인공지능응용 기계공학 텀프로젝트

세종 랜드마크 챌린지
(Sejong Landmark Challenge)

4조

14011541 김 형준

16011504 임 진욱

18011230 배 대위

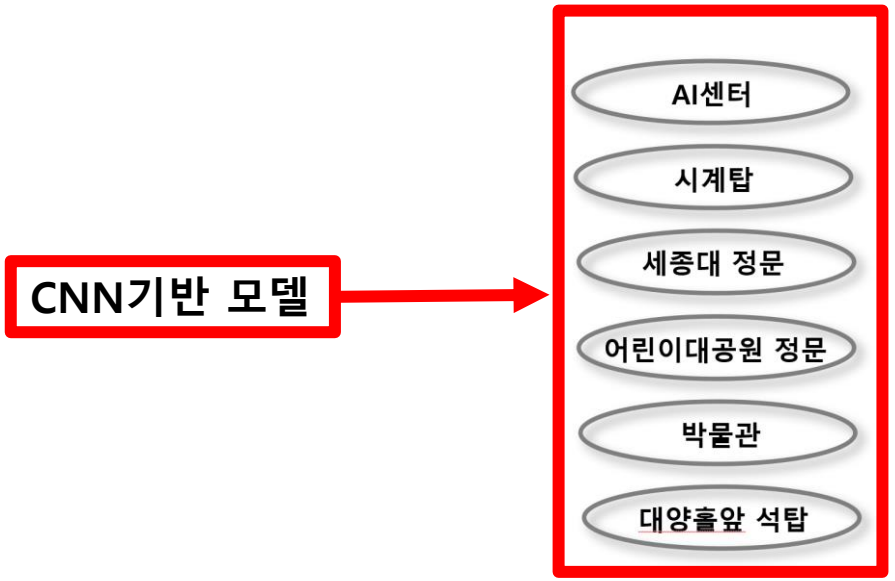
2020. 12. 17 최종발표

목차

I. Sejong Landmark Challenge (SLC)

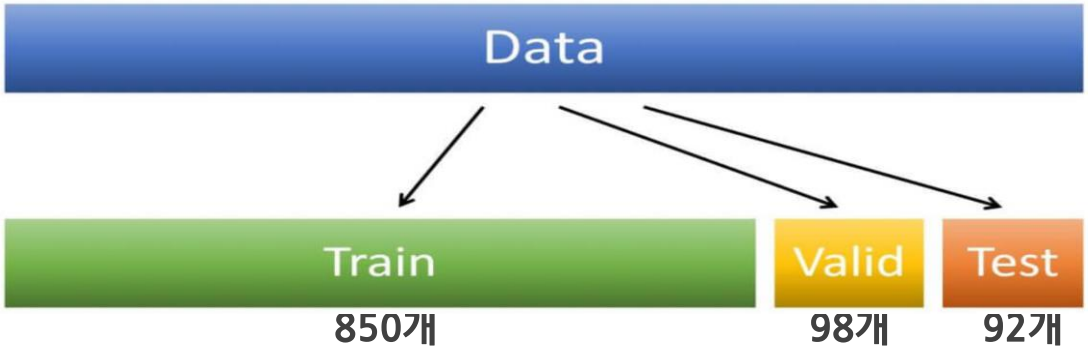
Sejong Landmark Challenge란?

- 세종대학교 건물 이미지를 분류하는 알고리즘 제작



이미지를 6개의 라벨로 분류

II. SLC Dataset



AI센터

시계탑

세종대 정문

어린이대공원 정문

박물관

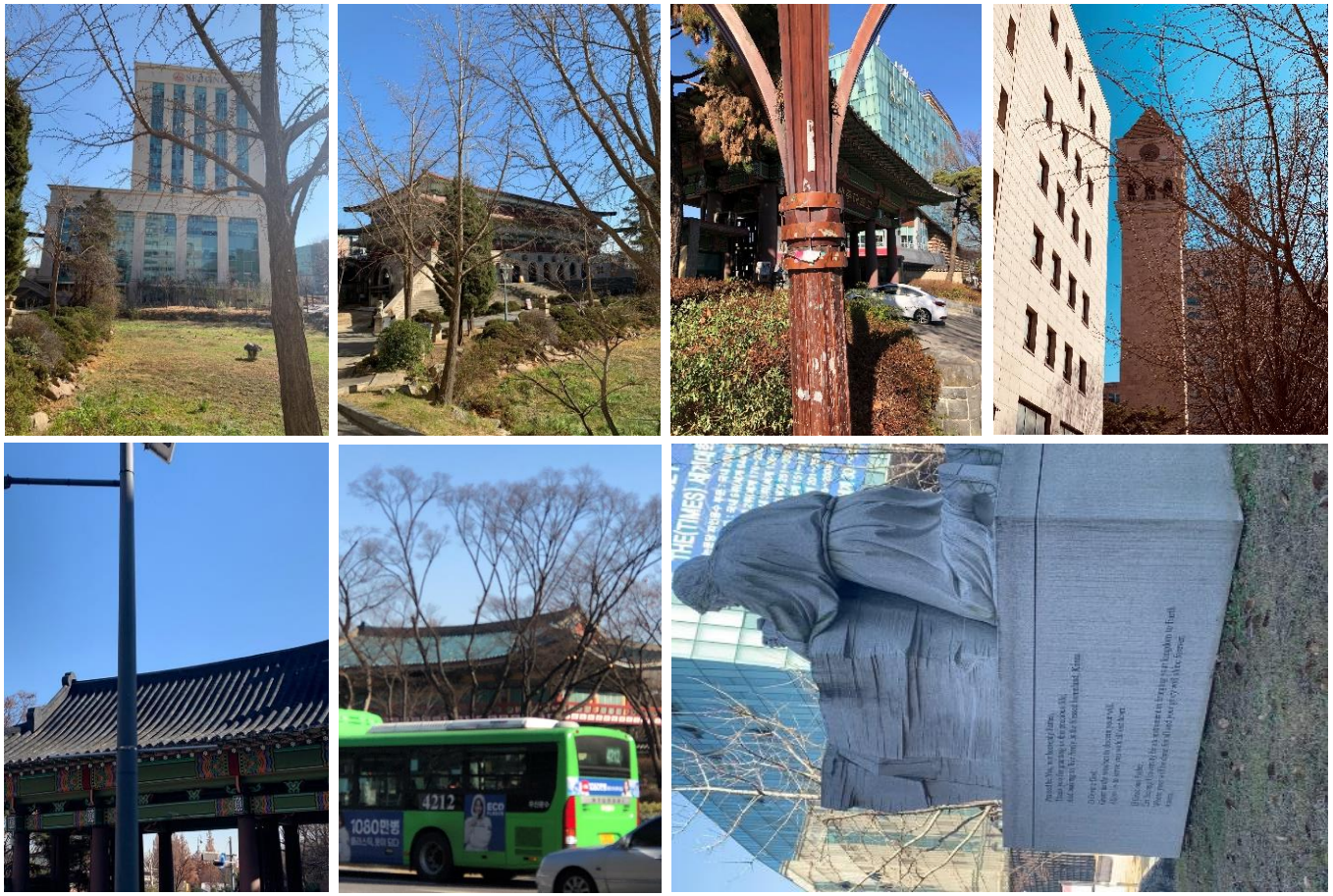
대양홀앞 석탑

Train: 웹크롤링

Validation: 직접찍은 이미지 + 웹크롤링

Test: 직접찍은 이미지

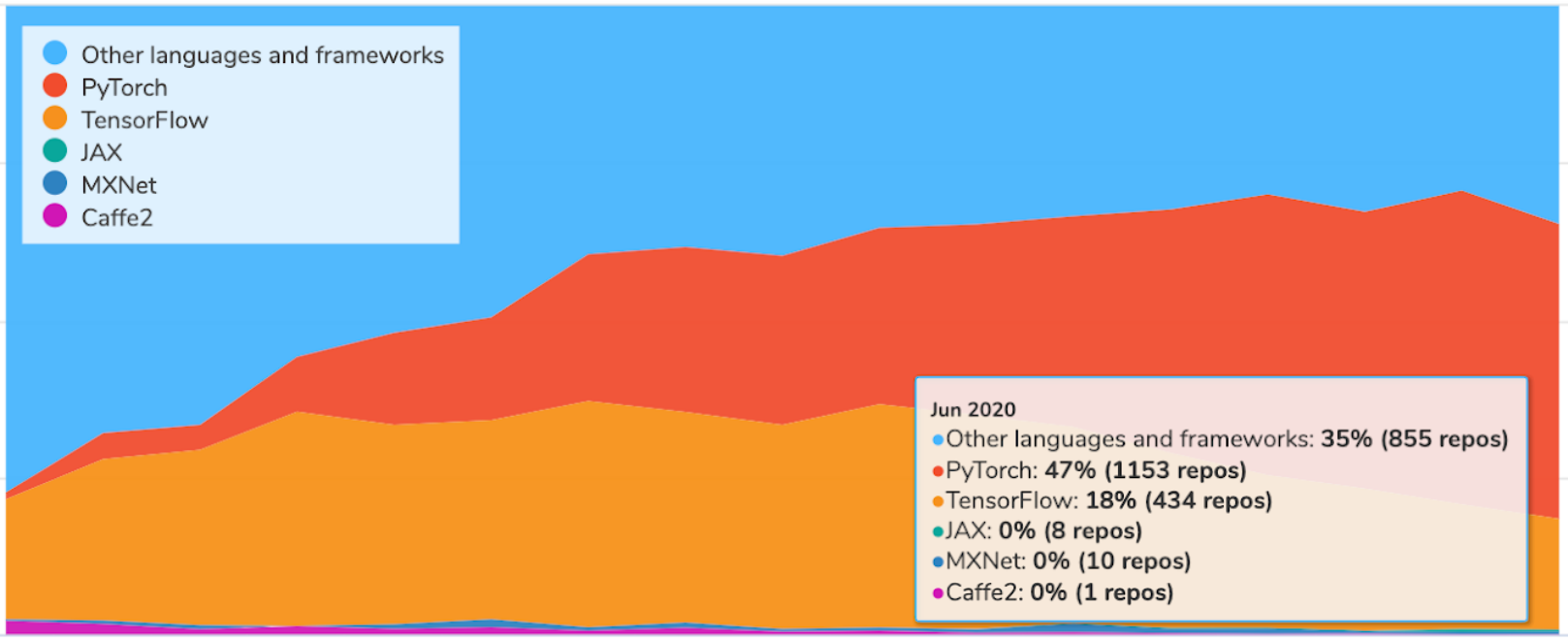
II. SLC Dataset



Test set에 회전, 광도, 스케일, 방해물등 노이즈를 넣어 Challenging하게 구성

III. 작업 환경

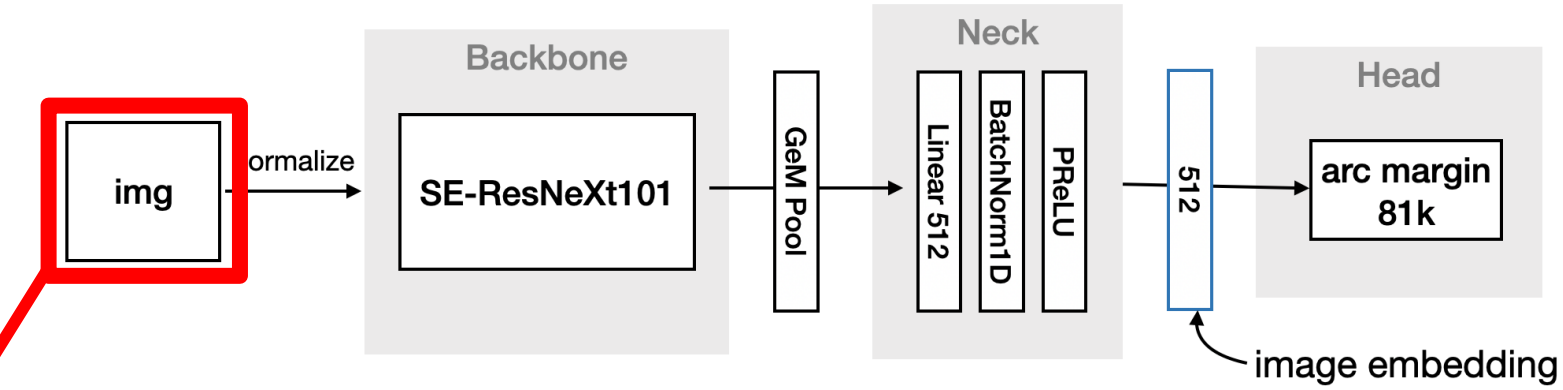
텐서플로우 vs 파이토치



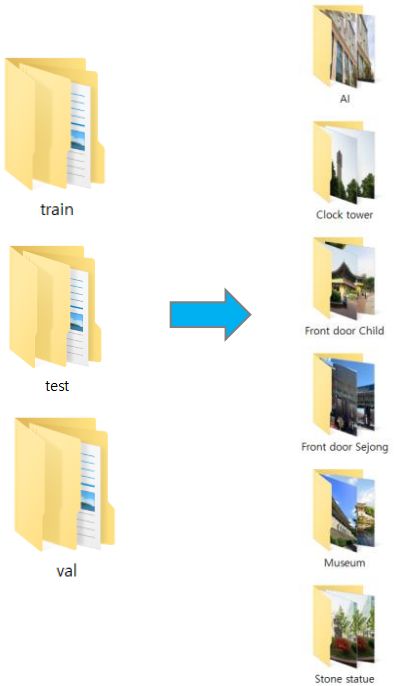
Github상 논문 구현언어 파이토치 점유율 [11]

→ 이와 더불어 디버깅상의 이점으로 파이토치 사용

IV. 파이프라인



1. 데이터 로드 및 가공



Data Augmentation

- 커스텀 데이터로더 설계
- Data Augmentation 필요

Ⅳ. 파이프라인

(1) 커스텀 데이터로더

class Custom_dataloader(Dataset):
    def __init__(self, mode , datapath, img_w, img_h, transform=None):
        → 클래스 호출시 실행되는 init함수

    def full_load(self):
        → 이미지가 저장된 폴더로부터 전체이미지 불러오기

    def csv_exist(self):
        → 이미지가 저장된 csv파일이 있으면 호출

    def __len__(self):
        return len(self.images)
        → 저장된 이미지의 길이 반환

    def __getitem__(self, idx):
        data = { ' image ' :img, ' label ' :label}
        return data
        → index값을 받아 해당 index의 이미지와 라벨을 딕셔너리 형태로 반환

IV. 파이프라인

(2) Data Augmentation

	A 0.4.2	Imgaug 0.3.0	Torchvision 0.4.1	Keras 2.3.1	Augmentor 0.2.6	Solt 0.1.8
HorizontalFlip	**2183**	1403	1757	1068	1779	1031
VerticalFlip	**4217**	2334	1538	4196	1541	3820
Rotate	**456**	368	163	32	60	116
ShiftScaleRotate	**800**	549	146	34	-	-
Brightness	**2209**	1288	405	211	403	2070
Contrast	**2215**	1387	338	-	337	2073
BrightnessContrast	**2208**	740	193	-	193	1060
ShiftRGB	**2214**	1303	-	407	-	-
ShiftHSV	**468**	443	61	-	-	144
Gamma	**2281**	-	730	-	-	925
Grayscale	**5019**	436	788	-	1451	4191
RandomCrop64	**173,877**	3340	43,792	-	36,869	36,178
PadToSize512	**2906**	-	553	-	-	2711
Resize512	663	506	**968**	-	954	673
RandomSizedCrop64_512	**2565**	933	1395	-	1353	2360
Equalize	**759**	457	-	-	684	-

2020년 발표된 [4] 논문 참고 ImageNet의 validation set 2000장에 대한

→ 이미지 처리속도가 가장 빠른 Albumentations 사용

IV. 파이프라인

(2) Data Augmentation

Torchvision 코드

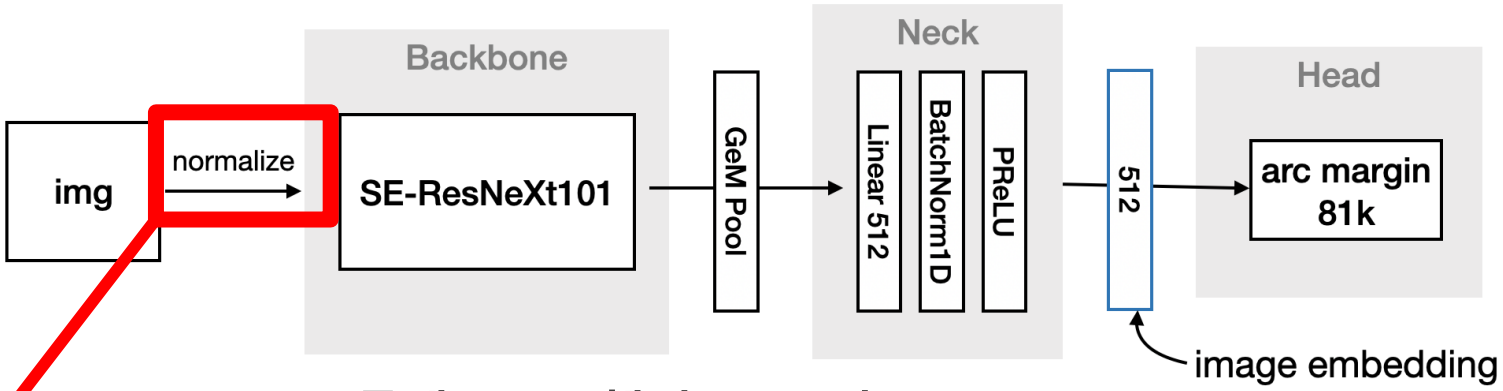
```
torchvision_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.RandomCrop(224),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
])
```

Albumentations 코드

```
albumentations_transform = albumentations.Compose([
    albumentations.Resize(256, 256),
    albumentations.RandomCrop(224, 224),
    albumentations.HorizontalFlip(),
    albumentations.pytorch.transforms.ToTensor()
])
```

→ torchvision 라이브러리랑 호환성이 좋음

IV. 파이프라인



구글랜드마크 챌린지 1등팀 논문 [2] 원복

2. Standardization으로 데이터 표준화

$$\frac{x-\mu}{\sigma} \quad (\mu : 평균, \sigma : 표준편차)$$

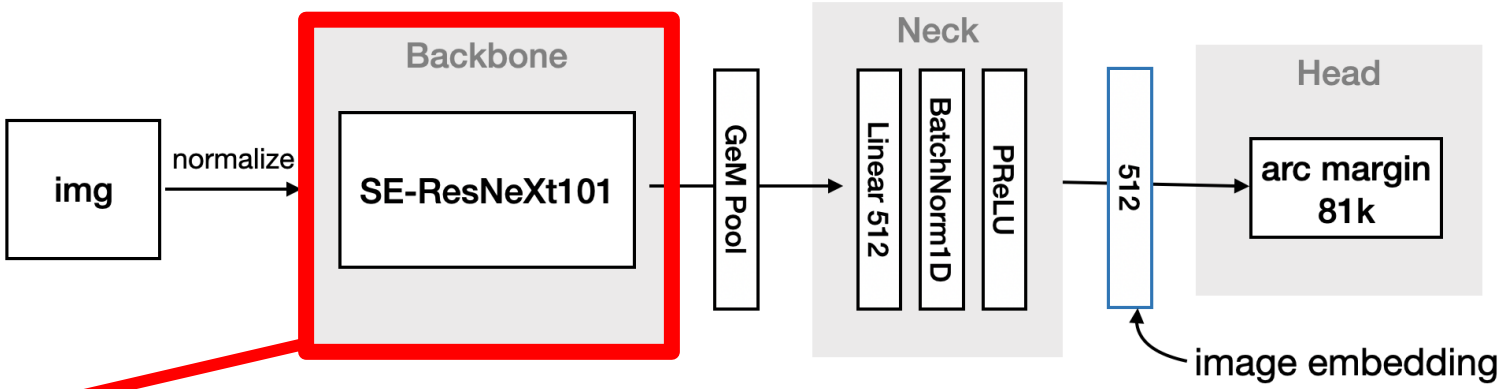
Train + validation 데이터 CSV파일로 가공

numpy사용 평균과 표준편차

→ 평균: [124.5, 125.58, 118.67]

→ 표준편차: [70.97, 70.39, 78.2]

IV. 파이프라인



3. Timm 라이브러리 사용 Backbone 교체실험

참고사항

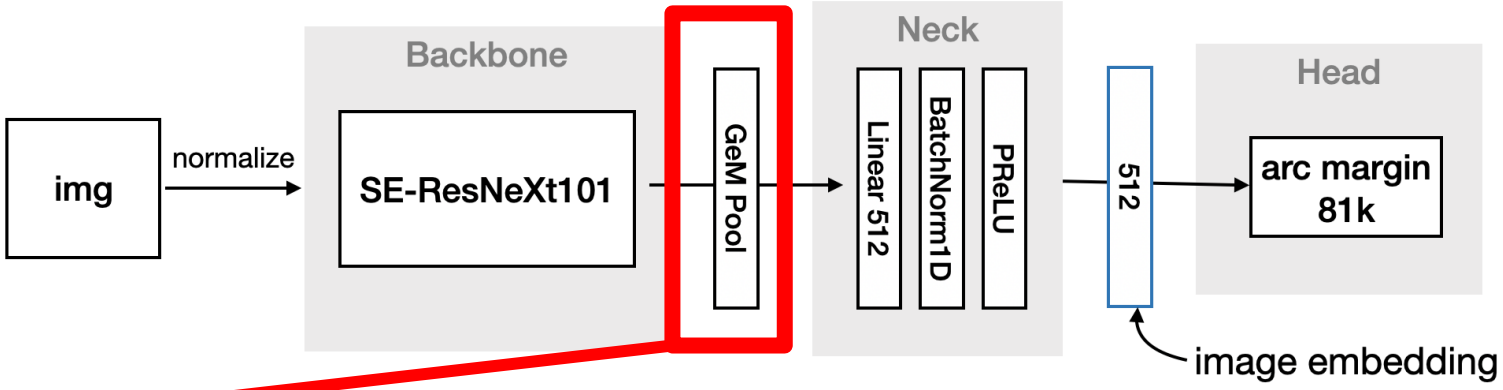
① 구글랜드마크 1등 팀 논문 [2]

② ImageNet 기준 모델 순위 [5]

③ 코랩 작업환경 고려 실험모델 선정

→ [5]참고 ImagNet 기준 상위에 랭크된 모델 중 너무 무겁지않은 모델들 사용

IV. 파이프라인



4. GeM Pooling 사용

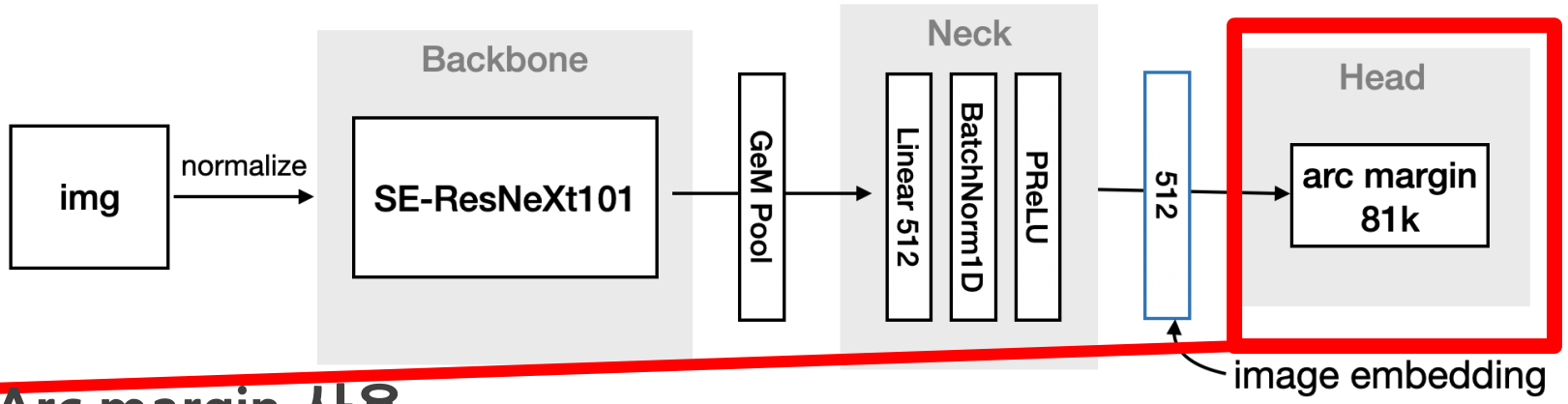
$$\mathbf{f}^{(g)} = [\mathrm{f}_1^{(g)} \dots \mathrm{f}_k^{(g)} \dots \mathrm{f}_K^{(g)}]^\top, \quad \mathrm{f}_k^{(g)} = \left(\frac{1}{|\mathcal{X}_k|} \sum_{x \in \mathcal{X}_k} x^{p_k} \right)^{\frac{1}{p_k}}.$$

$P_k \rightarrow \infty$: Max pooling

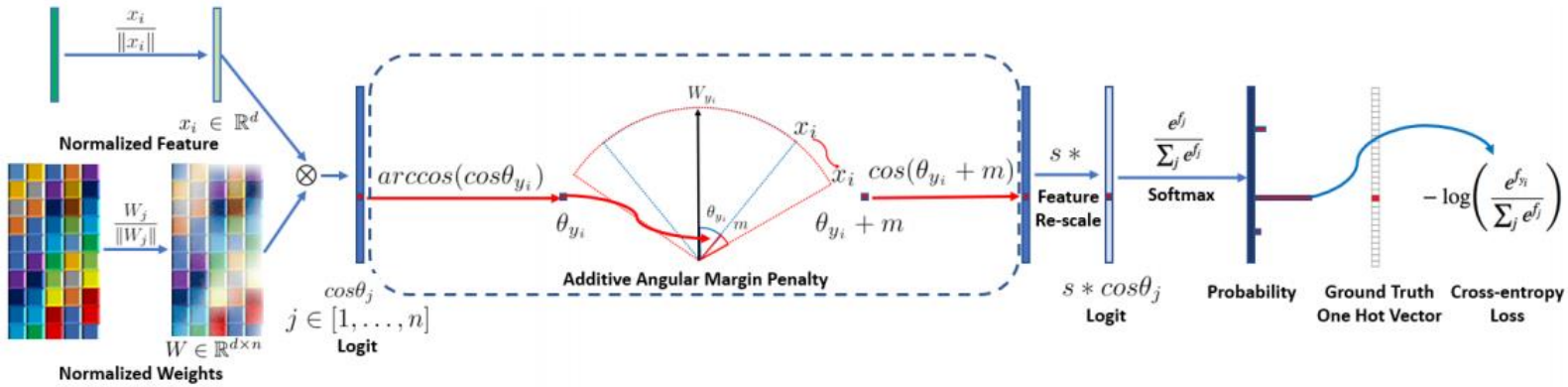
$P_k = 1$: Average pooling

미분가능 → P_K 학습가능

IV. 파이프라인



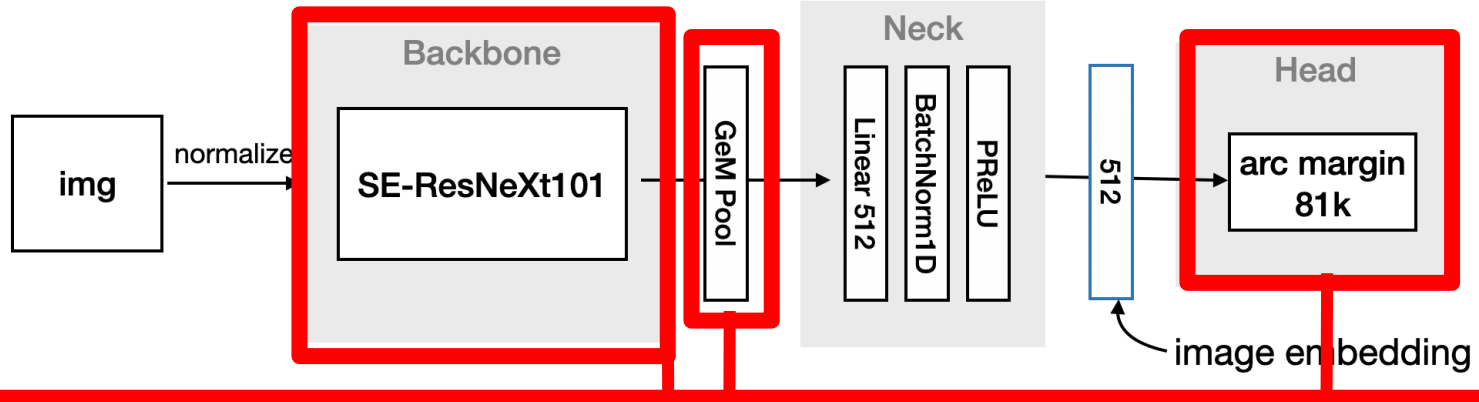
5. Arc margin 사용



[9] 논문 참고 Arc margin 부분적 구현

State of the Art(SOTA) 논문들에 주로 등장

IV. 파이프라인




```
class ArcMarginProduct(nn.Module):
    # Arc margin
```

```
class GeM(nn.Module):
    # Trainable GeM Pooling
```

```
class Backbone(nn.Module):
    # Backbone network 선언
```

총 4개의 class로 분할 (객체지향)

↓

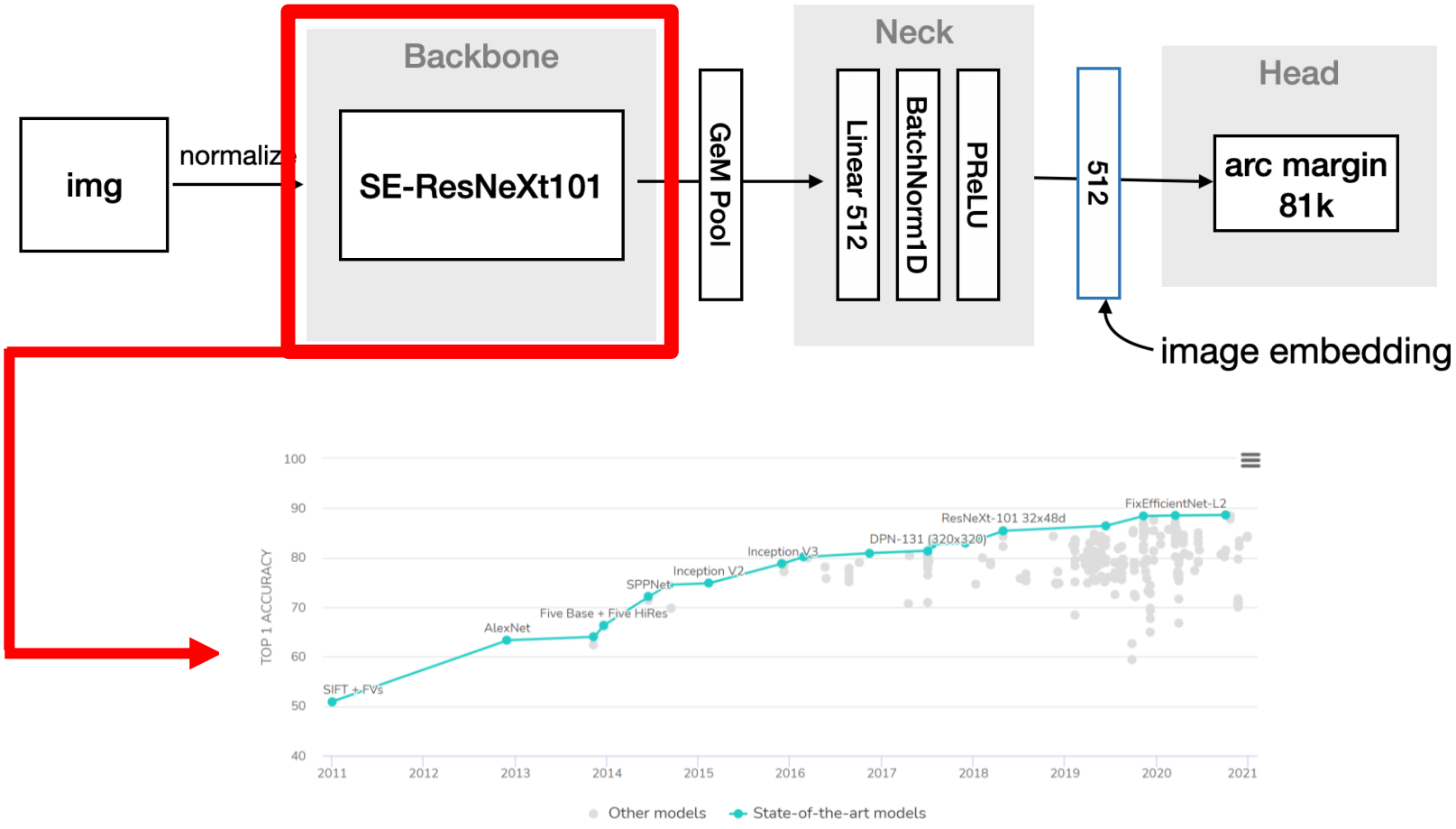
Net class에서 앞의 3개의 class를 불러옴

Feedforward 연산

```
class Net(nn.Module):
    def __init__(self, ...중략):
        # 위의 클래스들 불러오기
        # Neck설계 (sequential 이용)
    def forward(self, image, ...중략):
        # 이미지 불러온 클래스들 통과
        return x # 통과한 결과 반환
```

V. 모델 학습

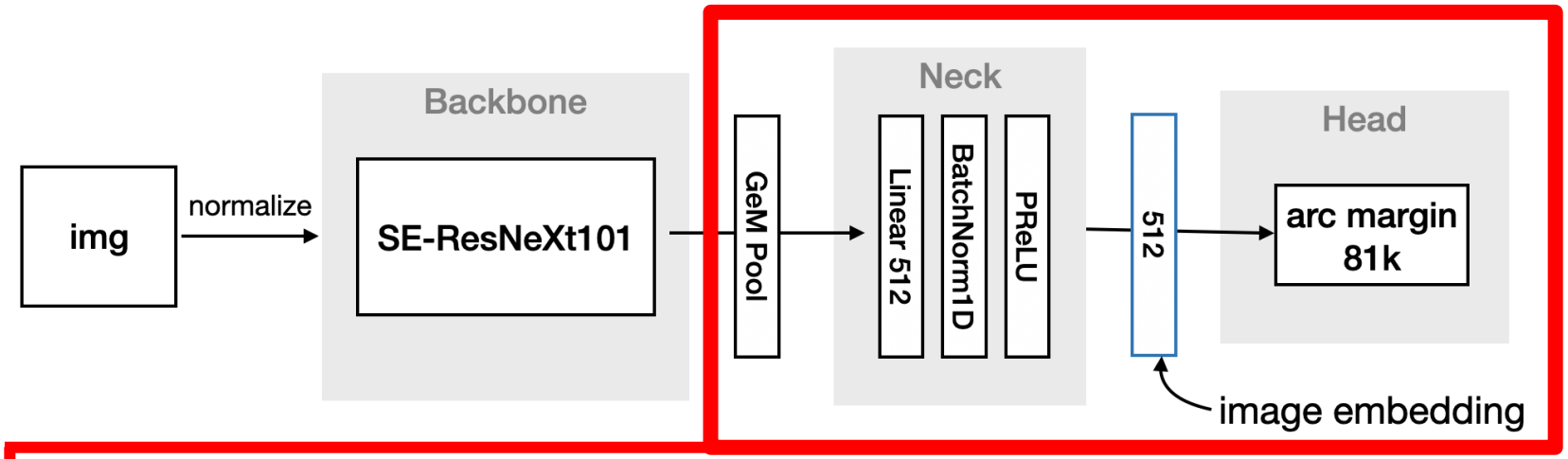
[5] 모델 순위, [2] 논문 기반 Backbone 선정



ImageNet 기준 모델 성능 순위[5]

V. 모델 학습

Freezing을 이용한 학습기법



Backbone은 pre-trained weight 사용

↓

빨간박스 부분만 학습 나머진 Freezing

↓

코랩 환경에서 큰 모델을 빠르게 학습시킬 수 있음

V. 모델 학습

Freezing을 이용한 학습기법 (코드)

1. 모델의 파라미터들 확인

```
names_pram=[] (파라미터들의 이름을 저장할 리스트)
for name_pram, _ in model.named_parameters():
    names_pram.append(name_pram)
pprint(names_pram)
```

2. names_pram에 저장된 파라미터 중 맨 뒤에서 9개만 빼고 나머진 Freeze

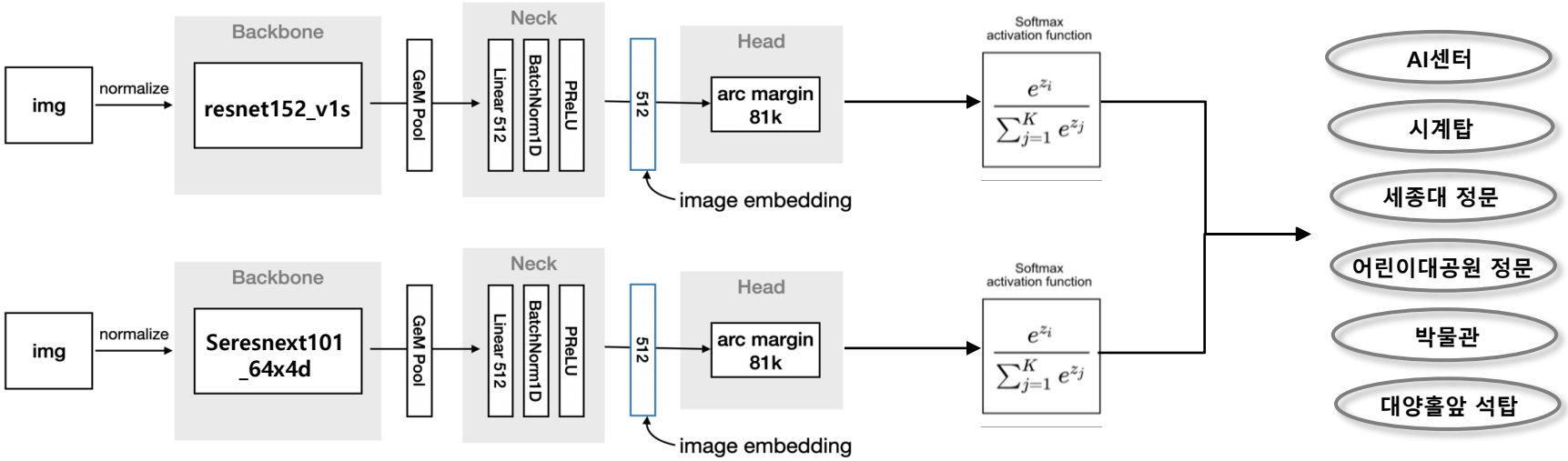
```
for name_pram, param in model.named_parameters():
    param.requires_grad = False
    if name_pram in names_pram[-9:]:
        param.requires_grad = True
```

'global_pool.p',
'neck.0.weight',
'neck.0.bias',
'neck.1.weight',
'neck.1.bias',
'neck.2.weight',
'neck.3.weight',
'neck.3.bias',
'head.weight'

해당 파라미터제외 Freeze

V. 모델 학습

앙상블



Softmax 결과를 더하여 argmax를 취해주었음

앙상블에 쓰일 모델들에 대한 Test Accuracy

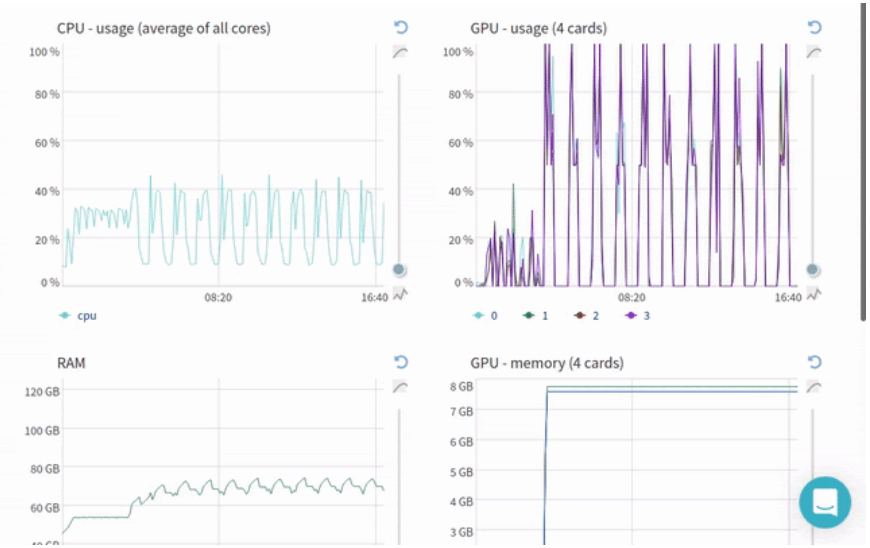
- **resnet152_v1s 90.21 %**
- **seresnext101_64x4d 88.04 %**

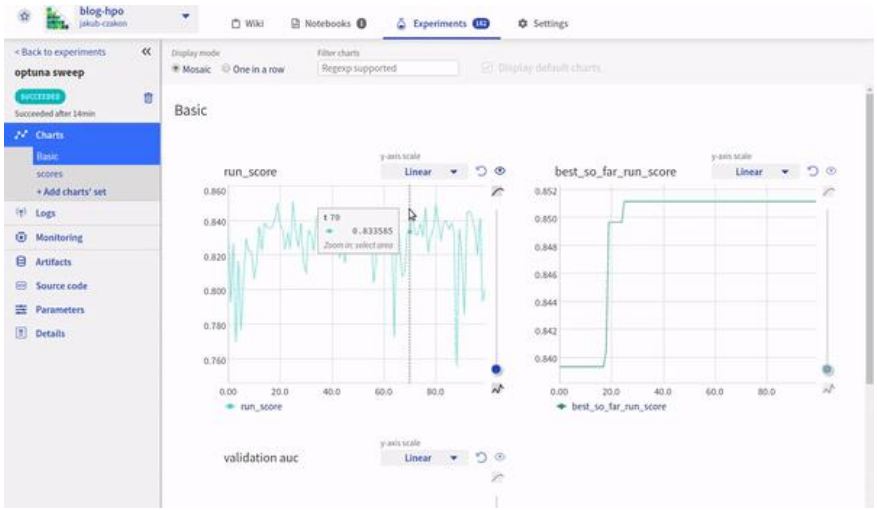
→ 실험을 통해 얻은 2개에 모델을 앙상블하여 92.38 %

VI. Neptune 시각화

Neptune의 장점

- 실험조건, 매트릭 백업 및 트랙킹
- 코드 백업
- GPU, CPU 사용량 기록
- 에러메시지 기록
- 원하는 변수로 실시간 Plotting
- 다양한 visulalization tool과 연동가능
- 코업환경에서 코업자와 교류가능





VI. Neptune 시각화

사용방법 소개

neptune.ai

1. 초기화

```
import neptune
# Neptune parameters
api_token="ANONYMOUS",
project_qualified_name='사용자명/저장소이름'
api_token='API 토큰 번호'
upload_source_files = '파일명.py' #백업할 파일
지정
# Neptune initialize
neptune.init(
    api_token=api_token,
    project_qualified_name=project_qualified_
name,
)
```

넵튠 공식사이트에서 회원가입 및 토큰발급 필요

VI. Neptune 시각화

사용방법 소개

2. 실험생성 및 실험변수 업데이트

```
def create_exp(name, params, upload_source_files):
    neptune.create_experiment(
    name=name,
    params=params,
    upload_source_files = upload_source_files
    )
```

3. 태그생성 및 매트릭 트랙킹

```
def create_tag(tag):
    return neptune.append_tags(tag)

def create_log_metric(name, val):
    return neptune.log_metric(name, val)
```

[1] https://neptune.ai 에서 **튜토리얼 제공** (동영상도 있음)

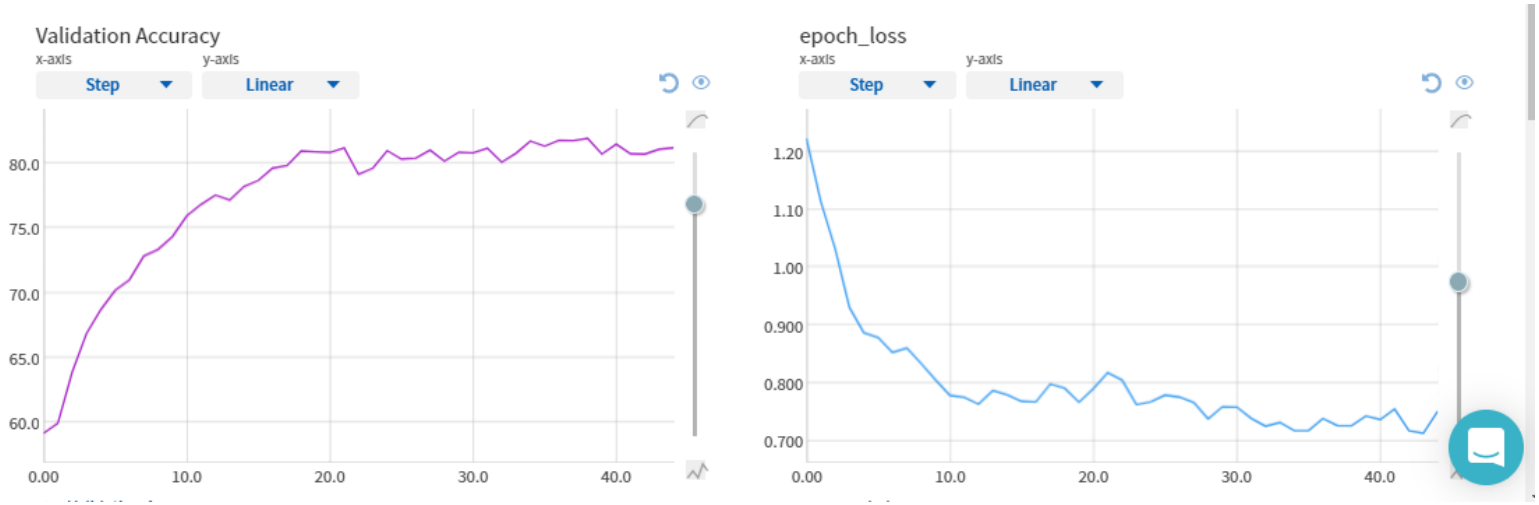
VI. Neptune 시각화

Search | Time: Any Time | Tags: Select tags | Experiment Status: Select status | Owned By: Select users

Go to advanced search BETA

	ID	Owner	Created	{string} Tags	number n_classes	number batch_size	number epoch_loss (last)
☐	SAN-29	hjkim	12 DEC 23:...	Second, Standardization applied	6	8	0.775936
☐	SAN-28	hjkim	12 DEC 20:...	First experiment	6	8	0.722849

실험 날짜, 조건, 변수, 주석 등 넵튠에 업데이트



원하는 변수 Tracing 및 실시간 plot

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
1	gluon_resnext101_64x4d	Standardization 적용	**90.22 %**
2	gluon_resnext101_64x4d	Data augmentation 적용	**67.39 %**

지나친 Data Augmentation으로 성능 감소 발생

확률값 p 낮춤

실험No.	모델명	주요 변동사항	Test Acc
3	gluon_resnext101_64x4d	Data agumentation 수정	**84.78 %**
4	gluon_resnext101_64x4d	Data agumentation 수정 Normalize로 변경	**81.52 %**
5	gluon_seresnext101_64x4d	Standardization로 다시 변경 백본 모델 변경	**88.04 %**
6	gluon_seresnext101_64x4d	Augmentation 수정	**86.96 %**

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
7	efficientnet_b3	Batch size 2	**66.30 %**

Efficientnet[8]은 Resolution과 Depth가 커야 성능이 좋음

But 코랩에서 GPU 메모리부족으로 성능↓

실험No.	모델명	주요 변동사항	Test Acc
8	gluon_resnet152_v1s	모델변경, batch size 128	**90.21 %**
9	gluon_resnet152_v1s	**[12]**근거 Test set 사이즈 2배	**73.91 %**
10	gluon_resnet152_v1s + gluon_seresnext101_64x4d	확률합 기반 **앙상블** (exp5+exp8)	**92.39 %**
11	gluon_resnet152_v1s + gluon_seresnext101_64x4d + gluon_resnext101_64x4d	확률합 기반 **앙상블** (exp1+exp5+exp8)	**92.39 %**

VII. 모델평가

정량적 평가

실험No.	모델명	주요 변동사항	Test Acc
12	gluon_resnet152_v1d	모델 변경	95.65 %
13	gluon_resnet152_v1s + gluon_seresnext101_64x4d + gluon_resnet152_v1d	확률합 기반 **앙상블** (exp5+exp8+exp12)	**93.48 %**

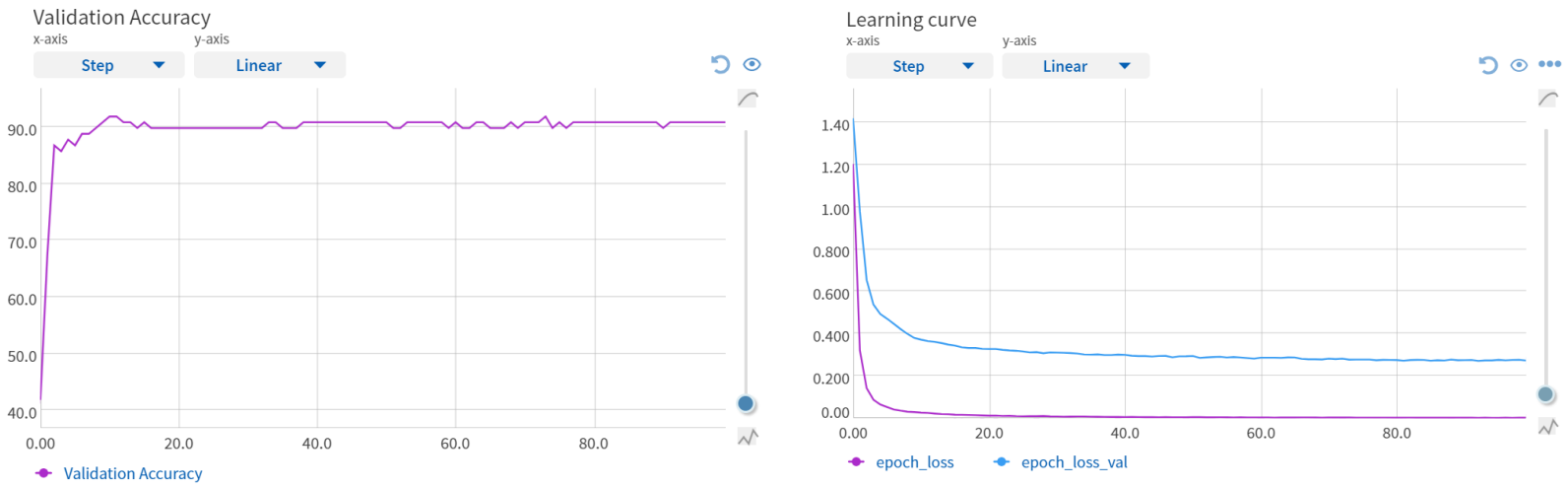
앙상블을 했음에도 *성능이 더 떨어짐*

∵ **성능이 낮은 모델과 높은모델의 조합**

실험결과: 가장 좋은 성능을 낸 모델은 "실험12"로 95.65% 기록

VII. 모델평가

정량적 평가

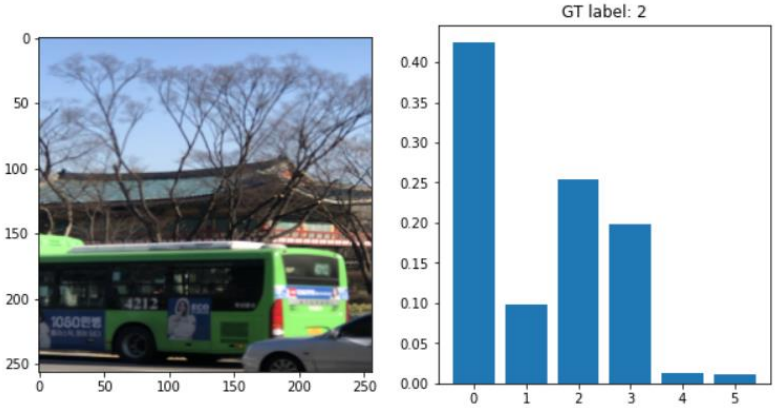


Experiment 8: gluon_resnet152_v1s에 대한 그래프 예시

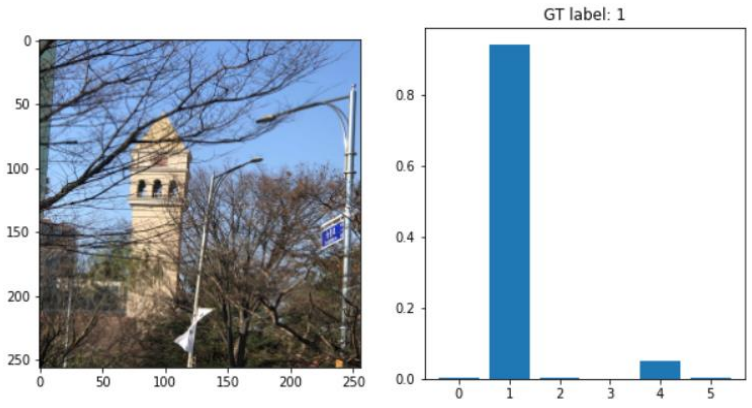
- **Validation loss를 기준으로 Best model을 저장**
- **모든 실험에 대하여 파라미터, 그래프, 체크포인트, CSV파일을 저장**
- **Learning curve를 참고하여 추가학습여부를 결정**

VII. 모델평가

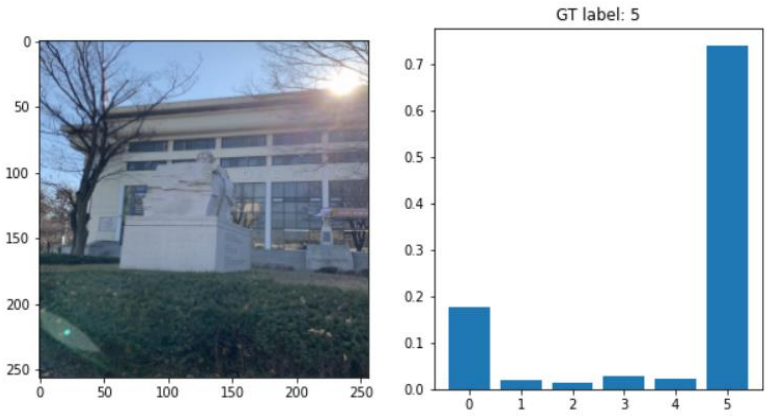
정성적 평가



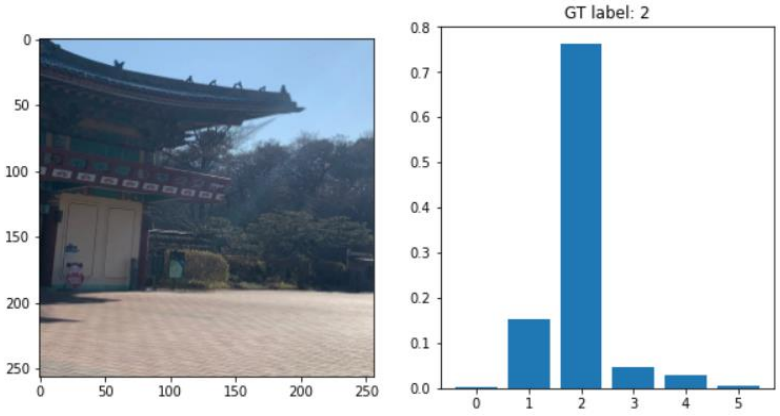
오답



정답



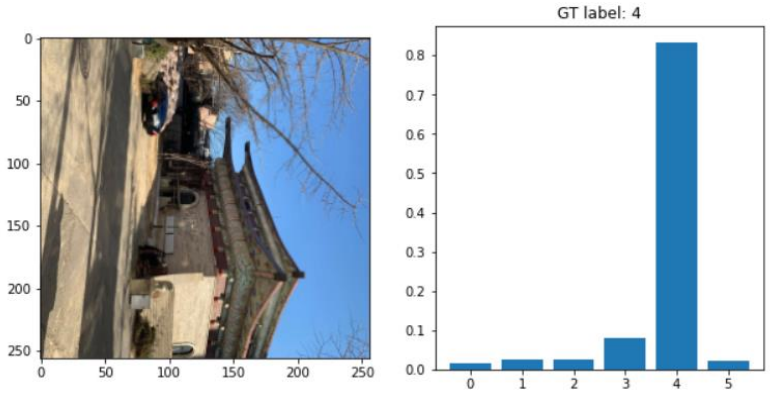
정답



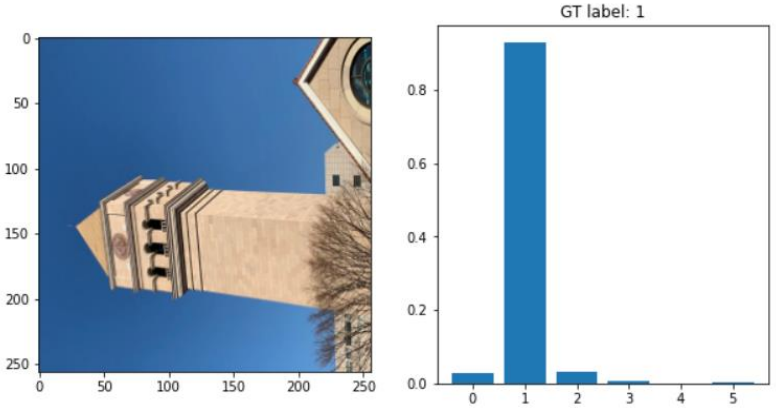
정답

VII. 모델평가

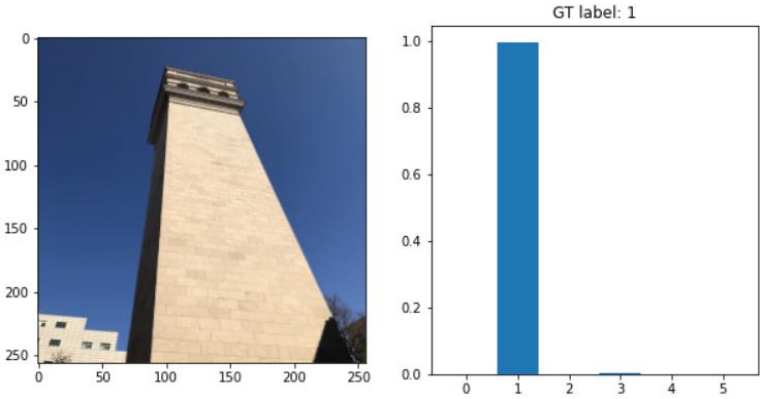
정성적 평가



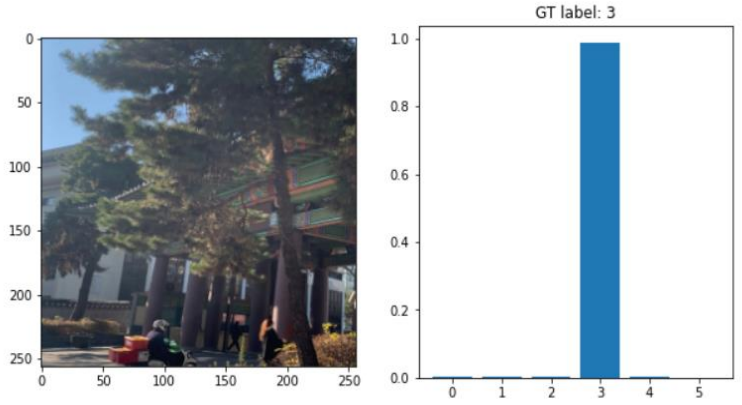
정답



정답



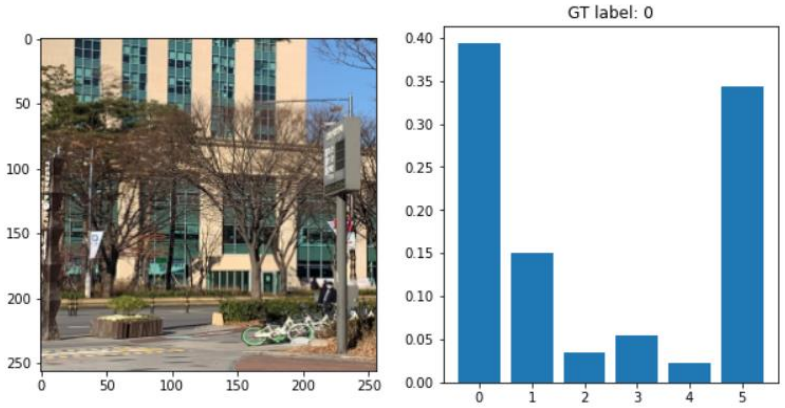
정답



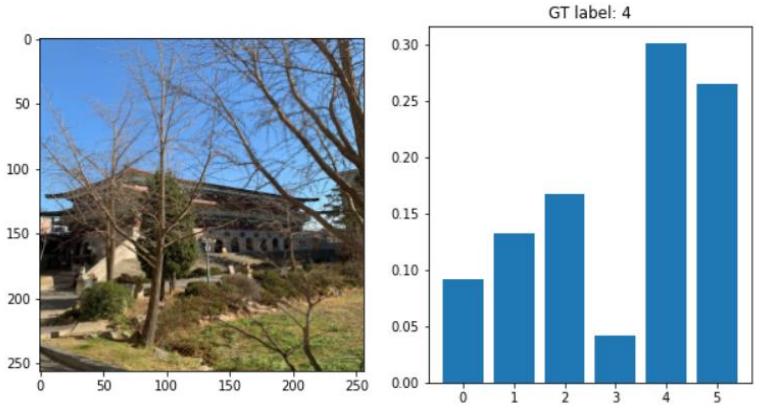
정답

VII. 모델평가

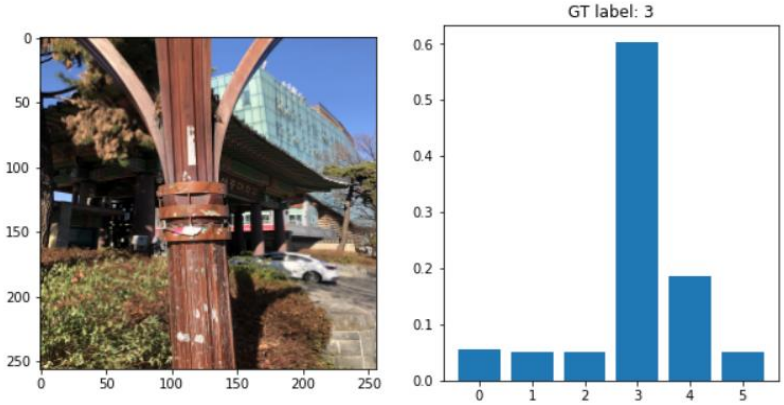
정성적 평가



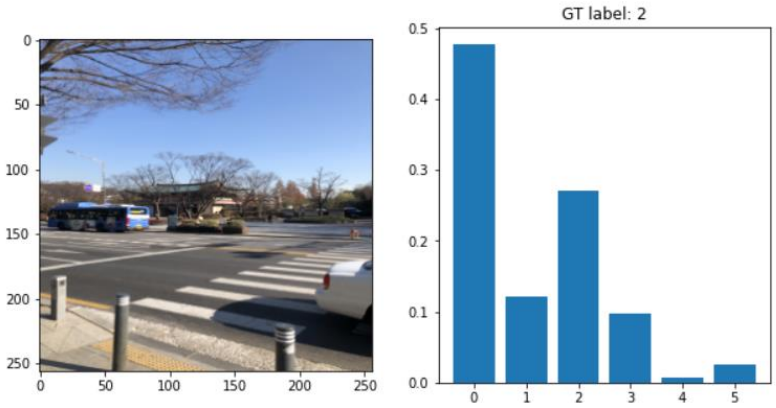
정답



정답



정답



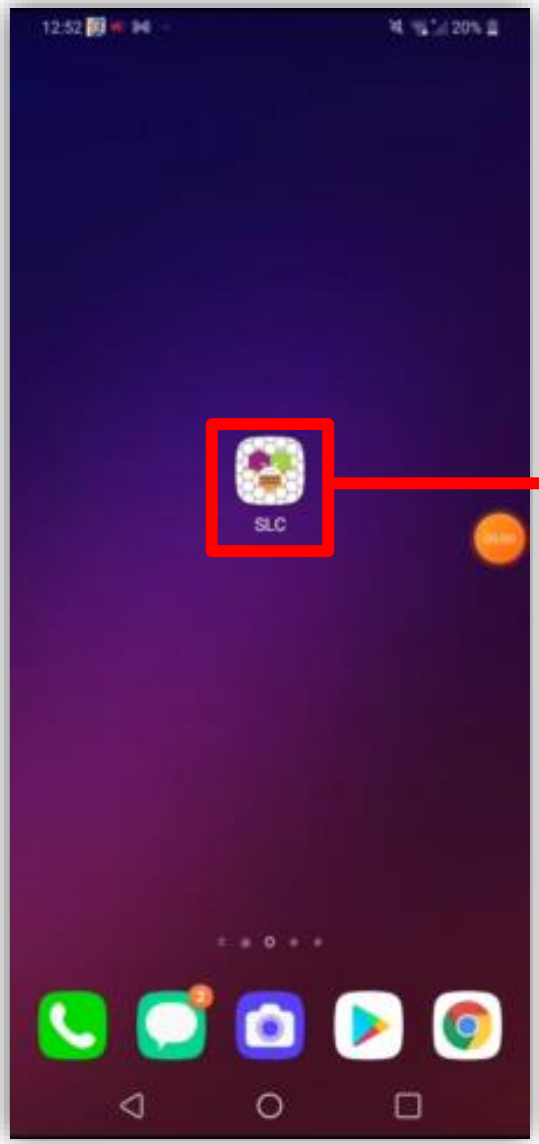
오답

개선 방안

1. 다른 앙상블기법을 이용
   - 각 모델마다 나온 feature를 L2 norm 적용 후 concatenate하여 새로운 feature로 사용
   - 다른 앙상블기법 (Voting, Bagging, Boosting) 사용
2. Cos similarity를 이용한 이미지유사도 비교
   - Trainset에서 Non-landmark 이미지 제거
3. Multi-GPU, 메모리 사용
   - 이미지해상도, Depth, Batch size를 키워서 efficientnet 학습
     - Metric learning 기반 loss 사용시 batch size 영향도 있을거라 예측
4. 데이터로더 설계시 파일경로만을 CSV로 가공하여 메모리 절약
   - 메모리 확보로 좀 더 다양한 실험가능
   - 데이터로더 선언부가 훨씬 빠르게 돌아감

참고 자료

앱인벤터로 개발한 안드로이드 어플 데모영상 (라이트모델 엣지 컴퓨팅)

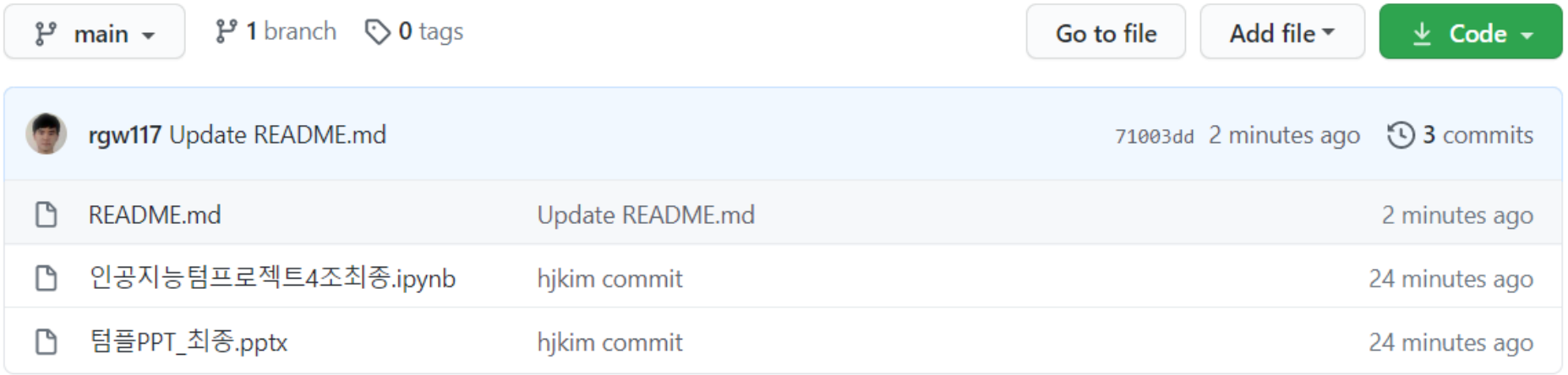


세종랜드마크
챌린지 어플

IX. 참고 자료

- **[1]** https://neptune.ai (넵튠 공식사이트)
- **[2]** Henkel, Christof, and Philipp Singer. "Supporting large-scale image recognition with out-of-domain samples." arXiv preprint arXiv:2010.01650 (2020) (구글랜드마크 챌린지 2020 1등 논문)
- **[3]** https://albumentations.readthedocs.io/en/latest/ (albumentation 라이브러리 다큐먼트)
- **[4]** Buslaev, Alexander, et al. "Albumentations: fast and flexible image augmentations." Information 11.2 (2020): 125. (data augmentation 관련 논문)
- **[5]** https://paperswithcode.com/sota/image-classification-on-imagenet (ImageNet 기준 모델 성능 순위)
- **[6]** https://amaarora.github.io/2020/08/30/gempool.html#gem-pooling (GeM pooling 관련 설명)
- **[7]** https://realblack0.github.io/2020/03/29/normalization-standardization-regularization.html (Normalize관련 설명)
- **[8]** Tan, Mingxing, and Quoc V. Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." arXiv preprint arXiv:1905.11946 (2019) (Efficientnet 논문)
- **[9]** Deng, Jiankang, et al. "Arcface: Additive angular margin loss for deep face recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2019 (Arcface 관련 논문)
- **[10]** http://www.aitimes.com/news/articleView.html?idxno=132756 (파이토치 관련 기사)
- **[11]** https://docs.google.com/presentation/d/1ZUimafgXCBSLsgbacd6-a-dqO7yLyzIl1ZJbiCBUUT4/edit#slide=id.g8b560ae0a6_0_49 (파이토치 관련 자료)
- **[12]** Touvron, Hugo, et al. "Fixing the train-test resolution discrepancy: FixEfficientNet." arXiv preprint arXiv:2003.08237 (2020) (Test set의 resolution 변화에 따른 성능에 관한 논문)
- **[13]** https://appinventor.mit.edu/ (MIT에서 제공하는 앱인벤터 공식사이트)

감사합니다



해당 깃허브에 코드공개됨, 데이터는 이메일문의

https://github.com/rgw117/Sejong-Landmark-Challeng-ME-termproject-

rgw117@naver.com

4조

14011541 김 형준

16011504 임 진욱

18011230 배 대위