

수화 이미지 분류 AI 해커톤

DL Team 1 |

윤성민 (19) 강동헌 (18)
박민지 (19) 정윤주 (18)

CONTENTS

01

대회 소개

(1) 콘테스트 목표 설정

02

데이터 전처리

(1) 데이터 셋 확인
(2) 추가적 이슈

03

모델 분석 및 학습


(1) GoogLeNet
(2) CNN + 10 Fold
(3) 목표 재정의
(4) 평가 및 개선

04

결론

(1) 인사이트

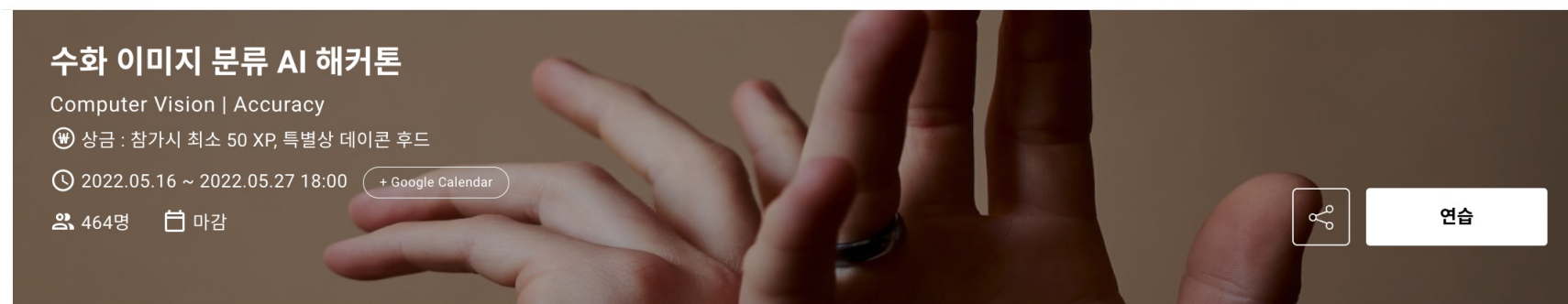




01. 대회 소개

01. 대회 소개

[DACON] 수화 이미지 분류 AI 해커톤



- 수화로 표현한 숫자 1~10 이미지를 학습시켜 분류 정확도를 높이는 것이 목적
 - DL Basic Study에서 학습한 내용을 활용할 수 있는 대회로 선정
-
- 팀을 2개로 나누어 높은 성능을 보인 수상자의 코드 중 GoogLeNet, CNN + 10 Fold 분석
 - 분석 내용을 바탕으로 기존 “성능보다 개선시키는 것”을 목표

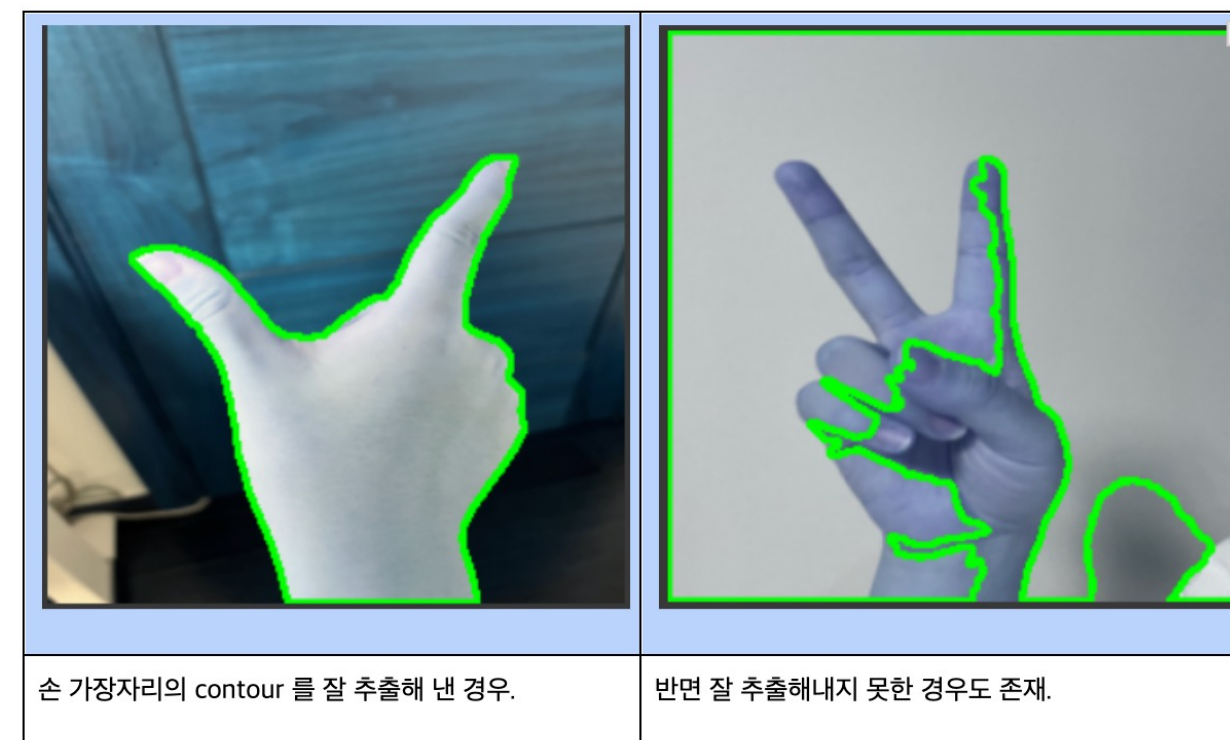


02. 데이터 전처리

02. 데이터 전처리 : 데이터셋 확인

데이터 파일이 없다?

- Github 링크를 통해 데이터 확보
- 수화 "10" 의 라벨링 통합 등 데이터 재정의
- PKL 형식으로 데이터 저장
- 가장 큰 Contour 추출 데이터 저장



2. HEIC 형식 파일에 대해 RGB 디코딩 진행

1. 각 폴더 내 이미지
파일을 순회하며 탐색

3. Tensor와 라벨 integer 의
튜플 형태로 PKL 파일 저장

02. 데이터 전처리 : 추가 이슈

(1) 데이터 불균형 및 혼재

- 수화 "10" 데이터 불균형 및 라벨링 혼재

(2) Test Data 문제

- Validation data accuracy 와 Test data accuracy 의 극심한 차이

과연 DAICON 데이터와 Github 데이터는 같은 데이터일까?

- DAICON의 코드를 동일하게 돌려봐도 현저한 성능 차이
- Test Data 를 폐기하고 Train data를 6:2:2 로 분할하기로 결정



03. 모델 분석 및 학습

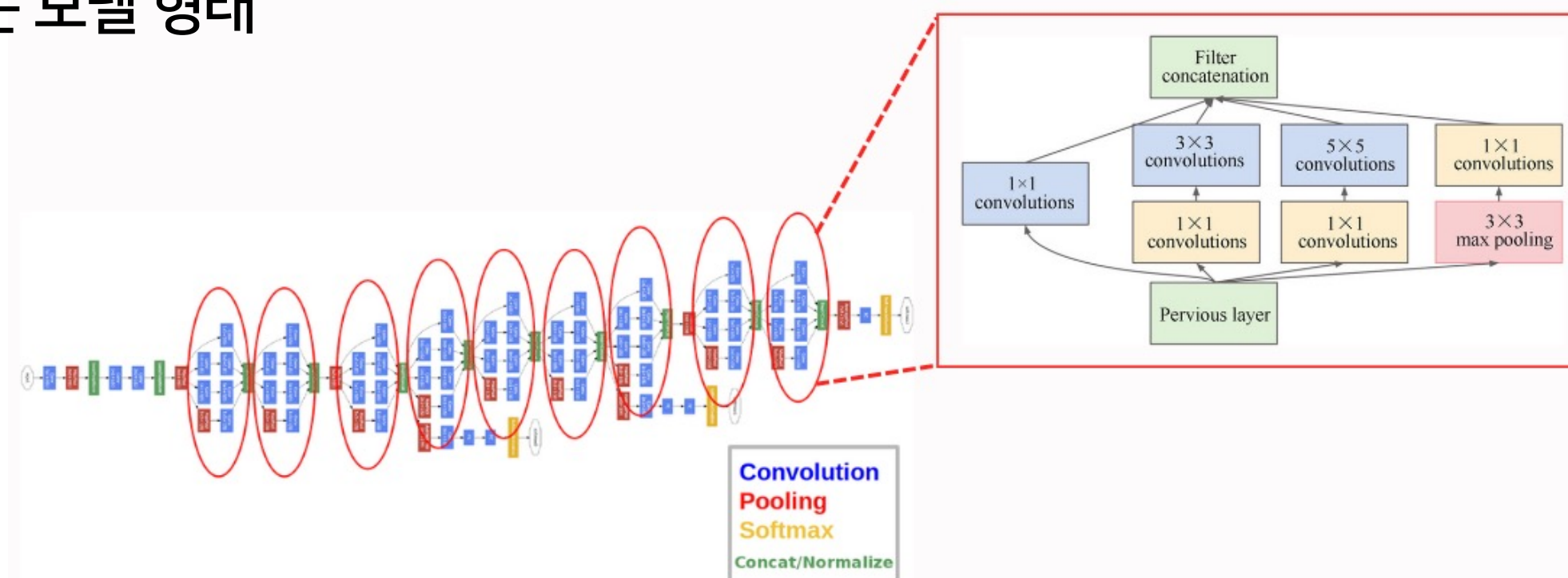
03. 모델 분석 및 학습 : GoogLeNet

2014년, ILSVRC14에서 VGGNET을 제치고 1위

- 네트워크 크기가 깊어지면서, 파라미터 수 증가로 인한 **과적합** 현상 발생
- 이에 따라 **연산량 / 컴퓨터 리소스**의 과도한 투입의 근본적 해결 필요
- GoogLeNet 은 1개의 최종 output 과 2개의 중간 Auxiliary classifiers output

총 3개의 output 을 가지는 모델 형태

GoogLeNet 에서 사용한 Inception 모듈



03. 모델 분석 및 학습 : GoogLeNet

DACON 코드 분석

- 3개의 output 각각의 loss 의 linear combination 으로 전체 loss를 정의할 수 있음
- Label 별로 accuracy를 비교한 뒤, 낮은 accuracy를 갖는 label에 대하여 RegNet으로 evaluation 한 output 출력

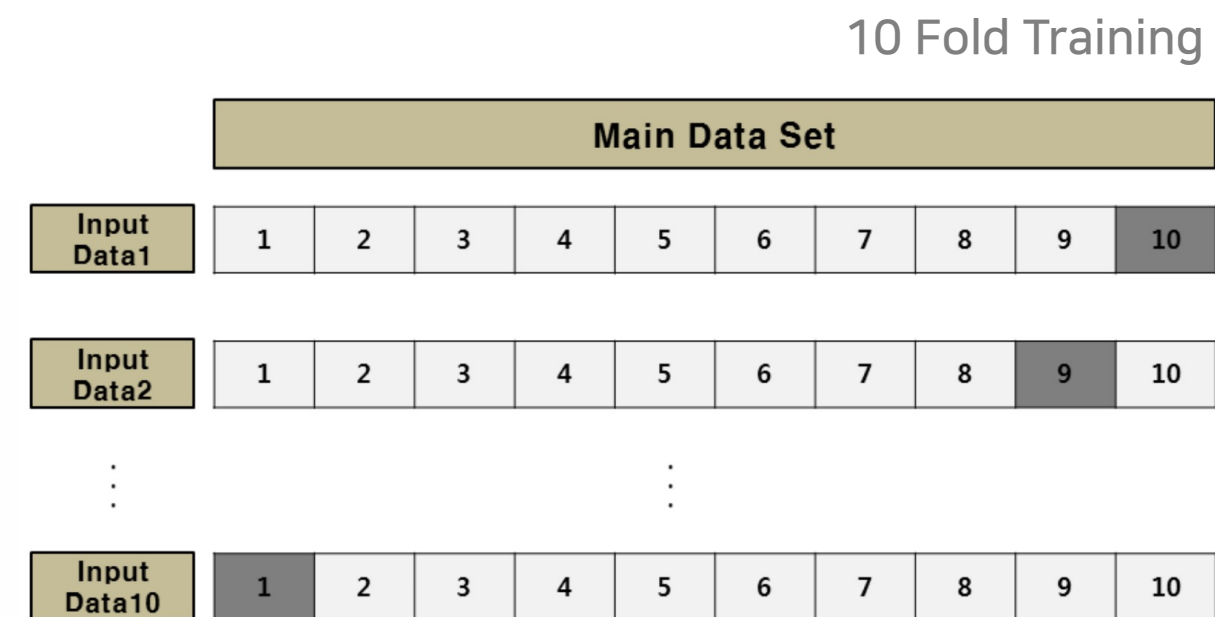
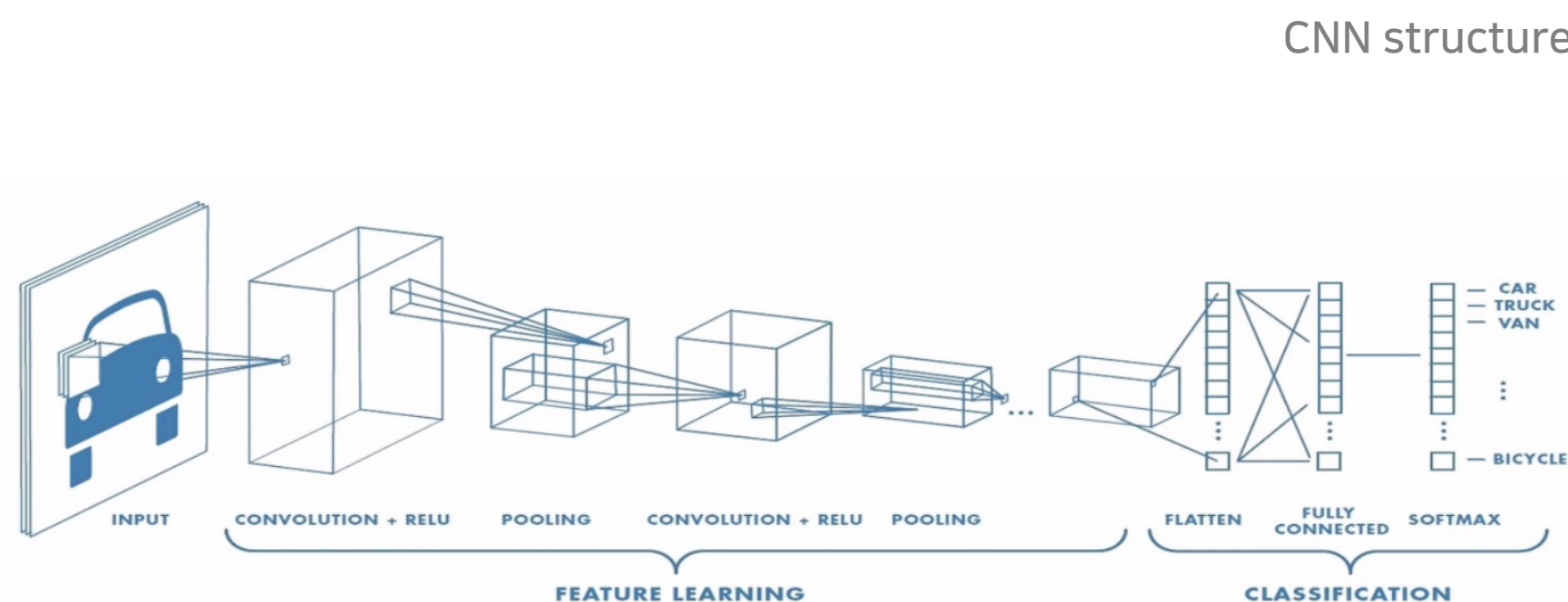
코드 개선 시도

- Optimizer 초기 learning rate /learning rate dacay factor / 종류 변경
- Transform 인자 변경 및 추가
- GoogLeNet의 output loss combination 의 변경

03. 모델 분석 및 학습 : CNN + 10 Fold

2012년, ILSVRC14 에서 1위를 차지한 AlexNet의 등장으로 보편화되기 시작

- Feature를 추출하는 구조를 convolution layer 과 pooling layer 를 쌓아 구성
- Classification 을 위해 Fully Connected Layer 를 추가
- Main dataset 을 10 Fold 로 나누어 학습



03. 모델 분석 및 학습 : 평가 및 개선

DACON 코드 분석

- 데이터 변환 중 normalize를 진행하기 위해 일괄적 정규화 대신 **채널별 평균과 표준편차** 적용
- **Data Augmentation** (이미지 크기 조정, Gaussian Noise, 밝기/대조 변경, 수평 뒤집기, 회전, 정규화)
- **Early Stopping** (loss 가 개선되는지에 따라 조기 중단할 수 있도록 학습 효율성 증가)
- **10 Fold** (학습 시 각 epoch 마다 검증 데이터 손실을 계산하고, 이전 최소 값보다 개선되지 않는 경우 early stopping 카운터 증가)

코드 선택 이유

- 앞선 GoogLeNet 모델의 **일반화 능력을 더 잘 보존할 수 있는 장점**이 있어 최종 평가 모델로 선정
- Loss 가 가장 작은 모델들의 평균 정확도와 평균 손실 출력

03. 모델 분석 및 학습 : 평가 및 개선

문제점

- GoogLeNet 모델은 특정 label 에 대한 성능이 매우 떨어짐
- CNN + 10 Fold 모델은 전반적으로 낮은 accuracy 를 보임
- 데이터 문제로 여전히 Test data 에서의 성능이 개선되지 않음

목표 재정의 : 모델 결합 계획

- GoogLeNet 모델을 사용해 Train data 학습을 진행
- Validation data 기준 accuracy 가 낮은 특정 label에 대해 기존 모델에서 추가로 10 Fold 교차 검증을 수행

03. 모델 분석 및 학습 : 평가 및 개선

파이프라인 및 평가

- 10 Fold training 에서 **Early Stopping** 기준으로 저장된 모델과
- 각 Fold 학습이 완료된 후의 모델을 사용하여 최종 성능 평가

문제점

Validation – Test set accuracy

10 Fold Training 이후

```
check_accuracy(val_loader, model)

Got 138 / 156 with accuracy 88.46153846153845
138

check_accuracy(test_loader, model)

Got 203 / 330 with accuracy 61.51515151515151
203
```

Validation accuract

```
Overall Accuracy: 87.32%
Accuracy for label 0: 93.75%
Accuracy for label 1: 83.33%
Accuracy for label 2: 92.31%
Accuracy for label 3: 73.33%
Accuracy for label 4: 94.74%
Accuracy for label 5: 100.00%
Accuracy for label 6: 88.24%
Accuracy for label 7: 84.62%
Accuracy for label 8: 66.67%
Accuracy for label 9: 91.67%
```

Dataset

Test accuracy

```
model.load_state_dict(torch.load('./best_model_fold_spec'))
#최종평가
evaluate_and_calculate_accuracy(model, test_loader)

Overall Accuracy: 52.89%
Accuracy for label 0: 43.33%
Accuracy for label 1: 46.67%
Accuracy for label 2: 45.83%
Accuracy for label 3: 73.33%
Accuracy for label 4: 70.00%
Accuracy for label 5: 93.33%
Accuracy for label 6: 63.33%
Accuracy for label 7: 40.00%
Accuracy for label 8: 63.33%
Accuracy for label 9: 23.08%
```


03. 모델 분석 및 학습 : 평가 및 개선

최종 코드 개선

1. Test 데이터 폐기 (기본 Train dataset을 6:2:2 (Train / Val /Test)로 분할)
2. Train 데이터 증강
 - Vertical Flip된 Train 데이터를 추가 (2배 증강)
 - 기본 + Vertical Flip + Contour + Contour Vertical Flip(4배 증강)

“Train time – accuracy 의 Trade off”

03. 모델 분석 및 학습 : 평가 및 개선

최종 코드 개선

- Horizontal Flip, Affine 변환 미적용
- GoogLeNet 에서 성능이 낮은 라벨 3, 8, 10 을 대상으로 10 Fold 학습
- GoogLeNet 의 3개 output 중 첫번째 주 output의 Loss 만 활용하여 10 Fold 학습

“Training Robustness, Accuracy”



04. 결론

04. 결론 및 인사이트

- Train Data 증강을 통한 Dataset 문제 해결
- 2개 모델을 결합해 일반화 성능 향상 시도

최종 결과

10 Fold Training 이후
Validation Set Accuracy

```
Overall Accuracy: 91.34%  
Accuracy for label 0: 84.48%  
Accuracy for label 1: 95.74%  
Accuracy for label 2: 74.19%  
Accuracy for label 3: 92.06%  
Accuracy for label 4: 100.00%  
Accuracy for label 5: 96.55%  
Accuracy for label 6: 87.93%  
Accuracy for label 7: 91.84%  
Accuracy for label 8: 94.34%  
Accuracy for label 9: 98.25%
```

특정 Label 추가 학습 후 해당 Label 의 Accuracy

```
Overall Accuracy: 97.67%  
Accuracy for label 0: 100.00%  
Label 1 does not appear in the test set.  
Accuracy for label 2: 93.33%  
Accuracy for label 3: 100.00%  
Label 4 does not appear in the test set.  
Label 5 does not appear in the test set.  
Label 6 does not appear in the test set.  
Label 7 does not appear in the test set.  
Label 8 does not appear in the test set.  
Label 9 does not appear in the test set.
```

최종 Test Accuracy

```
Overall Accuracy: 91.89%  
Accuracy for label 0: 91.38%  
Accuracy for label 1: 100.00%  
Accuracy for label 2: 81.43%  
Accuracy for label 3: 83.64%  
Accuracy for label 4: 100.00%  
Accuracy for label 5: 96.00%  
Accuracy for label 6: 92.19%  
Accuracy for label 7: 88.24%  
Accuracy for label 8: 94.92%  
Accuracy for label 9: 94.64%
```



Thank You