Image Classification

데이터사이언스개론(3207)

201711719 응용통계학과 심은선

목차

1. Modeling

Training 환경

Dataset

-label: food(0), interior(1), exterior(2)

Food
Label: 0

Interior
Label: 1

Exterior
Label: 1

-dimension: (300, 300, 3)
-# of data: 45000장
-train : validation = 7 : 3 (with shuffle)

1. Modeling

Training 환경

Activation

-CNN논문에서 많이 사용하는 Relu 사용
-RNN에서 사용되는 tanh와 sigmoid는 gradient vanishing 위험

Pooling

-가장 많이 사용하는 MaxPooling 사용

Environment

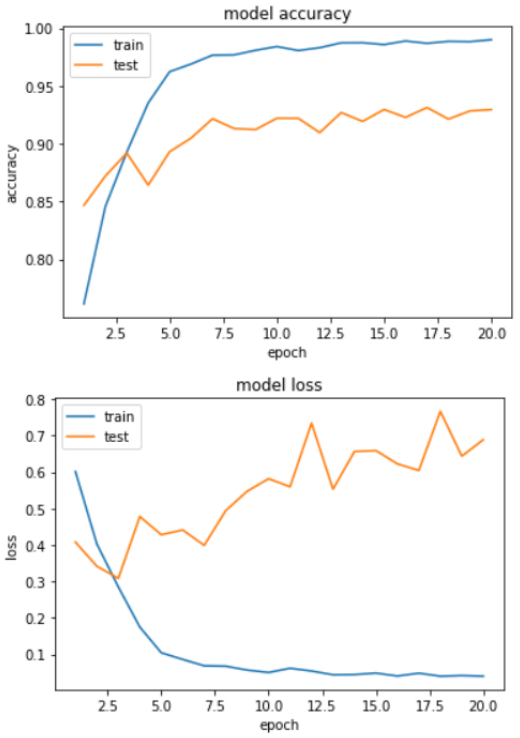
-Python 3.7
-Tensorflow 2.3.1

1. Modeling

A. Stride 실험

Model1

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model1	3	5,5,3	1	2	1	32	20	O	91~92



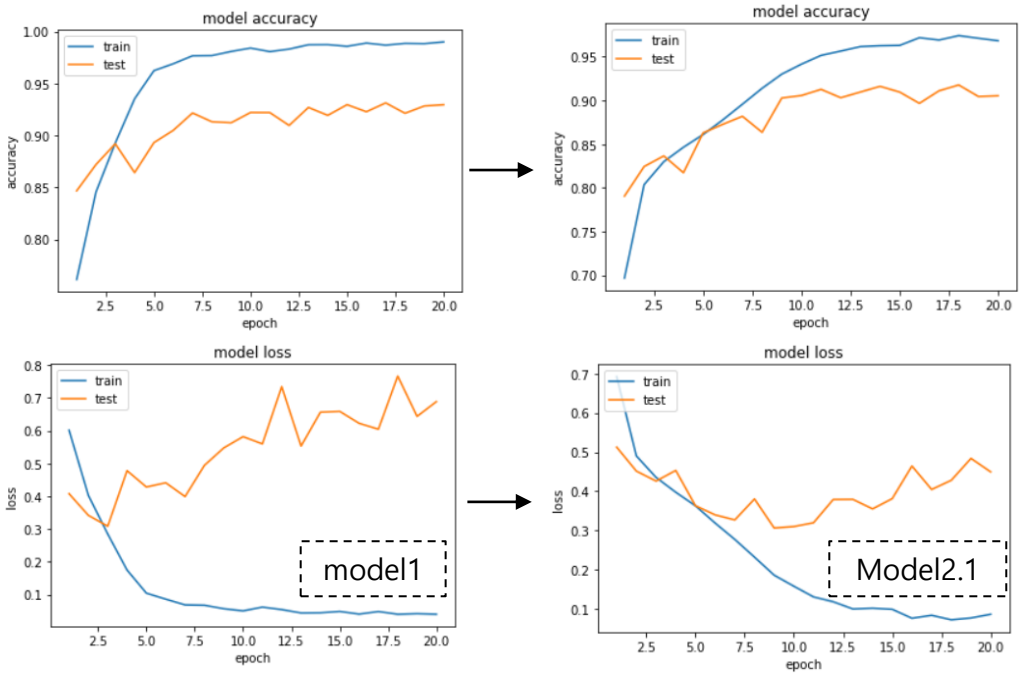
-baseline 모델
-filter 개수: 32, 128, 512
-성능: 0.92 (overfitting 이전)
-parameter 개수가 많아 overfitting발생, 학습 속도 느림

1. Modeling

A. Stride 실험

Model2.1

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model1	3	5,5,3	1	2	1	32	20	O	91~92
Model2.1	4	5,5,3,3	2,2,1,1	2	1	32	20	O	90



[stride 증가, filter 개수 감소]
-파라미터 개수 감소
-학습 속도 개선: 1시간 30분 -> 50분
-overfitting 완화
-filter 개수: 16, 32, 64, 128

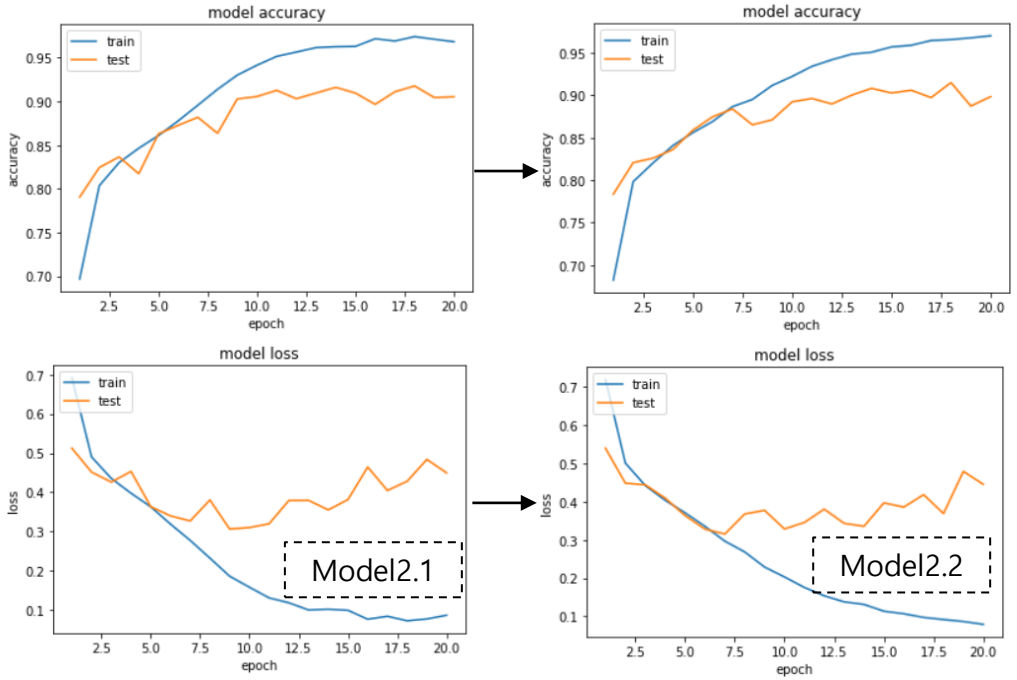
-stride가 커서 정보 손실 발생, accuracy 1~2% 감소

1. Modeling

B. Dense Layer 실험

Model2.2

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.1	4	5,5,3,3	2,2,1,1	2	1	32	20	O	90
Model2.2	4	5,5,3,3	2,2,1,1	2	2	32	20	O	90



[Dense Layer 추가]
-classification하는 layer 이전에 FC layer 추가

-효과: accuracy, loss 차이 X
-파라미터 개수만 증가

1. Modeling

A. Stride 실험

- stride를 늘리면 학습 시간이 감소하나, accuracy도 감소

→stride=1 사용

B. Dense Layer 실험

- dense layer 추가 효과 x

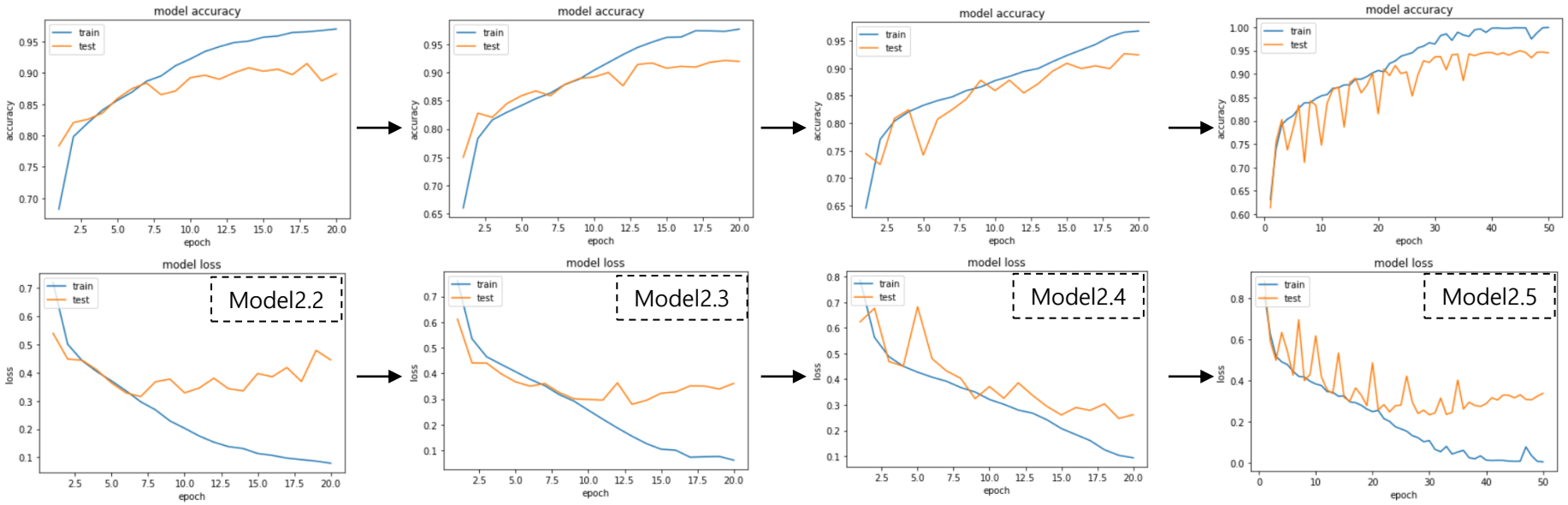
→dense layer=1

1. Modeling

C. Batch Size 실험

Model2.3~2.5

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.3	4	5,5,3,3	2,2,1,1	2	2	64	20	O	91
Model2.4	4	5,5,3,3	2,2,1,1	2	2	128	20	O	92
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94



1. Modeling

C. Batch Size 실험

Model2.3~2.5

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.3	4	5,5,3,3	2,2,1,1	2	2	64	20	O	91
Model2.4	4	5,5,3,3	2,2,1,1	2	2	128	20	O	92
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94

[속도]
Model2.3: 한 epoch당 2초 빠름
Model2.4: 한 epoch당 10초 빠름
Model2.5: 한 epoch당 50초 빠름

[성능]
Model 2.3: accuracy 1% 상승
Model 2.4: accuracy 2% 상승
Model 2.5: accuracy 4% 상승

1. Modeling

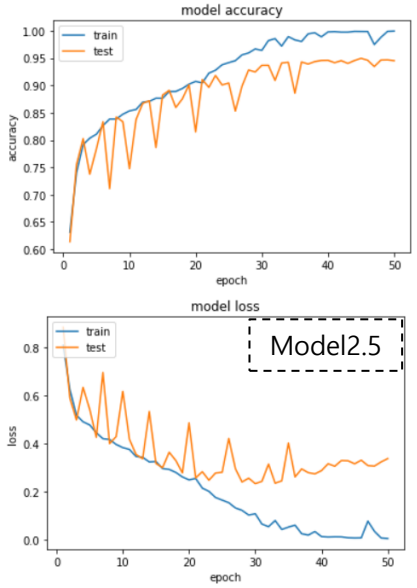
C. Batch Size 실험

Model2.3~2.5

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.3	4	5,5,3,3	2,2,1,1	2	2	64	20	O	91
Model2.4	4	5,5,3,3	2,2,1,1	2	2	128	20	O	92
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94

[Overfitting]
Batch size를 늘릴수록 overfitting 완화
하지만 loss가 수렴하기 위해 epoch수 늘려야 함

Batch size를 늘릴수록 epoch수도 늘려야함
Trade-off를 고려해 batch size=256, epoch=50

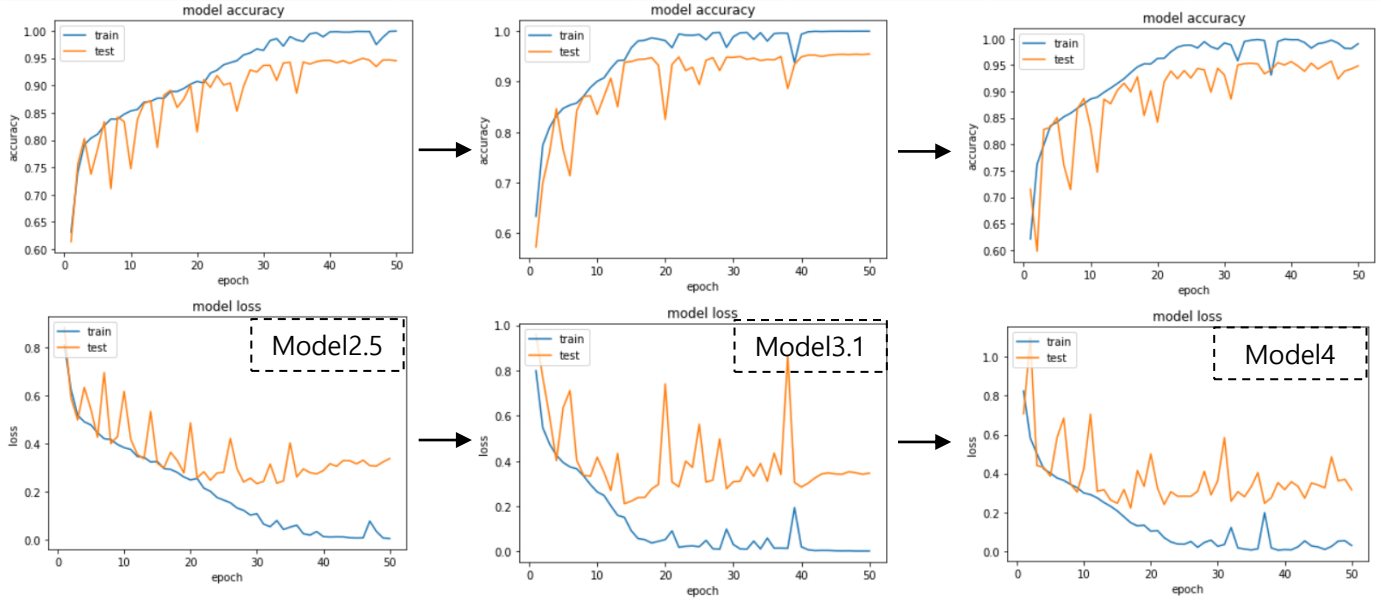


1. Modeling

D. Convolution Layer 실험

Model3.1, Model4

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95
Model4	7	5,5,4,3,3,2,2	1	2	1	256	50	O	94~95



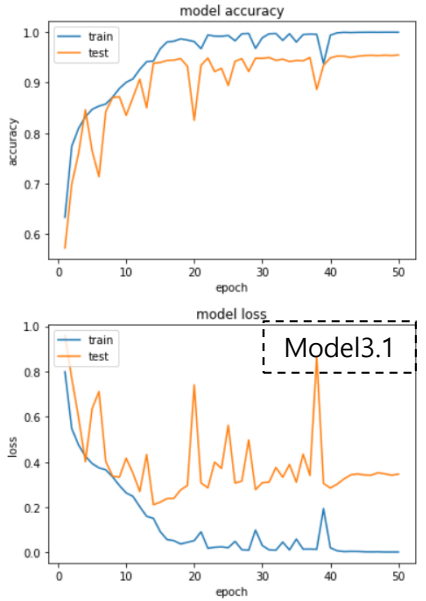
1. Modeling

D. Convolution Layer 실험

Model3.1

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95
Model4	7	5,5,4,3,3,2,2	1	2	1	256	50	O	94~95

-overfitting 완화
-성능: accuracy 1% 상승 (model 3.1)



1. Modeling

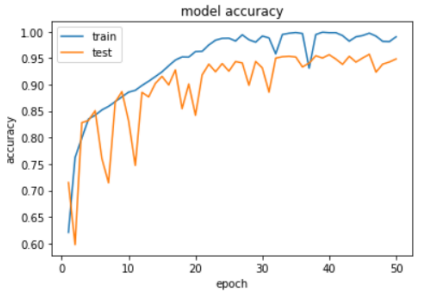
D. Convolution Layer 실험

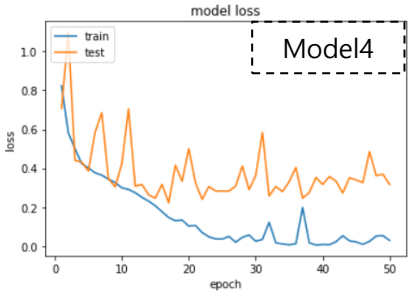
Model4

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model2.5	4	5,5,3,3	2,2,1,1	2	2	256	50	O	94
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95
Model4	7	5,5,4,3,3,2,2	1	2	1	256	50	O	94~95

-overfitting 완화
-성능: accuracy 0~1% 상승

-쌓을 수 있는 가장 깊은 layer 개수 (with pooling)
-차원 유지 padding 사용
-model 3.1에 비해 epoch당 학습 속도 50초 증가
-model 3.1에 비해 loss, accuracy가 불안정



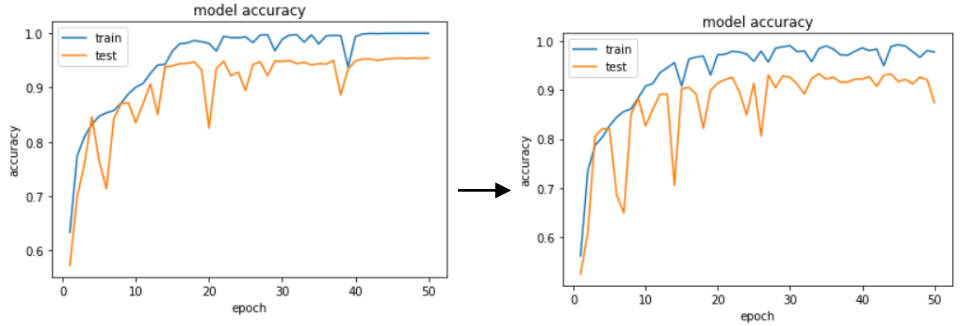


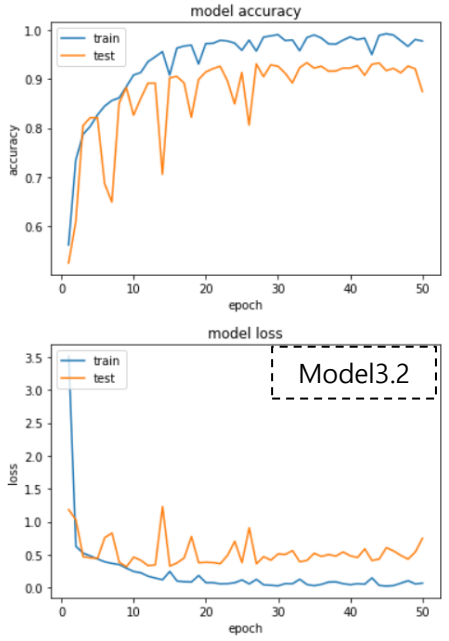
1. Modeling

E. Normalize 실험

Model3.2

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95
Model3.2	6	5,5,5,3,3,3	1	2	1	256	50	X	91~92

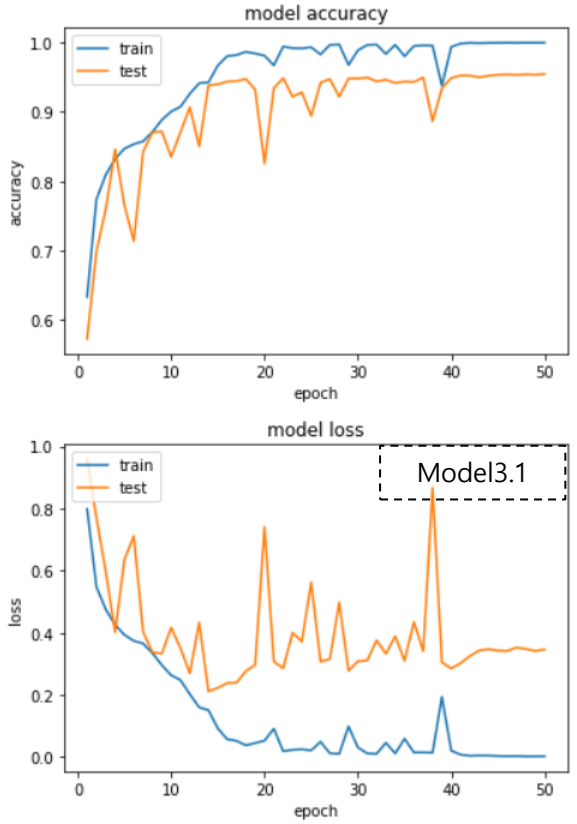




-성능: accuracy 3~4% 하락

1. Modeling

보완 사항



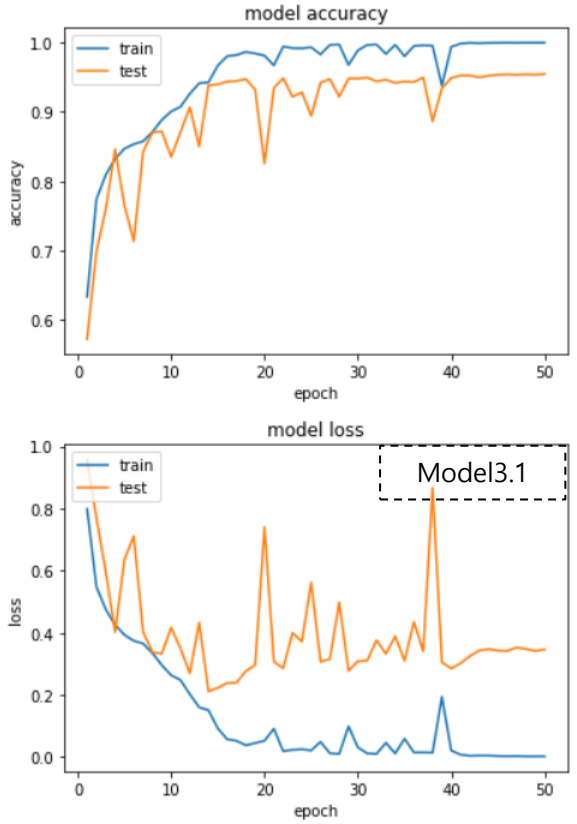
-Validation accuracy, loss가 train에 비해 크게 튐
-모델 일반화를 위한 장치 필요
→batch normalization, dropout 적용 필요

[Batch normalization]
-convolution(또는 FC) 연산 후 정규화해 분포를 강제
-정규화한 값을 activation 함수에 적용
-weight 초기값에 영향을 받지 않고, 학습이 빨리 수렴함

[Dropout]
-몇 개의 노드를 끄고(0으로) 다음 layer에 입력
-노이즈에 강하며 모델의 일반화

1. Modeling

보완 사항



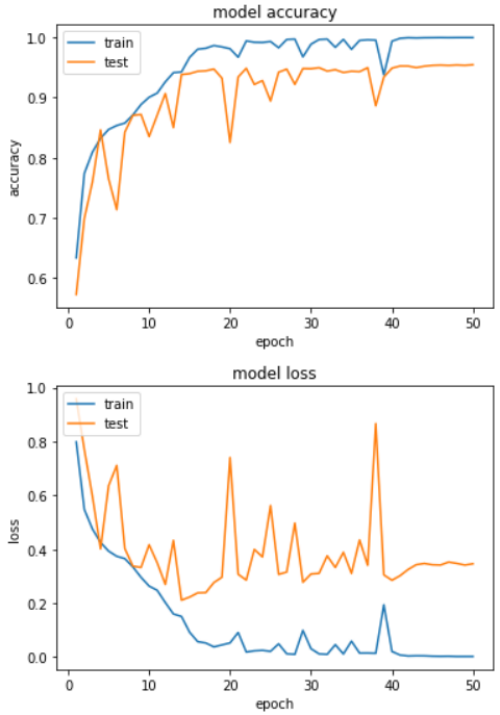
-Validation accuracy, loss가 train에 비해 크게 튐
-모델 일반화를 위한 장치 필요
→batch normalization, dropout 적용 필요

코드 작성했으나
학습 시간이 오래걸려 실험하지 못함

-몇 개의 노드를 끄고(0으로) 다음 layer에 입력
-노이즈에 강하며 모델의 일반화

2. 최종 Model

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95



모델 선택 이유

-overfitting X
-accuracy가 높음
-loss와 accuracy가 수렴
-상대적으로 빠른 시간에 학습
(model4에 비해 epoch당 50초 빠름)

3. 최종 Model 분석

Model	#Conv2d	Kernel	Stride	Pooling	#Dense	Batch	Epoch	Normal	val_acc
Model3.1	6	5,5,5,3,3,3	1	2	1	256	50	O	95

	precision	recall	f1-score	support
food	0.97	0.98	0.98	6000
interior	0.95	0.93	0.94	4500
exterior	0.94	0.93	0.94	3000
accuracy			0.95	13500
macro avg	0.95	0.95	0.95	13500
weighted avg	0.95	0.95	0.95	13500

4. Prediction 시연

감사합니다