

[캡스톤디자인 결과보고서]

■ 연구과제

과제명 (작품명)	딥러닝을 이용한 농작물 질병 분류 모델 성능 측정 및 비교	참여학기	2021 년 1 학기
--------------	----------------------------------	------	-------------

■ 강좌정보

과목명	소프트웨어융합캡스톤디자인	학수번호	SWCON401
과제기간	2021 년 03 월 02 일 ~ 2021 년 06 월 11 일	학점	3

■ 팀구성

팀명				팀구성 총인원	1명
구분	성명	학번	소속학과	학년	
대표학생	정환석	2018110660	소프트웨어융합학과	4	
참여학생					

■ 지도교수 확인


지도교수	성명	이대호	직급	전임교수
	소속학과	소프트웨어융합학과		성명 : (인)

■ 불임

[첨부 1] 과제 요약보고서
[결과물] 최종결과물 (최종작품 사진/도면/발표자료 등)

본 팀은 과제를 성실히 수행하고 제반 의무를 이해하여 이에 따른 결과보고서를 제출합니다.

일자 : 2021 년 06 월 11 일

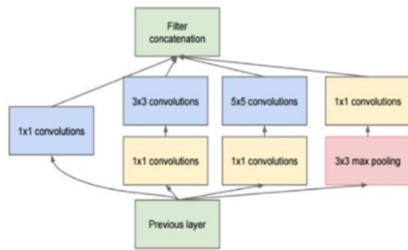
신청자(또는 팀 대표) _____ 정환석 

[캡스톤디자인 과제 요약보고서]

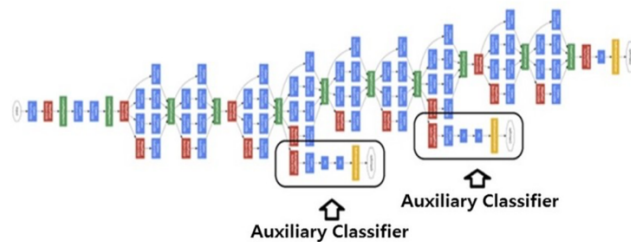
과 제 명	딥러닝을 이용한 농작물 질병 분류 모델 성능 측정 및 비교
<div>1. 과제 개요</div> <div>가. 과제 선정 배경 및 필요성</div> <p>한국의 기후 특성상 여름철 긴 장마 후 생육이 불량하고 고온 다습한 환경에서농작물 병해충으로 인한피해가 빈번히 발생한다. 병해충에 걸린 농작물은 잎, 줄기, 열매 등에서 증상이 나타나는데 병해충으로인한 피해를 줄이기 위해서는 감염된 작물의 증상을 통해 초기에 진단하여 신속한 조치를 취해 확산을막는 것이 가장 중요하다. 하지만 작물별 병해충의 종류는 매우 다양하고 그 진행 양상도 다르므로 인간이다양한 종류의 작물에 걸린 병을 모두 진단하는데는 한계가 있다. 따라서 농작물의 증상을 통해 병을진단하는 연구가 필요하다.</p> <div>나. 과제 주요내용</div> <ul style="list-style-type: none">- 병해충에 걸린 농작물 데이터 수집- 이미지 분류에서 높은 성능을 보인 VggNet, GoogLeNet, ResNet 모델의 사용- Drop Out, Batch Normalization, Data Augmentation 등 성능 향상을 시도- 학습에 사용한 모델의 성능 측정 및 비교 <div>다. 최종결과물의 목표</div> <ol style="list-style-type: none">1) 농작물의 질병을 분류하는 딥러닝 모델의 성능을 90% 이상의 정확도를 목표로 한다.2) Data Augmentation 추가하여 성능을 비교한다.3) 학습에 사용한 각 모델별 성능을 비교한다.	
<div>2. 과제 수행방법</div> <div>가. CNN Architecture</div> <div>1) VGGNet</div> <p>VGGNet 은 ILSVRC 2014 대회에서 2 위를 차지한 모델로 VGGNet 연구팀의 논문에 따르면 신경망의 깊이가 어떤 영향을 주는지에 대해서 연구하기 위해서 개발되었다. 따라서 간단하게 3x3 컨볼루션을 깊게 중첩한 것이 VGGNet 은 특징이다.</p> <div><p><그림 1> VGGNet</p></div>	

2) GoogLeNet

이전까지 CNN 모델에서 컨볼루션 연산을 하면 네트워크가 깊어질수록 학습해야하는 파라미터 수와 연산량이 증가하는 문제점이 있었다. 하지만 GoogLeNet 은 총 22 개의 층을 가지고 Inception module 이라는 block 구조를 통해 연산량을 대폭 줄일 수 있다. Inception module 은 4 가지 서로 다른 연산을 거친 뒤 feature map 을 channel 방향으로 연쇄적으로 합친다. 이 방법을 통해서 연산량을 절반이상으로 줄일 수 있다.



<그림 2> Inception Module

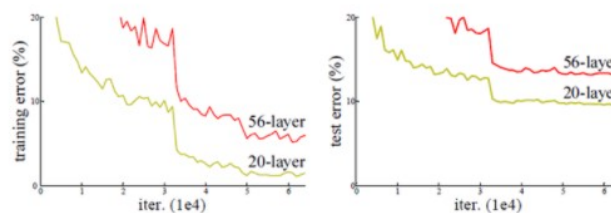


<그림 3> GoogLeNet

그리고 총 9 개의 inception module 을 쌓는데 3 번째와 6 번째에 보조분류기를 추가하여 총 3 개의 분류기를 사용하였다. 모델이 깊어질수록 마지막에 하나의 분류기만 존재한다면 앞쪽까지 gradient 가 잘 전파되지 않을 수 있는데, 네트워크의 중간부분에 보조분류기를 사용하여 gradient 소멸 문제를 줄일 수 있다고 한다. 보조 분류기로 구한 loss 는 보조이므로 마지막 분류기 보다 영향력을 적게 주기 위해서 0.3 의 가중치를 주어 total loss 에 더한다.

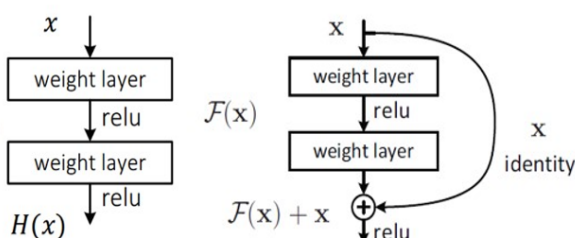
3) ResNet

일반적으로 딥러닝에서 신경망을 깊게 설계하는 것만으로도 성능을 올릴 수 있다고 생각할 수 있지만 <그림 4>에서 56-layer 보다 20-layer 의 신경망의 성능이 더 좋게 나오듯이 무조건 깊게 쌓는다고 성능이 좋아지는 것이 아니다.

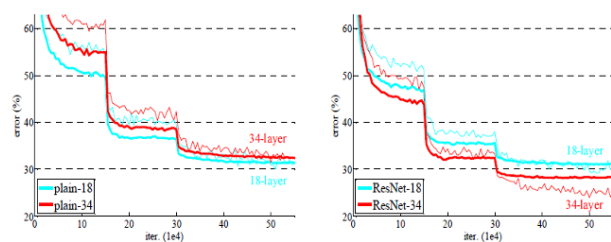


<그림 4> 일반적인 신경망의 깊이에 따른 에러

그 이유는 층이 깊어질수록 gradient vanishing 문제가 발생하기 때문이다. 따라서 ResNet 은 이전 Layer 의 Feature Map 을 다음 Layer 의 Feature Map 에 더해주는 Residual Block 을 도입한 모델로 신경망이 깊어지더라도 최소 gradient 로 1 이상의 값을 갖게 하여 gradient vanishing 문제를 해결한 것이다.



<그림 5> Residual Block



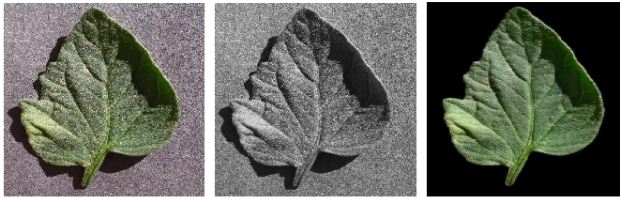
<그림 6> 일반적인 신경망과 ResNet의 깊이에 따른 에러

<그림 6>과 같이 Residual Block 이 없는 모델과 있는 모델의 층의 깊이에 따른 성능 차이를 통해서 ResNet 은 신경망의 깊이가 깊어질수록 성능이 좋다는 특징을 찾을 수 있다.

나. Plant Village 데이터셋

1) 데이터 수집 & 전처리

kaggle 에서 PlantVillage 데이터셋을 수집하였다. PlantVillage 데이터셋은 질병에 걸린 작물의 잎을 촬영한 이미지로 256x256 크기의 color, grayscale, segmented 데이터로 구성되어있다. 과제에서는 color 이미지를 224x224로 Resize 하고 0~255 범위의 픽셀 값을 정규화를 통해 0~1 범위를 가지게 하였다. 전처리된 데이터는 60:20:20으로 train, valid, test 데이터셋으로 나누었다.



Class	Name	Count
0	Bacterial_spot	2127
1	Early_blight	1000
2	Healthy	1591
3	Late_blight	1909
4	Leaf_Mold	952
5	Septoria_leaf_spot	1771
6	Spider_mites	1676
7	Target_Spot	1404
8	Tomato_Yellow_Leaf_Curl_Virus	5357
9	Tomato_mosaic_virus	373
총 계		16569

<그림 7> PlantVillage 데이터와 메타 데이터

2) PlantVillage 클래스들의 실제 데이터 수집

PlantVillage 데이터는 토마토의 질병에 걸린 잎을 잘라서 촬영된 이미지이다. 이와 다르게 실제 토마토 농가에서 촬영된 병충해 이미지를 통해 모델의 분류 성능을 측정하기 위해서 데이터를 추가로 수집하였다.

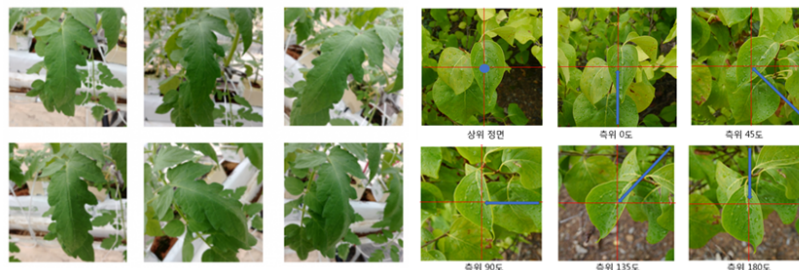


<그림 8> 수집한 실제 농가에서 촬영한 데이터

다. AIHub 데이터셋

1) 데이터 수집 & 전처리

국내외에서 보유한 학습용 데이터를 공개하는 AIHub 에서 농작물 잎을 수집하였다. 이 데이터는 PlantVillage 데이터와 달리 실제 농가에서 촬영된 이미지로 질병에 걸린 토마토의 잎을 다각도에서 촬영한 이미지이다.



<그림 9> AIHub 데이터

수집한 데이터는 병해충에 따라서 데이터의 개수가 다른데 너무 적은 클래스로 모델을 학습시키는 것은 겨로가 적으로 무의미하다 생각하여 데이터가 100 개 이하인 클래스는 제거하였다. 따라서 총 7 개의 클래스를 이용하였다.

Class	Name	Count
0	꺾양병	229
1	시들음병	5
2	오이모자이크바이러스	6
3	잎곰팡이병	768
4	점무늬병	2067
5	줄기속썩음병	2
6	토마토위흑바이러스	90
7	황화잎말림바이러스	427
8	흰가루병	49
9	녹응애	6
10	담배가루이	3
11	아메리카잎굴파리	3357
12	차면지응애	5
13	청벌레	139
14	아메리카잎굴파리	274
15	정상	8009
총 계		15436

➔

Class	Name	Count
0	꺾양병	229
1	잎곰팡이병	768
2	점무늬병	2067
3	황화잎말림바이러스	427
4	아메리카잎굴파리	3357
5	청벌레	139
6	정상	8009
총 계		14996

<그림 10> AIHub 메타데이터와 학습에 사용할 데이터 정보

학습을 위해서 고해상도이고 각각 사이즈가 다른 데이터를 224x224 크기로 resize 해주고, 0~255 범위의 값을 가지는 픽셀값을 0~1 의 범위를 가지게 정규화하였다.

2) Data Augmentation

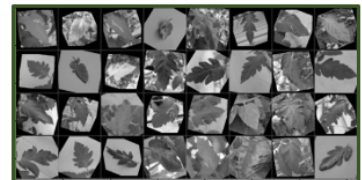
모델 성능 향상을 위해서 Flip, Rotation, Affine transform, Crop, Grayscale 기법을 이용하였다. 총 3 가지 버전으로 Data Augmentation 을 진행하였는데 V1 은 Flip, Rotation, Affine transform 만을 사용하였다. V2 는 V1 에 Crop 을 추가한 것이고 V3 는 V1 에 Gray Scale 을 추가한 것이다.



a. V1



b. V2



c. V3

- a) V1: Flip, Rotation, Affine transform
- b) V2: Flip, Rotation, Affine transform, Crop
- c) V3: Flip, Rotation, Affine transform, Gray Scale

라. 모델 성능 평가 지표

- 1) Confusion Matrix: 예측 값이 실제 값을 얼마나 정확히 예측했는지 보여주는 행렬
- 2) Accuracy: 전체 중에 정답을 맞춘 비율
- 3) Precision: Positive 로 예측한 것 중에서 실제 Positive 의 비율
- 4) Recall: 실제 Positive 인 것 중에서 Positive 로 예측한 비율
- 5) F-1 Score: Precision 과 Recall 의 조화 평균 (* Imbalanced Data 인 경우)

3. 수행결과

가. 과제수행 결과

1) Plant Village 데이터셋

가) GoogLeNet (Test Data / Real Data)

Actual \ Predicted	Predicted									
	0	1	2	3	4	5	6	7	8	9
0	421	0	1	0	0	0	0	1	0	2
1	0	193	0	6	0	0	0	1	0	0
2	0	0	318	0	0	0	0	0	0	0
3	0	2	0	379	0	0	0	0	0	0
4	0	0	0	0	187	0	2	0	0	1
5	0	0	0	1	0	353	0	0	0	0
6	0	0	0	0	0	0	333	2	0	0
7	0	1	0	1	0	1	6	271	0	0
8	0	0	0	0	0	0	0	0	74	0
9	0	0	0	0	0	0	0	1	0	1070

<Test Data>

Accuracy = 0.992007
Precision = 0.991369
Recall = 0.989268
F1-score = 0.990281

Actual \ Predicted	Predicted									
	0	1	2	3	4	5	6	7	8	9
0	0	0	0	7	0	3	0	0	0	0
1	0	9	0	1	0	0	0	0	0	0
2	0	0	0	9	0	1	0	0	0	0
3	0	2	0	7	0	1	0	0	0	0
4	0	1	0	8	0	0	0	0	0	1
5	0	7	0	3	0	0	0	0	0	0
6	0	0	1	9	0	0	0	0	0	0
7	0	2	0	5	0	2	0	0	0	1
8	0	2	0	8	0	0	0	0	0	0
9	0	1	0	9	0	0	0	0	0	0

<Real Data>

Accuracy = 0.16
Precision = 0.0481061
Recall = 0.16
F1-score = 0.713622

나) ResNet (Test Data / Real Data)

Actual \ Predicted	Predicted									
	0	1	2	3	4	5	6	7	8	9
0	421	0	0	1	0	0	0	1	0	2
1	2	196	0	0	0	0	0	1	0	1
2	0	0	318	0	0	0	0	0	0	0
3	2	5	0	370	0	2	1	1	0	0
4	0	0	1	1	188	0	0	0	0	0
5	0	0	0	1	0	352	0	1	0	0
6	0	1	1	0	0	0	332	0	1	0
7	1	1	0	0	1	2	6	269	0	0
8	0	0	0	0	0	0	1	0	73	0
9	1	0	0	0	1	0	0	0	0	1069

<Test Data>

Accuracy = 0.988975
Precision = 0.986092
Recall = 0.986192
F1-score = 0.986102

Actual \ Predicted	Predicted									
	0	1	2	3	4	5	6	7	8	9
0	0	4	0	6	0	0	0	0	0	0
1	0	7	0	3	0	0	0	0	0	0
2	0	0	0	9	0	0	0	0	0	1
3	0	2	0	8	0	0	0	0	0	0
4	0	1	0	9	0	0	0	0	0	0
5	0	4	0	4	0	2	0	0	0	0
6	0	0	1	9	0	0	0	0	0	0
7	0	3	0	5	0	1	0	0	0	1
8	0	3	0	6	0	0	0	0	0	1
9	0	5	0	5	0	0	0	0	0	0

<Real Data>

Accuracy = 0.17
Precision = 0.103305
Recall = 0.17
F1-score = 0.0882883

2) AIHub 데이터셋

가) VGGNet

VGGNet

Actual \ Predicted	Predicted						
	0	1	2	3	4	5	6
0	19	0	8	7	4	2	2
1	0	118	5	0	10	0	13
2	0	1	336	1	22	0	27
3	3	0	2	44	15	0	13
4	0	2	10	5	624	2	84
5	2	0	11	2	4	5	8
6	0	1	16	6	37	0	1526

Accuracy = 0.893347
Precision = 0.807797
Recall = 0.668663
F1-score = 0.713845

Aug1 VGGNet

Actual \ Predicted	Predicted						
	0	1	2	3	4	5	6
0	30	0	1	4	6	0	1
1	0	126	1	0	9	1	9
2	2	2	369	0	7	1	6
3	2	0	2	58	4	0	11
4	0	1	9	1	681	1	34
5	1	0	6	1	6	12	6
6	0	2	6	2	20	3	1547

Accuracy = 0.943831
Precision = 0.884075
Recall = 0.796411
F1-score = 0.832381

Aug2 VGGNet

Actual \ Predicted	Predicted						
	0	1	2	3	4	5	6
0	11	1	4	13	5	0	8
1	1	115	0	0	5	0	25
2	2	1	329	2	12	0	41
3	3	0	6	57	3	0	8
4	3	1	19	0	524	0	180
5	4	0	1	0	10	0	17
6	0	1	6	1	5	0	1567

Accuracy = 0.870277
Precision = 0.697836
Recall = 0.621787
F1-score = 0.651829

Aug3 VGGNet

Actual \ Predicted	Predicted						
	0	1	2	3	4	5	6
0	27	0	7	4	1	0	3
1	1	123	7	0	4	0	11
2	1	10	351	3	4	0	18
3	1	1	3	59	2	0	11
4	2	0	7	0	702	4	12
5	0	0	1	1	3	20	7
6	1	2	13	9	4	0	1551

Accuracy = 0.947175
Precision = 0.881588
Recall = 0.818684
F1-score = 0.846327

나) GoogLeNet

GoogLeNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	40	1	0	1	0	0	0
	1	0	137	1	0	3	2	3
	2	1	0	384	0	2	0	0
	3	0	0	1	72	0	1	3
	4	0	3	2	0	709	4	9
	5	1	1	1	0	1	21	7
	6	0	0	3	3	8	0	1566

Accuracy = 0.979271
Precision = 0.937273
Recall = 0.920097
F1-score = 0.928199

Aug1 GoogLeNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	40	0	1	0	0	0	1
	1	0	138	2	0	1	1	4
	2	0	0	386	0	0	0	1
	3	0	0	1	73	0	1	2
	4	0	2	4	0	716	0	5
	5	1	1	0	1	0	26	3
	6	0	0	0	5	4	0	1571

Accuracy = 0.986292
Precision = 0.967091
Recall = 0.947818
F1-score = 0.956804

Aug2 GoogLeNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	37	0	0	2	0	2	1
	1	0	135	0	0	6	0	5
	2	0	1	381	0	1	0	4
	3	0	0	0	71	0	1	5
	4	1	1	1	0	713	6	5
	5	0	0	0	0	4	26	2
	6	0	0	0	1	5	153	1421

Accuracy = 0.930792
Precision = 0.859576
Recall = 0.914971
F1-score = 0.85232

Aug3 GoogLeNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	33	1	6	2	0	0	0
	1	0	133	2	0	4	0	7
	2	2	1	371	0	3	0	10
	3	1	0	8	57	0	2	9
	4	0	0	2	0	720	2	3
	5	0	0	1	1	2	22	6
	6	0	4	12	3	0	1	1560

Accuracy = 0.968238
Precision = 0.925954
Recall = 0.865829
F1-score = 0.893094

다) ResNet

ResNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	32	1	0	3	2	0	4
	1	1	117	0	0	7	1	20
	2	0	0	337	4	22	0	24
	3	0	0	2	64	0	1	10
	4	1	0	7	1	655	2	61
	5	0	0	3	1	9	9	10
	6	5	2	11	6	21	1	1534

Accuracy = 0.918756
Precision = 0.860263
Recall = 0.774049
F1-score = 0.805398

Aug1 ResNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	37	1	0	2	1	0	1
	1	2	135	4	0	2	0	3
	2	0	3	377	0	3	1	3
	3	1	0	0	70	0	1	5
	4	0	2	3	0	714	2	6
	5	0	1	2	1	4	20	4
	6	0	0	5	0	1	0	1574

Accuracy = 0.978602
Precision = 0.943311
Recall = 0.898883
F1-score = 0.918766

Aug2 ResNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	35	1	0	1	0	0	5
	1	2	122	1	0	5	0	16
	2	0	0	371	1	4	0	11
	3	0	0	1	63	0	0	13
	4	1	1	3	0	687	2	33
	5	1	0	2	0	6	14	9
	6	0	0	0	0	2	0	1578

Accuracy = 0.949545
Precision = 0.947231
Recall = 0.832429
F1-score = 0.877319

Aug3 ResNet

		Predicted						
		0	1	2	3	4	5	6
Actual	0	22	1	10	6	1	0	2
	1	1	134	1	0	4	0	6
	2	0	8	364	1	5	0	9
	3	0	0	9	51	0	0	17
	4	1	4	6	0	706	0	10
	5	0	1	4	0	5	9	13
	6	0	5	9	3	6	0	1557

Accuracy = 0.950518
Precision = 0.923939
Recall = 0.754619
F1-score = 0.801247

나. 최종결과물 주요특징 및 설명

PlantVillage 로 학습한 모델로 테스트셋을 예측한 결과 Accuracy 와 F-1 Score 모두 높게 나왔지만 수집한 실제 농가에서 촬영된 데이터셋에서는 성능이 매우 낮았다. 실제 농가 데이터를 수집하는데 한계가 있어 클래스당 10 개씩만 사용하여 소량으로 예측해서 낮게 나온 것일 수도 있지만 PlantVillage 를 학습한 모델이 잘라진 잎 부분만 학습하여 실제에서는 적용이 안되는 듯 했다.

실제 농가에서 적용할 수 있는 모델을 만드는 것이 목표였기 때문에 AIHub 에서 구한 데이터로 다시 학습하였다. VGGNet, GoogLeNet, ResNet 을 사용한 결과 GoogLeNet 에서 F-1 Score 가 0.92 로 가장 좋게 나왔다. 그 다음 ResNet 이 0.80, VGGNet 이 0.71 이 나왔다.

추가적으로 성능을 향상시키기 위하여 Data Augmentation 을 진행하였다. Data Augmentation 은 총 세가지 버전으로 진행하였는데 Crop 을 추가한 버전 2 를 적용한 VGGNet 과 GoogleNet 을 보면 Data Augmentation 을 하지 않았을 때 보다 성능이 하락한 것을 볼 수 있는데 Crop 은 특정 부분만 자른 후 데이터 증강시키는 것으로 질병의 특징이 있는 부분이 제거가 되어 성능이 떨어졌다고 생각한다.

최종적으로 결과는 Flip, Rotation, Affine transform 만 적용한 GoogleNet 이 0.95 로 가장 높게 나왔다.

4. 기대효과 및 활용방안

일반적으로 농가에서는 병해충에 대한 자기 진단이 어려워 전문가의 도움을 필요로 한다. 하지만, 농작물 질병 분류 모델을 통해서 작물의 병해충을 초기에 인식하고, 그 병해충에 걸맞는 조치를 취해 병해충의 확산을 신속하게 방지할 수 있게 된다. 이러한 결과로, 수확량을 증가시킬 수 있을 것으로 예상되며, 농작물의 품질 또한 개선될 것으로 예측된다. 19 년 한국 농가의 병해충 피해액은 약 342 억으로 집계됐다. 농작물 질병 분류 모델의 활용은 이러한 한국 농가의 경제적 부담, 그리고 정부의 농가 지원금 부담을 조금은 덜어줄 것이라고 예상된다. 또한 현재 농업 분야에서 고령화 문제가 대두되고 있는 만큼, 이러한 농작물 질병 분류 모델은 농업의 진입장벽을 낮춰 젊은 층의 유입을 기대해 볼 수 있다.

5. 결론 및 제언

PlantVillage 로 학습한 모델은 테스트셋에서는 좋은 성능을 보였지만 추가적으로 수집한 실제 농가의 데이터에서는 분류를 거의 하지 못하는 것을 보았다. 따라서 PlantVillage 에서 수집한 데이터로 학습한 모델로는 실제 농업 현장에서 사용하는데 어려움이 있을 것으로 판단된다.

AIHub 에서 수집한 데이터는 실제 농가에서 촬영된 데이터이기 때문에 이것으로 학습한 모델은 실제 농업현장에서 사용 가능 할 것으로 예상된다. 이 데이터로 학습한 모델의 최종 결과를 보면 Data Augmentation V1 을 사용한 GoogleNet 이 가장 좋은 성능을 보였다. 그리고 다른 결과들을 통해 Data Augmentation 만으로 모델의 성능을 향상시킬 수도 있지만, 부적절한 기법을 사용할 경우 반대로 성능이 떨어질 수 있다는 것을 확인하였고 특정 Data Augmentation 기법을 적용한 모델의 성능이 향상되었다고 다른 모델에 같은 기법을 적용했을 때도 성능이 무조건 향상되는 것은 아니라는 것을 알 수 있었다.

※ 본 양식은 요약보고서이며, 최종결과물을 필히 추가 제출하여야 함.

팀 학생대표 성명 : _____ 정환석

정환석