W16 멀티미디어 프로그래밍 텀 프로젝트

현재 플라스틱의 종류 및 상태를 판별해주는 CNN모델

[redacted]

01.프로젝트 개요

인식 대상 : 여러 종류의 플라스틱

입력 영상

음식용기

1. 세척이 완료된 상태의 짜장면, 김치찜 등의 포장용기

2. 동일한 종류의 포장용기 중 세척을 하지 않아 음식물이 심하게 묻어 있는 상태의 음식용기

3. 동일한 종류의 포장용기 중 세척을 하였으나 음식물이 조금 묻어 있는 상태의 음식용기

투명 페트병

4. 여러 종류의 페트병 중 내용물이 묻어 있고 라벨이 붙어있는 투명 페트병
5. 여러 종류의 페트병 중 내부가 세척되어 있고, 라벨이 붙어있는 투명 페트병
6. 여러 종류의 페트병 중 내부가 세척되어 있고, 라벨을 제거한 투명 페트병
7. 여러 종류의 페트병 중 내용물이 묻어 있고, 라벨을 제거한 투명 페트병

유색 페트병

8. 여러 종류의 유색 페트병 중 내용물이 묻어 있고 라벨이 붙어있는 페트병
9. 여러 종류의 유색 페트병 중 내부가 세척되어 있고, 라벨이 붙어있는 페트병
10. 여러 종류의 유색 페트병 중 내부가 세척되어 있고, 라벨을 제거한 페트병
11. 여러 종류의 유색 페트병 중 내용물이 묻어 있고, 라벨을 제거한 페트병

응용 분야

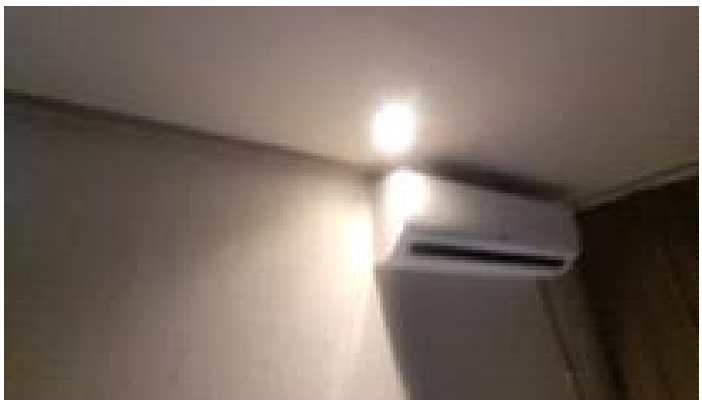
분리수거시 미흡한 점을 알려 주어 분리배출이 용이하게 한다.
또한 재활용 업체에서 수작업으로 분리할 때 해당 모델을 사용하여 미흡한 플라스틱을 종류별로 분리해주어 재활용없이 소각되는 비율을 줄일 수 있다.

02. 데이터 취득 과정

각 데이터는 Train의 경우 class별로 600장 씩 6600장
Test의 경우 150장씩 1650장으로 4:1의 비율을 맞추어 주엇다.

각 class의 600장은 분리 수거환경을 거려하여 어두운 환경, 어두운 환경에서의 황색등, 어두운환경에서의 백색등을 이용하여 데이터의 다양성을 늘려주었다.

페트병의 경우 2종류의 페트병을 사용하여
각 100장 X(2가지 종류)X(조명환경 3가지) = 600장을 맞추어 주었다.

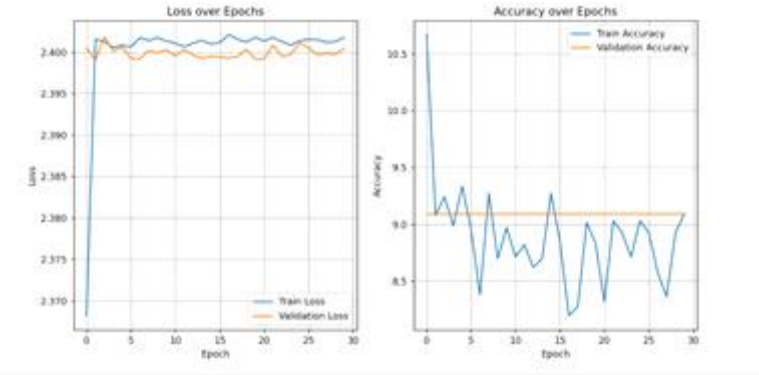


<각 순서대로 어두운 환경, 백색등, 황색등>

<11개 class에 사용된 데이터 예시>

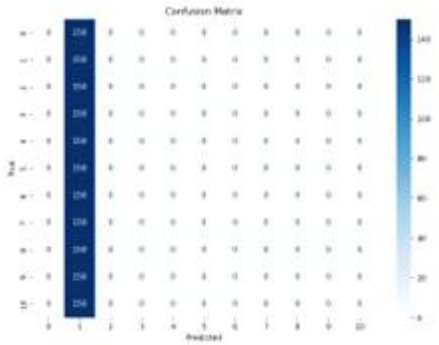
웹캠과 삼각대를 사용하여 다각도 촬영을 진행

03. 신경망을 그대로 사용한 1차결과



<train,test의 loss와 Accuracy>

<혼동행렬>

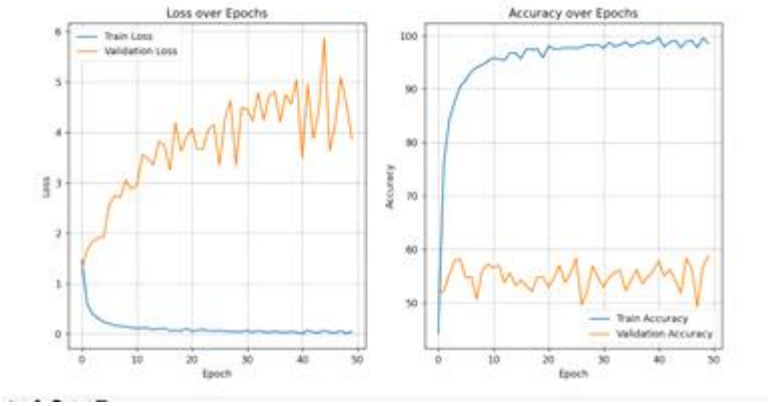


파라메터
Batch size = 8
Learning rate = 0.01
Epochs = 30

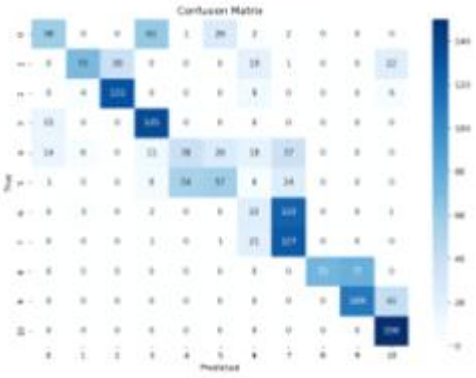
train과 test모두 9%의 정확도를 보였다. 이는 6600장의 데이터 셋에 비해 epoch이 작고 learning rate가 너무 커서 학습이 진행되지 않았다고 볼 수 있다. 2차 학습에서는 해당 결과를 바탕으로 두가지 파라미터를 수정 해 주었다.

04. 신경망의 각종 파라메터 및 학습 변수 변경 후 결과

2차 실험 <파라메터 변경>



<train,test의 loss와 Accuracy>

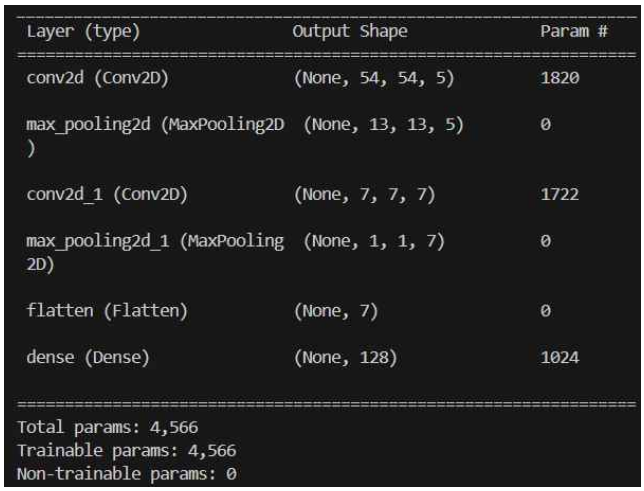


<혼동행렬>

파라메터
Batch size = 8
Learning rate = 0.001
Epochs = 50

1차 학습 결과를 바탕으로 learning rate를 줄이고 epoch을 늘려주었다.
train 98.05% test 58.78%의 정확도를 확인 할 수 있었다.
loss그래프와 Accuracy 그래프를 보면 30epoch이후 train 정확도는 수렴하여
1차 학습의 좋지 못한 결과는 learning rate로 인한 문제가 컸음을 알 수 있다.
또한 loss를 통해 overfitting의 문제가 발생하고 있음을 확인 할 수 있다.

3차 실험<모델 구조 수정>



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 54, 54, 5)	1820
max_pooling2d (MaxPooling2D)	(None, 13, 13, 5)	0
conv2d_1 (Conv2D)	(None, 7, 7, 7)	1722
max_pooling2d_1 (MaxPooling2D)	(None, 1, 1, 7)	0
flatten (Flatten)	(None, 7)	0
dense (Dense)	(None, 128)	1024

Total params: 4,566
Trainable params: 4,566
Non-trainable params: 0

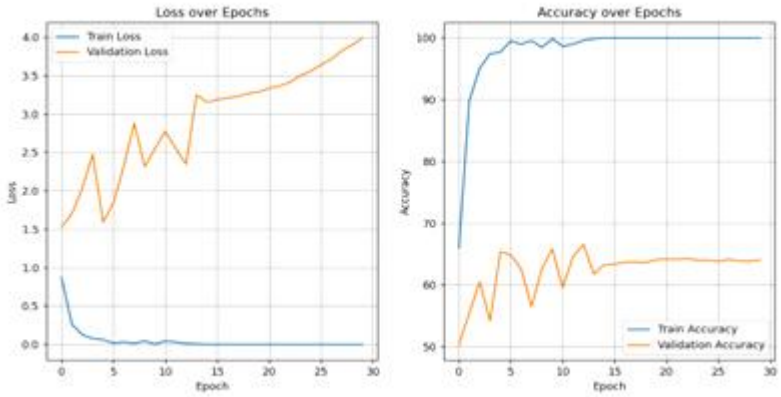
<기존 모델구조>

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 32)	896
leaky_re_lu (LeakyReLU)	(None, 62, 62, 32)	0
max_pooling2d (MaxPooling2D)	(None, 31, 31, 32)	0
conv2d_1 (Conv2D)	(None, 29, 29, 64)	18496
leaky_re_lu_1 (LeakyReLU)	(None, 29, 29, 64)	0
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_2 (Conv2D)	(None, 12, 12, 128)	73856
leaky_re_lu_2 (LeakyReLU)	(None, 12, 12, 128)	0
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 128)	0
conv2d_3 (Conv2D)	(None, 4, 4, 128)	147584
leaky_re_lu_3 (LeakyReLU)	(None, 4, 4, 128)	0
max_pooling2d_3 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262656
leaky_re_lu_4 (LeakyReLU)	(None, 512)	0
dense_1 (Dense)	(None, 11)	5643

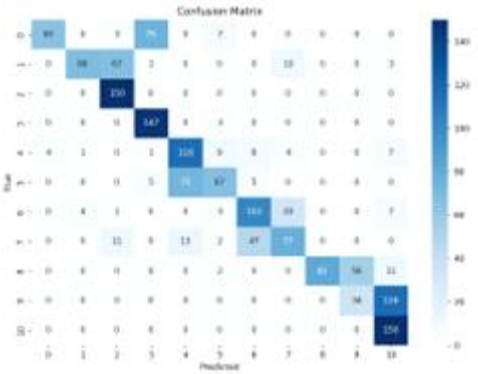
Total params: 509,131
Trainable params: 509,131
Non-trainable params: 0

<수정된 모델구조>

conv2d레이어를 추가하여 복잡한 패턴학습이 가능하도록 모델을 수정하였다.
또한 활성화 함수를 leaky_relu로 변경해주어 비선형성 학습을 강화해 주었으며 kernel size를 3x3으로 수정해 더 작은 범위의 특징을 세밀하게 학습이 가능하도록 모델의 구조를 수정해 주었다.



<train,test의 loss와 Accuracy>



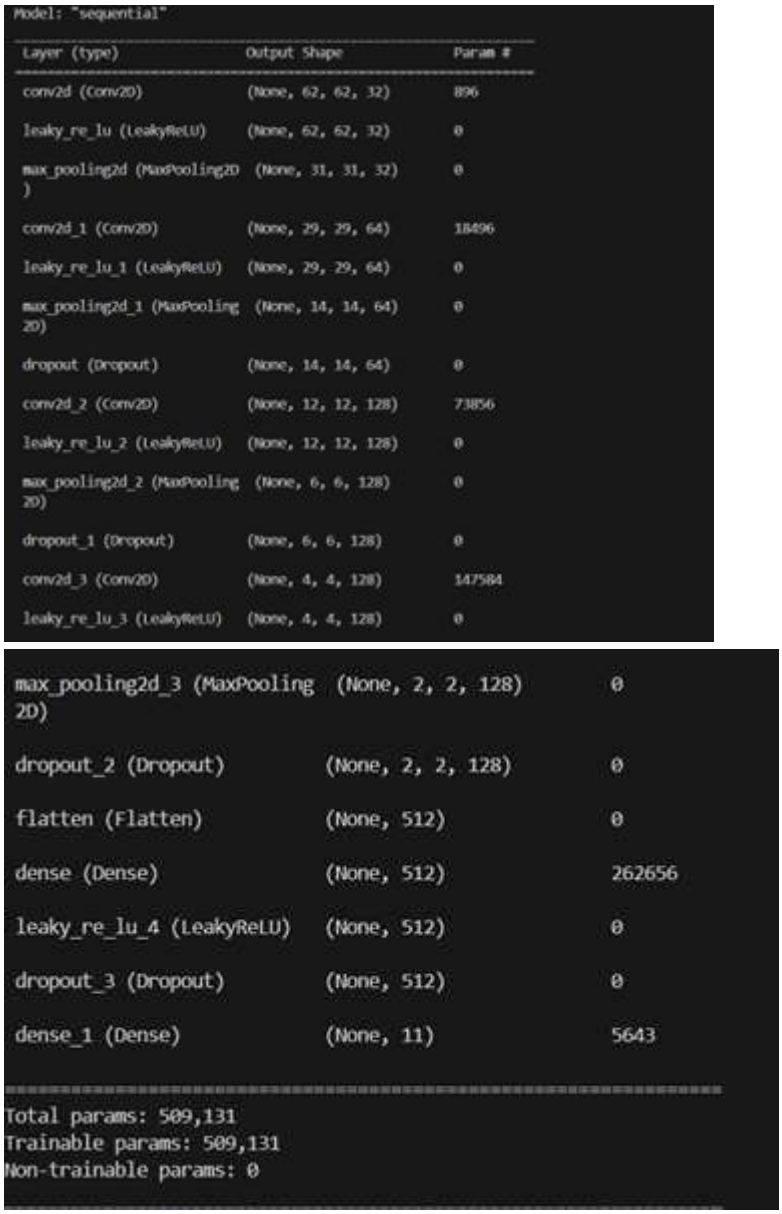
<혼동행렬>

파라메터

Batch size = 16
Learning rate = 0.001
Epochs = 30

작은 batch size는 큰 epochs를 필요로 하기 때문에 batch size를 16으로 늘려 주었다.
train 100% Test 64.06%로 모델의 성능이 올랐으나 15epoch를 기준으로 train 정확도는 수렴하고 test loss는 계속 상승하는 것을 통해 모델이 여전히 일반화가 부족하고 overfitting이 발생하는 것을 확인 할 수 있었다.

4차 실험<모델 구조 수정>



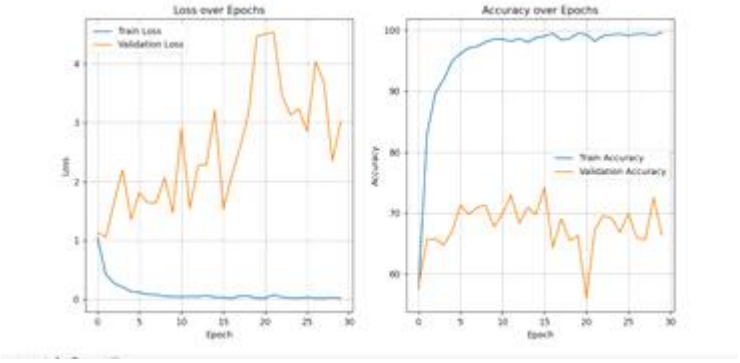
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 leaky_re_lu (LeakyReLU)     (None, 62, 62, 32)        0

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 leaky_re_lu_1 (LeakyReLU)   (None, 29, 29, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 leaky_re_lu_2 (LeakyReLU)   (None, 12, 12, 128)       0

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 128)         0

 conv2d_3 (Conv2D)           (None, 4, 4, 128)         147584

 leaky_re_lu_3 (LeakyReLU)   (None, 4, 4, 128)         0
```



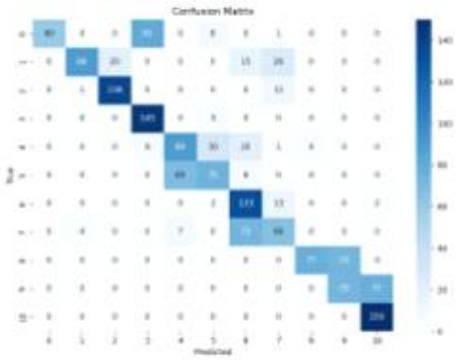
```
 max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 512)               262656

 leaky_re_lu_4 (LeakyReLU)   (None, 512)               0

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 11)                5643

=================================================================
Total params: 509,131
Trainable params: 509,131
Non-trainable params: 0
```

<기존 모델에 dropout을 추가>

첫 레이어는 중요한 특징을 학습하기 때문에 dropout 레이어를 추가하지 않았다. dropout 비율은 conv2d레이어는 0.2 Dense는 가중치가 상대적으로 더 많기 때문에 0.5의 파라메터를 적용해 주었다.



<train,test의 loss와 Accuracy>



<혼동행렬>

파라메터

Batch size = 16
Learning rate = 0.001
Epochs = 30
Dropout : 0.2,0.2,0.2,0.5

train 99% Test 66.32%로 모델의 성능이 올랐으나 test loss는 계속 상승하는 것을 통해 모델이 여전히 일반화가 부족하고 overfitting이 발생하는 것을 확인 할 수 있었다. 다만 꾸준히 상승하던 3차결과에 비해 값이 진동하는 등의 차이가 있었다. 모델의 성능이 증가하였어도 일반화가 부족하여 모델이 신뢰가 부족하기 때문에 최종적으로 overfitting을 줄이기 위해 test loss를 줄이는 방향으로 모델을 수정 하려 한다.

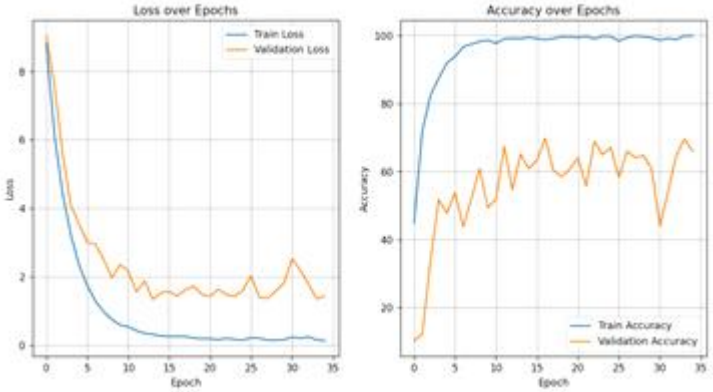
최종 실험<모델 구조 수정 및 파라메터 변경>

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 62, 62, 32)        896

 leaky_re_lu (LeakyReLU)     (None, 62, 62, 32)        0

 max_pooling2d (MaxPooling2D  (None, 31, 31, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 29, 29, 64)        18496

 leaky_re_lu_1 (LeakyReLU)   (None, 29, 29, 64)        0

 max_pooling2d_1 (MaxPooling  (None, 14, 14, 64)       0
 2D)

 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_2 (Conv2D)           (None, 12, 12, 128)       73856

 batch_normalization (BatchN  (None, 12, 12, 128)      512
 ormalization)

 leaky_re_lu_2 (LeakyReLU)   (None, 12, 12, 128)       0

 max_pooling2d_2 (MaxPooling  (None, 6, 6, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 6, 6, 128)         0

 conv2d_3 (Conv2D)           (None, 4, 4, 128)         147584

 batch_normalization_1 (Batc  (None, 4, 4, 128)        512
 hNormalization)

 leaky_re_lu_3 (LeakyReLU)   (None, 4, 4, 128)         0

 max_pooling2d_3 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 dropout_2 (Dropout)         (None, 2, 2, 128)         0

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 1024)              525312

 leaky_re_lu_4 (LeakyReLU)   (None, 1024)              0

 dropout_3 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 11)                11275

=================================================================
Total params: 778,443
Trainable params: 777,931
Non-trainable params: 512
```

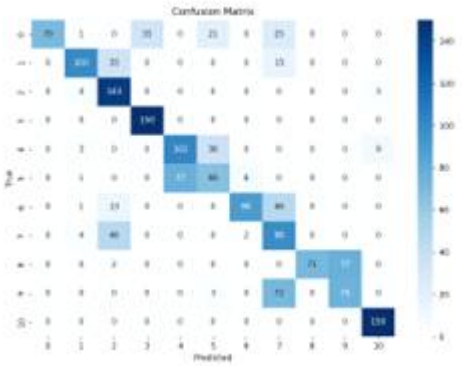
<기존 모델에 batch normalization을 추가>

```
 kernel_regularizer=regularizers.l2(0.003)))
01))
```

<conv2d레이어에 kernel regularizer l2 정규화추가>



<train,test의 loss와 Accuracy>



<혼동행렬>

파라메터

Batch size = 32
Learning rate = 0.00025
Epochs = 35
Dropout : 0.3,0.3,0.3,0.5
Kernel_regularizer :0.003

최종적으로 overfitting을 해결하기 위해 l2정규화를 통해 가중치의 크리를 제한하고 Dropout파라메터를 수정하여 주었으며 배치사이즈를 늘리고 Learning Rate를 줄여주었다.

test 정확도는 67%로 큰 차이가 없었지만 Loss가 증가하던 이전 모델들과 다르게 안정적으로 감소하는 모습을 보여주었다.

즉 train셋 뿐만 아니라 test 셋에도 가장 일반화가 잘 되었음을 알 수 있다.

05. 결과분석

모델을 학습하는 과정을 거치며 데이터셋의 크기에 따라 너무 큰 학습률은 학습이 잘 안 되는 결과를 확인 할 수 있었으며 한가지 파라메터를 변경할 경우 다른 파라메터도 영향을 받는 것을 확인 할 수 있었다.
데이터셋이 6600장으로 적지 않은 양이 었기 때문에 너무 작은 Batch size는 overfitting의 원인이 되었으며 여러 가지 파라메터를 수정하여 최적의 결과를 얻는 과정을 수행하였다.

Conv2d레이어의 커널사이즈와 가중치 정규화 과정 및 Dropout과 batch normalization 등의 레이어를 추가해 주는 것 엮시 모델의 성능을 향상하는 데 도움이 되는 것을 실제 학습을 진행하며 확인 할 수 있었다. 또한 전체적으로 필터의 개수를 늘려주어 많은 특징을 추출한 것이 결과가 좋았음을 알 수 있다.

과적합을 방지하기위해 loss를 줄이는 단계에서 dropout보다 효과가 좋았던 것은 l2정규화였다. 큰 가중치가 존재하면 모델의 복잡도가 증가하는 단점이 있어 가중치의 크기를 줄여 일반화 성능을 향상 시킬 수 있었다.

또한 Kernel mask size의 경우는 단순하게 이미지의 외형을 통해 분류할 수 있는 데이터 셋이 아니라 각 유사한 데이터들간 라벨의 유무 및 잔여물의 유무에 따라서 다르게 분류를 해주어야 했는데 Mask size가 큰 경우 전체적인 이미지의 특징을 추출하며 세세한 이미지의 정보를 추출해야하는 실험에서 부적합 하였기 때문에 3X3 size의 커널 사이즈가 가장 결과가 좋았다고 생각된다.