Paddy Disease

Classification

최종 발표

구름다리

목차

구름다리 팀 구성

- 팀　장 : 박종민
- 부팀장 : 손희경
- 팀　원 : 윤형석, 백진선, 박규리

프로젝트 소개

대회 소개

벼 경작은 여러 질병·해충으로 최대 70%의 수확량 손실을 초래할 수 있기 때문에 일관된 감독이 필요
→ 컴퓨터 비전 기반 자동화된 질병 식별 프로세스 필요

주어진 벼 이미지를 9개의 질병 범주 중 하나 또는 정상 잎으로 정확하게 분류하는 딥러닝 모델 개발

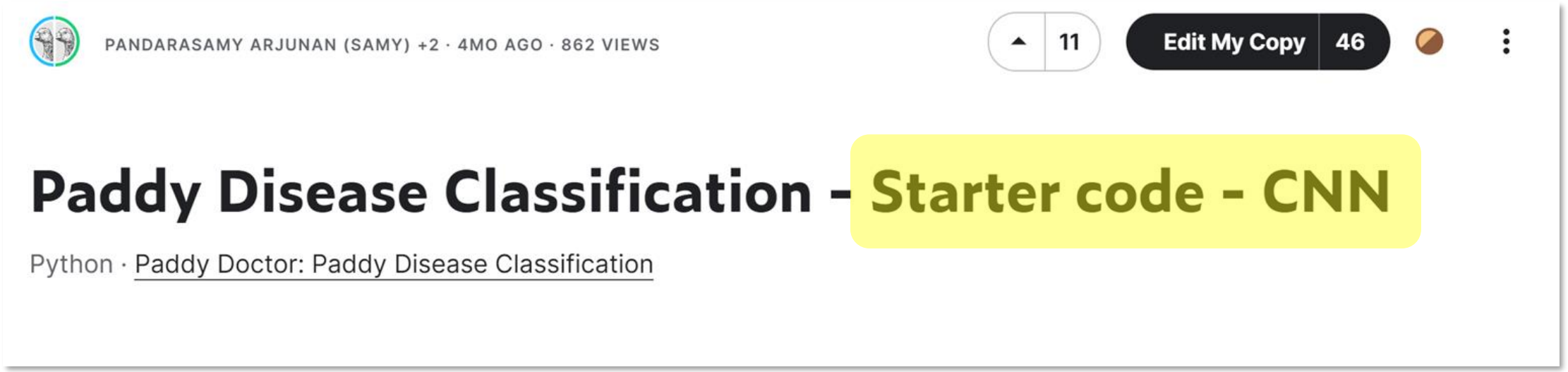
팀 프로젝트 내용

캐글 대회의 코드를 베이스 라인으로,
모델을 수정하여 **성능을 향상**하여
최적의 최종 모델 찾기

첫 번째 모델

(중간 발표 모델)

초기에 사용한 모델 (중간 발표 모델)



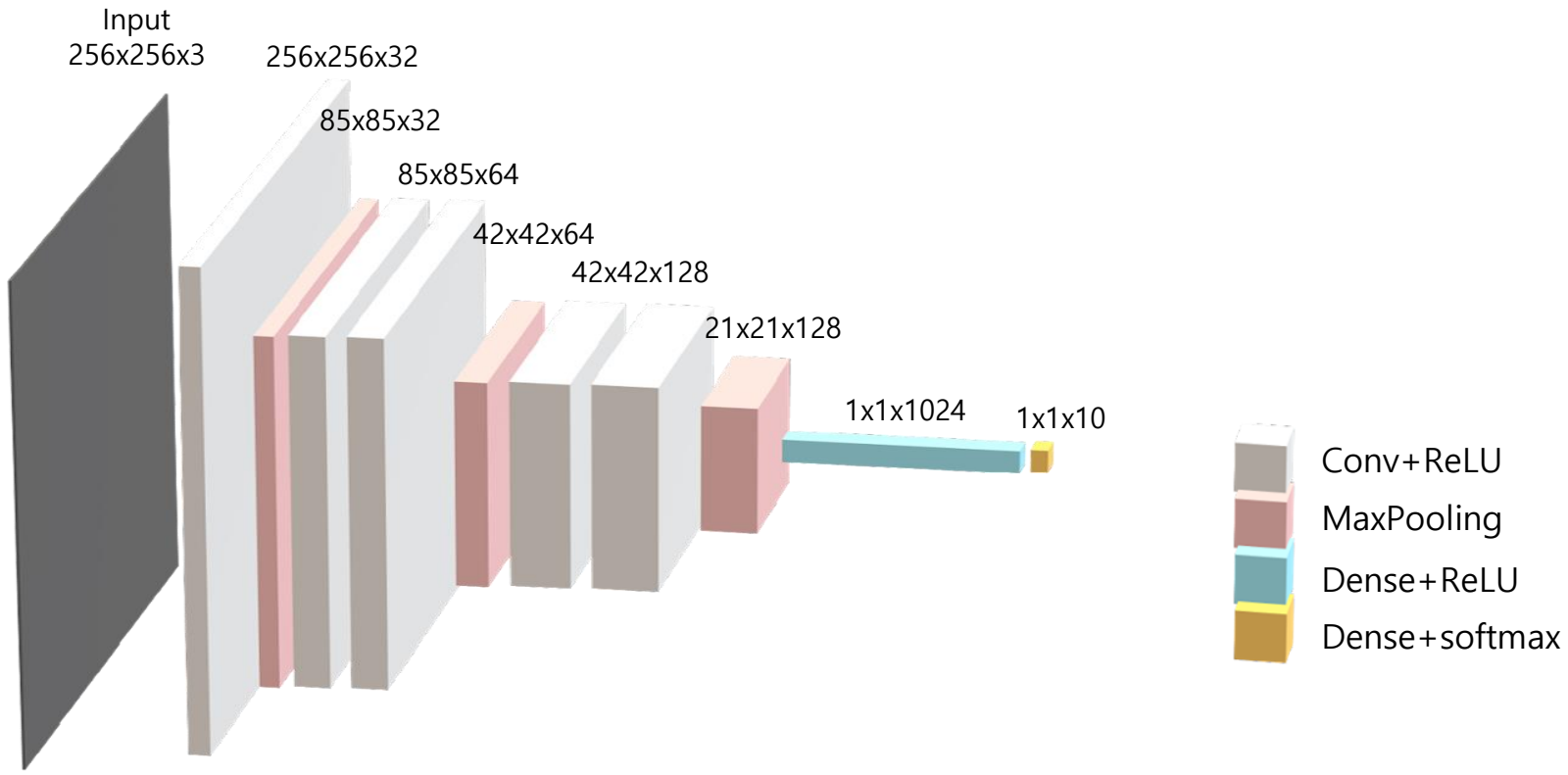
캐글에서 CNN starter code 참고

https://www.kaggle.com/code/samy101/paddy-disease-classification-starter-code-cnn

초기에 사용한 모델 (중간 발표 모델)

모델 구조 도식화

(BatchNormalization, Dropout 그림에서 생략)



https://www.kaggle.com/code/samy101/paddy-disease-classification-starter-code-cnn

초기에 사용한 모델 (중간 발표 모델)

이후 모델 수정으로 정확도를 개선하고자 함.

Epochs 조절

Dropout

Batch size 조절

Image size 조절

Batchnormalization

활성화함수 변경
(ReLU → LeakyReLU)

Padding 조절

But 정확도 개선에 **한계 존재**

(**최대 94.886%**, 이외 43%~93%으로 다양)
(중간 결과 발표 때 결과 정리하였음)

전이학습 시도

가설 설정

기존 CNN 모델의 수정에도 정확도 개선에 한계가 있었지만, **사전훈련모델을 이용하는 전이학습**을 통해 정확도를 높일 수 있을 것이며, 학습속도가 빨라질 것임.

이후 전이학습에 사용한 사전훈련모델

<1>

VGG16

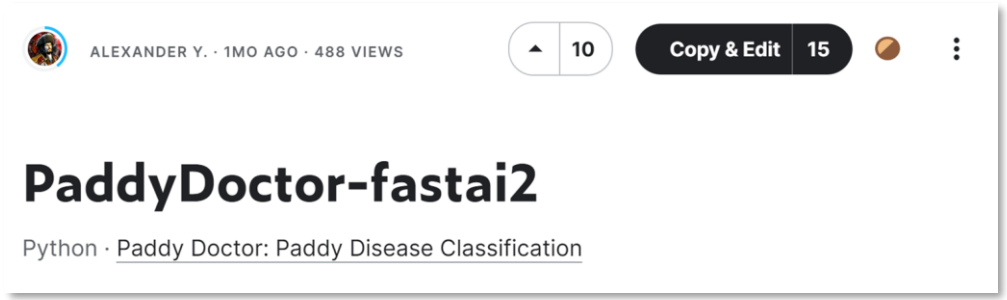
(Keras 사용)

https://www.kaggle.com/code/amanbahuguna/vgg16-paddy-disease-classify-92-81-acc

<2>

ResNet50

(Fastai2 사용)

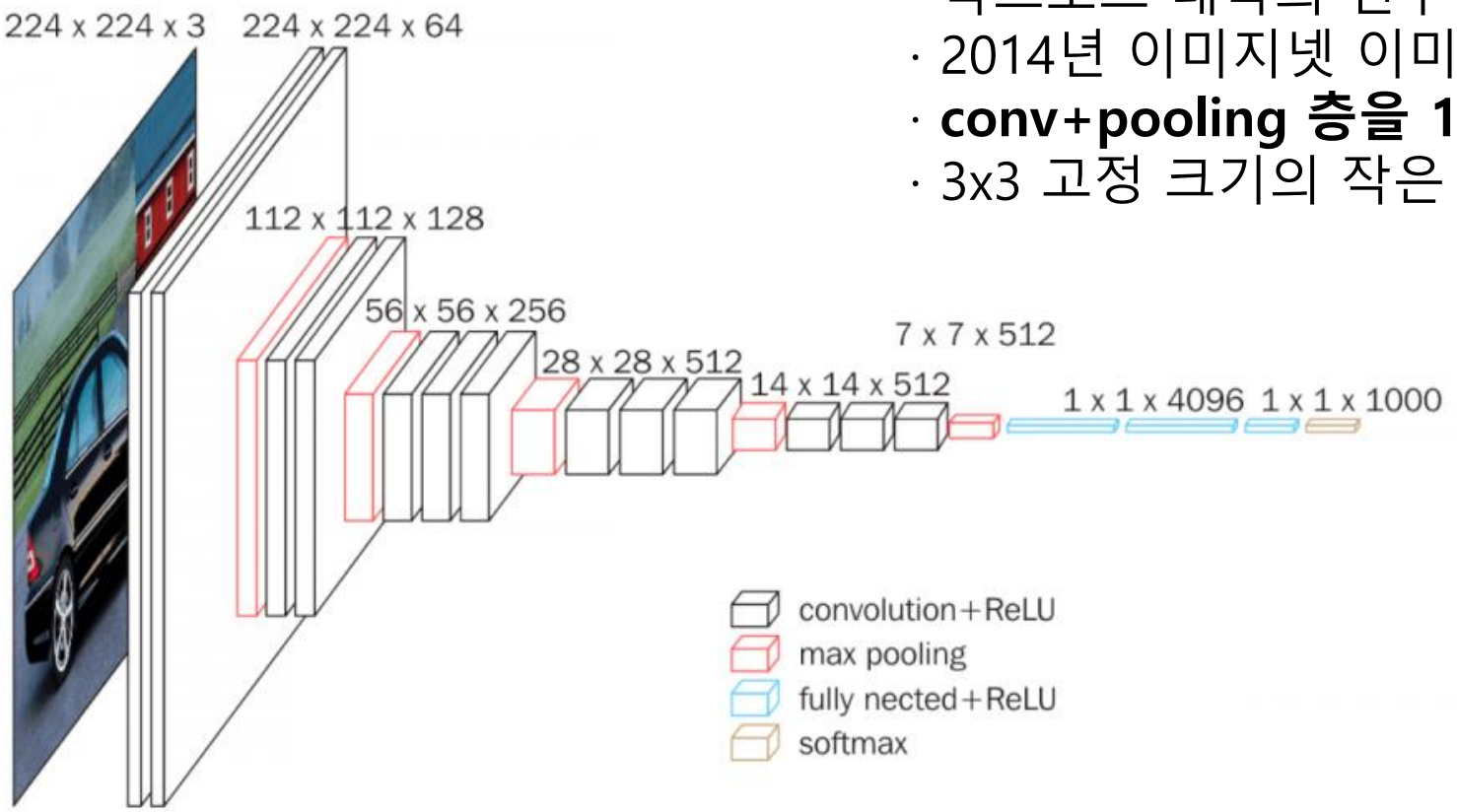


https://www.kaggle.com/code/alexanderyyy/paddydoctor-fastai2

VGG 16

(전이학습 모델 1)

VGG16 설명



· 옥스포드 대학의 연구팀 VGG에 의해 개발된 모델
· 2014년 이미지넷 이미지 인식 대회에서 준우승
· **conv+pooling 층을 16층으로 심화**
· 3x3 고정 크기의 작은 필터

VGG16 전이학습 결과 정리

Base Model 정보

batch_size = 32
image_size = (224, 224)
epochs = 50
trainable_layer = 2

batch size

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	1590.5s	0.3824	0.92772		
batch size 32 → 64	1125.2s	0.3894	0.92349	↑	↓
batch size 32 → 16	1703.2s	0.3882	0.91464	↓	↓
batch size 32 → 128	901.5s	0.4211	0.93002	↑	↑
batch size 32 → 256	1199.1s	0.3678	0.93271	↑	↑

VGG16 전이학습 결과 정리

epochs

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	1590.5s	0.3824	0.92772		
epochs 50 → 100	1926.3s	0.3563	0.93156	↓	↑

Trainable_layer

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	1590.5s	0.3824	0.92772		
Trainable layer 2 → 3	1795.2s	0.2840	0.94963	↓	↑
Trainable layer 2 → 1	1786.3s	0.8180	0.79546	↓	↓

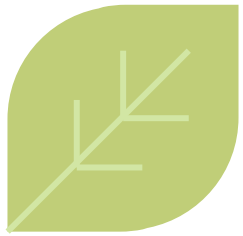
VGG16 전이학습 결과 정리

가장 정확도가 좋았던 버전

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	1590.5s	0.3824	0.92772		
Trainable layer 2 → 3	**1795.2s**	**0.2840**	**0.94963**	↓	↑

94.963%

(이는 첫번째 모델 최고정확도 94.886% 과 큰 차이 X)



ResNet 50

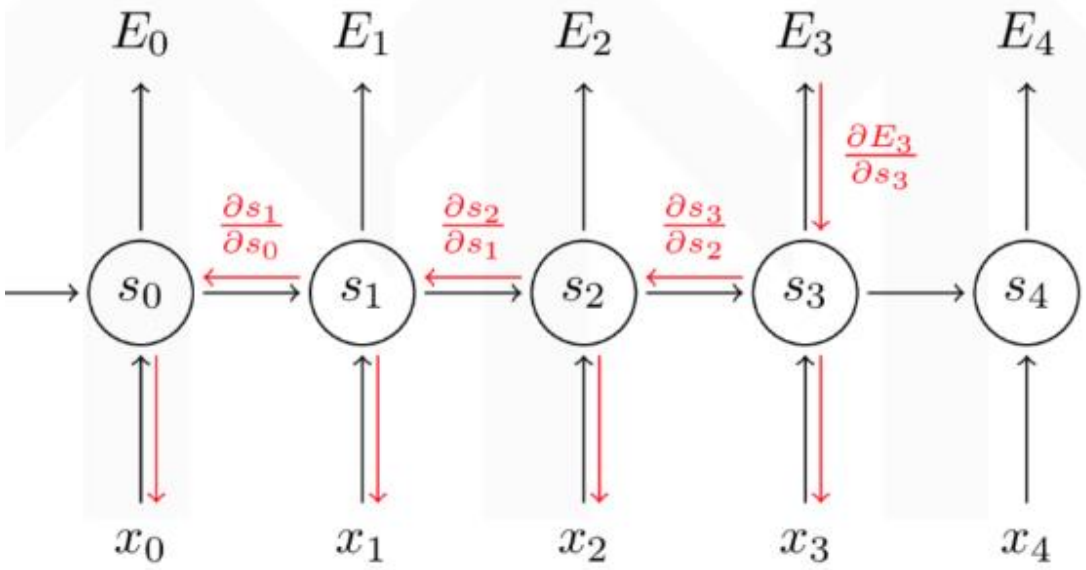
(전이학습 모델 2)

Fastai 설명

- pytorch를 기반으로 설계된 **라이브러리** (Pytorch의 상위 wrapper)
- fastai2 = fastai v2
- 이 라이브러리는 딥러닝 모델을 만드는 코드 스킬 없이
  빠르게 딥러닝 모델을 학습시켜서 사용할 수 있도록 하는 것을 목표로 개발되어서,
  복잡한 구현없이 딥러닝 모델을 생성할 수 있다.

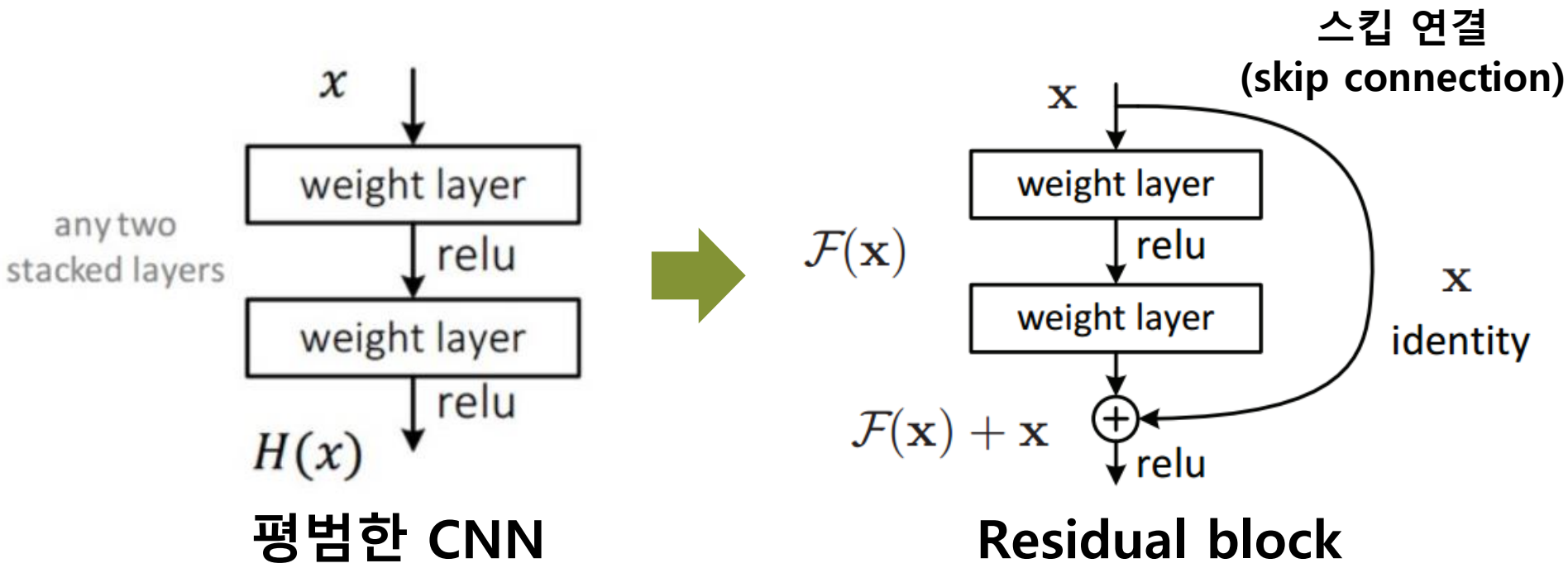
ResNet 설명

· 마이크로소프트 연구팀에 의해 개발된 모델
· 2015년 152개의 Conv layer를 채용하면서도 ImageNet test(Top-1)에서 80.72%의 정확도 달성
· 층을 깊게 함에 따라 생기는 **Vanishing Gradient Problem**를 **Residual Block**으로 해결



Vanishing Gradient Problem

Residual Block 설명



평범한 CNN

Residual block

ResNet50 설명

Residual block을 바탕으로 한 ResNet50의 구조

- 총 50개의 layer 가짐, 각각의 conv에 있는 []가 residual block.
- ResNet50은 layer마다 다른 residual block 형태가 반복되어 학습되는 과정을 거침

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

ResNet50 전이학습 결과 정리

Base Model 정보

image_size 224
batch_size 32
epochs 30
freeze_epochs 1
tta n 32

batch size

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
batch size 32 → 64	6844.2s	0.039010	0.98346	↑	-
batch size 32 → 16	6608.8s	0.172993	0.98116	↑	↓

ResNet50 전이학습 결과 정리

Freeze_epochs

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
freeze_epochs 1 → 2	6963.2s	0.026470	0.98193	↓	↓
freeze_epochs 1 → 3	6780.4s	0.079041	0.98346	↑	-

Image_size

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
Image_size 224 → 300	7985.3s	0.006672	0.98385	↓	↑
Image_size 224 → 448	9397.6s	0.002149	0.98462	↓	↑

ResNet50 전이학습 결과 정리

epochs

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
epochs 30 → 60	10917.1s	0.123482	0.985	↓	↑
epochs 30 → 100	17755.3s	0.016897	0.98615	↓	↑

TTA n

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
tta n 32 → 64	7832.4s	0.028258	0.98462	↓	↑
tta n 32 → 16	6125.3s	0.030517	0.98423	↑	↑

ResNet50 전이학습 결과 정리

여러 요소 한번에 조절

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
epochs 30 → 60 tta n 32 → 64	14789.9s	0.073718	0.985	↓	↑
epochs 30 → 60 tta n 32 → 64 image_size 224 → 300	14479.9s	0.018465	0.985	↓	↑
epochs 30 → 60 image_size 224 → 448	19200.6s	0.019863	0.98462	↓	↑
epochs 30 → 100 **image_size** **224 → 300** **tta n 32 → 64**	21786.1s	0.076063	0.98462	↓	↑

ResNet50 전이학습 결과 정리

가장 정확도가 좋았던 버전

구분	소요시간 (GPU)	검증 데이터 셋 loss	제출 최종 결과 (정확도)	insight	
				속도	정확도
Base model	6955.6s	0.072411	0.98346		
epochs 30 → 100	**17755.3s**	**0.016897**	**0.98615**	↓	↑

98.615%

**(이는 첫번째 모델 최고정확도 94.886%,
VGG16 전이학습 모델 최고정확도 94.963%에서 많은 향상)**

최종 모델 결정

ResNet 50

image_size 224
batch_size 32
epochs 100
freeze_epochs 1
tta n 32

소요시간 (GPU)

17755.3s

제출 최종 결과 (정확도)

98.615%

코드 분석 및 설명

In [1]:

```
import numpy as np
import pandas as pd

from fastai.vision.all import *

import albumentations as Alb
```

fastai.vision에 속해있는 모든 라이브러리를 가져온다.
(4개의 하위모듈 → image, transform, data, learner)

데이터 증강 라이브러리.
회전, 확대, 왜곡, 반전 등의 함수를 사용할 수 있다.

In [2]:

```
INPUT_PATH = '../input/paddy-disease-classification'
TRAIN_PATH = INPUT_PATH + '/train_images'
TEST_PATH = INPUT_PATH + '/test_images'
```

데이터 경로

코드 분석 및 설명

In [3]:

```python
# 이미지 증식
class AlbTransform(Transform):
    def __init__(self, aug):
        self.aug = aug

    def encodes(self, img: PILImage):
        aug_img = self.aug(image=np.array(img))['image']
        return PILImage.create(aug_img)
```

```python
def get_augs():
    return Alb.Compose([
        Alb.ShiftScaleRotate(rotate_limit=20, border_mode=0),
        Alb.Transpose(),
        Alb.Flip(),
        Alb.RandomRotate90(),
        Alb.RandomBrightnessContrast(),
        Alb.HueSaturationValue(
            hue_shift_limit=5,
            sat_shift_limit=5,
            val_shift_limit=5
        )
    ])
```

- ShiftScaleRotate

무작위로 아핀 변환을 적용
즉, 입력을 변환, 축척 및 회전

- Transpose()

행과 열을 바꿔서 입력

- Flip()

입력을 수평, 수직 또는 수평과 수직 둘 다로 뒤집음

- RandomRotate90()

임의로 입력을 0번 이상 90도 회전

- RandomBrightnessContrast()

입력 이미지의 밝기와 대비를 임의로 변경

- HueSaturationValue

입력 이미지의 색상, 채도 및 값을 임의로 변경
hue_shift_limit=5, #색상 default : (-20, 20)
sat_shift_limit=5, #채도 default : (-30, 30)
val_shift_limit=5 #값 default : (-20, 20)

코드 분석 및 설명

```
item_tfms = [Resize(224), AlbTransform(get_augs())]
batch_tfms = Normalize.from_stats(*imagenet_stats)
```

이미지 크기 변경

이미지 정규화

In [4]:

```
# create datalodaer from the folder structure
dls = ImageDataLoaders.from_folder(
    TRAIN_PATH,
    train='.',
    valid=None,
    valid_pct=0.01,
    item_tfms=item_tfms,
    batch_tfms=batch_tfms,
    bs=32,
    shuffle=True
)
```

- ImageDataLoaders
  - item_tfms: batching 전에 변형
  - batch_tfms: batches 에 적용
  - bs: batch size
  - val_bs: batch size of validation (default - bs)
  - shuffle_train: shuffle or not
- from_folder
  - train, valid: subfolders in path
  - valid_pct: random split with percentage

In [5]:

```
print('train items:', len(dls.train.items), 'validation items:', len(dls.valid.items))
dls.vocab
```

코드 분석 및 설명

- vision_learner
  - resnet50
  - loss func - FocalLoss
    - crossentropy의 클래스 불균형 문제 다루기 위한 개선된 버전
      - Cross Entropy Loss는 잘못 예측한 경우에 대하여 페널티 부여하는 것에 초점
    - 어렵거나 쉽게 오분류되는 케이스에 대하여 더 큰 가중치를 주는 방법

In [6]:

```
learn = vision_learner(
    dls,
    resnet50,
    path='.',
    loss_func=FocalLoss(),
    metrics=[accuracy]
)
```

```
Downloading: "https://download.pytorch.org/models/resnet50-0676ba61.pth" to /root/.ca
che/torch/hub/checkpoints/resnet50-0676ba61.pth

100% 97.8M/97.8M [00:07<00:00, 20.9MB/s]
```

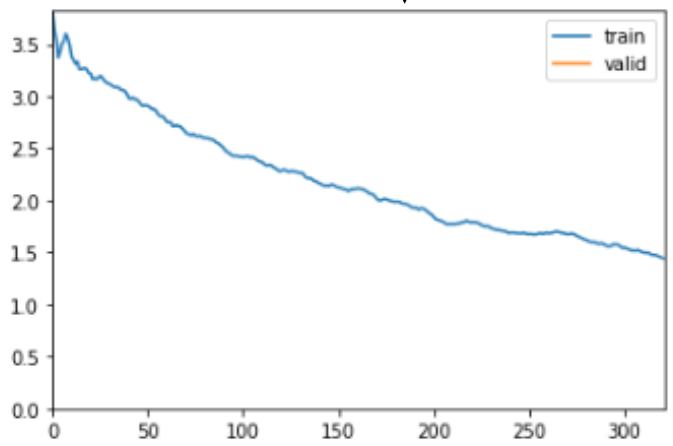
코드 분석 및 설명

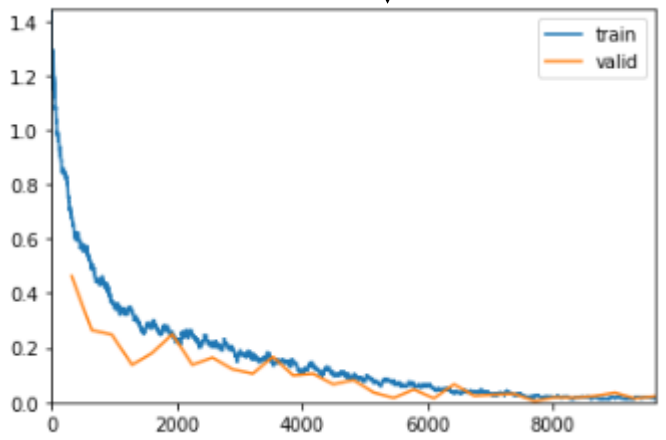
In [7]:
```
learn.fine_tune(30, freeze_epochs=1, cbs=[ShowGraphCallback()])
```

- fine_tune(epochs, base_lr=0.002, freeze_epochs=1, lr_mult=100, pct_start=0.3, div=5, ...)
  - freeze -> one_cycle -> unfreeze -> one_cycle 한번에 진행
  - freeze: 각 레이어의 trainable을 False (freeze_to(-1)가 default)
  - classifier 빼놓고 앞단의 레이어들의 trainable=False
  - freeze_epochs 동안 classifier 빼놓고 앞단의 레이어들의 trainable=False => classifier learning
  - epochs 동안 모든 레이어 trainable=True and learning

In [8]:
```
ftest = get_image_files(TEST_PATH)
print('Testing', len(ftest), 'items')
```

```
Testing 3469 items
```





코드 분석 및 설명

In [9]:

```
# make dataloader for test data
tst_dl = dls.test_dl(ftest, with_labels=False, shuffle=False)
tst_dl.show_batch(max_n=12)
```

- DataLoader.test_dl
  - Create a test dataloader from *test_items* using validation transforms of *dls*
  - with_labels: Whether the test items contain labels
  - shuffle: Whether to shuffle data passed to

코드 분석 및 설명

In [10]:

```
%%time

preds = learn.tta(dl=tst_dl, n=32, use_max=False)
```

TTA : 예측하는 동안 데이터 증식을 사용해서 모델의 정확도를 개선하는 과정

- Learner.tta(ds_idx=1, dl=None, n=4, item_tfms=None, batch_tfms=None, beta=0.25, use_max=False)
  - Test Time Augmentation을 사용하여 ds_idx 데이터 세트 또는 dl에 대한 예측 반환
  - 훈련 세트의 변환으로 n번 예측하고 평균을 냄
  - 최종 예측은 (1-베타) * 평균 + 베타 * 데이터세트의 변환으로 얻은 예측

```
CPU times: user 2min 26s, sys: 9.67 s, total: 2min 36s
Wall time: 25min 36s
```

In [11]:

```
predss = learn.dls.vocab[np.argmax(preds[0], axis=1)]
subm_df = pd.DataFrame()
subm_df['image_id'] = [item.name for item in tst_dl.items]
subm_df['label'] = predss
subm_df.to_csv('fasiai2_submission.csv', header=True, index=False)
```

감사합니다

구름다리