



2023 데이터 크리에이터 캠프

DATA CREATOR CAMP

대학부 **수정** Team

팀장

박지호

백예은

오수아

유소현

최정빈

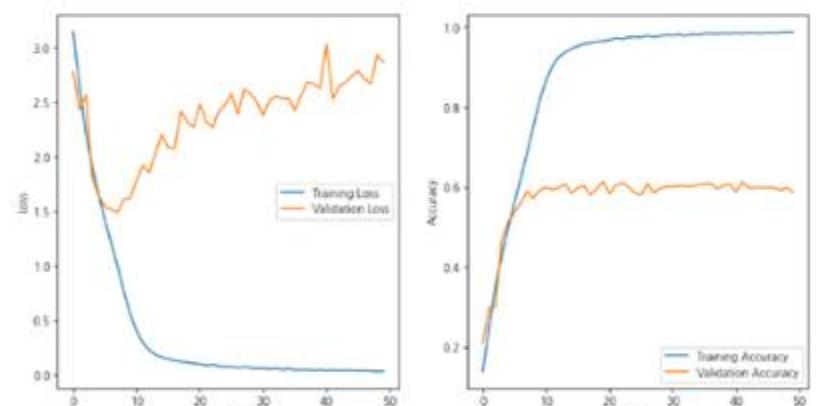
예선

- 예선 결과 요약
- 예선 발전 사항

1.1 예선 결과 요약

Mission 1

ResNet-18 모델 학습



Cross Entropy를
Loss function으로 지정
Adam optimizer를 사용한
ResNet-18 모델 학습

Val Acc : 0.6471

Test Acc : 0.6246

Mission 2

Mission 1의 결과 분석을 기반으로 한국 음식 이미지 데이터셋 분류

Architecture ResNet - 18

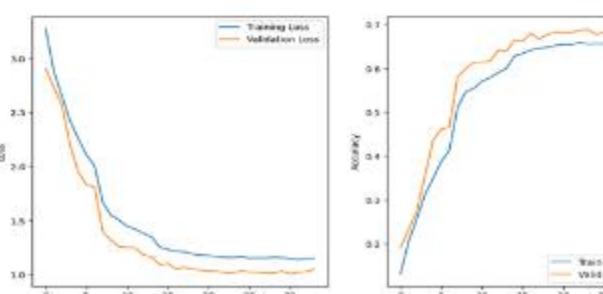
Agmentation RandomResizedcrop
RandomHorizontalFlip

Regularization minmax(-1, 1)

Optimizer SGD (momentum = 0.99)

Learning Rate StepLR(0.001)

Regularization L2 regularization



Val Acc

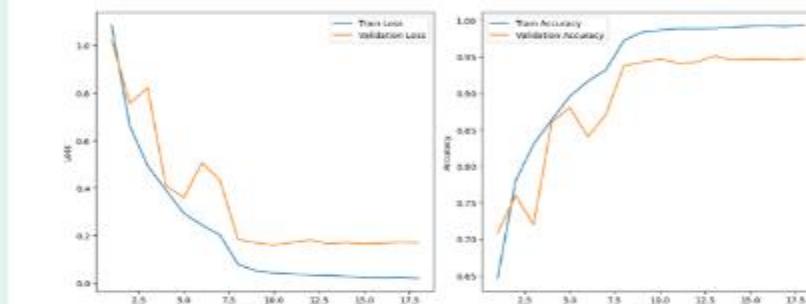
0.6910

Test Acc

0.6775

Mission 3

Mission 2모델을 기반으로 건강 관리를 위한 음식 이미지 분류 모델 학습



Val Acc
0.9510

Test Acc
0.9354

Architecture ResNet - 18

fine - tuning all - layers

weight decay 0.01

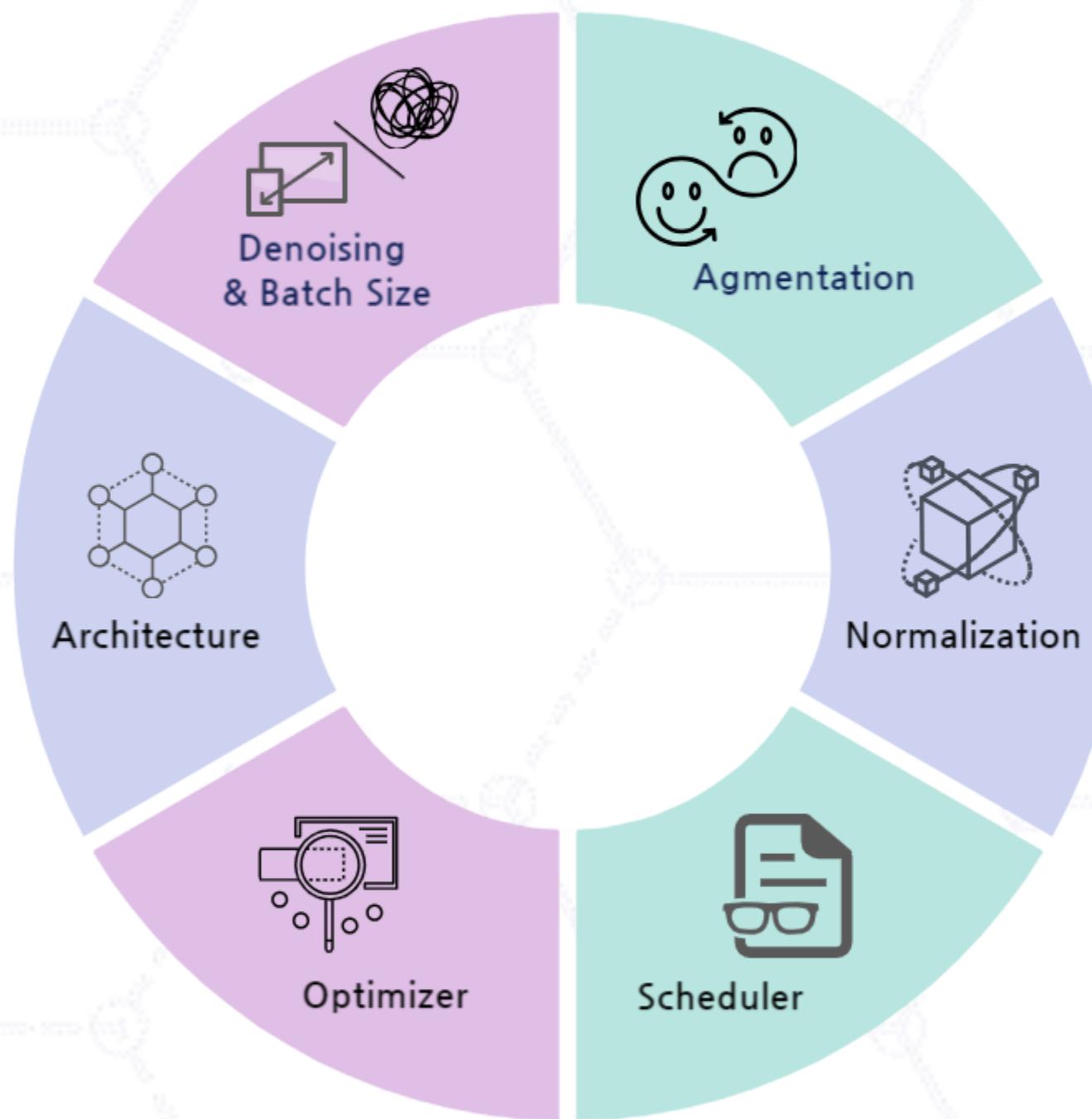
Optimizer Nadam

Learning Rate 0.001

Sampling Under Sampling

1.2 예선 결과 발전 사항 & 성능 비교 기본 가정

예선 결과 발전 사항



> 예선에서 비교하지 않은 추가 사항 추가 비교

성능 비교를 위한 기본 가정

1. Early Stopping 사용

- 모델의 훈련과정 중 일부 조건이 충족되면 훈련을 더 이상 진행하지 않고 종료하는 기술
- 5 epoch 이상 정확도 개선이 안될 시 학습 종료

2. 하드웨어 한계로 계산 불가능한 모델은 제외

- ex) InceptionV3 등

3. 분류 모델이므로 criterion = CrossEntropyLoss 사용

Mission 2

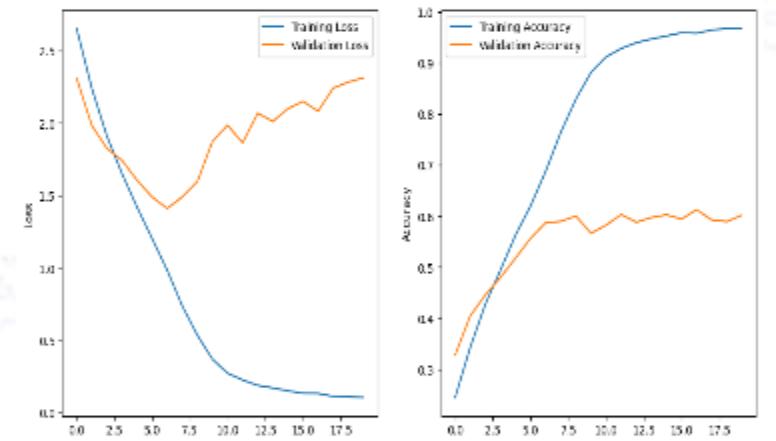
- Denoisiong & Batch Size
- Agmentation
- Architecture
- Optimizer
- Scheduler & L2 정규화 & Drop out & Batch Normalization

2.1 Denoising & Batch Size

Denoising : 무작위한 노이즈 제거

Blur 처리

이미지의 각 픽셀을 주변 픽셀
가중치 평균으로 대체하여
이미지가 흐릿해지는 효과

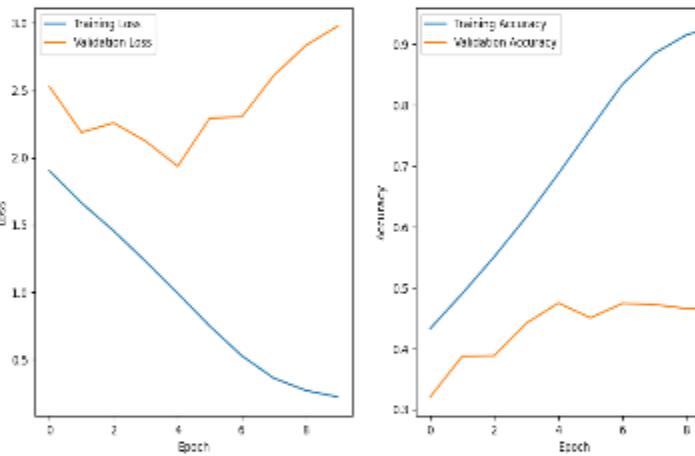


기본 모델의
Val Acc : 0.6185
Val Loss : 1.5217

Denoising을 사용하지 않은 기본 모델에 비해서 성능 감소
✓ Denoising 기법은 사용하지 않는 것이 낫다고 판단!

모폴로지 연산

이미지의 형태나 구조를 조작
특정 형태나 패턴을 추출 또는
제거하므로써 노이즈 제거



Batch Size : 한 번의 모델 학습에 사용되는 데이터 샘플 수 조정

16

Val Acc
0.6660

32

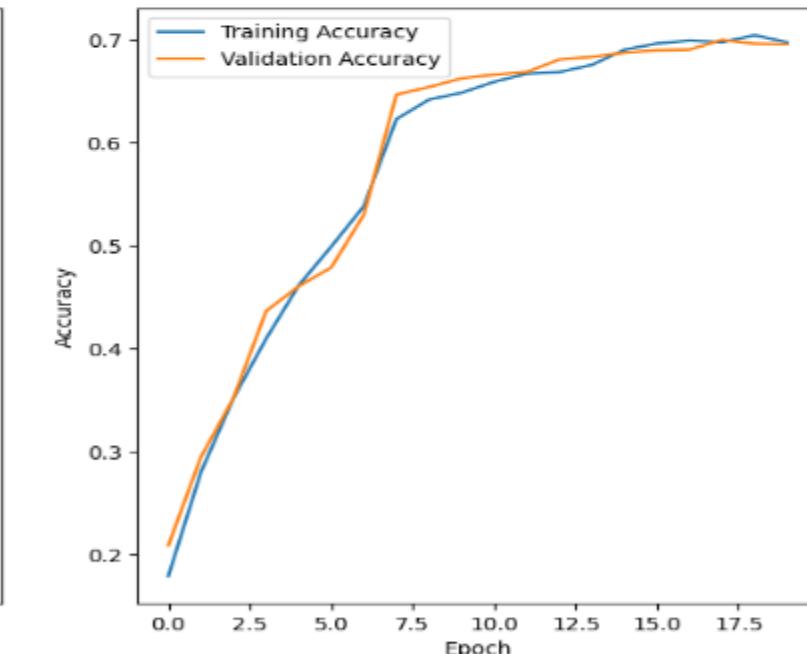
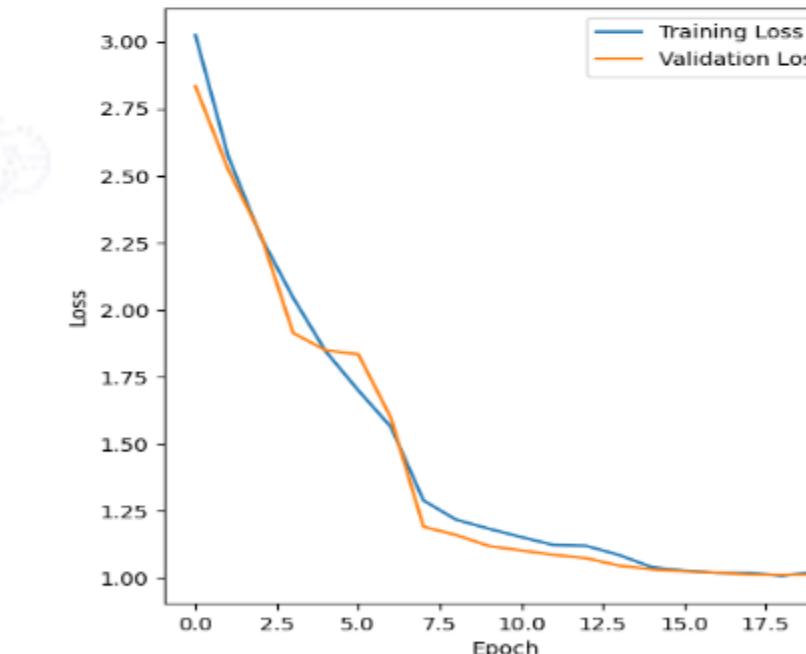
Val Acc
0.6922

64 ✓

Val Acc
0.6998

128

20에폭을 기준으로 일반화 성능이 가장 좋을 때의
batch size를 최적으로 선정



- batch size가 32인 모델과 64인 모델의 Val Acc 차이 미묘
- 20에폭을 기준 Val Acc와 Test Acc 모두 64 모델이 우세
- 두 모델의 성능 차이가 미미하기에 계산의 효율성 측면에서 64모델 선택

2.2 Agmentation : 학습 데이터셋을 다양한 방법으로 확장하여 모델의 성능을 향상시키는 기술

2.2.1 단일 증강

Random Crop

무작위로 영역을 잘라내는 기법

- val acc가 0.6121 → 0.6758로 6%p 가량 상승
- 학습률 그래프 또한 눈에띠게 안정화됨

2.2.2 다중 증강 - 2개

Random Crop

✓ Rotation Range

Height shift Range

Shear Range

Horizontal Flip

Vertical Flip

Zoom Range

무작위 회전

이미지 높이 무작위 이동

무작위 전단시켜 왜곡 유발

수평 뒤집기

수직 뒤집기

무작위 확대 및 축소

Val ACC

0.6845

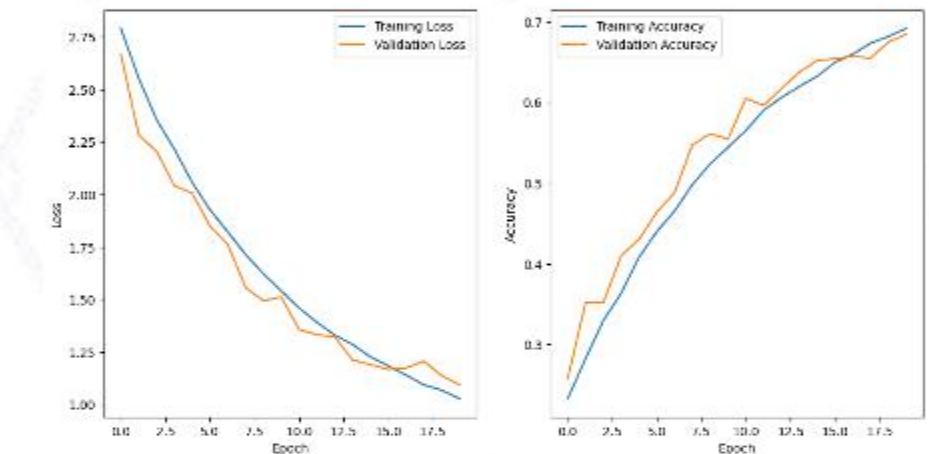
0.6555

0.6590

0.6733

0.6374

0.6711



Random Crop에 Rotation Range를 함께 적용시켰을 때 성능이 향상됨

Val Loss : 1.0952

Test Acc : 0.6856

2.2.3 다중 증강 - 3개

Random Crop

+

Rotation Range

+

✓ Horizontal Flip

- Val Acc = 0.6725로 flip 기법만 추가로 사용한 경우보다는 정확도 상승
- 최종적으로 Rotation Range만 적용한 다중 증강의 경우와 3개 기법을 모두 사용한 경우를 함께 비교

2.3 Architecture

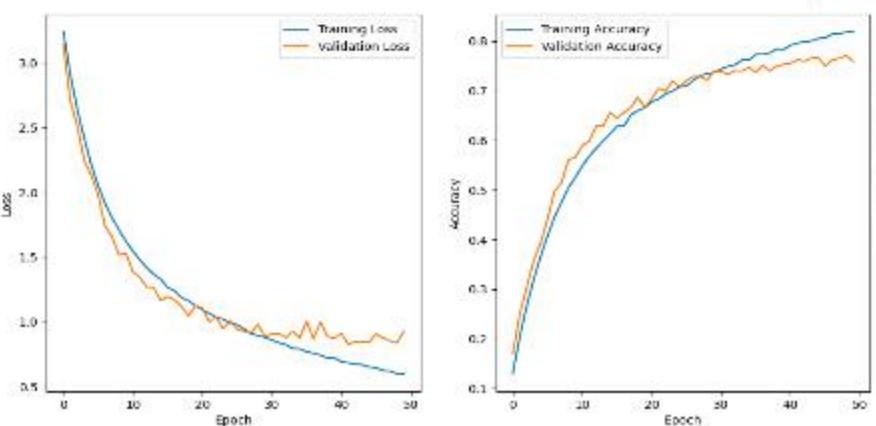
Val Acc : 0.6828
Val Loss : 1.0667
Test Acc : 0.6815

더 깊은 모델은 하드웨어
한계로 계산 불가

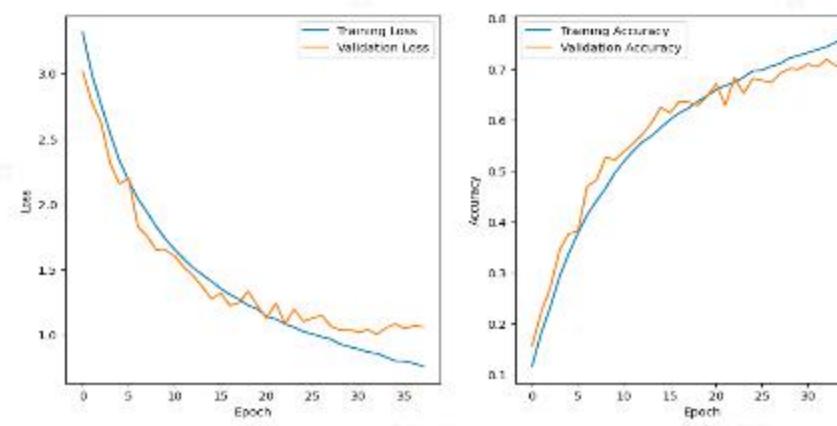
ResNet - 18
Val Acc : 0.6845

ResNet - 34
Val Acc : 0.6616

ResNet - 50
Val Acc : 0.6373



DenseNet121 모델



ResNet - 18 모델

DenseNet121

- 밀집 연결 구조를 가진 딥러닝 아키텍처
- 그래디언트 전파가 효율적이고, 특징 재사용이 강조

EfficientNet

- 성능과 효율성을 균형있게 조절하기 위해 compound scaling을 사용
- AutoML을 통해 찾은 구조

Deep Learning Architecture

ResNet

- 잔여 블록의 구조
- 잔여 연결(residual connection)을 사용하여 레이어를 건너뛰어 그래디언트 소실 문제를 완화

MobileNet

- Depthwise Separable Convolution을 사용하여 모델의 파라미터 수를 줄임
- 경량 모델

모델 깊이에 따라 두 가지 고려

EfficientNet B0
Val Acc : 0.5733

EfficientNet B1
Val Acc : 0.5773

가장 최근에 발표된 V2 모델
V1의 후속작으로 더 나은
성능과 효율성을 보임

Val Acc : 0.6162
Val Loss : 1.2775
Test Acc : 0.6055

- ResNet -18 모델과 DenseNet121모델 모두 좋은 성능을 보임
- 정밀한 비교를 위해 두 모델만 50에폭까지 실행 후 비교

DenseNet121
Validation Accuracy : 0.7662
Validation Loss : 0.8416
Test Accuracy : 0.7682



ResNet - 18
Validation Accuracy : 0.7190
Validation Loss : 1.0033
Test Accuracy : 0.7234

최종적으로 DenseNet121 모델을 사용

2.4 Optimizer : 모델 가중치 최적화

SGD Optimizer

RMSProp

Adam

Nadam

가장 기본적인 최적화 알고리즘

Val Acc : 0.2173

예선 최적화 파라미터 (50에폭 기준)
(momentum = 0.99, nesterov = T)
Val Acc : 0.6686

과거 gradient들의 영향을 감소시켜
보폭을 줄이며 최적화

Val Acc : 0.6197
Val Loss : 1.2780
Test Acc : 0.6258

RMSProp에 Momentum을
함께 적용시킨 알고리즘

Val Acc : 0.6543
Val Loss : 1.1824
Test Acc : 0.6458

Adam에 Momentum 대신
NAG를 적용시킨 알고리즘

Val Acc : 0.6660
Val Loss : 1.1214
Test Acc : 0.6665

" Nadam은 기존 Adam 모델에 빠른 수렴 속도와 안정성을 더욱 강조한 알고리즘이 만큼 정확도와 성능이 가장 뛰어남 "
-> Nadam의 파라미터를 튜닝하여 SGD와 비교

Momentum 조절

Beta 1

그래디언트의
방향성 결정

(Beta1, Beta2)

(0.95, 0.999)
(0.999, 0.95) →
(0.999, 0.999)

Beta 2

그래디언트의
크기 결정

Val Acc

0.6766
0.3913
0.7031

ϵ 조절

수치 안정화를 위한
작은 상수

epsilon

1e-10

1e-6

✓ 1e-8 (defalt)

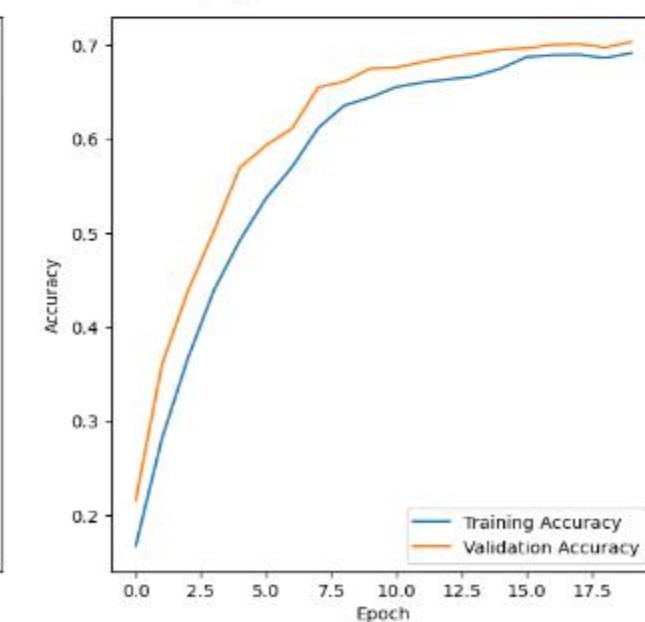
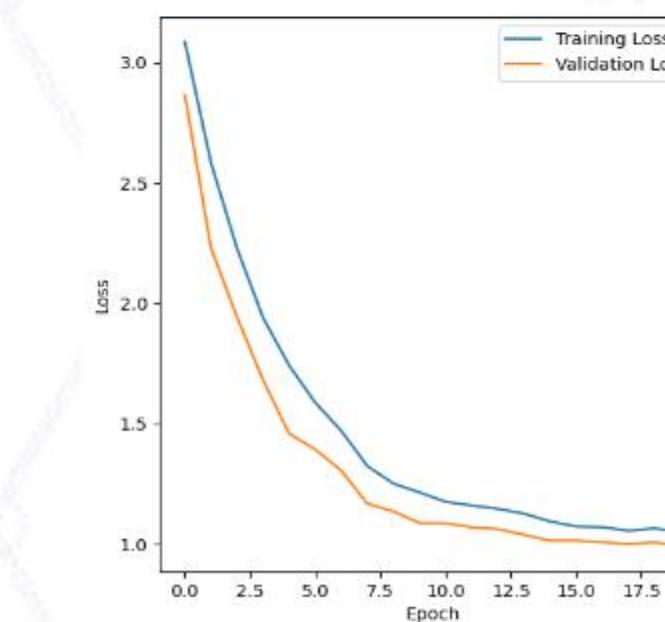
Val Acc

0.7016

→ 0.7052

0.7031

유의미한 차이가 보이지 않아
기본값으로 사용
> 수치적으로 불안정한 학습 유발 방지



> Nadam(beta = (0.999, 0.999), eps = 1e-8)
Val Loss : 0.9916
Test Acc : 0.7032

2.5 Scheduler & L2 정규화 & Drop out & Batch Normalization

Scheduler

StepLR

학습률을 epoch 단위로 조절

VAI ACC : 0.6952

ExponentialLR

학습률을 지수적으로 감소시킴

VAI ACC : 0.7345



CosineAnnealingLR

학습률이 코사인 함수의 주기에 따라 감소

VAI ACC : 0.7550

Non - Scheduler

초기 LR 할당값을 학습 종료까지 사용

VAI ACC : 0.7525

L2 정규화

- 손실 함수에 가중치의 제곱에 비례하는 항을 추가하여 가중치의 크기를 제한
- λ 의 값을 변경하여 정규화 정도 조절

Lambda

$\lambda = 0.01$
✓ $\lambda = 0.001$
 $\lambda = 0.0001$

Val Acc

0.6218
0.7635
0.7698

> 50에폭 기준 $\lambda = 0.001$ 성능이 더욱 뛰어남

Drop out & Batch Normalization

- 과적합 방지와 훈련 안정성 증대를 위한 기법

Drop Out

Drop Out + Batch Normalization

0.7050

0.7451

2.6 최종 비교

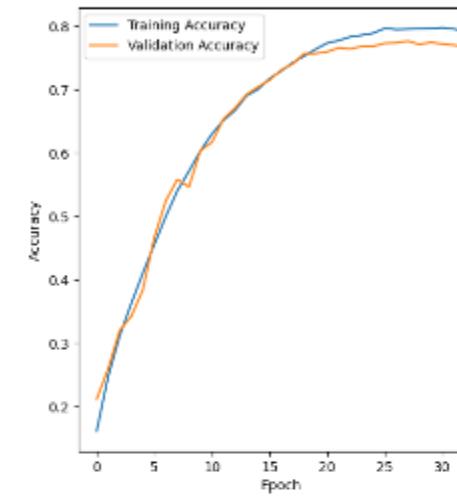
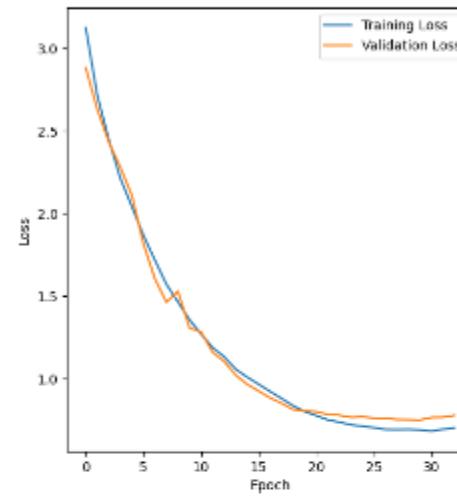


최종 모델

Val Acc
0.7771

Val Loss
0.7511

Test Acc
0.7730

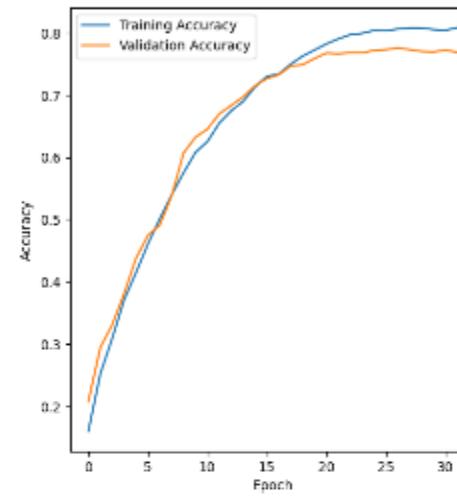
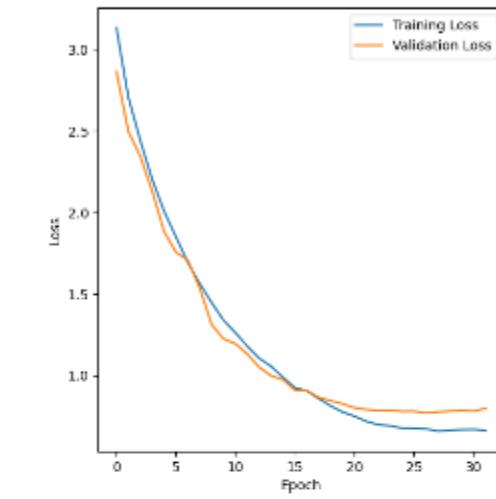


최종 모델 - flip 증강 제거

Val Acc
0.7757

Val Loss
0.7684

Test Acc
0.7730

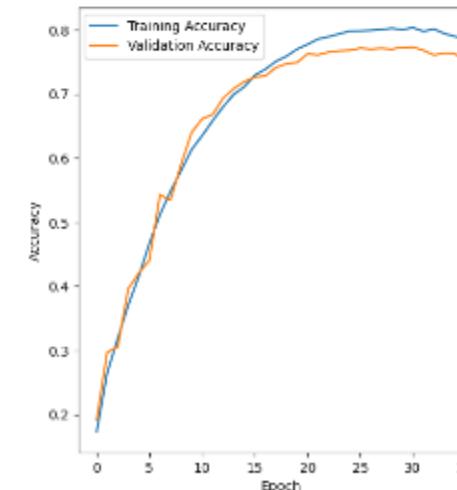
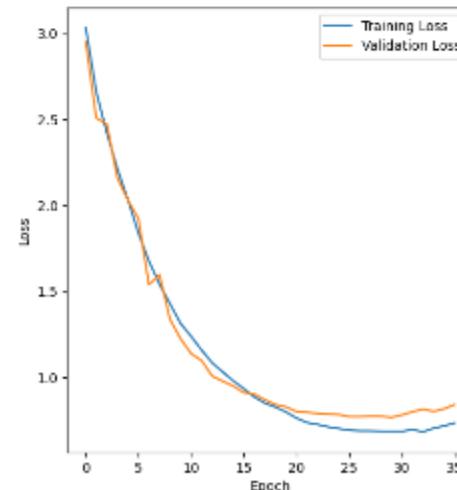


최종 모델 - Batch Normalization & Drop Out 제거

Val Acc
0.7733

Val Loss
0.7801

Test Acc
0.7654

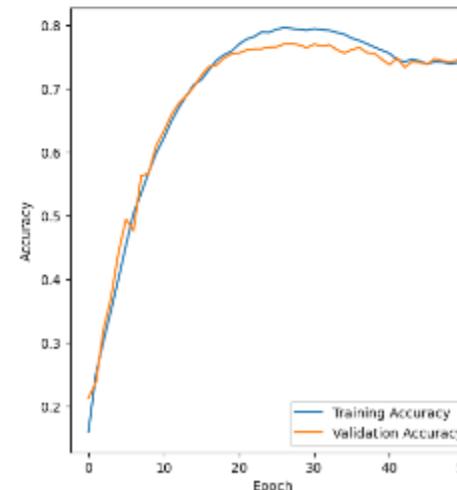
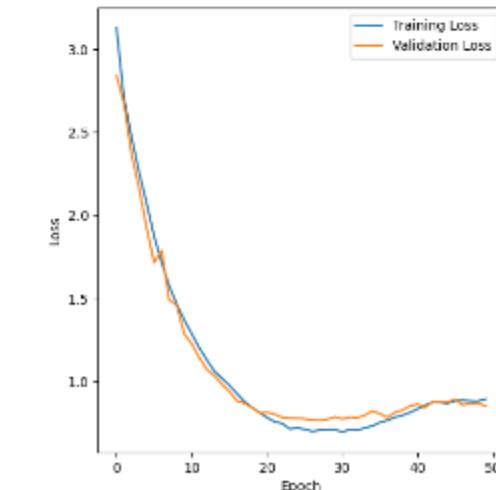


최종 모델 - Early Stopping 제거

Val Acc
0.7705

Val Loss
0.7638

Test Acc
0.7651



- early stopping 없이 50에폭 작동시 일정 구간 성능 감소를 보임

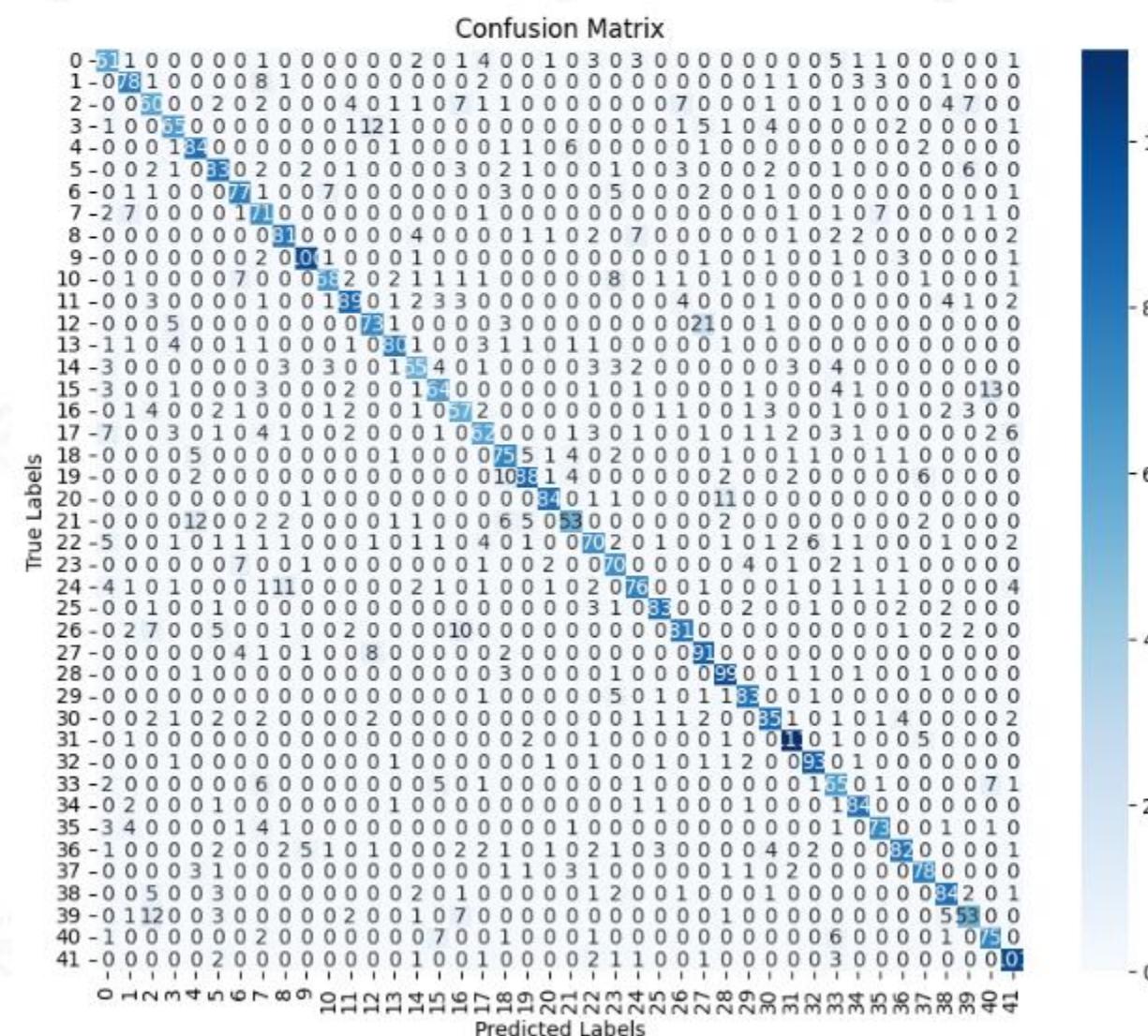
2.7 최종 모델

- 6 -

Densenet121 - minmax(-1~1) + (random crop + rotation range + horizontal flip) + Nadam(beta=(0.999, 0.999), eps=defalt + batch size=64 + CosineAnnealingLR(0.001~0.00001) + L2 regularization(weight_decay=0.001) + batch normalization + drop out 적용)

—

confusion matrix



› 대부분의 클래스를 고르게 잘 분류

분류 정확도 상위 top-3



시래기 91.67%



육개장
90.98%



잡곡밥
90.29%

분류 정확도 하위 top-3



떡갈비
60.19%



감자전
60.61%



북엇국
61.63%

2.7 최종 모델 - Class Activation Mapping(CAM)

"

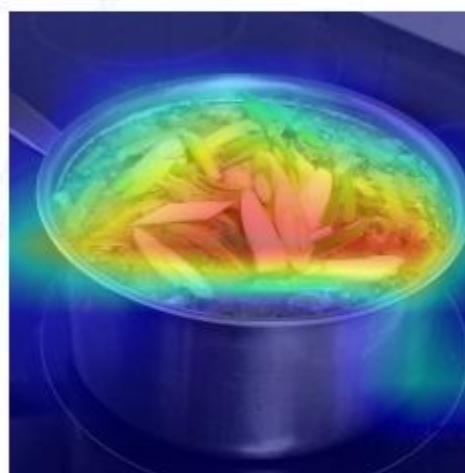
Densenet121 - minmax(-1~1) + (random crop + rotation range + horizontal filp) + Nadam(beta=(0.999, 0.999), eps=defalt + batch size=64 + CosineAnnealingLR(0.001~0.00001) + L2정규화(weight_decay=0.001) + batch normalization + drop out 적용

"

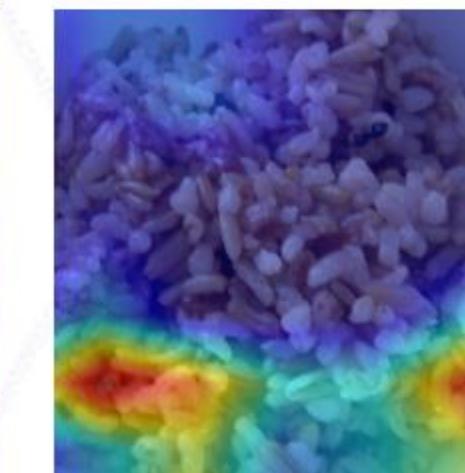
분류 정확도 상위 top-3



육개장



시래기국



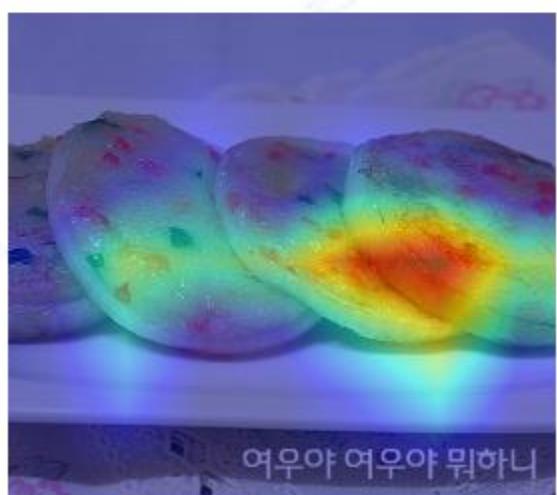
잡곡밥



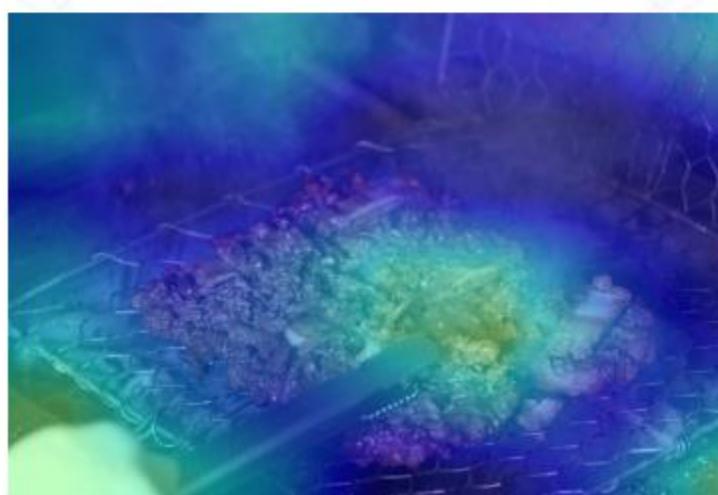
Class Activation Mapping (CAM)

컨볼루션 신경망 (CNN)의 특성 맵에서 어떤 부분이 분류 결정에 주요한 역할을 하는지 시각적으로 확인하는 기법

분류 정확도 하위 top-3



감자전



떡갈비



복어국

- 정확도가 높은 이미지들은 포인트 부분을 집중적으로 잘 찾아냄
- 분류 정확도가 낮은 이미지들은 포인트 부분을 잘 찾아내지 못하거나 잘못된 부분에 집중됨

Mission 3

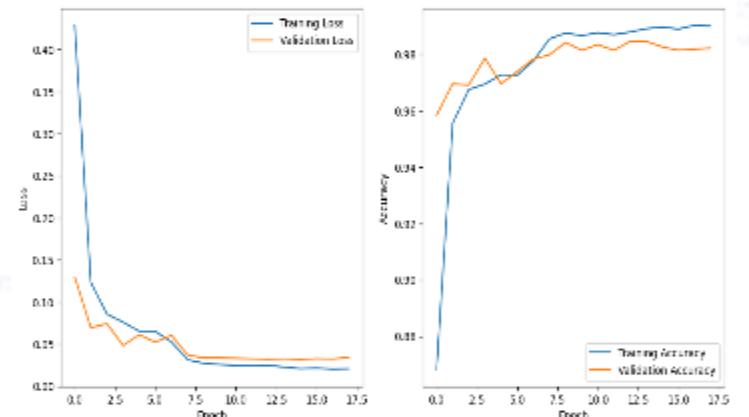
- Sampling & Batch Size
- Agmentation & Optimizer
- Optimizer Parameter
- Scheduler
- 최종 모델 비교

3.1 Sampling & Batch Size

Sampling : 클래스별 샘플 수 불균형 해소

Over Sampling

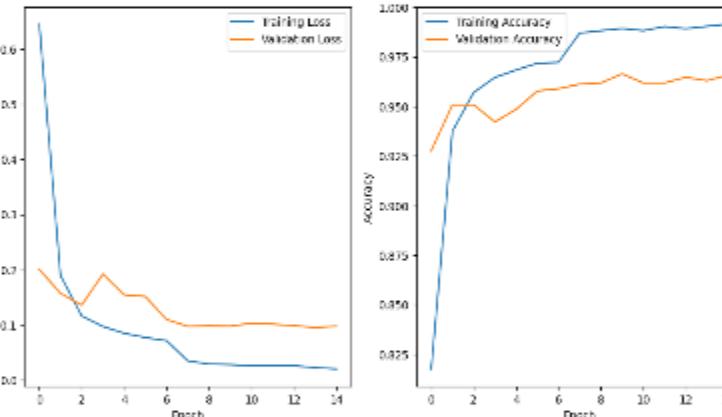
소수 클래스의 샘플 수를 늘려서 클래스 간의 균형을 맞추는 방법



Val Acc : 0.9745

Under Sampling

다수 클래스의 샘플 수를 줄여서 클래스 간의 균형을 맞추는 방법



Val Acc : 0.9665

No Sampling Val Acc : 0.9762
Sampling Val Loss : 0.0497

Sampling을 사용하지 않은 기본 모델에 비해서 성능 감소
Sampling 기법은 사용하지 않는 것이 낫다고 판단!

Batch Size : 한 번의 모델 학습에 사용되는 데이터 샘플 수 조정

✓ 16

Val Acc
0.9717

32

Val Acc
0.9702

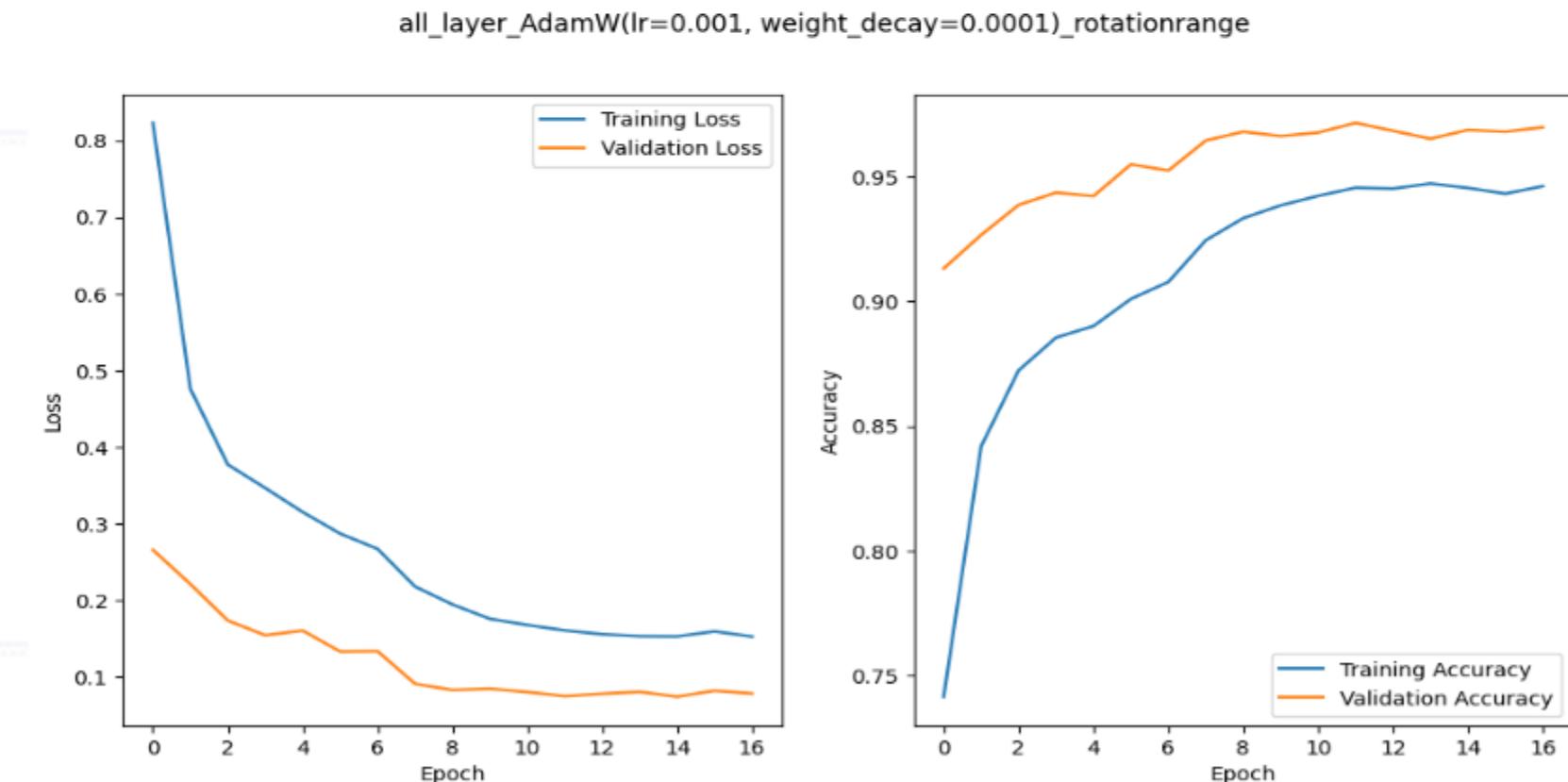
64

Val Acc
0.9649

128

증가할수록
성능 감소 목격

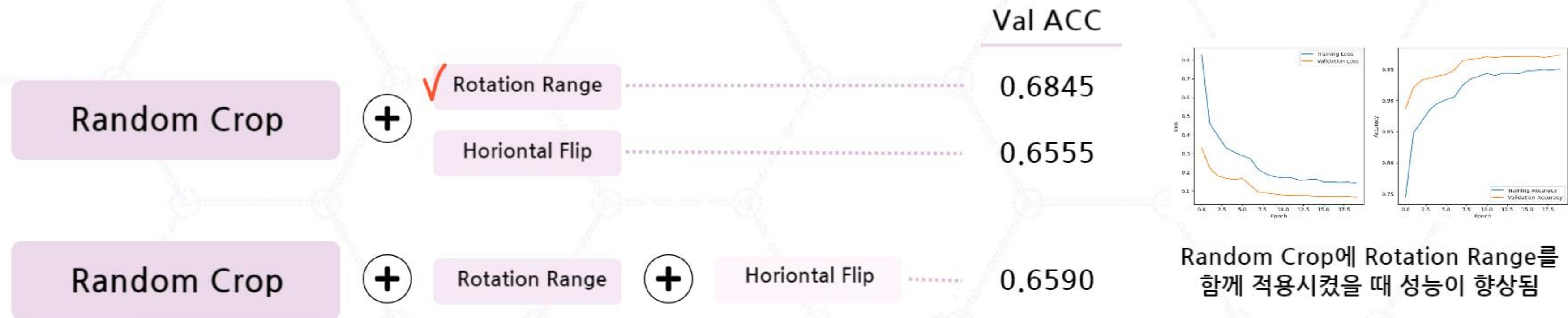
20에폭을 기준으로 일반화 성능이 가장 좋을 때의
batch size를 최적으로 선정



- batch size = 16 일때 가장 높은 정확도와 안정성을 보임

3.2 Agmentation & Optimizer

Agmentation : Mission2 증강 비교 결과를 기반으로 3가지 조합만 비교



Optimizer : 모델 가중치 최적화 알고리즘

SGD Optimizer

가장 기본적인 최적화 알고리즘

Val Acc : 0.6702
Val Loss : 1.9115
Test Acc : 0.6463

✓ Adam

RMSProp에 Momentum을 함께 적용시킨 알고리즘

Val Acc : 0.9759
Val Loss : 0.0514
Test Acc : 0.9768

AdamW

Adam의 변형으로 가중치 감쇠를 적용

Val Acc : 0.9717
Val Loss : 0.0751
Test Acc : 0.9705

Nadam

Adam에 Momentum 대신 NAG를 적용시킨 알고리즘

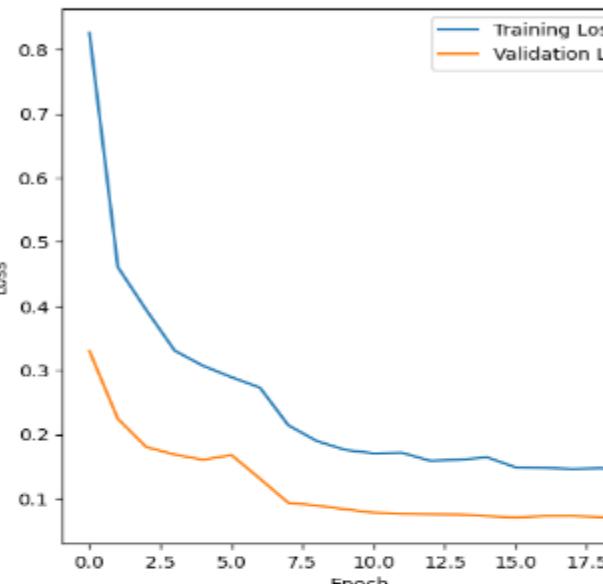
Val Acc : 0.9667
Val Loss : 0.0819
Test Acc : 0.9671

3.3 Optimizer parameter & Scheduler

Learning Rate

모델이 가중치 및 편향을 업데이트할 때
보폭(Step size)을 제어

LR	Val Acc
0.01(defalt)	0.9550
✓ 0.001	0.9731

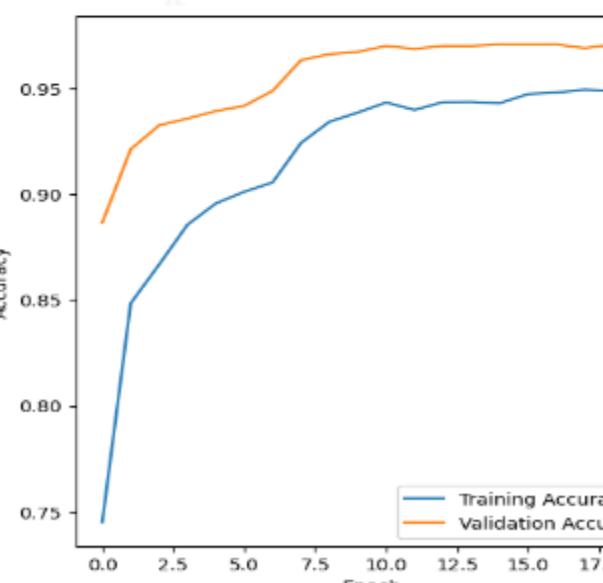


Weight Decay & Momentum

그래디언트에 가중치 감쇠 항을 추가하여
모델의 복잡도 조절

weight decay	Val Acc
✓ 0 (defalt)	0.9731
0.001	0.9472
0.0001	0.9660

> Loss 그래프



> Accuracy 그래프

Scheduler

✓ stepLR

val acc
0.9731

Exponential
LR

val acc
0.9717

CosineAnnealingLR

val acc
0.9720

> 스케줄러 3가지만 비교

> 가장 높은 정확도를 보인 stepLR 사용

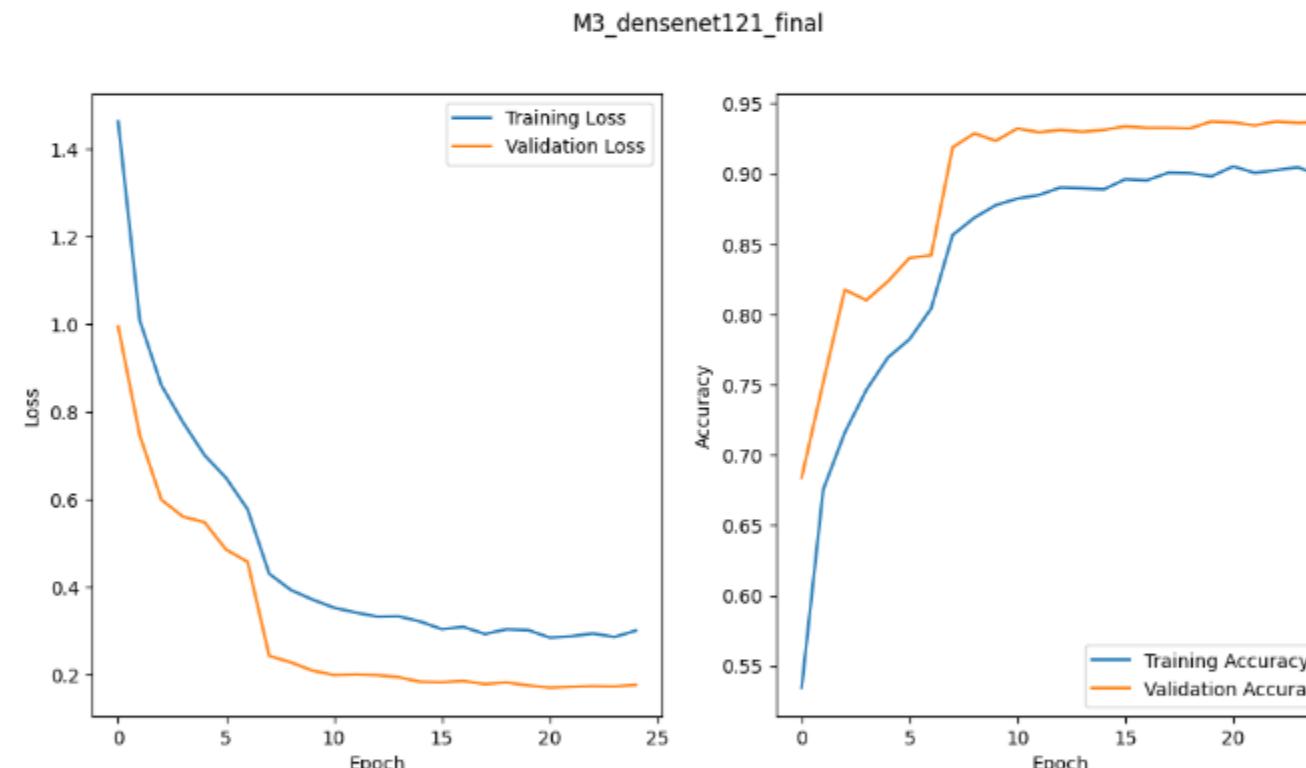
Momentum 조절

(Beta1, Beta2)	Val Acc
✓ (0.9, 0.999)	0.9731
(0.999, 0.999)	0.9663

3.4 최종 비교

Densenet121 - minmax(-1~1) + 데이터 증강(random crop + rotation range) + Adam($lr=0.001$, $weight_decay=0$, $betas=(0.9, 0.999)$) + batch size=16 + stepLR + fine-tuning(all layers) + 분할 비율(0.8)

Mission 2 최종 모델을 기반으로 학습시킨 모델

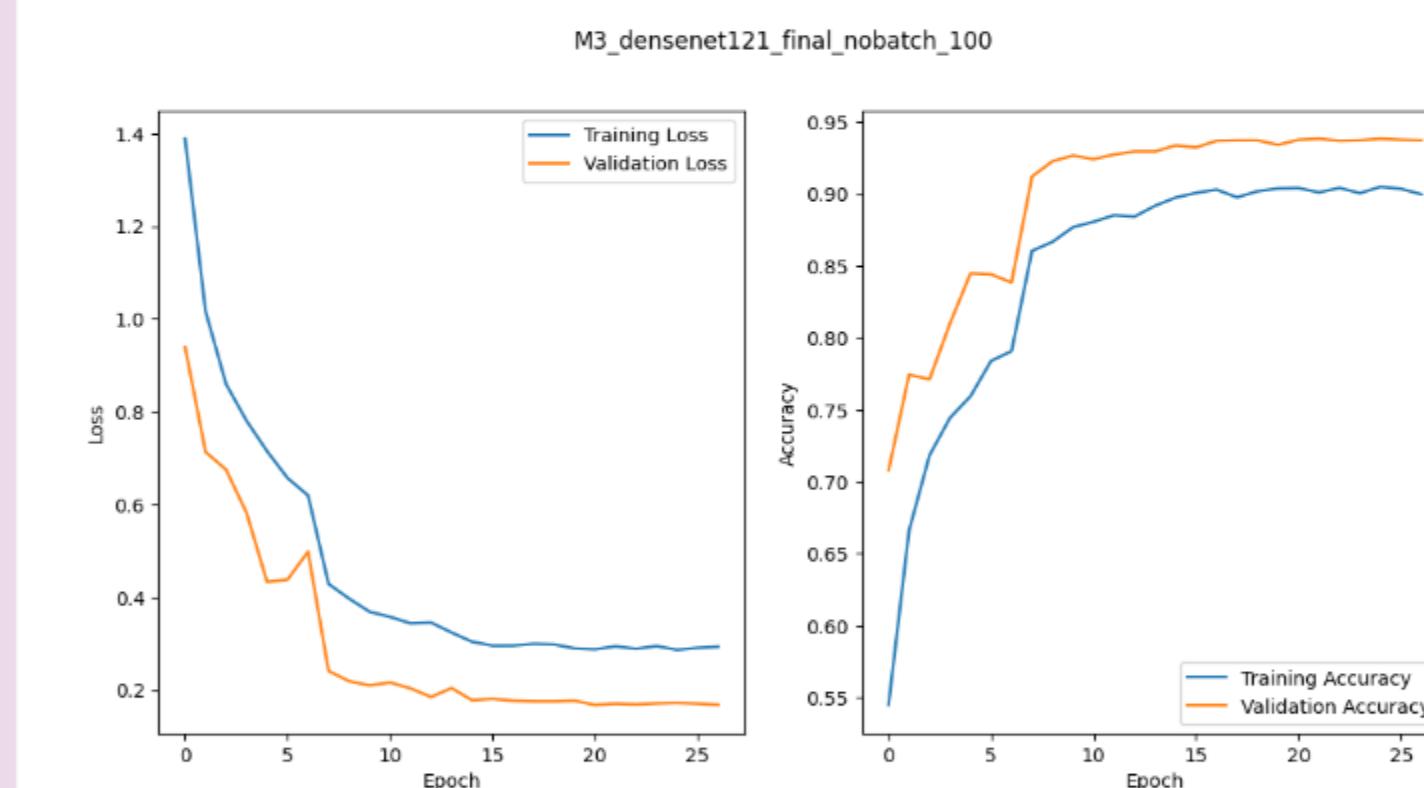


Val Acc
0.9373

Val Loss
0.1742

Test Acc
0.9325

Mission 2에 batch normalization + drop out을 제거한 모델을 기반으로 학습시킨 모델



Val Acc
0.9384

Val Loss
0.1700

Test Acc
0.9450

- Mission2의 성능 최고 모델 기반으로 학습시키기 보단, batch normalization + drop out을 제거한 모델을 기반으로 학습시켰을 때 성능이 더욱 높아짐

3.4 최종 모델

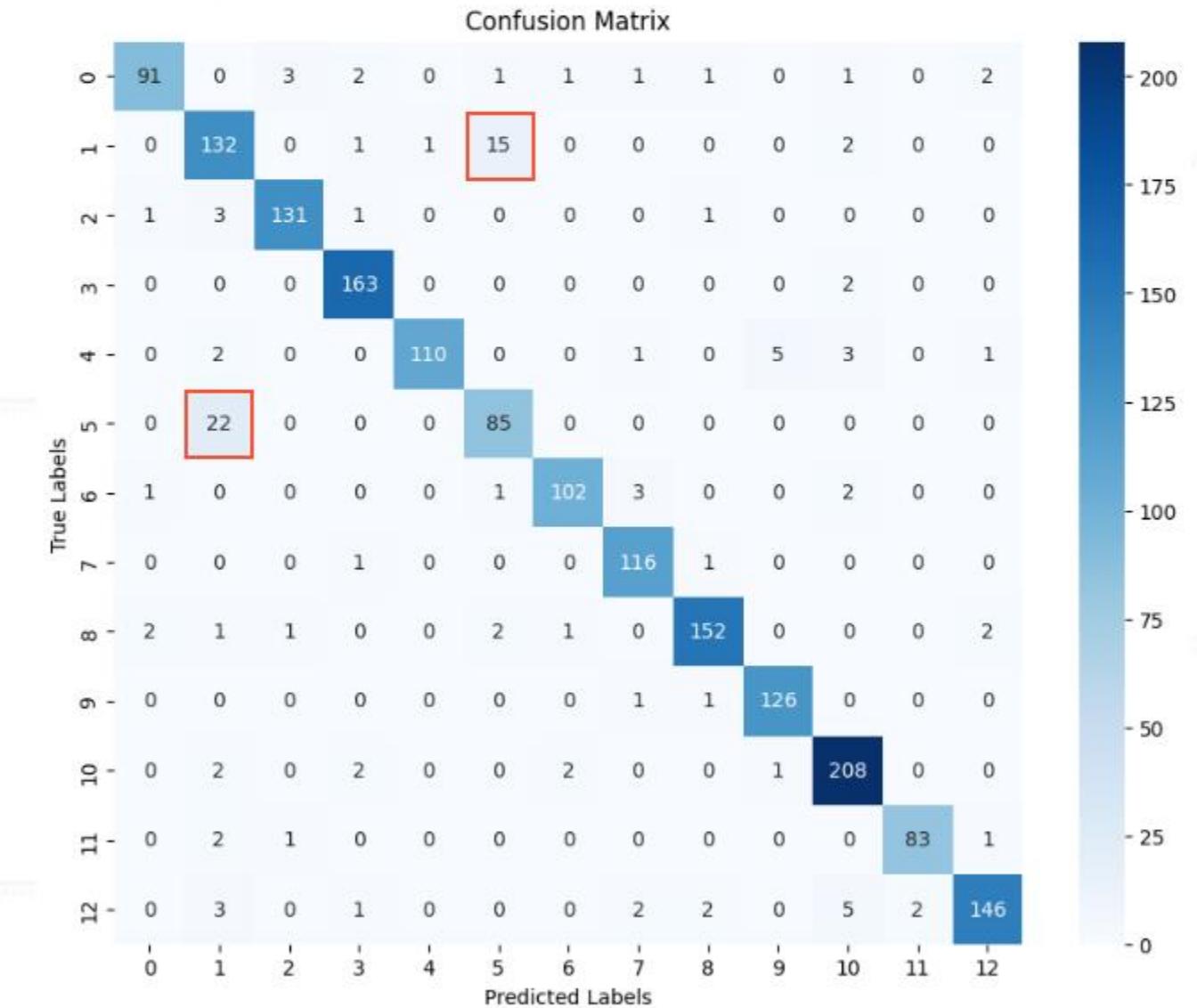
Densenet121 - minmax(-1~1) + 데이터 증강(random crop + rotation range) + Adam(lr=0.001, weight_decay=0, betas=(0.9, 0.999)) + batch size=16 + stepLR + fine-tuning(all layers) + 분할 비율(0.8)

Test 데이터 중 틀린 샘플



> 실제 사진은 잘못 예측된 클래스와 색감이 비슷한 경향이 있음

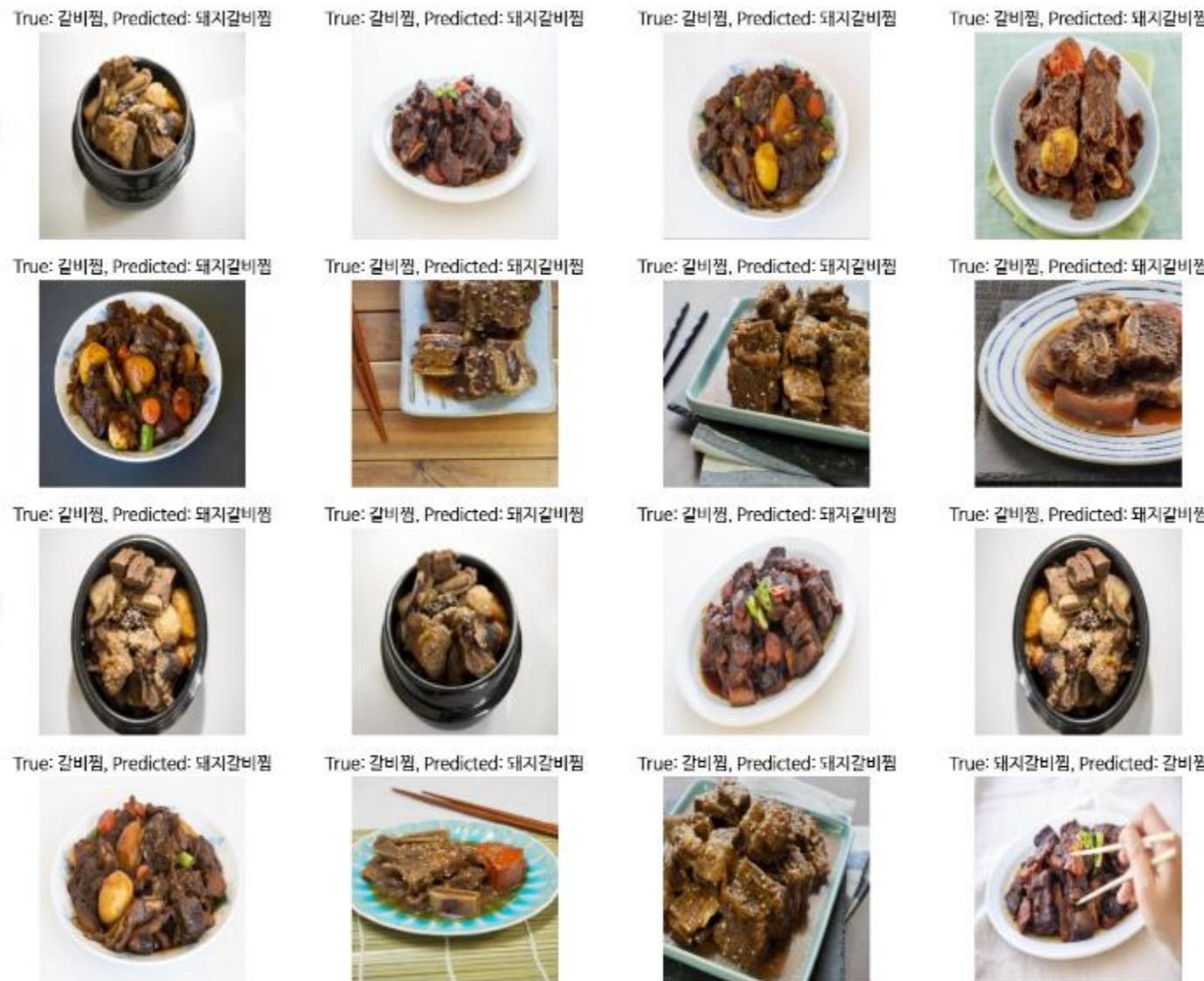
confusion matrix



> 특정 클래스만 오분류율이 심한 것을 발견

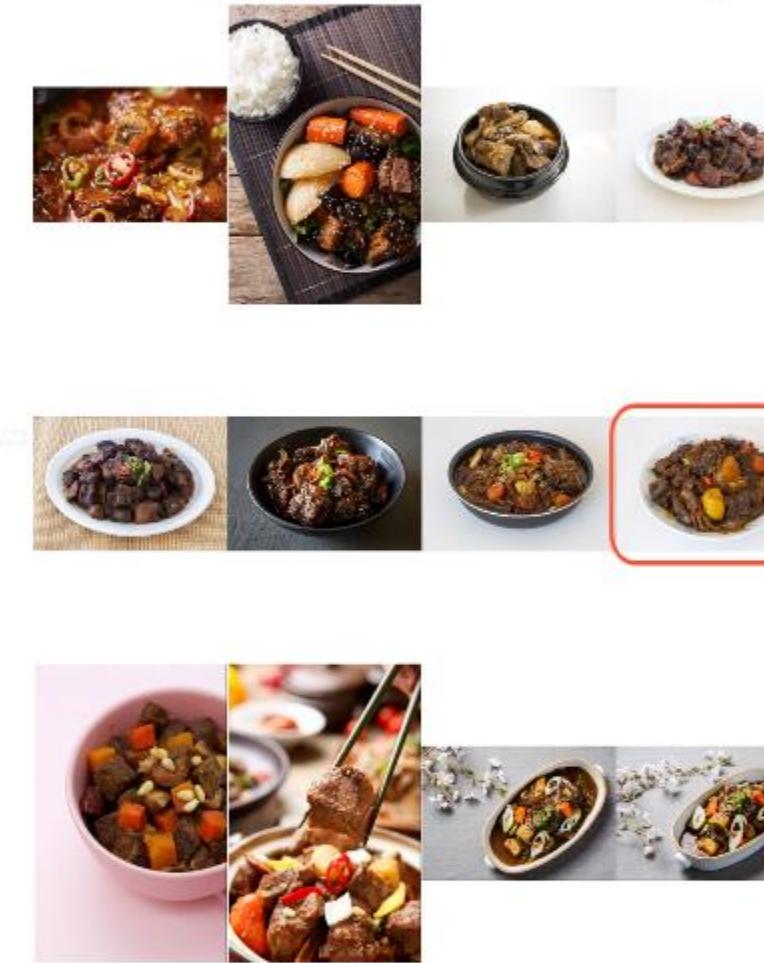
3.5.1 추가 분석 - 데이터 관찰

갈비찜 - 돼지갈비찜 혼동



- 갈비찜을 돼지 갈비찜으로 가장 많이 오분류
- 돼지 갈비찜을 갈비찜으로 가장 많이 오분류

Test Set의 갈비찜 클래스



- 동일한 음식의 다른 구도 이미지의 경우 두 클래스에 모두 속하는 것을 발견
- 두 클래스에 모두 속하는 데이터가 무시할 수 없을 정도로 존재
- 데이터셋 자체가 정확히 분류되어 있지 않기 때문에 모델 학습도 정확히 불가능!

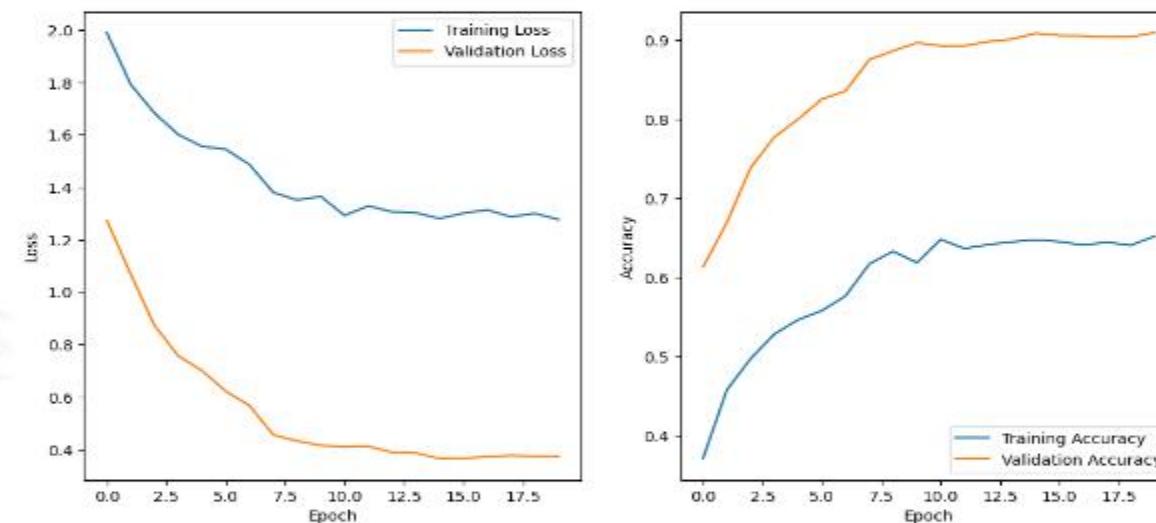
Test Set의 돼지갈비찜 클래스



3.5.2 추가 분석 - 새로운 모델 적합

Mixup을 적용 시킨 모델

데이터 증강(Data Augmentation)과 정규화(regularization)를 동시에 수행하는 기술 중 하나로, 훈련 데이터셋에서 두 개의 이미지와 레이블을 섞어서 새로운 샘플을 만드는 방법



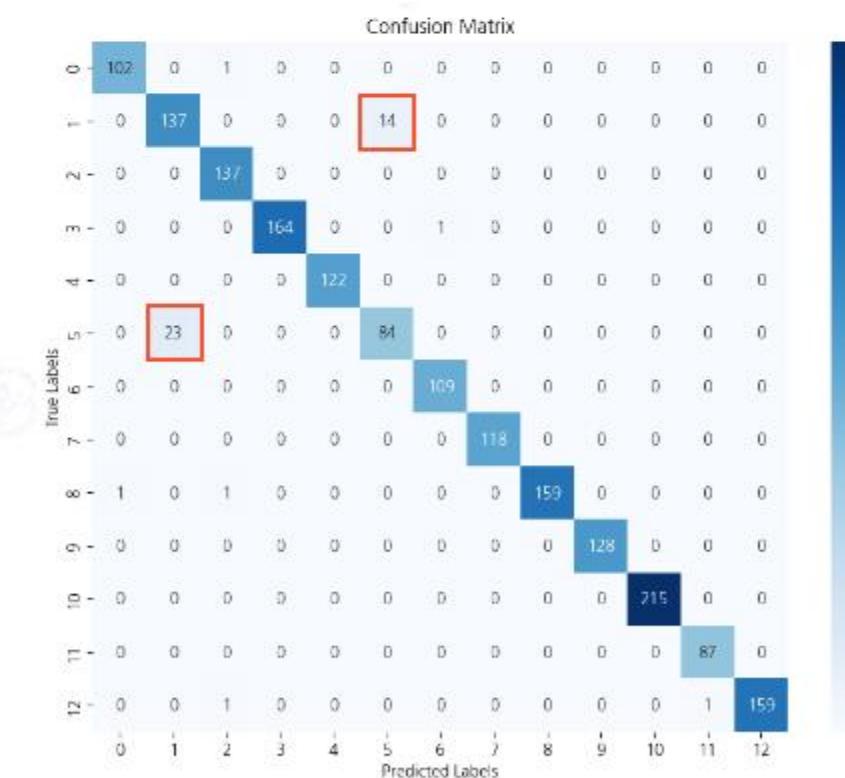
- 눈에 띠는 성능 개선으로 이어지진 않음



오분류율이 높은 두 클래스에 높은 가중치 설정

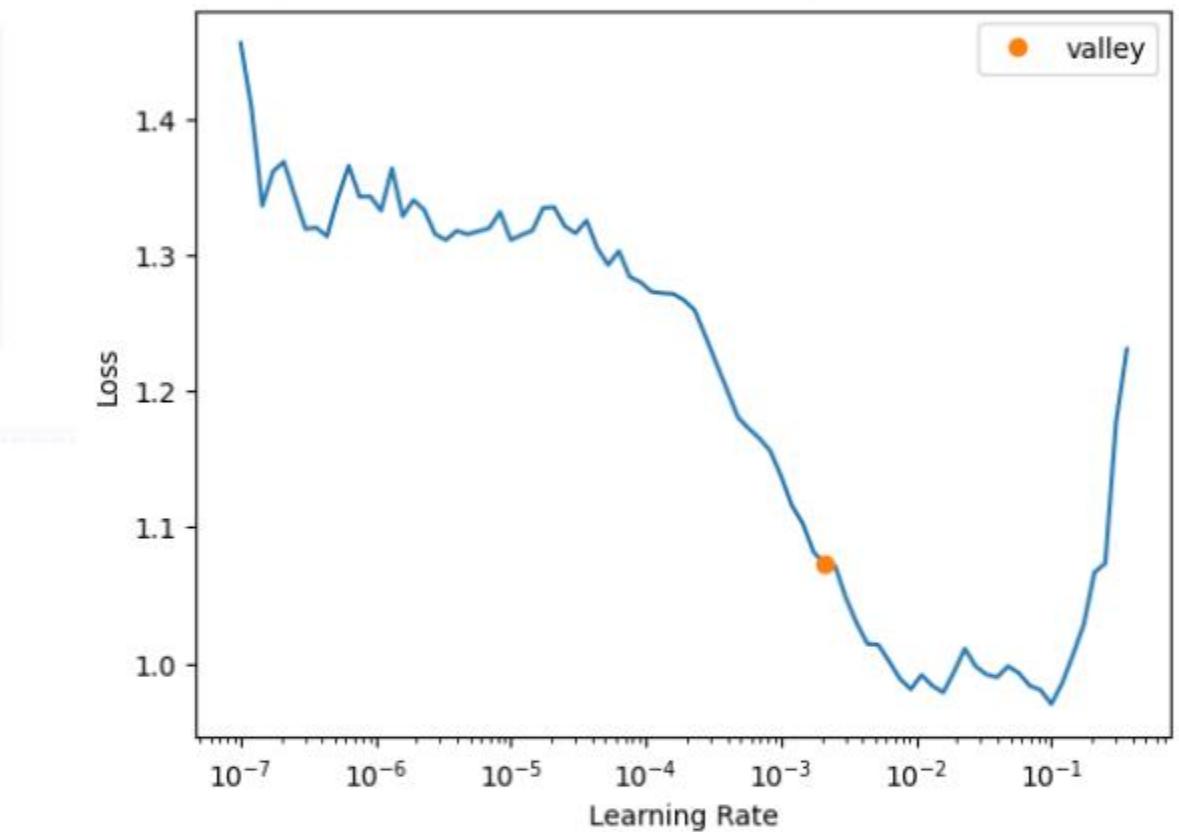
```
# 손실 함수 및 옵티마이저 설정
class_weights = torch.ones(num_classes)
class_weights[1] = 2.0
class_weights[5] = 2.0
criterion = nn.CrossEntropyLoss(weight=class_weights)
```

- 갈비찜-돼지갈비찜에 대한 오분류가 심해 높은 가중치를 지정해주어 오분류 감소를 기대



본래 최종 모델의 성능과 크게 달라지지 않음

갈비찜-돼지갈비찜 이진 분류 CNN 모델을 학습



- 적절한 학습률 범위를 식별

```
learn.lr_find()
learn.recorder.plot_lr_find()
```

> Val Acc
0.8910

random choice보다 높은 확률로
정답을 분류

3.6 한계점

1. 시간 관계상 모델의 하이퍼파라미터(hyperparameter)에 대한 최적의 조합을 찾기 위해 가능한 모든 조합을 시도해보진 못함

- ex) 그리드 탐색(Grid Search) : 가능한 모든 조합을 시도하므로, 하이퍼파라미터가 많거나 각각의 실험이 오랜 시간이 걸릴 경우에는 수많은 시간이 소요

2. GPU가 처리할 수 있는 연산량이나 메모리 용량에 제한이 있어 더 복잡한 모델 고려 불가

- ex) batch size = 128이상, ResNet - 101등

3. 갈비찜 - 돼지갈비찜 같은 분류가 어려운 데이터 존재 시 더 많은 데이터를 학습시킴으로써 어느정도 성능 개선을 기대할 수 있으나 주어진 데이터의 한계로 시도 불가



감사합니다.

대학부

수정

팀장 박지호

백예은

오수아

유소현

최정빈

2023 데이터 크리에이터 캠프

DATA CREATOR CAMP