc: 0.2665
Epoch 3/20
422/422 [=====] - 1s 2ms/step - loss: 1.7407 - acc: 0.3181 - val_loss: 1.6251 - val_acc: 0.3378
Epoch 4/20
422/422 [=====] - 1s 2ms/step - loss: 1.5870 - acc: 0.3513 - val_loss: 1.5251 - val_acc: 0.3982
Epoch 5/20
422/422 [=====] - 1s 2ms/step - loss: 1.4500 - acc: 0.3845 - val_loss: 1.4809 - val_acc: 0.3907
Epoch 6/20
422/422 [=====] - 1s 2ms/step - loss: 1.3130 - acc: 0.4177 - val_loss: 1.2166 - val_acc: 0.5797
Epoch 7/20
422/422 [=====] - 1s 2ms/step - loss: 1.1760 - acc: 0.4509 - val_loss: 0.9362 - val_acc: 0.7223
Epoch 8/20
422/422 [=====] - 1s 2ms/step - loss: 1.0390 - acc: 0.4841 - val_loss: 0.5084 - val_acc: 0.8652
Epoch 9/20
422/422 [=====] - 1s 2ms/step - loss: 0.9020 - acc: 0.5173 - val_loss: 0.4169 - val_acc: 0.9005
Epoch 10/20
422/422 [=====] - 1s 2ms/step - loss: 0.7650 - acc: 0.5505 - val_loss: 0.3200 - val_acc: 0.9342
Epoch 11/20
422/422 [=====] - 1s 2ms/step - loss: 0.6280 - acc: 0.5837 - val_loss: 0.2745 - val_acc: 0.9442
Epoch 12/20
422/422 [=====] - 1s 2ms/step - loss: 0.4910 - acc: 0.6169 - val_loss: 0.2327 - val_acc: 0.9472
Epoch 13/20
422/422 [=====] - 1s 2ms/step - loss: 0.3540 - acc: 0.6501 - val_loss: 0.2181 - val_acc: 0.9478
Epoch 14/20
422/422 [=====] - 1s 2ms/step - loss: 0.2170 - acc: 0.6833 - val_loss: 0.1835 - val_acc: 0.9567
```

학습 경과를 파악하기 어려움

학습 결과 지표를 단순히 출력



2. 보편적인 지표 모니터링 방법 (2)

```
import matplotlib.pyplot as plt

def show_graph(history_dict):
    accuracy = history_dict['acc']
    val_accuracy = history_dict['val_acc']
    loss = history_dict['loss']
    val_loss = history_dict['val_loss']

    epochs = range(1, len(loss) + 1)

    plt.figure(figsize=(16, 1))

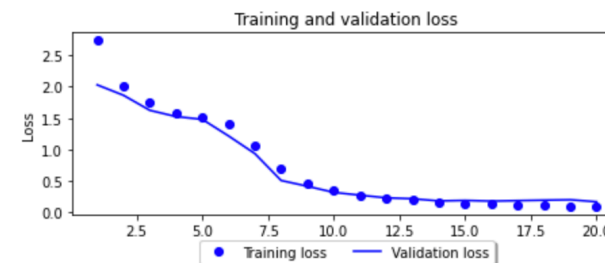
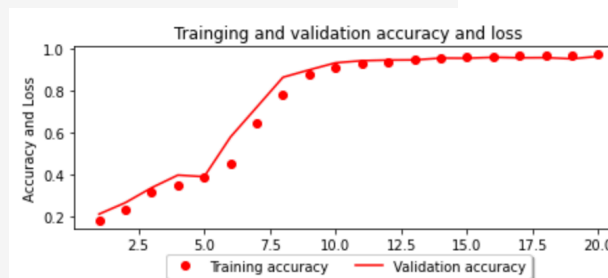
    plt.subplot(121)
    plt.subplots_adjust(top=2)
    plt.plot(epochs, accuracy, 'ro', label='Training accuracy')
    plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')
    plt.title('Trainging and validation accuracy and loss')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy and Loss')

    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
              fancybox=True, shadow=True, ncol=5)

    plt.subplot(122)
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
              fancybox=True, shadow=True, ncol=5)

    plt.show()

show_graph(history.history)
```



라이브러리를 이용해 직접 그래프를 작성



2. 보편적인 지표 모니터링 방법 (2)

```
import matplotlib.pyplot as plt

def show_graph(history_dict):
    accuracy = history_dict['acc']
    val_accuracy = history_dict['val_acc']
    loss = history_dict['loss']
    val_loss = history_dict['val_loss']

    epochs = range(1, len(loss) + 1)

    plt.figure(figsize=(16, 1))

    plt.subplot(121)
    plt.subplots_adjust(top=2)
    plt.plot(epochs, accuracy, 'ro', label='Training accuracy')
    plt.plot(epochs, val_accuracy, 'r', label='Validation accuracy')
    plt.title('Training and validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('Accuracy and Loss')

    plt.legend(loc='upper center', bbox_to_anchor=(0.5, 1.1),
              fancybox=True, shadow=True, ncol=5)

    plt.subplot(122)
    plt.plot(epochs, loss, 'bo', label='Training loss')
    plt.plot(epochs, val_loss, 'b', label='Validation loss')
    plt.title('Training and validation loss')
    plt.xlabel('Epochs')
    plt.ylabel('Loss')
    plt.legend(loc='upper center', bbox_to_anchor=(0.5, -0.1),
              fancybox=True, shadow=True, ncol=5)

    plt.show()

show_graph(history.history)
```

매번 그래프를 그리기 불편함



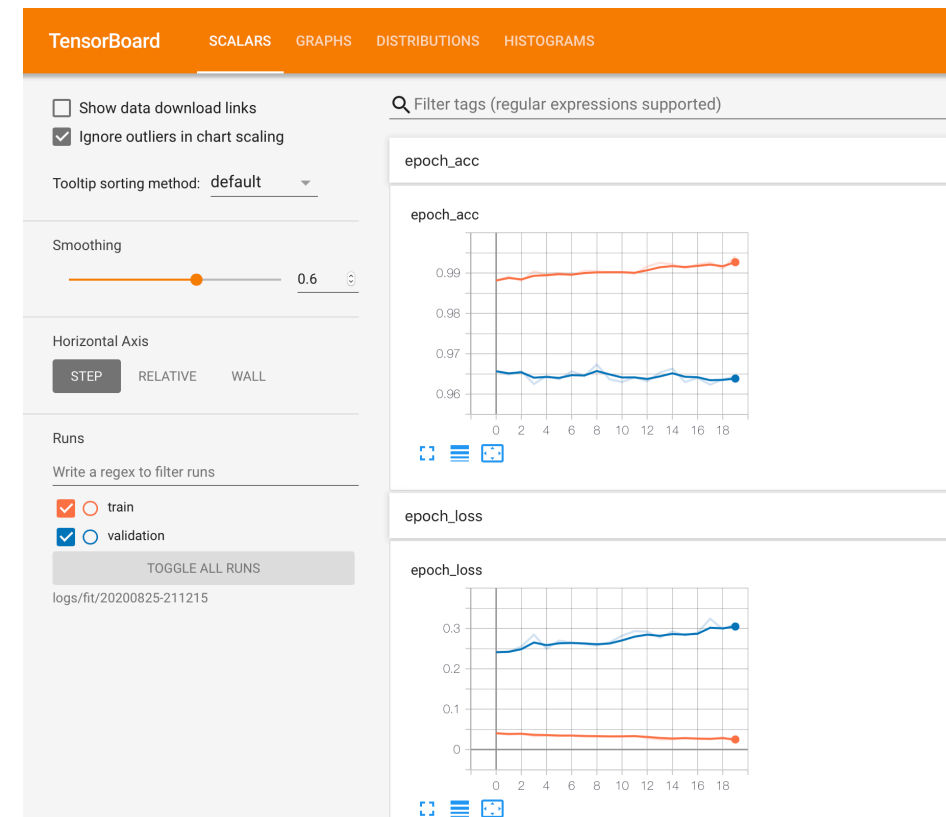
라이브러리를 이용해 직접 그래프를 작성

2. 보편적인 지표 모니터링 방법 (3)

```
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

#training
history = model.fit(
    reshape_x_train,
    y_train,
    batch_size=128,
    epochs=20,
    validation_split=.1,
    callbacks=[tensorboard_callback]
)

#visualization
%load_ext tensorboard
%tensorboard --logdir {log_dir}
```



학습 시 코드에서 로그를 남겨 Tensorboard와 연동한 뒤 로컬에서 시각화



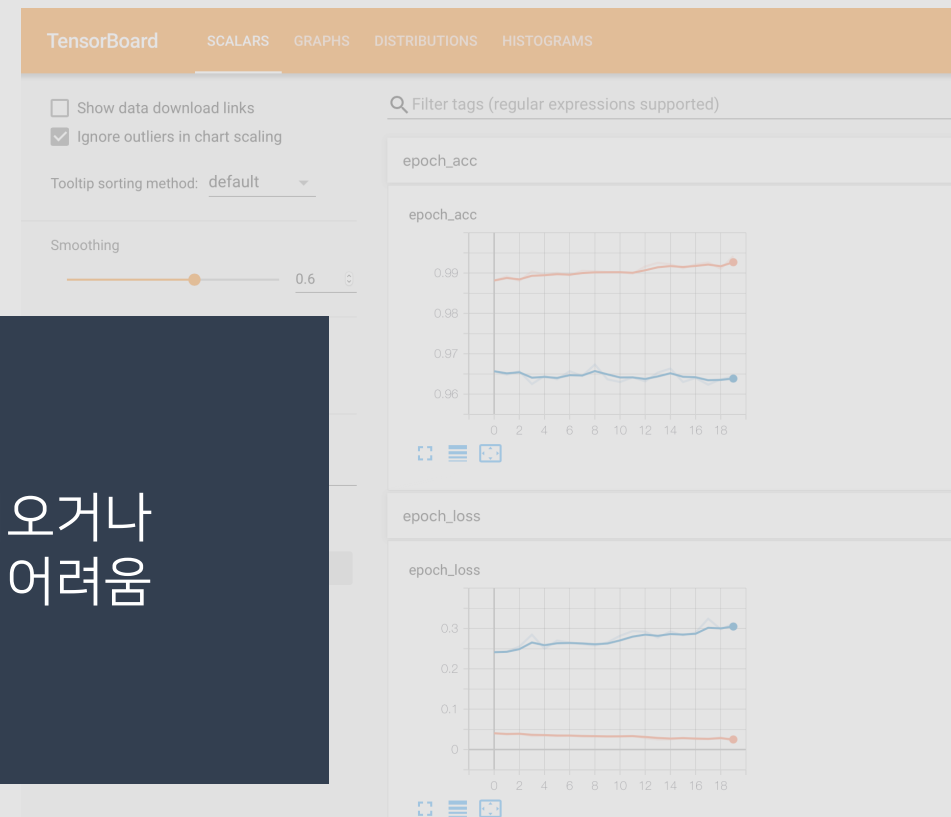
2. 보편적인 지표 모니터링 방법 (3)

```
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir, histogram_freq=1)

#training
history = model.fit(
    reshape_x_train,
    y_train,
    batch_size=128,
    epochs=20,
    validation_split=.1,
    callbacks=[tensorboard_callback]
)

#visualization
%load_ext tensorboard
%tensorboard --logdir {log_dir}
```

나중에 결과를 다시 불러오거나
다른 사람과의 공유하기 어려움



학습 시 코드에서 로그를 남겨 Tensorboard와 연동한 뒤 로컬에서 시각화

II. 시스템 요청사항



Nutella

3. Nutella 기능 목표 - 시각화

```
import NutellaML as nutella

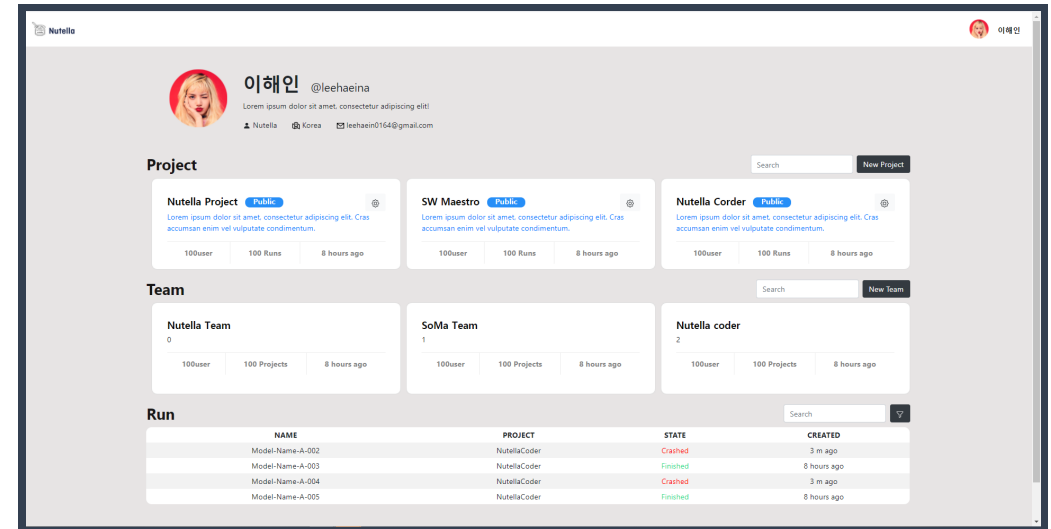
nutella.init(modelName="project_test", projectKey="6edBe32ESdfABETS3evdcjke")
nutella.config(epochs=epochs, batchSize=batchSize)

history = model.fit(
    reshape_x_train,
    y_train,
    batch_size=128,
    epochs=20, #50,
    validation_split=.1
)

acc = history.history['acc']
nutella.log(accuracy=acc)
```

간편함

코드 상에서 NutellaML을 import 하기
만 하면 사용 가능



용이함

한 번 시각화 한 자료는 언제 어디서나, 누구나 (등록한
유저에 한해서) web page 에서 접근 가능
=> 팀원 간 공유 및 학습 후 재 접근 용이



3. Nutella 기능 목표 - HPO

간편함

코드 상에서도 웹페이지에서도 config
를 설정할 수 있고 이에 따른 쉬운 함수
로 구성되어 있음

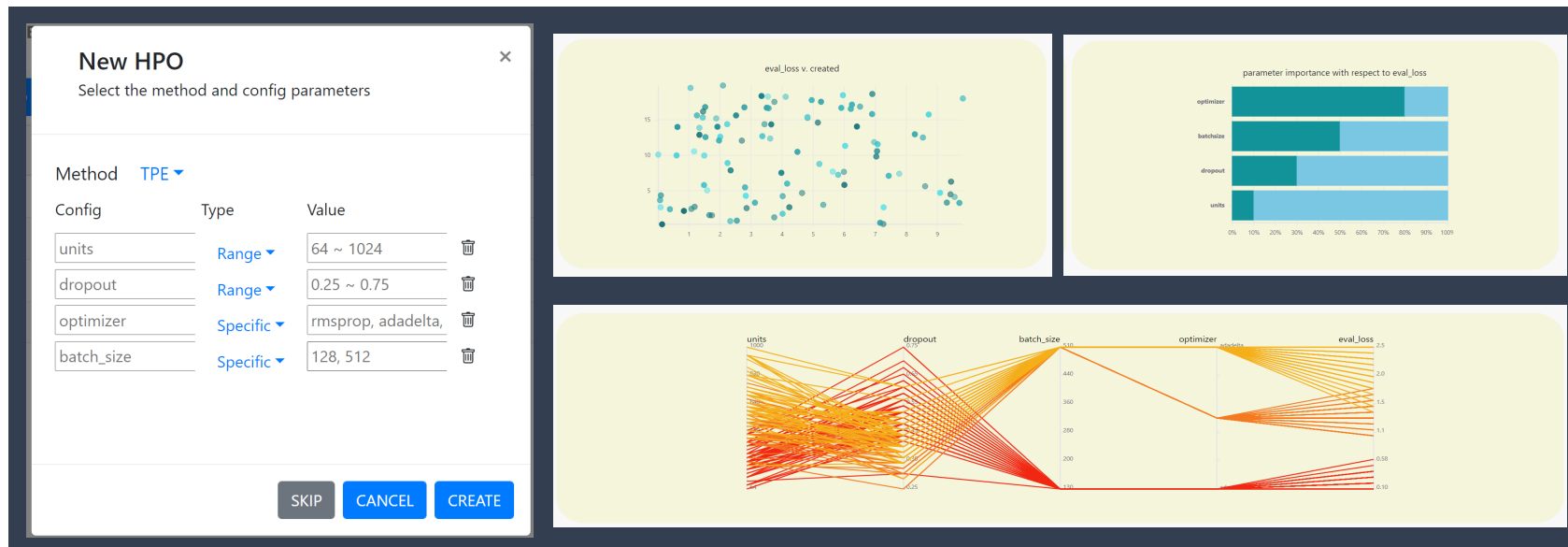
```
# 웹 페이지에서 config 설정한 경우
nu_simple_fmin(objective = lstm_model)

# 코드에서 config 설정한 경우
nu_fmin(objective = lstm_model, space = search_space, algo = tpe, trials = Trials())
```

Python ▾

용이함

HPO를 실행하면 자동으로 하이퍼 파라미
터들의 중요도를 계산해주고 이를 시각화
해서 보여주므로, 결과 해석과 모델 최적화
에 매우 용이함.





Nutella

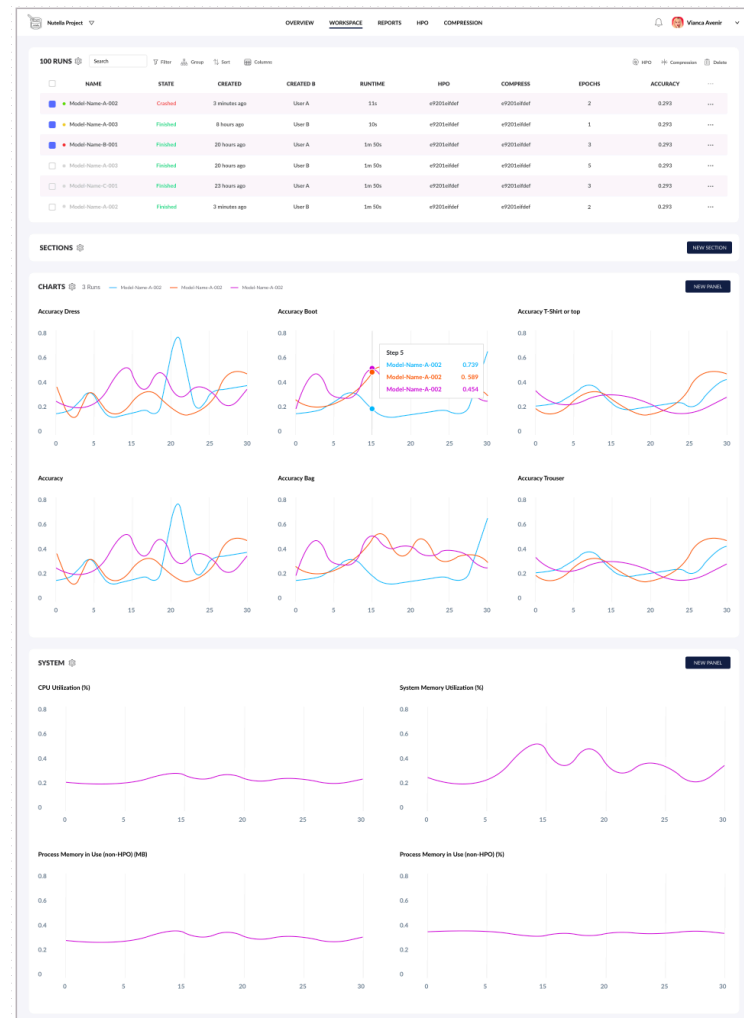
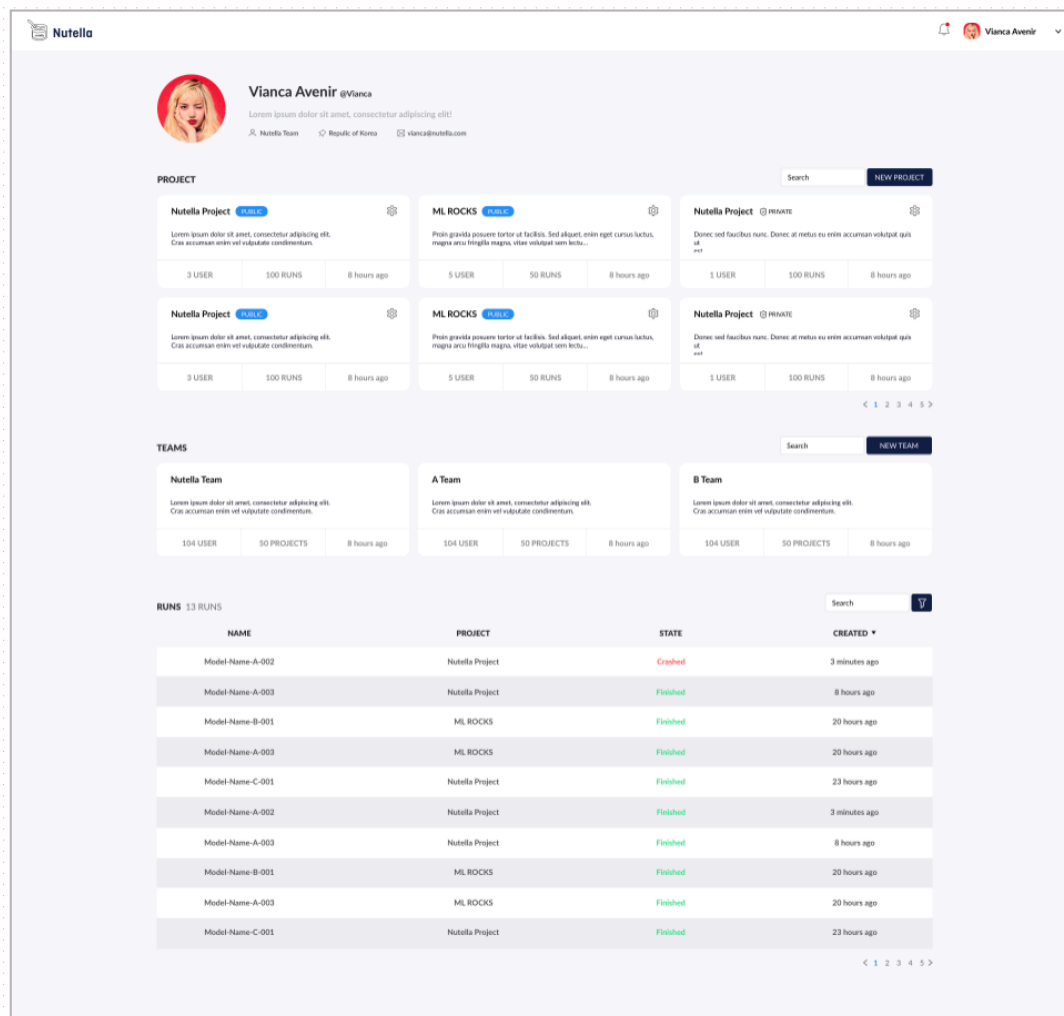
III . 시스템 Image

III. 시스템 Image



Nutella

1. 시각화 Image



III. 시스템 Image



2. HPO Image

New HPO
Create a New HPO Task

HPO Name

h9201eifdef

Description

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras accumsan enim vel vulputate condimentum.

CANCEL

NEXT

New HPO
Create a New HPO Task

Your HPO API Key is

...Xxxx0xxx-d-xdxxxxxxxxdxdxdx...

DONE

New HPO
Select the method and config parameters

Method

TPE

+ NEW CONFIG

Config	Type	Value
epochs	Range	1 - 10
bagging_fraction	Specific	0.6, 0.7, 0.8
bagging_freq	Specific	7
handle	Specific	clemendelangue
lr_scheduler	Specific	constant,linear, cosine
Parameter name	Specific	e.g. 0.6, 0.8 and linear

SKIP

CANCEL

CREATE

Nutella Project > HPO > h9201eifdef

OVERVIEW

CHARTS

SYSTEM

h9201eifdef

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras accumsan enim vel vulputate condimentum.

Privacy: PUBLIC

Created By: Vianca Avenir

Created Time: 2020-07-01 10:00:00

State: Running

Run Count: 20

Compute Time: 18 hours

HPO CONFIG 6 CONFIGS

CONFIG	TYPE	VALUE	BEST VALUE
method	-	TPE	
epochs	Range	1 - 10	1.3
bagging_fraction	Specific	0.6, 0.7, 0.8	0.6
bagging_freq	Specific	7	3
handle	Specific	clemendelangue	clemendelangue
lr_scheduler	Specific	constant, linear, cosine	linear

23

감사합니다.

Q & A