

2022. 07. 26.

< PANDAS >

## 팀프로젝트 최종 발표

ICT 이노베이션 스퀘어 -인공지능 프로젝트반





## Member

최연우(팀장)

심연수(부팀장)

김태경

장윤서

## Role

MobileNet 정리, 코드 성능 개선(filter size), PPT 제작

AlexNet 정리, 코드 성능 개선(Conv2D, MaxPooling), PPT 제작

VGG16 모델 정리, 코드 성능 개선(epoch & batch\_size, 발표

코드 성능 개선(img\_dim & batch\_size, learning rate, filter size, 발표

# CONTENTS

01 Paddy Doctor:  
Paddy Disease  
Classification 대회

02 CNN 모델 소개

03 초기모델 코드 분석

04 초기 모델 성능 분석

05 VGG16 vs MobileNet

06 VGG16 모델 소개

07 MobileNet 조정 결과

08 MobileNet 조정 결과

09 결론

10 좋았던 점 / 아쉬웠던 점



Image source: www.knowledgebank.irri.org and stock.adobe.com



### 대회 : Paddy Doctor : Paddy Disease Classification

내용 : 벼 잎 이미지에 존재하는 질병 유형 식별

#### train.csv

- image\_id : 고유한 이미지 식별자는 train\_images 디렉터리에 있는 이미지 파일 이름(.jpg)에 해당
- label : paddy disease의 유형. 대상 클래스. 10개의 범주가 있음.
  - bacterial\_leaf\_blight, bacterial\_leaf\_steak, bacterial\_panicle\_blight 등
- variety : 논 품종의 이름
  - ADT45, Ponni, Surya
- age : 벼의 나이(days)

#### sample\_submission.csv

- image\_id : 200001.jpg ~ 203469.jpg 3,469개의 이미지명.
- label : 예측할 품종



# 01 | Paddy Doctor: Paddy Disease Classification 대회

- result

In [17]:

```
results['label'].value_counts()
```

Out[17]:

normal	643
blast	618
hispa	543
dead_heart	462
tungro	376
brown_spot	274
downy_mildew	167
bacterial_leaf_blight	153
bacterial_leaf_streak	119
bacterial_panicle_blight	114
Name: label, dtype: int64	

### CNN 모델

---

1. AlexNet



초기 사용 모델과 유사한 구조

2. VGGNet

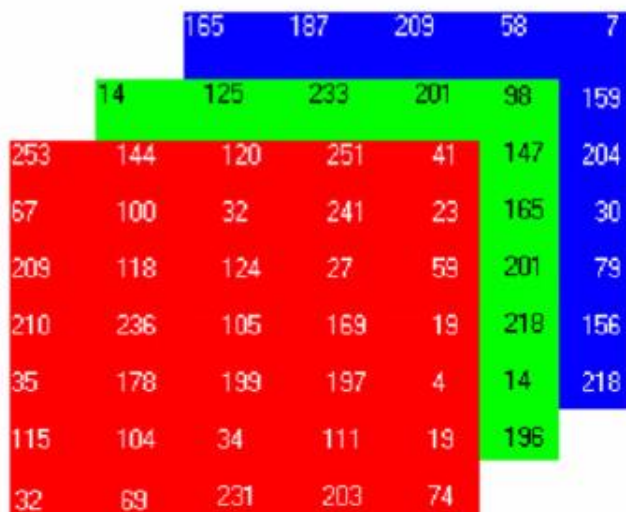
3. MobileNet



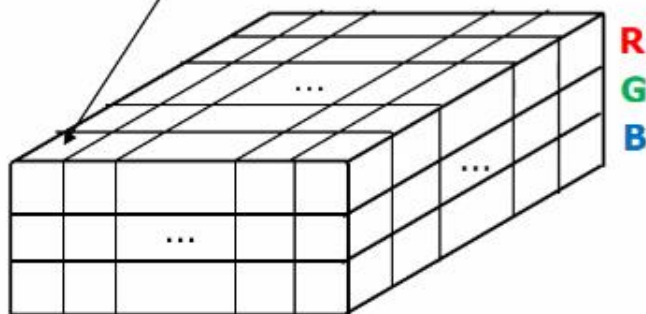
최종 사용 모델

## CNN 모델이란?

<https://yeong-jin-data-blog.tistory.com/entry/%EC%82%AC%EC%A0%84%ED%95%99%EC%8A%B5-%EB%AA%A8%EB%8D%B8-CNN>



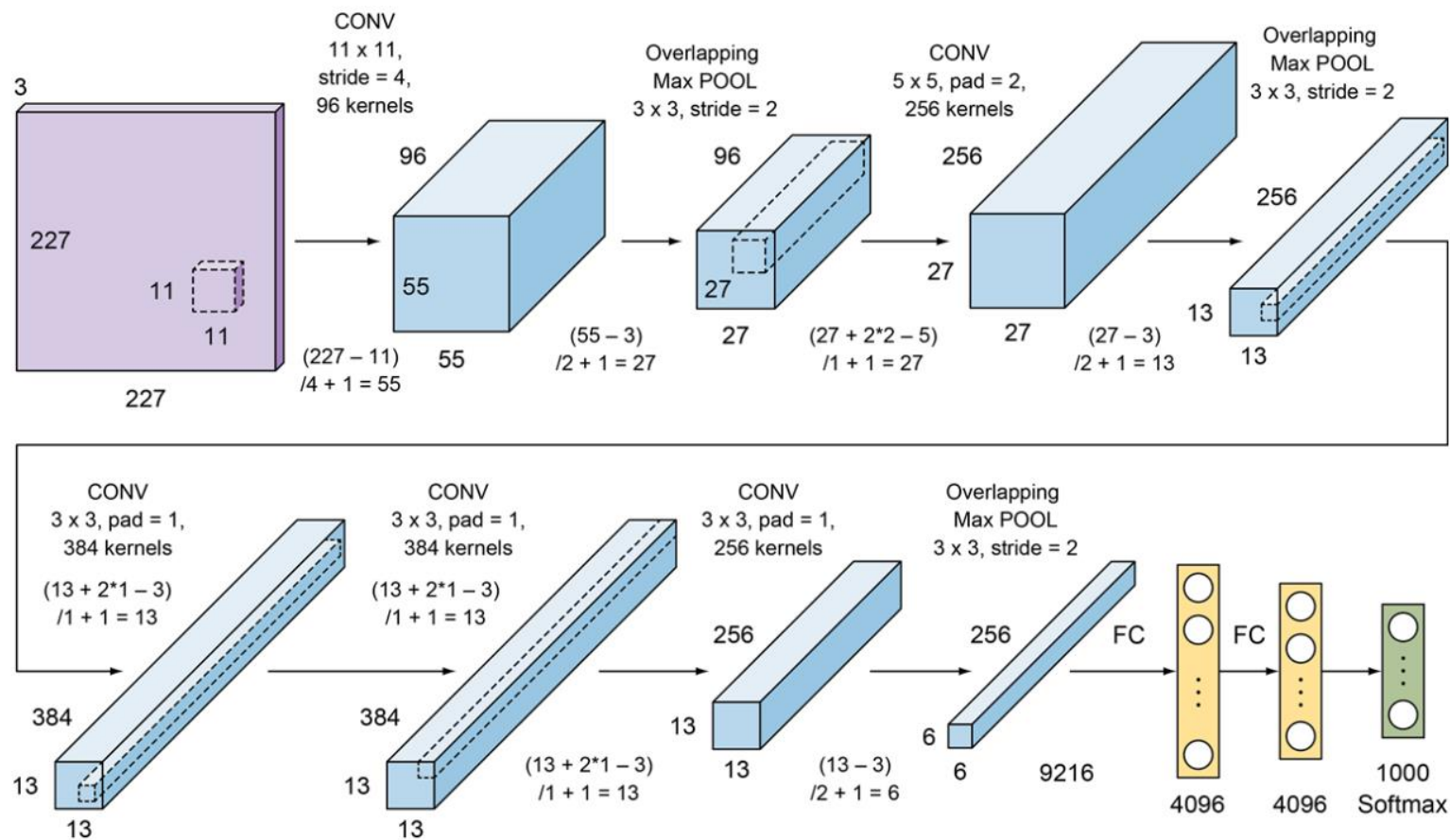
각 셀이 하나의 색상 정보를 지님  
각 셀이 하나의 feature (독립변수)의 값이 됨



- 3차원 이미지 그대로 사용
- 필터를 이용해 이미지의 정보 추출(합성곱 연산)을 반복
- 일반 신경망의 한계 극복
  - > 공간정보 보존
  - > 파라미터 ↓ (과적합 방지)

## AlexNet 모델이란?

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 1097-1105.



Alex Net 구조

- 여러 개의 합성곱 필터와 풀링 레이어를 가짐.
- 활성화 함수로 ReLU 사용
- MaxPooling 사용



## 코드 분석

---

### ✅ 이미지 제너레이터를 활용한 데이터 증식

→ ImageDataGenerator : 만나지 못한 데이터 추가함으로써 좀 더 성능 좋게

```
img_datagen = tf.keras.preprocessing.image.ImageDataGenerator(  
    rescale=1.0/255.0,  
    validation_split=0.1,  
    rotation_range=5,  
    shear_range=0.3,  
    zoom_range=0.3,  
    width_shift_range=0.05,  
    height_shift_range=0.05,  
    horizontal_flip=True,  
    vertical_flip=True,  
)
```

## 코드 분석

---

### ✓ Early\_stopping

```
early_stopping_callback = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                                            min_delta=0,
                                                            patience=5,
                                                            verbose=0,
                                                            mode='auto')

lr_reducer = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss',
                                                    factor=0.5,
                                                    patience=5)

callbacks = [early_stopping_callback, lr_reducer]
```

## 코드 분석

✓ ReLU, softmax 활성화 함수 사용

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(2048, activation='relu' ),
    tf.keras.layers.Dense(1024, activation='relu' ),
    tf.keras.layers.Dense(128, activation='relu' ),
    tf.keras.layers.Dense(N_CLASS, activation='softmax' )
])
```

- AlexNet 구조와 유사 (ReLU, MaxPooling)
- softmax : 결과를 10개 범주로 출력(가장 높은 확률 값 선택)

# 04 | 초기 모델 성능 분석

## 성능 분석 - epoch, img\_dim

### Epochs

표

#### epochs 조정

**epochs**   소요시간(GPU)   # 제출 최종 결과(Score)   Medal

10		0.65936	
29	3774.0s	0.8612	
30	3908.1s	0.86082	
35	3834.9s	0.8662	🏆
41	4250.5s	0.84775	
100	4285.5s	0.86466	🏆

### Img\_dim

표 +

#### img\_dim 조정 ...

**img\_dim**   epochs   소요시간(GPU)   # 제출 최종 결과(정확도)

128	31	3908.1s	0.86082
256	28	4500.0s	0.89926



epochs와 img\_dim은 커질 수 록 Score가 올라감이 명확

## 성능 분석 - batch\_size

### Batch\_size

Show All

#### ↗ Batch\_size 조정

<u>Aa</u> batch_size	소요시간(GPU)	제출 최종 결과(Score)	Medal
<u>8</u>	4024.2s	0.8366	
<u>16(basic)</u>	3080.4s	0.86466	
<u>32</u>	3434.9s	0.89696	

batch\_size 작을 수록, 소요시간 커짐.

batch\_size 클 수록, Score 커짐.

## 성능 분석 - Conv2D, MaxPooling

basis

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(2048, activation='relu'),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(N_CLASS, activation='softmax')
])
```

# 04 | 초기 모델 성능 분석

## Conv2D, MaxPooling\_Up

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(256, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(N_CLASS, activation='softmax')
])
```

## Conv2D, MaxPooling\_Down

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu'),
),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(8192, activation='relu'),
    tf.keras.layers.Dense(4096, activation='relu'),
    tf.keras.layers.Dense(2048, activation='relu'),
    tf.keras.layers.Dense(1024, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(N_CLASS, activation='softmax')
])
```

## 성능 분석

### Conv2D, MaxPooling,Dense 조정

Conv2D,MaxPooing	Dense	소요시간(GPU)	제출 최종 결과(Score)	파라미터 개수	Medal
<u>4개(basic)</u>	3개	3908.1s	0.86082	11,767,338	
<u>5개</u>	2개	3234.9s	0.79546	984,362	
<u>3개</u>	5개	3038.2s	0.84429	146,972,074	

Conv2D, MaxPooling layer 적을 수록, 소요시간 줄어듬.

Conv2D, MaxPooling layer 많을 수록, Parameter 줄어듬.

Conv2D, MaxPooling layer의 증가와 Score 사이에 상관관계가 명확하지 않고, layer들 사이의 균형이 중요해 보임.



## 04 | 초기 모델 성능 분석

### 성능 분석 - filter size

#### Basis

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (3,3), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
```

#### Filter size\_up

```
model = tf.keras.Sequential([
    tf.keras.layers.InputLayer(input_shape=(img_dim, img_dim, 3)),
    tf.keras.layers.Conv2D(16, (5,5), activation='relu' ),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(32, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2,2),
```

## 성능 분석 - filter size

### filter 크기 조정

<u>Aa</u> filter size	 소요시간(GPU)	 제출 최종 결과(정확도)	 파라미터 개수
<u>3*3</u>	3080.4s	0.86466	448
<u>5*5</u>	2986.3s	0.86966	1216

💡 파라미터 개수 3배 증가에 반해, 정확도 증가는 미미

💡 Filter size 클 수록, 무조건 Score 증가?

## 추가 성능 분석 - batch\_size

### ▼ basic

epoch : 35

img\_dim : 256

batch\_size : 32

filter size : (4,4) → 모든 Conv2D 필터 변경

- 6888.9s - GPU
- Score: 0.92272

### batch\_size ( 32 → 64 )

- 6443.2s - GPU
- Score: 0.89734

### batch\_size ( 32 → 16 )

- 7917.2s - GPU
- Score: 0.91503

💡 batch\_size와 성능 사이에 명확한 상관관계 X

(일반적으로 배치사이즈는 작게 줄 수록 성능이 올라감) &  
(adam optimizer는 배치사이즈가 커질수록 좋음.)

⇒ 두가지 특징이 충돌. 중간지점 필요










🔍 batch\_size와 성능의 상관관계는 아직 정확하게 규정되어있지 않고, task나 데이터에 따라 그 기준이 달라짐.


✅ paddy disease의 데이터는 32 사이즈가 적당해 보임


# 04 | 초기 모델 성능 분석

## highest Score 조합 & 원인 분석

### Batch\_size & filter size 조정


 batch_size	 filter size	 epoch	 img_dim	 Time -GPU (s)	 Score	 Medal
32	(3,3)	35	256			
32	(4,4)	35	256	6888.9	0.92272	
32	(5,5)	35	256	7480.8	0.92156	
64	(4,4)	35	256	6443.2	0.89734	
16	(4,4)	35	256	7917.2	0.91503	
16	(3,3)	35	256	5676.5	0.89504	
64	(5,5)	35	256	7081.7	0.90849	
16	(5,5)	35	256	5124.1	0.92772	
64	(3,3)	35	256			

 batch\_size와 filter의 크기가 크거나 작다고 무조건 성능이 좋아지는 것이 아님.

 batch\_size와 filter 크기 사이의 균형이 중요한 것으로 보여짐.

## 05 | VGG16 vs MobileNet

### Paddy Doctor: 논 질병 분류를 위한 시각적 이미지 데이터 세트

2022년 5월 23일 · Petchiammal A, Briskline Kiruba S, D. Murugan, Pandarasamy A ·  소셜 미리보기 수정

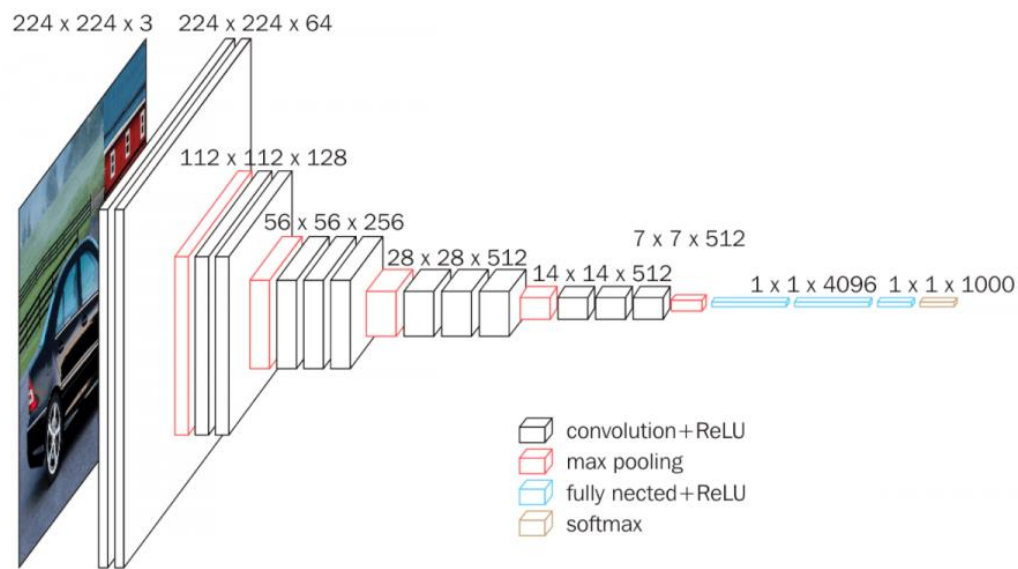
논농가가 직면하는 중요한 생물학적 스트레스 요인 중 하나는 박테리아, 곰팡이 및 기타 유기체로 인한 질병입니다. 이러한 질병은 식물의 건강에 심각한 영향을 미치고 심각한 작물 손실을 초래합니다. 이러한 질병의 대부분은 전문가의 감독하에 정기적으로 잎과 줄기를 관찰하여 식별할 수 있습니다. 광대한 농업 지역과 제한된 작물 보호 전문가가 있는 국가에서 벼 질병을 수동으로 식별하는 것은 어렵습니다. 따라서 이 문제에 대한 솔루션을 추가하려면 질병 식별 프로세스를 자동화하고 효과적인 작물 보호 조치를 가능하게 하기 위해 쉽게 액세스할 수 있는 의사 결정 지원 도구를 제공해야 합니다. 그러나 자세한 질병 정보가 포함된 공개 데이터 세트의 가용성 부족으로 정확한 질병 감지 시스템의 실제 구현이 제한됩니다. 본 논문에서는 논병을 식별하기 위한 영상 데이터셋인 Paddy Doctor를 제시한다. 우리의 데이터 세트에는 10개 클래스(9개 질병 및 정상 잎)에 걸쳐 주석이 달린 벼 잎 이미지 13,876개가 포함되어 있습니다. 우리는 CNN(Convolutional Neural Network)과 두 가지 전이 학습 접근 방식인 VGG16 및 MobileNet을 사용하여 Paddy Doctor를 벤치마킹했습니다. **실험 결과 MobileNet이 93.83\%의 가장 높은 분류 정확도를 달성함을 보여줍니다.** 커뮤니티 사용을 위해 오픈 소스로 데이터 세트와 재현 가능한 코드를 공개합니다. 실험 결과 MobileNet이 93.83\%의 가장

출처 : <https://paperswithcode.com/paper/paddy-doctor-a-visual-image-dataset-for-paddy>

VGG16 VS MobileNet

# 06 | VGG16 모델 소개

## ▲ VGG16



### VGG16의 구조

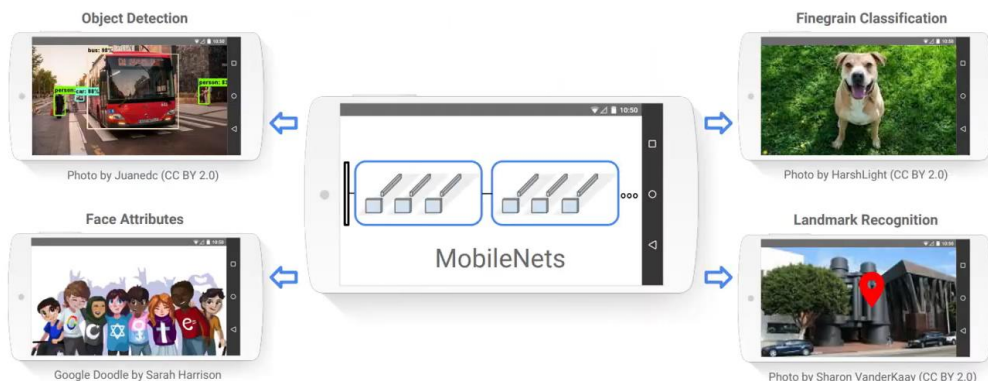
(13 Convolution Layers + 3 Fully-connected Layers)

출처 : <https://youtu.be/MaDakbMDBrl>

- VGG16은 16개의 층(Layer)로 구성된 모델을 의미
- 필터를 가장 작은 사이즈인 3X3만 사용하는 이유
  - ✓ 파라미터의 개수가 줄어드는 효과와 ReLU가 활성화 함수로 들어갈 수 있는 곳이 많아진다는 장점
- VGG19는 층이 더 많은 만큼 메모리등의 자원을 많이 소모
  - ✓ 성능은 VGG16과 비슷하거나 떨어지는 편

# 07 | MobileNet 모델 소개

## ▲ Mobilenet



- 목적 : 성능 저하를 최소화하면서 딥러닝 모델 파일의 크기를 줄이는 것
- 구글에서 2017년에 발표하였고, Xception과 비슷함
- 최대한 성능을 보존하면서 컴퓨팅 속도를 빠르게, 정확성 보다는 효율성 관점에서 만들어 짐

## ▲ Mobilenet 선택 이유

- MobileNetV2: Paddy Disease Classifier  
(<https://www.kaggle.com/code/abhishek123maurya/mobilenetv2-paddy-disease-classifier>)
- VGG16\_Paddy\_Disease\_Classify  
(<https://www.kaggle.com/code/amanbahuguna/vgg16-paddy-disease-classify-92-81-acc>)



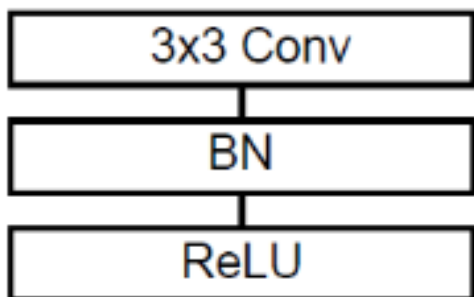
Time : 1320.0s(GPU), Score : 0.93810



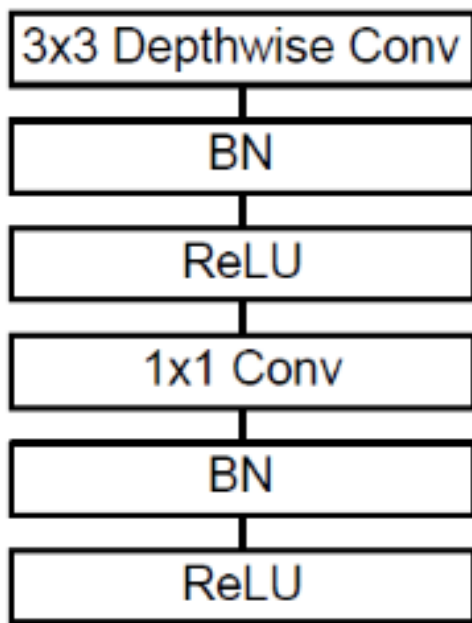
Time : 1646.5s (GPU), Score : 0.92349

# 07 | MobileNet 모델 소개

## ▲ Mobilenet 구조



기존 CNN 구조



MobileNet 구조

```
def get_model(base, preprocessor, img_size):
    inputs = tf.keras.Input(shape=(img_size, img_size, 3))
    x = RandomFlip('horizontal')(inputs)
    x = preprocessor(x)
    x = base(x)

    x = tf.keras.layers.Flatten()(x)
    x = tf.keras.layers.Dense(1024, activation='relu')(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dense(512, activation='relu')(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dense(128, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.15)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    x = tf.keras.layers.Dense(64, activation='relu')(x)
    x = tf.keras.layers.Dropout(0.3)(x)
    x = tf.keras.layers.BatchNormalization()(x)
    outputs = tf.keras.layers.Dense(10, activation='softmax')(x)

    model = tf.keras.Model(inputs, outputs)

    return model
```

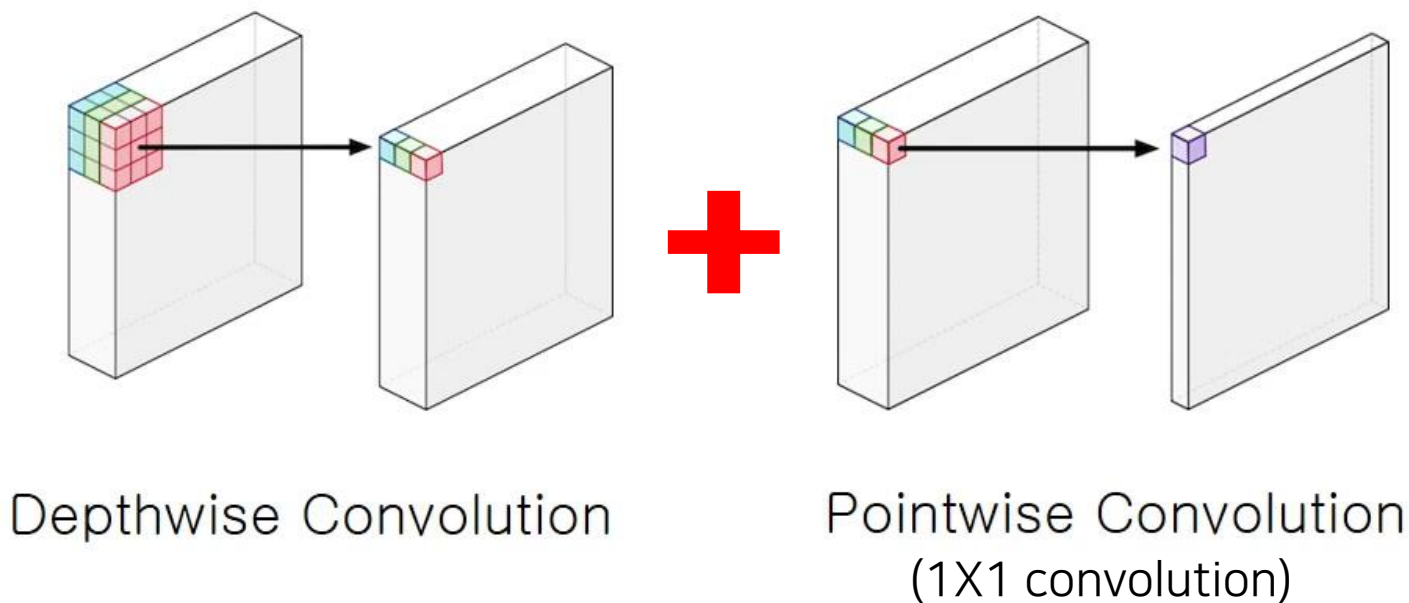
실제 code에 사용된 MobileNet 구조

- 3X3 → 1X1 순서의 conv 사용하며, BatchNormaliztion과 ReLU 활성화 함수가 두 Conv 사이에 있는 구조



# 07 | MobileNet 모델 소개

## ▲ Mobilenet의 핵심 : Depthwise Separable Convolution



출처 : <https://sotudy.tistory.com/15>

- 공간(spatial)에 대한 Convolution을 먼저 수행 후, 채널(channel)에 대한 Convolution 수행하며 연산량을 줄임

## 08 | MobileNet 조정 결과

### ▲ Batchsize 조정

Batch_size	Time-GPU(s)	Score	비고
16	1406.5	0.92041	
32	1410.0	0.91541	
64(basis)	1320.0	0.9381	🏆
128	1299.7	0.87966	
256	1419.2	0.76701	

- 너무 크지도 않고 작지도 않은 크기의 batch\_size가 성능이 가장 높았음. → 균형 중요


### ▲ Epochs 조정

Epochs	Batch_size	Time-GPU(s)	Score	비고
History : 5 History_tuned : 14 (basis)	64	1320.0	0.9381	🏆
History : 14 History_tuned : 30	64	2878.4	0.91503	

- MobileNet 사용 모델과 기존 AlexNet 사용 모델을 비교

- ✓ 비슷한 정확도에서 속도 훨씬 빠른 모습 확인

⇒ MobileNet의 Depthwise Separable Convolution의 구조상 특징으로 연산량을 줄였기 때문에 속도 빨라진 것

Batch_size	Time-GPU(s)	Score	비고
16	1406.5	0.92041	
32	1410.0	0.91541	
64(basis)	1320.0	0.9381	

MobileNet 사용 모델

- Time: 6888.9s - GPU
- Score: 0.92272

기존 AlexNet 사용 모델

- ✓ 성능 향상

⇒ 사용자가 직접 CNN모델을 구축한 방법이 아닌, 이미 방대한 양의 학습 데이터를 이용해서 학습을 완료한 모델인 **사전학습 모델 사용**, 해당 파라미터 최적값이 학습을 통해 정해져있으므로 성능향상으로 이어진 듯

## 10 | 좋았던 점 / 아쉬웠던 점

- **좋았던 점**

- ✓ CNN의 여러 종류의 모델의 구조를 배워볼 수 있었던 점
- ✓ Kaggle 대회 참여로 성능향상 방법에 대해 계속해서 고민해보고 시도해 본 점
- ✓ Notion을 사용하며 의견과 진행사항 공유가 잘 되었던 점

- **아쉬웠던 점**

- ✓ 시간상 MobileNet 모델의 성능향상을 위한 조정시도가 부족했던 점
- ✓ 그로 인해 MobileNet 모델의 성능향상이 이루어지지 않은 점

**감사합니다**