[Project #2]
Deep Learning for Image Classification with EM Algorithms

[오늘도 (#05)]

	이름	학번	학년	E-mail
팀장	박상은	20211522	3	pse0219@naver.com
조원 1	이찬회	20192923	3	klavier0823@naver.com
조원 2	민경준	20192899	3	cxz017@naver.com
조원 3	함정규	20211556	3	jgham0101@naver.com

1 Introduction

2 Main Structure

2.1 Deep Learning for Image Classification

Image Classification Learning

2.2 EM Algorithms for K-means

Design Loss Function using EM Algorithms

3 Experimental Results

3.1 Deep Learning for Image Classification

(1) Comparison of Hyper parameters

(2) Intermediate feature space & CAM Visualization

3.2 EM Algorithms for K-means

(1) Comparison of Proposed Model

(2) 3D Surface Visualization

4 Discussion and Conclusion

I. INTRODUCTION

이미지는 2 차원의 RGB 값으로 이루어져 있다. 일반적으로, 사람은 이미지를 봤을 때, 무엇인지 알고 있는 물체라면 바로 판단이 가능하다. 그러나 인공지능 모델은 이미지를 분류하기 위한 학습을 한 후, 그 패턴으로 판단하는 과정이 필요하다. 학습 과정에서 CNN 을 이용해 2 차원 공간을 유지하며 특징을 추출할 수 있고, 여러 layer 를 거치다가 마지막 분류 신경망을 통해 추출한 특징을 바탕으로 물체를 인식할 수 있다. 이번 프로젝트에서는 인공신경망 구조 제안 및 소개, 제안한 알고리즘을 통한 학습, intermediate feature space 와 CAM 가시화와 추가적으로 EM 알고리즘(K-means)을 이용한 loss 함수 설계, 결과 및 3D surface visualization 등을 설명하려 한다.

II. MAIN STRUCTURE

2-1. DEEP LEARNING FOR IMAGE CLASSIFICATION

Image Classification Learning

실험을 통해 가장 높은 정확도를 보여 모델로 선정한 최적의 알고리즘을 설명하려 한다. 다른 모델과의 비교는 3 에 작성하였다. 기본이 되는 모델은 keras 의 sequential model 인 순차적으로 layer 를 층을 더해주는 순차 모델이다. 이미지는 0 에서 255 사이 값을 가지는 RGB 이기에 0 에서 1 사이의 값으로 normalization 하고, label 도 one-hot encoding 을 수행하여 벡터로 표현한다. convolutional block 은 Conv-Conv-Pooling-Dropout 의 구조로 제작하였다. (1)첫 block 을 기준으로 2 개의 Conv 의 경우, 3x3 크기의 kernal 을 64 개 사용하며 stride 는 1 로 수행하였고, activation function 은 ReLU 활성화 함수를 사용하였다. 비교 연산 한번으로 1 차 도함수 계산이 빠르며 부드러운 의사결정을 수행하여 딥러닝에 적합하기 때문이다. 추가적으로 input 의 outline 에 적절한 개수의 0 을 추가하여 output feature map 이 input 과 동일한 공간 차원을 가지도록 하였다. input 의 경우 32 개의 channel 과 3x3 의 공간차원으로 지정되어 이에 따라 데이터가 처리되도록 하였다. (2)conv layer 사이에 배치 정규화를 추가함으로써 input 의 평균과 분산을 정규화하여 학습을 안정화하였다. (3)다음은 Pooling 이다. input 의 공간적인 정보는 보존하되 크기를 줄이고자 MaxPooling 방식으로 2x2 window 크기를 정하였다. (4)마지막으로

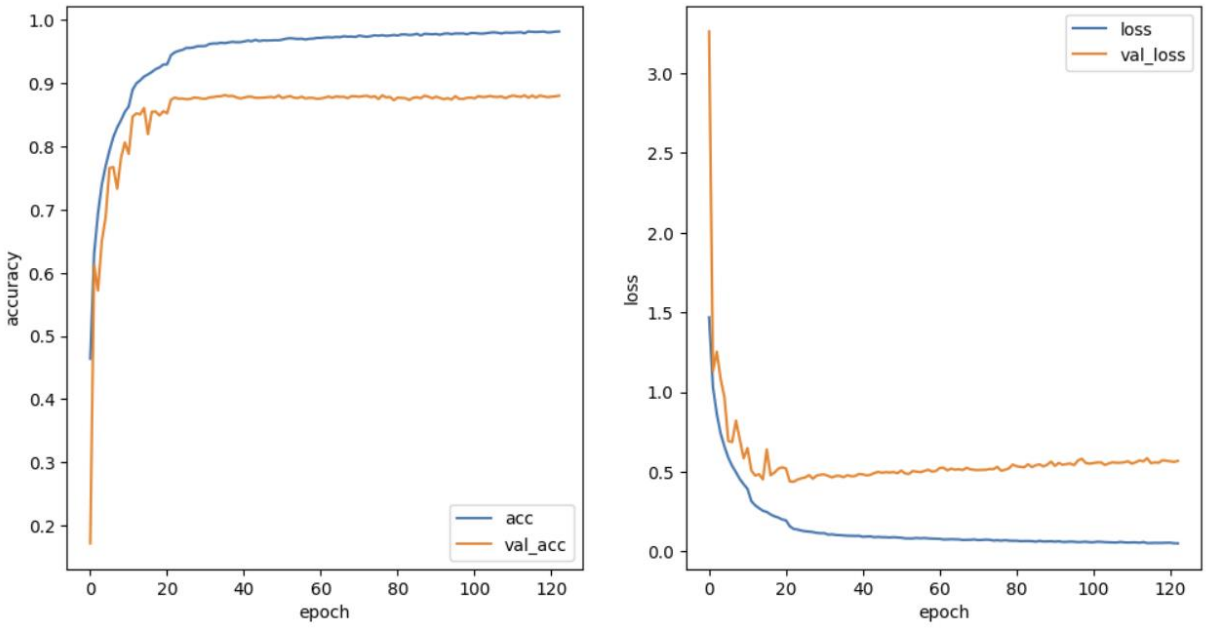
Dropout 이다. 학습과정에서 과적합을 방지하고 일반화 성능을 높이기 위해 0.25 인 25%의 unit 을 비활성화하도록 지정했다. 아래의 자료는 설명한 첫 block 과 더불어 모든 layer 층을 보여주는 summary 이다.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 64)        1792

 batch_normalization (BatchN  (None, 32, 32, 64)       256
 ormalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 64)        36928

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 dropout (Dropout)           (None, 16, 16, 64)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 128)       73856

 batch_normalization_1 (Batc  (None, 16, 16, 128)      512
 hNormalization)

 conv2d_3 (Conv2D)           (None, 16, 16, 128)       147584

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 8, 8, 128)         0

 conv2d_4 (Conv2D)           (None, 8, 8, 256)         295168

 batch_normalization_2 (Batc  (None, 8, 8, 256)        1024
 hNormalization)

 max_pooling2d_2 (MaxPooling  (None, 4, 4, 256)        0
 2D)

 dropout_2 (Dropout)         (None, 4, 4, 256)         0

 global_average_pooling2d (G  (None, 256)              0
 lobalAveragePooling2D)

 dense (Dense)               (None, 512)               131584

 dropout_3 (Dropout)         (None, 512)               0

 dense_1 (Dense)             (None, 10)                5130

=================================================================
Total params: 693,834
Trainable params: 692,938
Non-trainable params: 896
_________________________________________________________________
```

<그림 1. convolution summary>

loss function 은 직전에 activation function 으로 softmax 를 사용하여 softmax loss 인 categorical_crossentropy 을 사용하여 MSE loss 보다 더 빨리 수렴하도록 설정하였다. 경사하강법보다 더 빠른 수렴과 좋은 성능을 보여주기 때문에 optimazer 는 Adam 으로 학습 속도를 조절하였으며 learning rate 를 0.001 로 설정하였다. 다음은 epoch 에 따른 learning rate 설정이다. epoch 에 따라 10 을 초과하면 learning rate 를 0.0005 로, epoch 20 을 초과하면 0.0001, epoch 가 30 을 초과하는 경우 0.00005 로 동적으로 조정된다. 추가로 만약 학습이 100 번의 연속 epoch 동안 개선이 없는 경우 학습을 중단하도록 종료조건을 넣었다.

제안한 알고리즘을 기반으로 학습을 진행하였다. 최대 epoch 는 150 이나, 종료조건으로 인해 epoch 는 122 회 진행되었다. 학습 후 accuracy/loss graph 와 정확도를 출력하였고, 아래의 <그림 1>은 제안한 모델의 학습 후 test 한 값의 epoch 에 따른 accuracy(좌측) or loss(우측)의 graph 이다. 하단에는 loss, accuracy 의 값을 출력했으며 정확도는 0.8644, 즉 86.44%를 보였다.

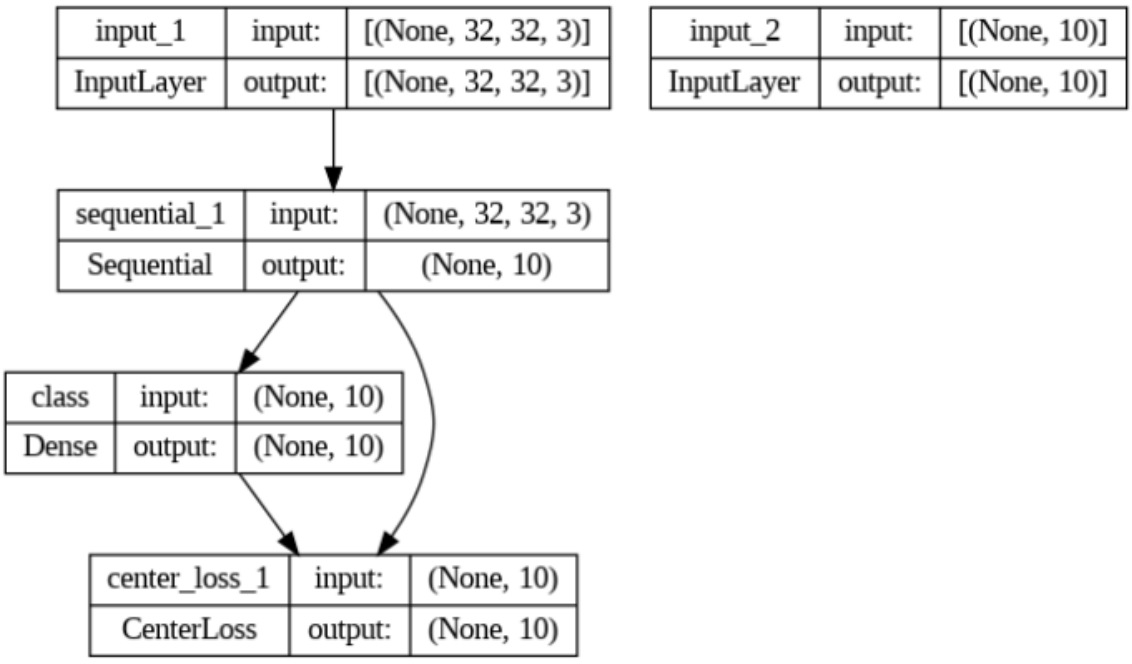


<그림 2. 제안한 CNN 모델의 accuracy/loss graph>

2-2. EM ALGORITHMS FOR GMM, K-MEANS

Design Loss Function using EM Algorithms

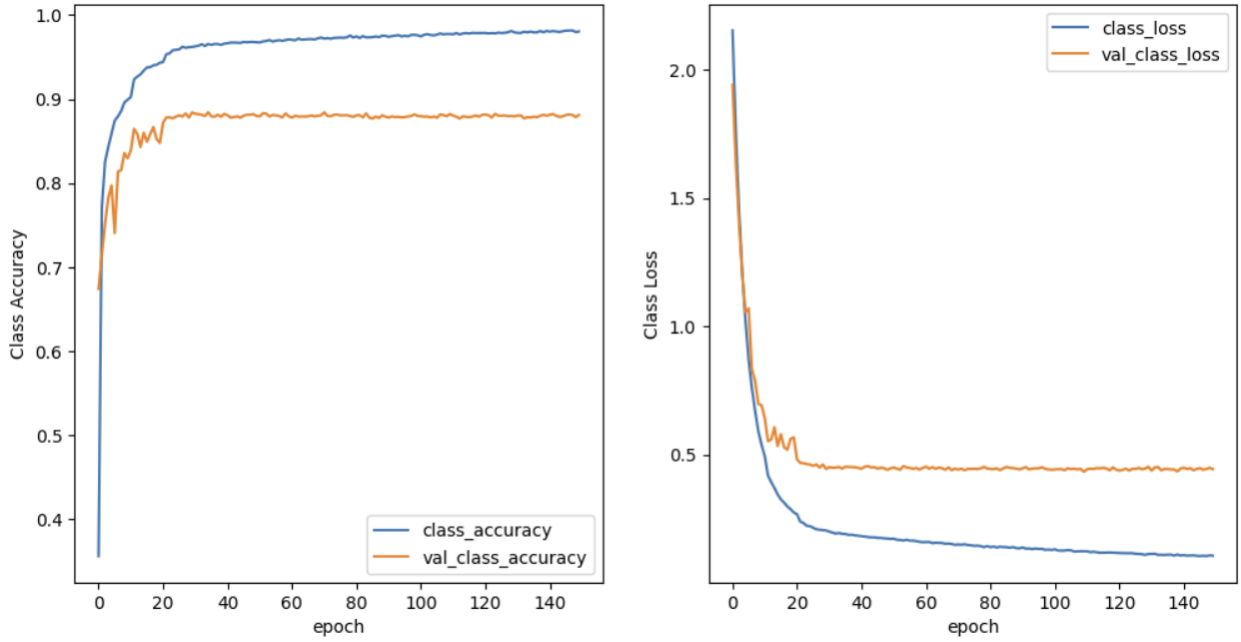
K-means clustering 결과를 바탕으로 class 간 거리를 증가하는 center-aligned loss function 을 설계하였다. class 내부 표본 간의 거리는 최소화하며 다른 class 간의 거리는 최대화하여 경계를 명확하게 만듦으로써 정확도를 높일 수 있다. class 의 center 을 계산하여 각 class 에 속하는 표본들의 feature space 에서의 mean location 으로 정의하고 center 를 초기화하기 위해 K-means 알고리즘을 사용한다. 이후, 표본의 feature vector 와 center 사이의 거리를 계산하여 loss 함수가 최소화하려는 값을 구하여 최종 손실 값을 얻었다. 이 모델로 학습하기 전, 기존의 loss function 과 고안한 loss function 을 결합하여 최적화하였다. 가중치 w 는 두 개의 loss function 의 gradient 를 함께 사용하며 class 내부 분산 감소, class 간의 분리를 개선하며 결국 새로운 test 에 대한 정확도를 높일 수 있었다. 3 에서 기존의 CNN 모델과 고안한 모델의 비교를 위해 epoch 에 따른 learning rate 와 종료조건은 동일하게 적용하였고, 최대 150epoch 중 150epoch 의 학습이 진행되었다.



<그림 3-1. Additional Loss layer>

기존 CNN 모델의 layer 는 sequential_1 에 변경없이 압축되었고, 'class', 'input_2', 'center_loss_1'의 정확도를 최적화하였다. 위의 layer 그림은 계산 그래프에서 loss layer 를 정의하기 위한 방법이다. loss 를 추가함으로써 모델이 좀 더 효과적으로 학습할 수 있었다.

아래의 <그림 6>은 제안한 모델의 학습 후 test 한 값의 epoch 에 따른 accuracy(좌측) or loss(우측)의 graph 이다. 하단에는 loss, accuracy 의 값을 출력했으며 정확도는 0.878, 즉 87.8%를 보였다.




```
313/313 - 1s - loss: 0.8131 - class_loss: 0.4297 - center_loss_2_loss: 0.1000 - class_accuracy: 0.8788 - center_loss_1_accuracy: 0.0951 - 1s/epoch - 4ms/step
Test accuracy: 0.8787999749183655
Test class loss: 0.4296596646308899
```

<그림 3-2. 새로운 loss function 을 적용한 모델의 accuracy/loss graph>

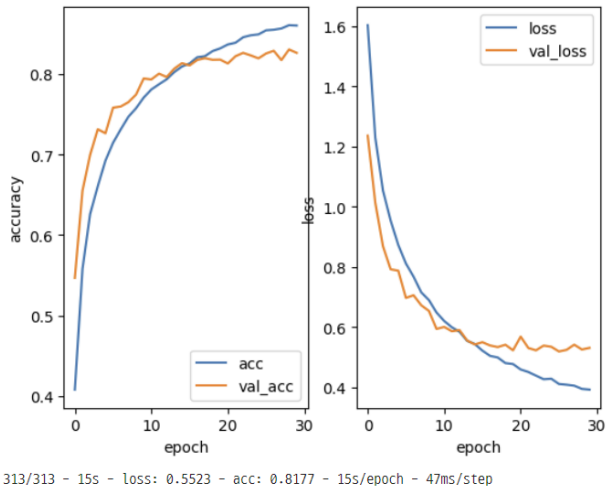
EXPERIMENTAL RESULTS

3-1. DEEP LEARNING FOR IMAGE CLASSIFICATION

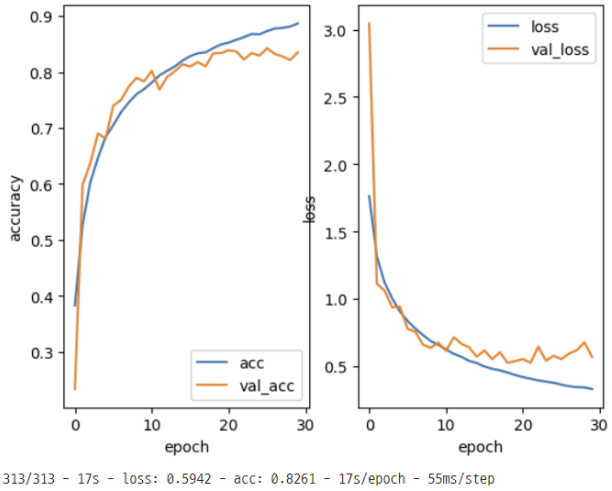
(1) Comparison of Hyper parameters

2 에서 소개한 CNN 모델은 hyper parameter 를 변형해가며 실험을 진행한 후 가장 높은 정확도를 보인 모델을 선택하였다. 유의미한 결과를 보인 6 가지 모델을 어떤 값을 변형하였는지, 그에 따른 accuracy/loss graph 와 정확도를 비교하려 한다. a) 기본 keras 의 sequential model b) 기본 모델의 Conv2D 사이 Batch normalization layer 추가 c) b + epoch 에 따른 learning rate 변형 d) b + c + training image 의 rotation & flip e) b + c + SGD optimize f) b + c + RMSprop optimize 의 모델을 생성하였고, 명확한 비교를 위해 epoch 는 30 으로 정한 후 학습을 진행하였다.

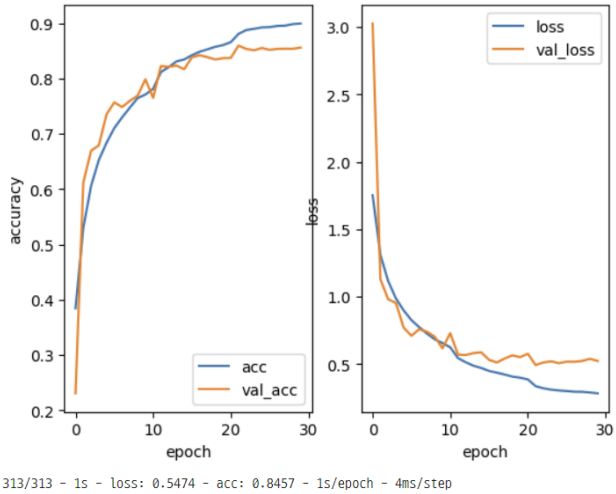
아래의 좌측 그림은 a) 기본 keras 의 sequential model 의 graph 이다. 순차적인 구조이기 때문에 각 layer 는 한 번에 하나의 input 을 받고, 하나의 output 을 생성하는 특징이 있다. 정확도는 81.77%를 보이며 loss 는 0.5523 으로 나타났다. 우측 그림은 b) 기본 모델의 Conv2D 사이 Batch normalization layer 추가 model 의 graph 이다. 배치정규화는 학습 시 평균과 분산을 조정하는 과정으로 변형된 분포가 나오지 않도록 조절할 수 있다. 정확도는 82.61%, loss 는 0.5942 가 나타났다.



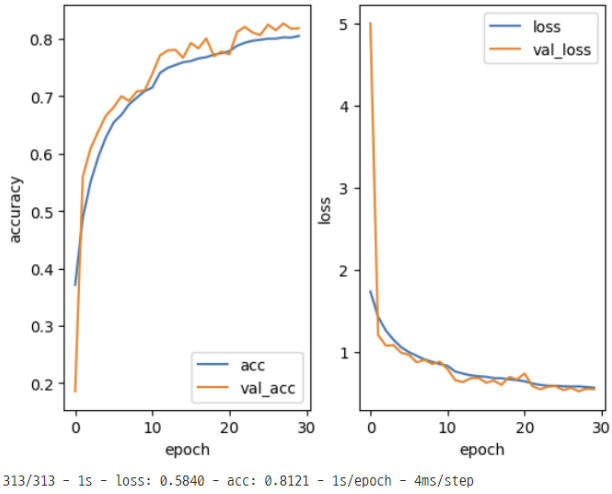
<그림 4-1. a model 의 accuracy/loss graph>



<그림 4-2. b model 의 accuracy/loss graph>



<그림 4-3. c model 의 accuracy/loss graph>

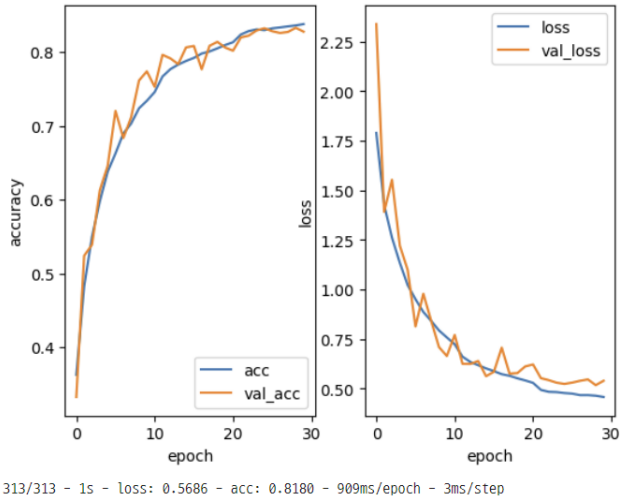


<그림 4-4. d model 의 accuracy/loss graph>

위의 좌측 그림은 c) b + epoch 에 따른 learning rate 변형 model 의 graph 이다. learning rate 는 적정 값을 초과하는 값이면 W 값이 발산하며 overshooting 이 발생한다. 반대로 learning

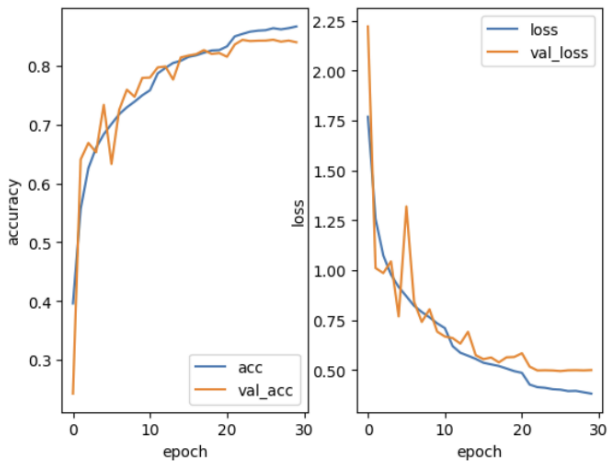
rate 가 적정 값보다 작은 값이면 수렴하기까지 너무 많은 iteration 으로 학습 속도에 영향을 미친다. 따라서 2-1 에서 설명했듯이 epoch 의 증가에 따라 learning rate 를 줄여가며 학습하도록 설정하였다. 정확도는 84.57%, loss 는 0.5474 이 나타났다. 우측 그림은 d) b + c + training image 의 rotation & flip model 의 graph 이다. training image 를 15° 이내로 rotation 하거나, 좌우 반전인 flip 을 적용한 image 를 추가하여 학습시켰다. 데이터의 정보량은 보존한 상태로 noise 를 주는 방식이기에 가지고 있는 정보량은 변하지 않고 정보량에 약간의 변화를 주는 것이다. 그러나 결론적으로 Data Augmentation 은 뚜렷한 장점을 보이지 않았다. 정확도는 81.21%, loss 는 0.5840 으로 a 기본 모델보다 정확도가 떨어졌음을 알 수 있다.

앞의 실험에서는 optimize function 으로 RMSprop 과 Momentum 방식을 합친 Adam 을 사용하였지만 남은 두 실험에서는 최적화 함수를 바꾼 모델을 제작하였다. SGD optimize 는 mini-batch 를 적용해서 크기만큼의 데이터 손실을 계산하는 방법이다. RMSprop optimize 는 학습이 진행될수록 learning rate 가 감소되는 Adagrad 의 문제점을 해결해주는 방법으로 g 를 제곱의 지수 이동 평균으로 나타냄으로써 g 가 최근 변화량의 변수 간 상대적인 크기 차이를 유지할 수 있다. 아래는 각각 e) b + c + SGD optimize f) b + c + RMSprop optimize model 의 graph 이다. SGD optimize 인 경우 정확도는 81.8%, loss 는 0.5686 를 보였고, RMSprop optimize 는 정확도 83.47%, loss 는 0.5215 를 나타냈다.



313/313 - 1s - loss: 0.5686 - acc: 0.8180 - 909ms/epoch - 3ms/step

<그림 4-5. e model 의 accuracy/loss graph>



313/313 - 1s - loss: 0.5215 - acc: 0.8347 - 1s/epoch - 4ms/step

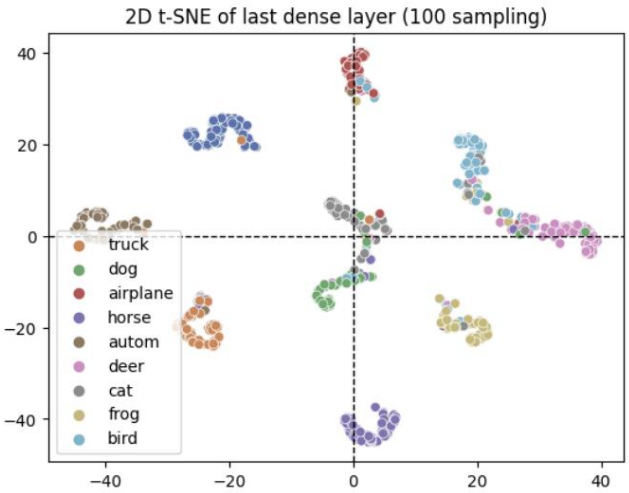
<그림 4-6. f model 의 accuracy/loss graph>

6 가지 모델 간의 비교를 통해 결론적으로 c) b + epoch 에 따른 learning rate 변형 model 을 최종 CNN model 로 선정하였다. 가장 높은 정확도를 출력했으며, optimize 는 Adam 을 사용함으로써 이전까지의 상황에 따라 방향과 step size 를 조정하는 방식으로 가장 안정적인 그래프를 보였기 때문이다.

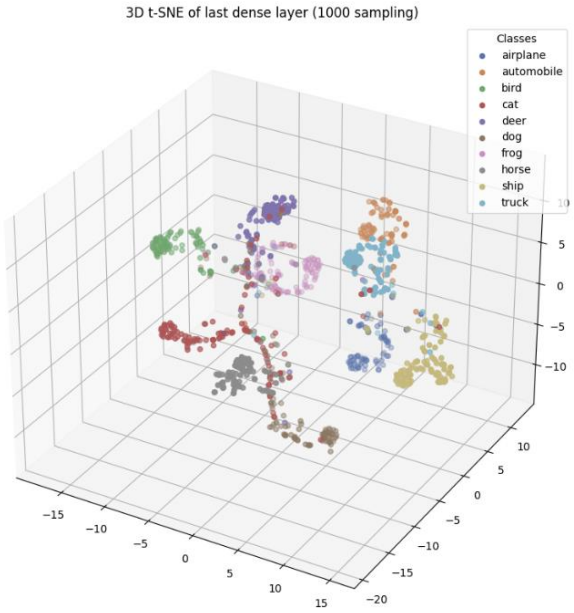
(2) Intermediate feature space & CAM Visualization

Intermediate feature space

t-SNE 는 비선형적 방법으로 높은 차원의 복잡한 데이터를 분포 확인 혹은 시각화를 위해 2 에서 3 차원으로 축소하는 방법이다. 3-2 (2)에서의 비교를 위해 마지막 Dense layer 의 output 을 2D, 3D 로 나타내었고 제안한 모델이 만드는 feature map 을 가져와 t-SNE 알고리즘과 PCA(주성분 분석)을 적용하였다. 주성분 분석은 데이터에 있는 정보를 유지하면서 차원을 줄이는 통계 방법이다. 적용 과정은 중간 layer output → 차원 축소를 위한 데이터 형태 변경 → PCA 를 사용한 차원 축소 → sampling 과 t-SNE →PCA 결과 sampling → t-SNE → 출력이다. 각 색상의 점들은 총 10 class 를 나타낸다. 다음은 CNN 모델의 마지막 Dense layer 의 output feature space 이다.



<그림 5-1. 2D t-SNE of last dense>

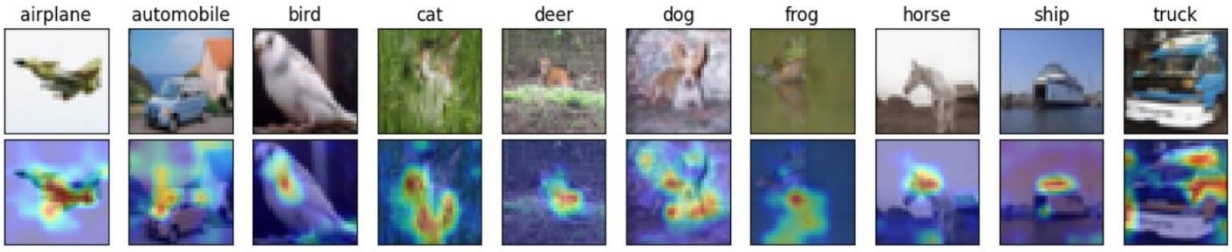


<그림 5-2. 3D t-SNE of last dense>

<그림 5-1>과 <그림 5-2>를 보면 class 별 표본들이 같은 색상이 모여있음을 확인할 수 있다.

CAM Visualization

CAM visualization 은 Grad-CAM 을 사용하였다. CNN 층에서 Grad-CAM 을 사용하기 위해 flatten 대신 GlobalAveragePooling2D 를 이용하였다. Grad-CAM 은 feature Map 이 어느정도 모델의 결과값에 기여했는지와 각 픽셀이 모델의 결과값에 어느정도 기여했는지를 곱하는 것이다. 결국, 각 픽셀이 feature map 을 고려하였을 때 결과값에 어느정도 영향을 미쳤는지를 계산할 수 있다. 특정 class 에 대한 loss 를 선택 후, loss 에 대해 연산그래프를 역전파하여 gradient 를 계산하고 가중치 w 를 얻었다. Heatmap 은 데이터의 상대적인 크기나 값을 색상의 강도로 나타내어 시각화하는 그래프인데, 붉은 부분이 CNN 모델에서 클래스의 확률에 큰 기여를 함을 의미한다. 아래의 <그림 6>는 각 클래스 별 Grad-CAM 적용 전 후 이미지의 heatmap 이다.

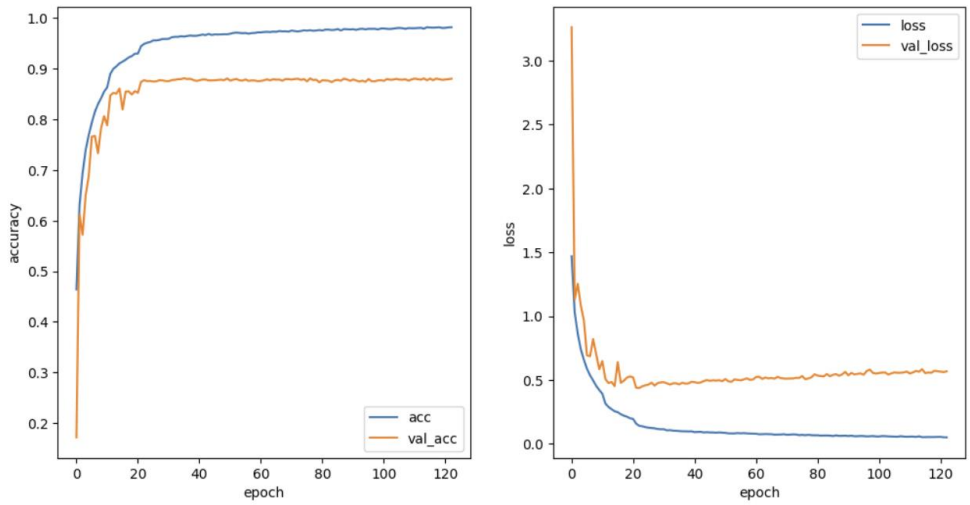


<그림 6. 각 class 별 heatmap>

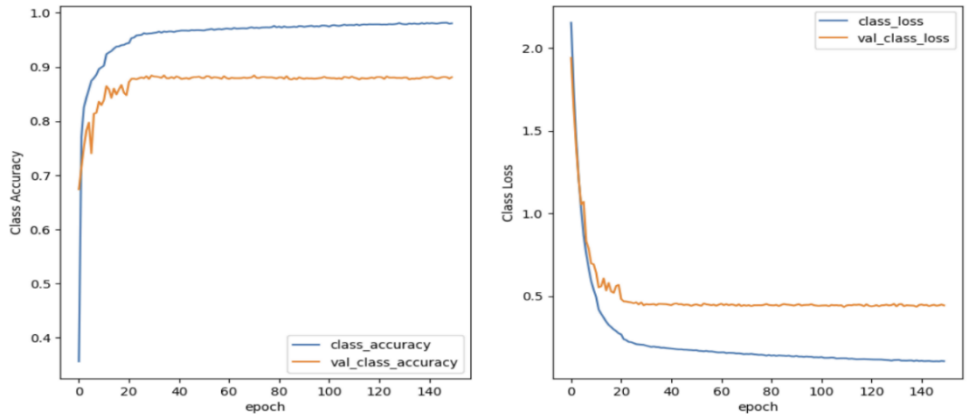
3-2. EM ALGORITHMS FOR GMM, K-MEANS

(1) Comparison of Proposed Model

제안한 CNN 모델과 새로운 loss function 을 적용한 모델을 비교한다. K-means clustering 의 사용으로 class 의 중심을 효과적으로 찾을 수 있고, 이를 통해 loss 계산에 유용하게 사용된다. clustering 결과에 새로운 loss function 을 기존의 loss function 과 결합해 도입하여 class 내부의 표본들 간 거리는 최소화하여 내부 분산 감소, class 간의 거리는 최대화하여 구분이 명확해짐으로써 정확도를 높일 수 있었다. 기존의 CNN 모델은 일반적인 loss function 을 사용하였기 때문에 class 내부의 데이터 간 거리를 고려하는 것에 약하고, class label 간의 분류 경계에 더 뛰어나다. class 내부 분산이 큰 경우, 표본들의 분류 오류가 커지는데, K-means 를 활용한 새로운 loss function 으로 향상된 정확도와 시각화를 얻을 수 있었다.



<그림 7-1. 기존 CNN 모델의 accuracy/loss graph>

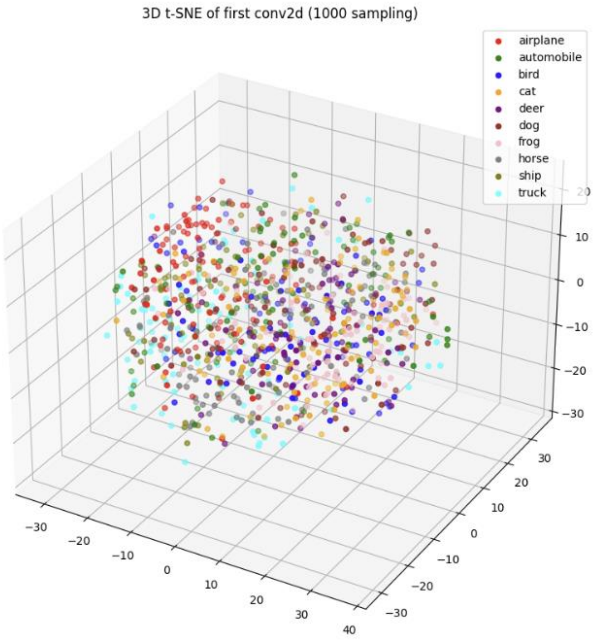


<그림 7-2. 새로운 loss function 적용한 CNN 모델의 accuracy/loss graph>

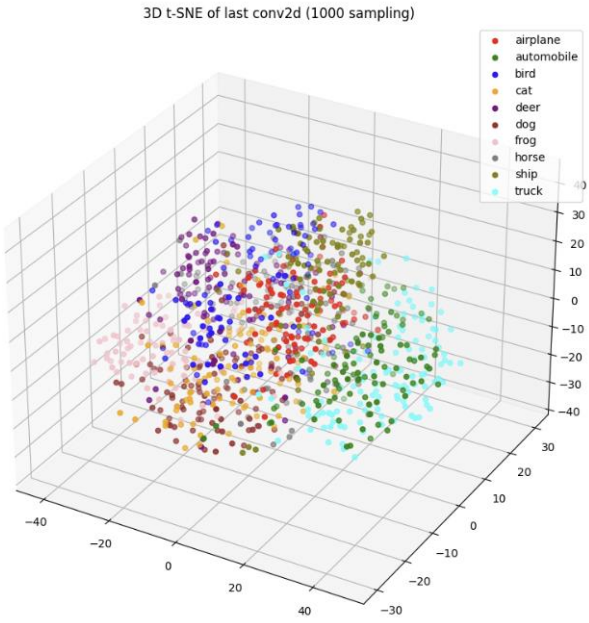
(2) 3D Surface Visualization

Intermediate feature space

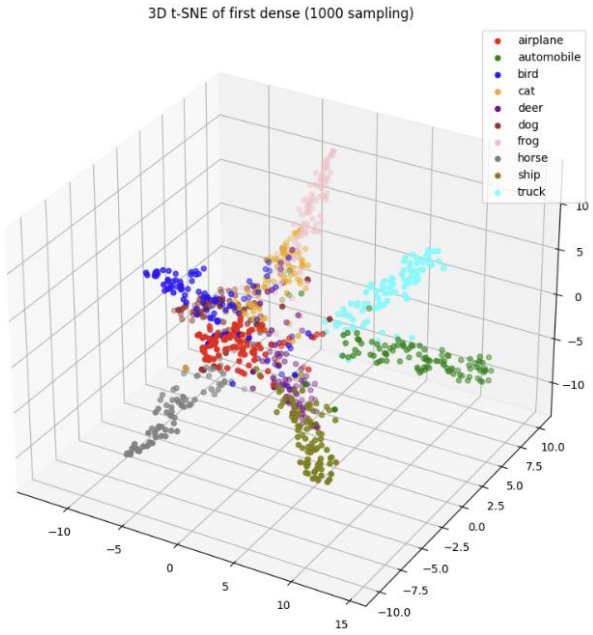
적용 과정은 기존의 t-SNE 와 마찬가지로 중간 layer output 가져오기 → 차원 축소를 위한 데이터 형태 변경 → PCA 를 사용한 차원 축소 → sampling 과 t-SNE 적용 →PCA 결과 sampling → t-SNE 적용 → 출력을 거친다. 총 4 개의 layer output 으로 3D t-SNE 를 적용, 가장 얕은 layer 부터 마지막 layer 순으로 그림을 나타내었다. <그림 8-1>과 <그림 8-2>는 첫 번째 Conv2D layer 와 마지막 Conv2D layer 의 output 을 시각화한 그림이다. 좌측 그림에 비해 우측 그림의 class 별 색상이 모여 있음을 확인할 수 있다. 아래의 <그림 8-3>과 <그림 8-4>는 첫 번째 Dense layer 와 Class layer 의 output 을 시각화한 그림이다. 같은 색상, 즉 같은 class 내부의 표본은 서로 더 가까워지고, class 간의 거리는 멀어짐을 확인할 수 있다.



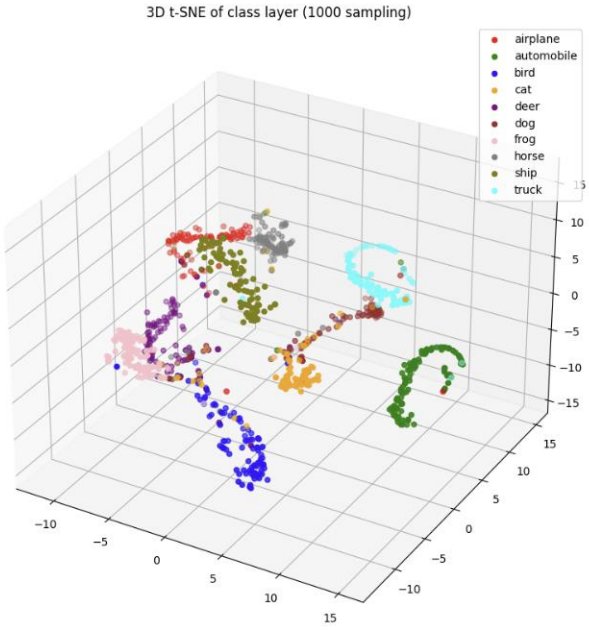
<그림 8-1. 3D t-SNE of first Conv2D>



<그림 8-2. 3D t-SNE of last Conv2D>

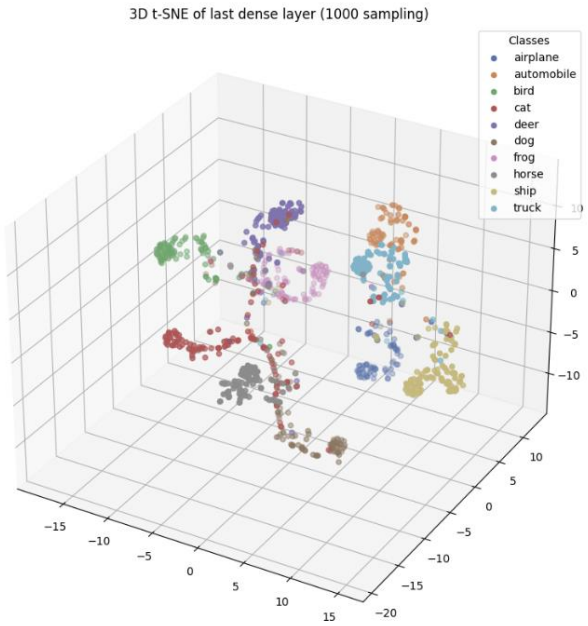


<그림 8-3. 3D t-SNE of first Dense>

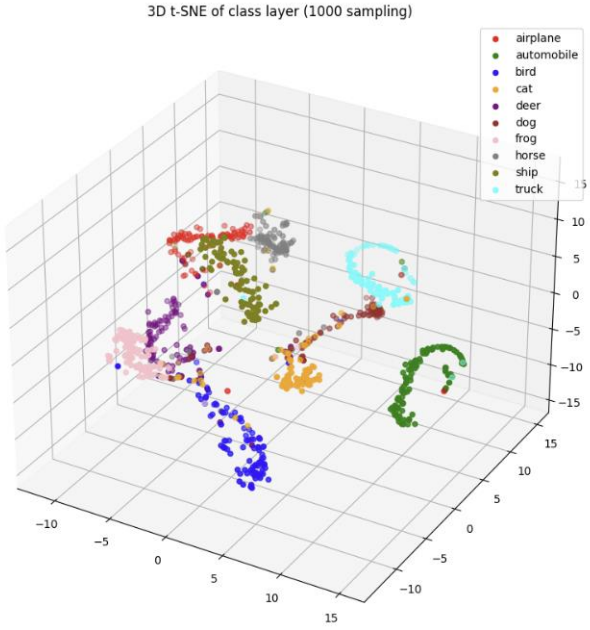


<그림 8-4. 3D t-SNE of Class layer>

기존 CNN 모델과의 t-SNE 비교이다. 아래의 좌측은 기존 CNN 모델의 last Dense layer 의 output 시각화이고 우측은 K-means 와 새로운 loss function 을 적용한 CNN 모델의 Class layer output 시각화이다. 각 class 간의 거리는 멀어지고, class 내부 표본들 간의 거리는 가까워진 것을 확인할 수 있다.



<그림 9-1. 기존 CNN 모델 3D t-SNE>



<그림 9-2. 새로운 CNN 모델 3D t-SNE>

DISCUSSION AND CONCLUSION

최적화 함수 변경, learning rate 조정, train 이미지 변환 등 hyper parameter 를 변경 실험을 통해 최적의 모델을 택하였다. 이후 제안한 CNN 모델을 학습시킨 후 t-SNE 가시화 및 Grad-CAM 을 이용한 heat-map 가시화를 함으로써 클래스 간의 분류를 시각적으로 확인하였다. 추가적으로 K-means clustering 후 새로운 loss function 을 설계, 적용한 모델을 학습시킨 후, 3D t-SNE 가시화를 4 개의 layer output 으로 진행하였으며 기존 CNN 모델과의 비교로 성능 향상을 확인하였다. 위의 각 요소의 선택 과정, 정확도 및 가시화 그림으로 비교 분석 및 과정을 살펴봄으로써 제안한 CNN 모델을 이용해서 이미지 분류 deep learning 모델을 개발할 수 있었다.