[Project #2]

Deep Learning for

Image Classification

[Team#2]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **이름** | **학번** | **학년** | **E-mail** |
| 팀장 | 고준영 |  |  | solbon1212@naver.com |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. Introduction

Deep Learning(심층 학습)은 Machine Learning(기계 학습)의 특정한 한 분야로서, 사람에게는 자연스러운 일, 즉 예시를 통해 학습하는 것을 컴퓨터가 수행할 수 있도록 가르치는 기법이다. 대부분의 딥러닝 방식은 인간의 뇌와 유사한 계층 구조에서 상호 연결된 노드 또는 뉴런을 사용하여 학습하는 적응형 시스템인 신경망 아키텍쳐를 사용한다. 인공 신경망은 인간의 뇌가 동작하는 것과 유사하게 데이터를 통해 패턴을 인식하고 데이터를 분류하며, 향후 사건을 예측하도록 훈련시킬 수 있다.

CNN(Convolutional Neural Networks)은 이러한 인공신경망 중 가장 널리 사용되는 유형 중 하나로, 입력 데이터에 대해 Convolution을 취함으로써 특징을 추출하며, 영상(image)에서 객체, 얼굴, 장면 인식을 위한 패턴을 찾을 때 특히 유용하게 사용된다. CNN은 다른 신경망과 마찬가지로 입력 계층과 출력 계층, 그리고 그 둘 사이의 여러 개의 은닉 계층으로 구성되며, 가장 일반적인 2가지의 계층으로는 Convolutional 계층, Pooling 계층을 들 수 있다. Convolutional 계층은 입력된 영상을 일련의 Convolution filter에 통과시켜 그 과정에서 특성 맵(feature map)을 출력하여 다음 계층으로 전달한다. Pooling 계층은 데이터의 세로 및 가로 방향의 공간을 줄이는 연산을 수행하며, 이미지 인식 분야에서는 주로 최대 Pooling을 사용한다.

본 프로젝트에서는 CIFAR-10 data set을 이용해 image classification을 위한 CNN 모델을 개발하는 것을 목표로 한다. 또한 그 과정에서 Intermediate feature space와 class activation map의 가시화를 시도하고 그 결과를 분석한다.

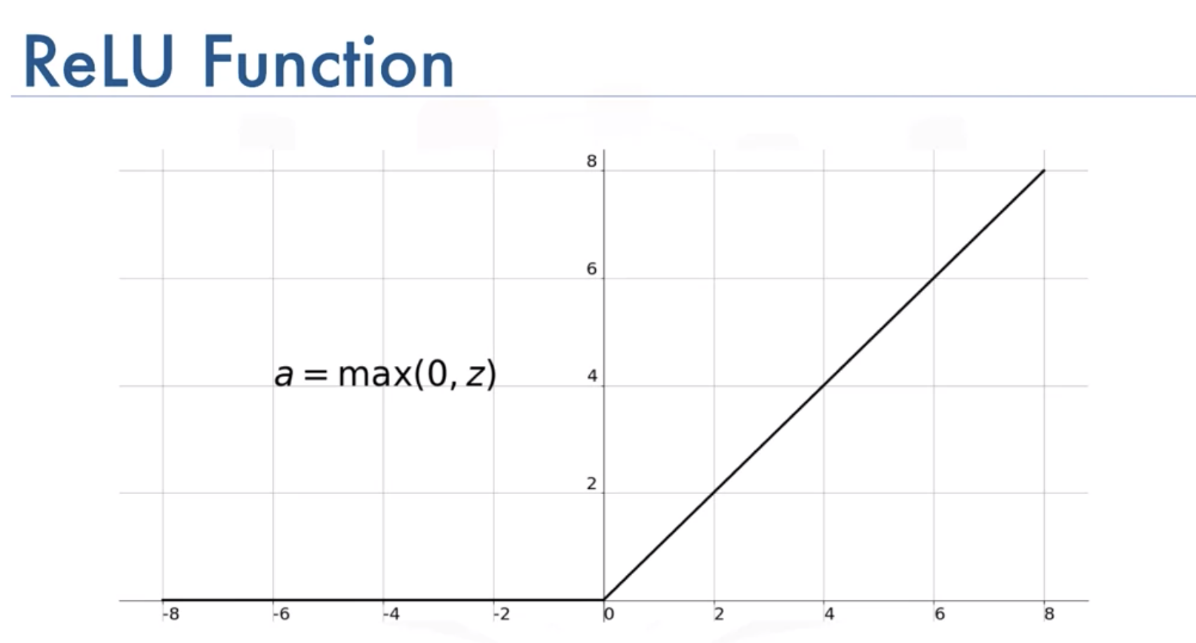
1. Network Design

1

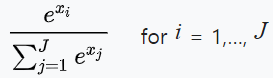
본 chapter에서 신경망 구성에 이용한 활성화 함수, optimizer, 기타 기법 등의 정의와 쓰임새는 다음과 같다.

1. ReLU / softmax

filters를 통해 feature map이 추출되면, CNN에서는 이 특징 맵에 활성화 함수를 적용하여 값을 활성화 시켜준다. 대표적인 활성화 함수로는 시그모이드(sigmoid)와 렐루(ReLU) 함수가 있는데, 시그모이드 함수를 사용하면 레이어가 깊어지면서 BackPropagation이 제대로 동작하지 않는 기울기 소실 문제가 발생하기 때문에 이를 방지하기 위해 ReLU 함수를 주로 사용한다.

[[1]](#footnote-1)

ReLU는 다른 활성화 함수에 비해 모든 뉴런을 동시에 활성화하지 않는다는 특징이 있다. 위의 plot에 따르면 입력이 음수이면 값이 0으로 변환되어 뉴런이 활성화되지 않는다. 즉, 한 번에 몇 개의 뉴런만이 활성화 되기 때문에 네트워크가 효율적으로 구성된다는 장점이 있다. 이러한 이유들로 우리 코드에서도 신경망을 구성할 때 ReLU 활성화 함수를 사용하기로 결정했다.

[[2]](#footnote-2)

softmax 함수는 로지스틱 함수를 다차원으로 일반화한 것으로, K개의 실수로 구성된 벡터 z를 입력으로 받아 입력 숫자의 지수에 비례하는 K개의 확률로 구성된 확률분포를 정규화하는 기능을 한다. 주로 출력층에서 사용하기 때문에 우리 코드에서도 출력층의 활성화 함수는 softmax를 사용하도록 구성했다.

1. Max Pooling

Convolutional 계층을 통해서 feature가 어느 정도 추출이 되었으면, 이 모든 feature들을 모두 학습에 사용할 필요는 없다. 그래서 여기서 추출된 Activation map을 resizing 하여 새로운 layer를 생성하는 작업을 진행하는데, 이것을 Pooling이라고 한다. Pooling의 종류에는 최댓값을 추출하는 Max Pooling, 평균값을 추출하는 Mean Pooling 등이 있는데, 최근 CNN에서는 주로 2\*2 size를 가지는 Max Pooling을 사용하고, 이렇게 하면 입력이 대폭 다운샘플링 되어 계산 효율성을 향상 시킨다는 장점을 가지므로, 우리 코드에서도 Max Pooling 방법을 채택했다.

1. Adam Optimizer

Optimizer는 딥러닝에서 학습 속도를 빠르고 안정적이게 해주는 Optimization을 수행하는 역할을 한다. Optimizer에는 Momentum, RMSProp, Adam 등 다양한 종류가 있지만 CNN에서는 Adam을 주로 사용하여 우리 코드에서도 Adam optimizer를 사용하기로 했다. Adam은 Adagrad와 RMSProp의 장점을 섞어 놓은 optimizer로, 확률적 경사 하강법을 확장한 알고리즘을 사용한다.

1. Data augmentation

Data augmentation 은 기존 데이터의 약간 수정된 사본이나 기존 데이터로부터 새롭게 생성된 합성 데이터를 추가하여 데이터의 양을 증가시키는 것을 의미한다. 딥러닝에서 정규화 역할을 하며 학습을 할 때 over-fitting 문제를 해결하는 데 도움이 된다. 우리 코드에서는 keras의 ImageDataGenerator() 함수를 사용해 약간의 Data augmentation을 수행해준다. 우리 코드에서는 rotation\_range 변수를 통해 회전 범위를 15도로 설정하고, width\_shift\_range와 height\_shift\_range 변수에 원본 가로/세로에 대한 비율값을 0.1로 지정하여 영상을 수평 또는 수직으로 랜덤하게 평행 이동 시켜주었다. 또한 horizontal\_flip 변수를 True로 설정하여, 50% 확률로 이미지를 수평으로 뒤집어주는 augmentation을 활용하였다.

1. Batch normalization

텍스트이(가) 표시된 사진

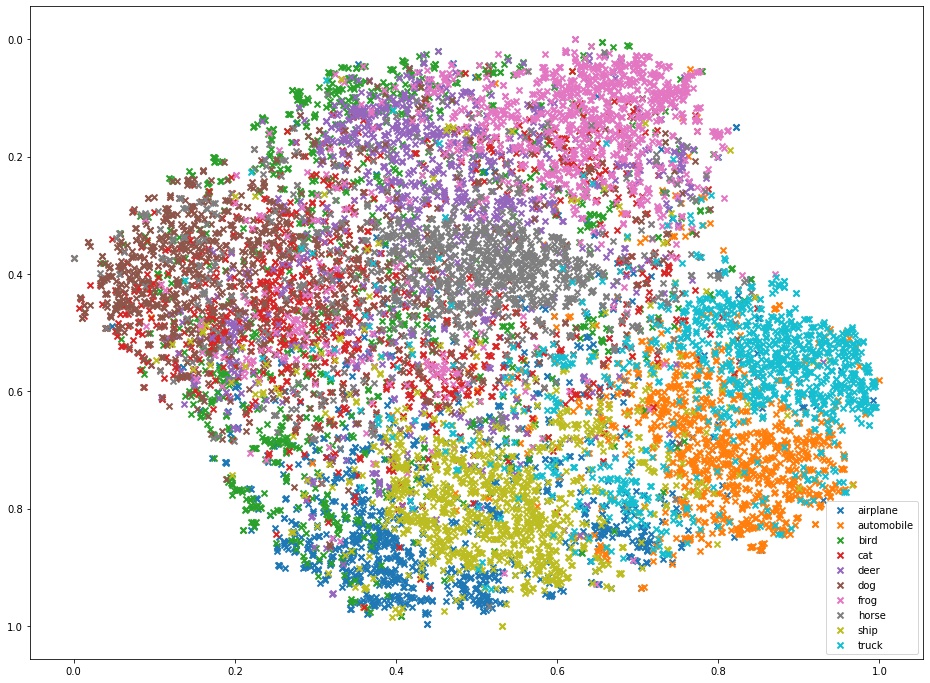
자동 생성된 설명

배치 정규화는 2015년에 발표된 "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift" 논문에서 등장한 개념이다. 일반적으로 정규화를 하는 이유는 학습 속도를 더 높이고 Local optimum에 빠질 가능성을 줄이기 위해서이다. 배치 정규화 논문에서는 학습에서 불안정화가 일어나는 이유를 'Internal Covariance Shift' 라고 주장하고 있는데, 이는 이전 layer의 파라미터 변화로 인해 현재 layer의 입력 분포가 바뀌는 현상인 Covariate Shift 현상이 layer를 통과할 때마다 발생하면서 입력의 분포가 조금씩 변하는 현상을 의미한다.

이 현상을 막기 위해 간단하게 각 layer의 입력의 분산을 평균이 0, 표준편차가 1인 입력값으로 정규화 시키는 방법을 생각해 볼 수 있는데, 이러한 방법을 Whitening이라고 한다. 하지만 이러한 Whitening 방법은 계산량이 많고 일부 파라미터들의 영향이 무시될 수 있다는 약점이 존재한다. 여기서 Whitening의 문제점을 해결하도록 한 트릭이 바로 배치 정규화이다. 배치 정규화는 평균과 분산을 조정하는 과정이 별도의 과정으로 분리된 것이 아닌, 신경망 안에 포함되어서 학습을 할 때 평균과 분산을 조정하는 과정 역시 같이 수행되도록 한다. 우리 코드에서는 keras에 내장된 BatchNormalization() 함수를 간단히 이용해 배치 정규화를 진행하였다.

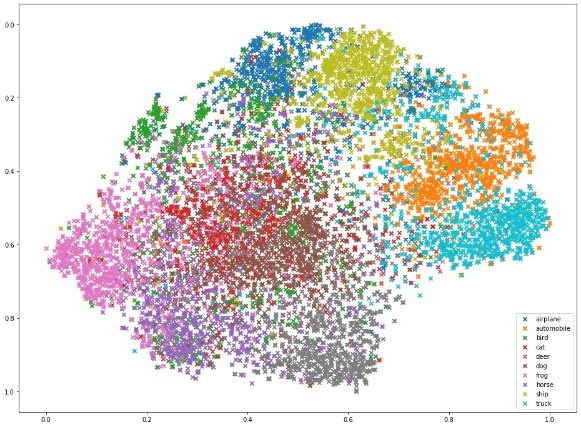
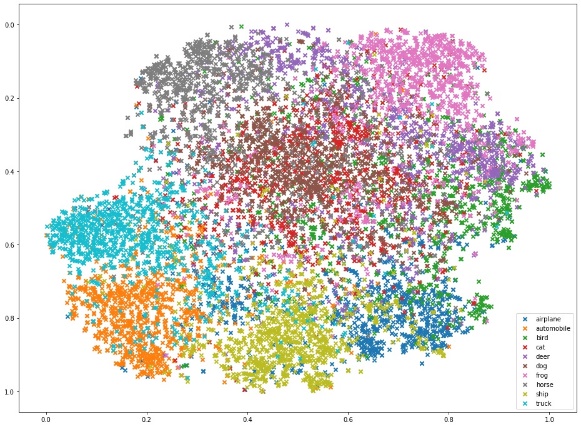
1. Intermediate Feature Space

다음은 학습 과정에서 생성되는 feature space를 저장하여 영상 분류가 어떤 식으로 이루어지고 정확도가 어떻게 향상되는지를 시각화를 통해 살펴보도록 하겠다. 우리는 각 epoch에서 최고 정확도가 갱신될 때마다 모델을 저장한 다음, t-SNE를 이용하여 그 모델에서 분류가 어떤 식으로 진행되었는지를 산점도 그래프로 나타내 보았다.

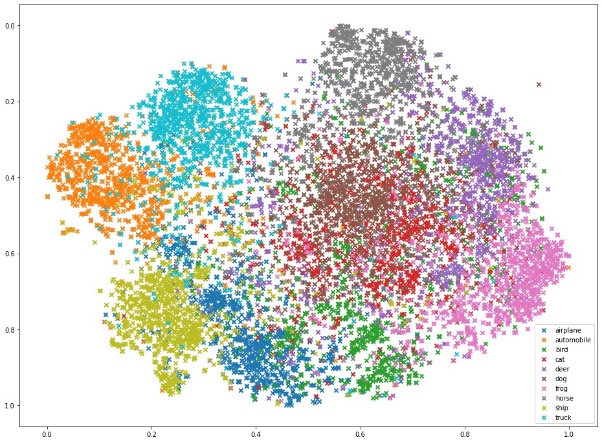
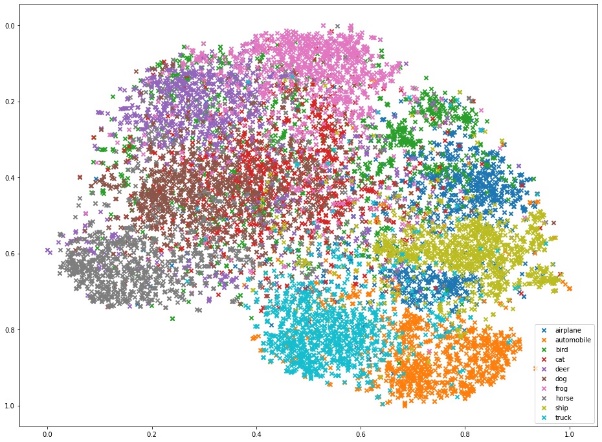


(epoch = 1 / accuracy = 0.5791)

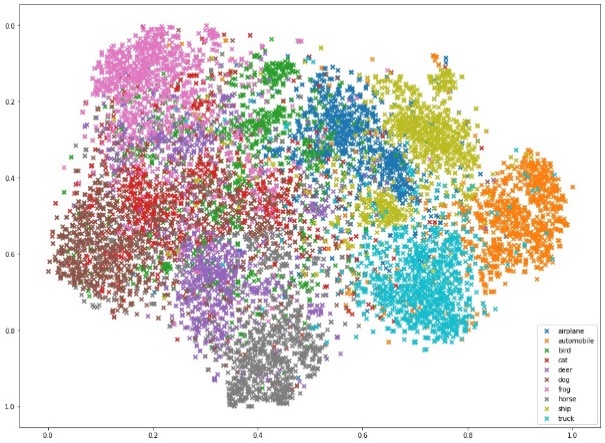
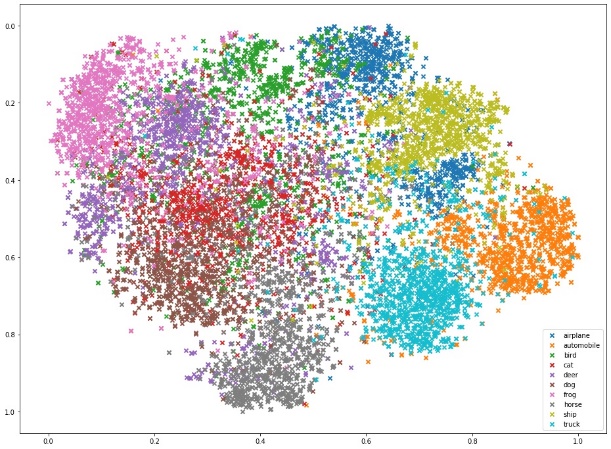
먼저 epoch=1일 때는 정확도가 약 0.5791로, 분류가 위와 같이 진행되었다. 어느 정도의 분류가 이루어졌지만, 아직 정확도가 낮기 때문에 제대로 분류가 이루어지지 않은 부분 역시 발견되었음을 확인할 수 있다.



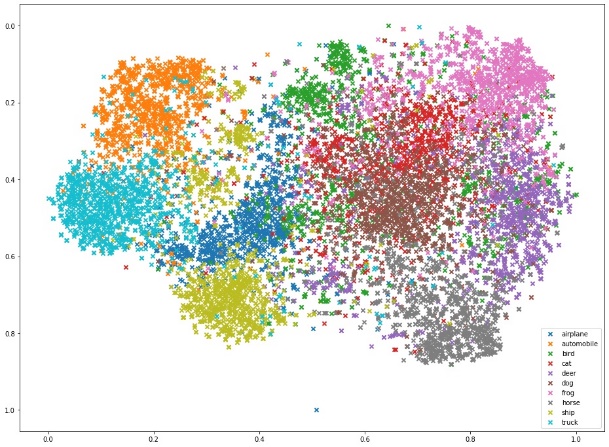
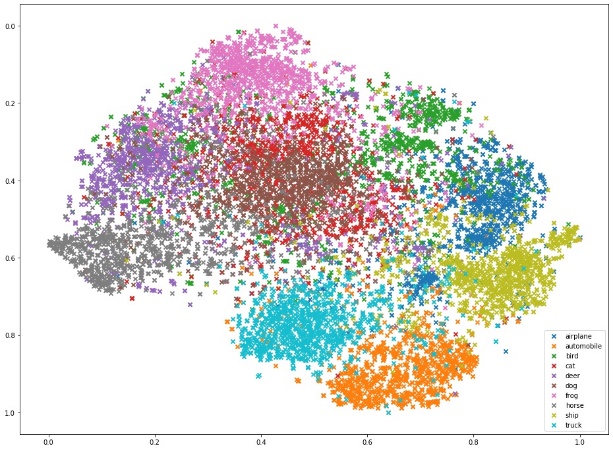
(epoch = 2 / accuracy = 0.6452) (epoch = 3 / accuracy = 0.7320)



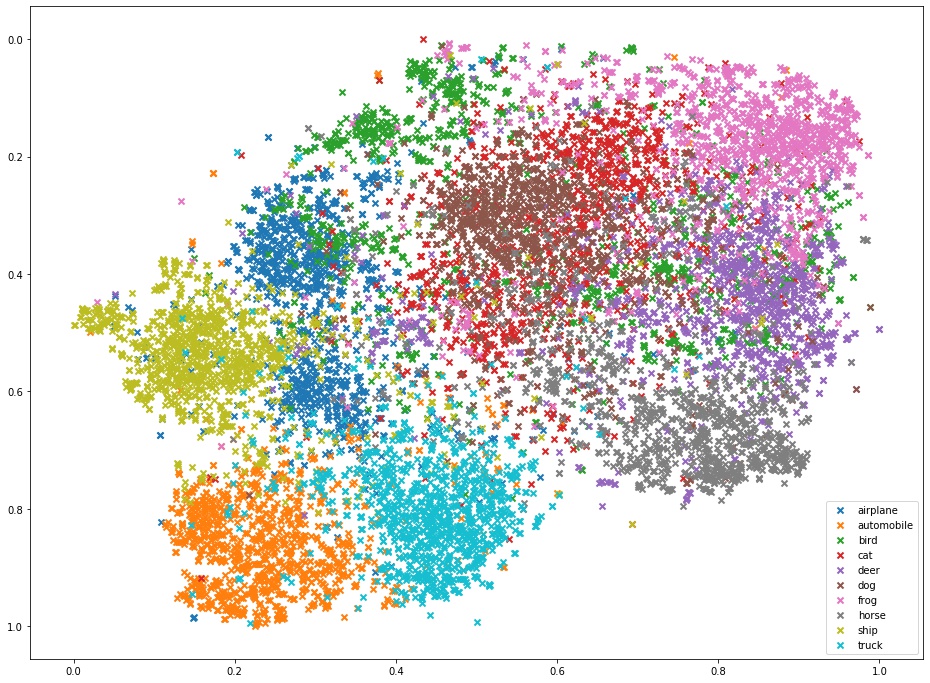
(epoch = 6 / accuracy = 0.7802) (epoch = 7 / accuracy = 0.8009)



(epoch = 8 / accuracy = 0.8093) (epoch = 16 / accuracy = 0.8364)



(epoch = 18 / accuracy = 0.8440) (epoch = 19 / accuracy = 0.8473)

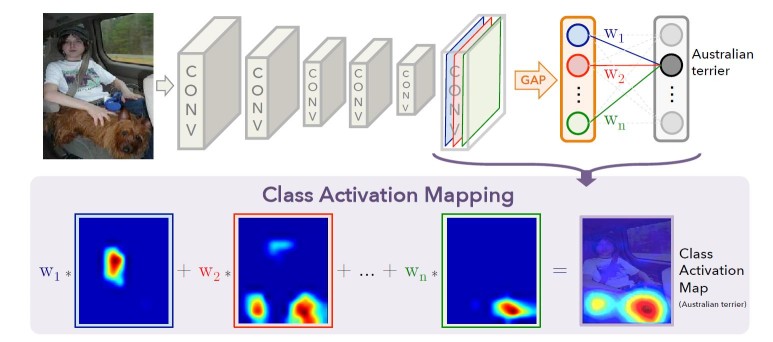


(epoch = 24 / accuracy = 0.8497)

최종적으로는 epoch가 증가하면서 정확도 역시 증가하고, 점점 영상 분류가 더 잘 이루어진다는 것을 시각화를 통해 확인할 수 있다.

1. Class Activation Map

Class Activation Mapping (CAM) 이란 CNN이 특정 클라스 이미지를 그 클라스라고 예측하게 한 그 이미지 내의 위치 정보를 의미한다.

[[3]](#footnote-3)

위 figure 는 CAM 과 CAM 의 네트워크 구조를 보여준다. 우선 기본적인 구조는 Network in Network 과 GoogleNet 과 흡사하다. 하지만 결정적인 차이점이 마지막 Conv Layer를 Fc-Layer 로 Flatten 하지 않고,GAP(Global Average Pooling) 을 통해 새로운 Weigh을 만들어 낸다. 마지막 Conv Layer 가 총 n 개의 channel 로 이루어져 있다면, 각각의 채널들은 GAP 를 통해 하나의 Weight 값으로 나타내고, 총 n 개의 Weight 들이 생긴다. 그리고 마지막에 Softmax 함수로 연결되어 이 Weight 들도 백프롭을 통해 학습을 시키는 것이다.

그러나 우리 연구에서는 cam을 같이 실험하지 않고, 따로 코드를 분리하여 실험을 진행했다. Conv Layer에서 추출한 1개의 샘플 데이터에서 Heat Map을 통해 CAM을 진행했다.

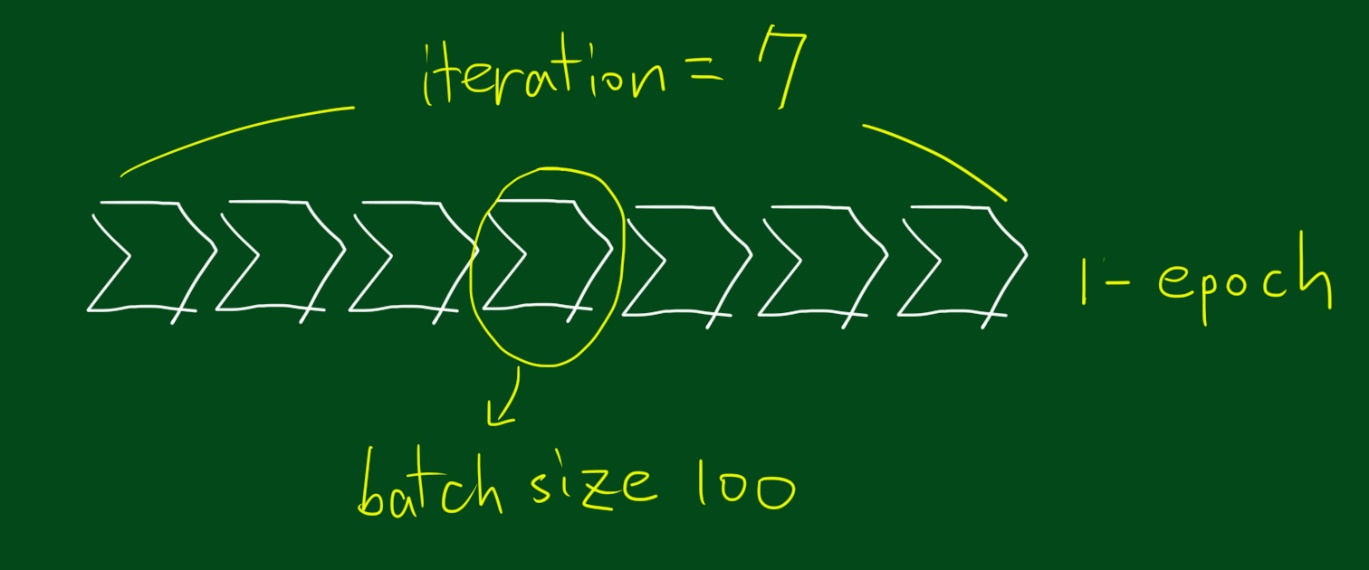


128x128 픽셀에서 32 사이즈의 이미지로 추출한 Heat Map은 위와 같다. 해당 사진을 통해, 이미지의 특징적인 부분에서 높은 값을 가지는 Heat Map을 확인할 수 있다.

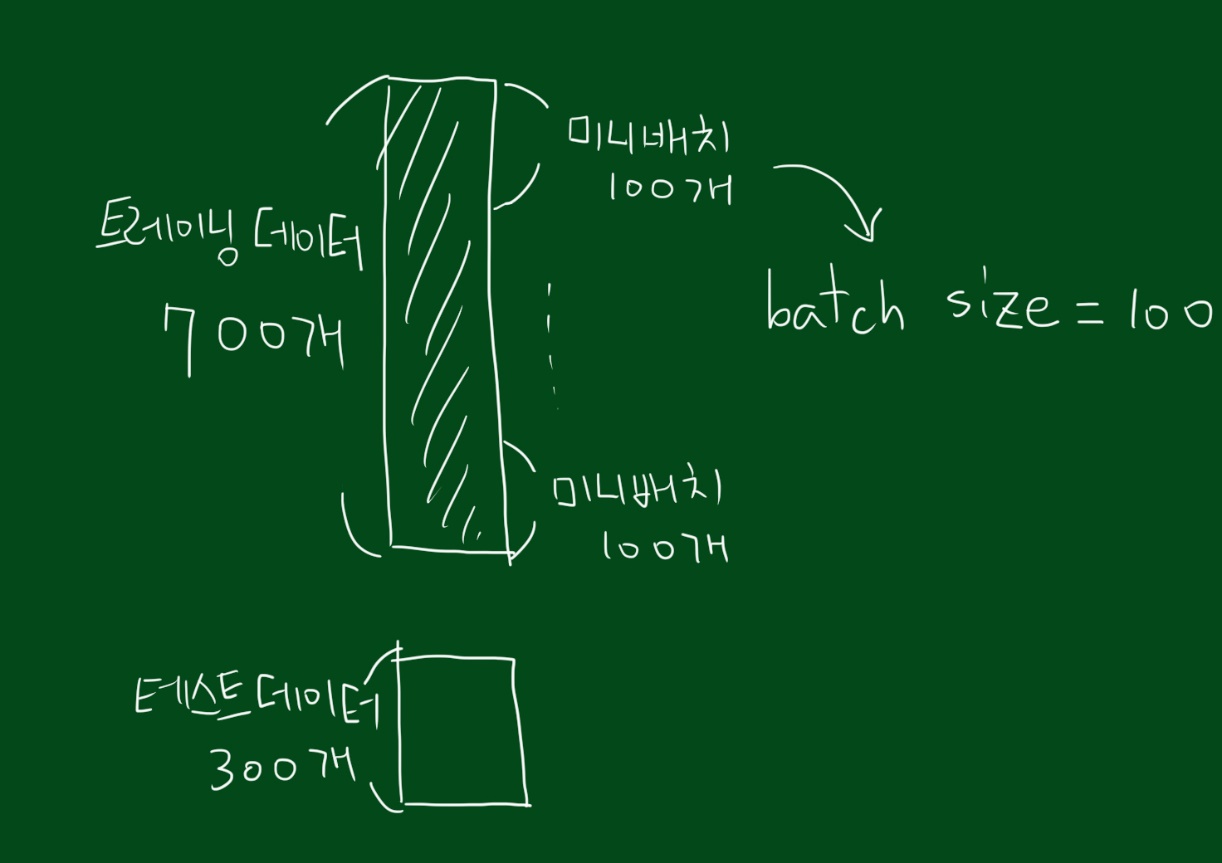
1. Accuracy and Loss

본 연구는 Epoch와 Batch Size을 조정하며 여러 실험을 진행했다.

여기서 Epoch란, 전체 트레이닝 셋이 신경망을 통과한 횟수를 의미한다. 예를 들어, 1-Epoch는 전체 트레이닝 셋이 하나의 신경망에 적용되어 순전파와 역전파를 통해 신경망을 한 번 통과했다는 것을 의미한다.

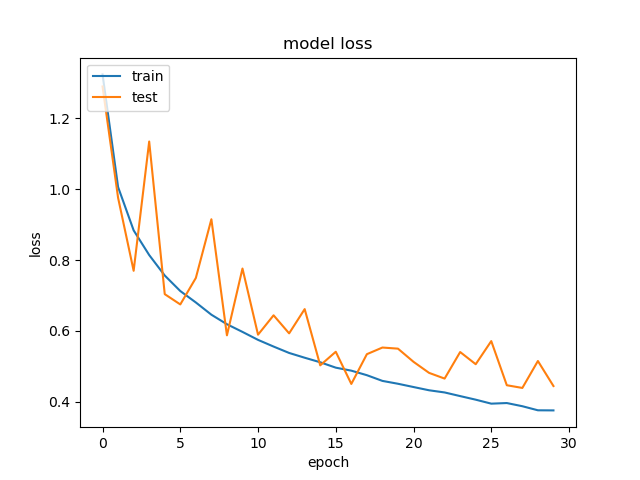
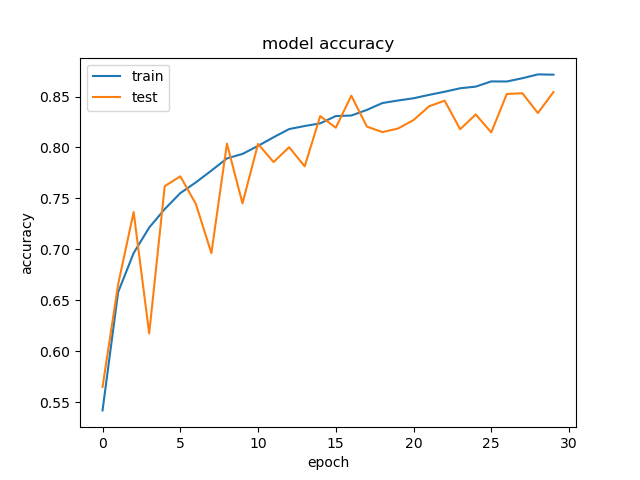
[[4]](#footnote-4)

Batch Size란, 전체 트레이닝 데이터 셋을 여러 작은 그룹으로 나눌 때, 그 한 소그룹의 속하는 데이터 수를 의미한다. 전체 트레이닝 셋을 작게 나누는 이유는 트레이닝 데이터를 통째로 신경망에 넣으면 비효율적인 리소스 사용으로 학습 시간이 오래 걸리기 때문이다.

[[5]](#footnote-5)

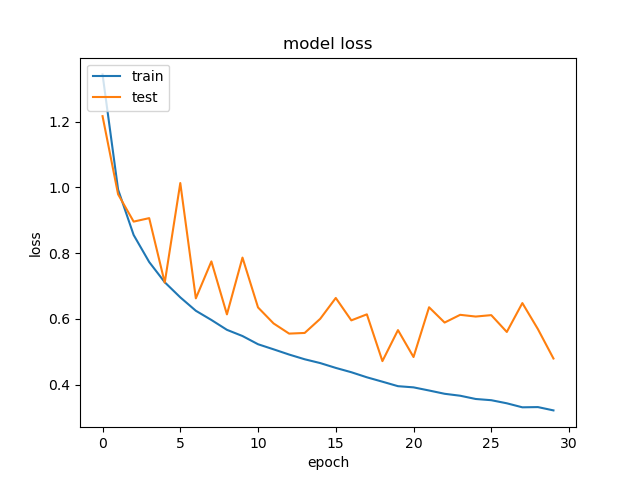
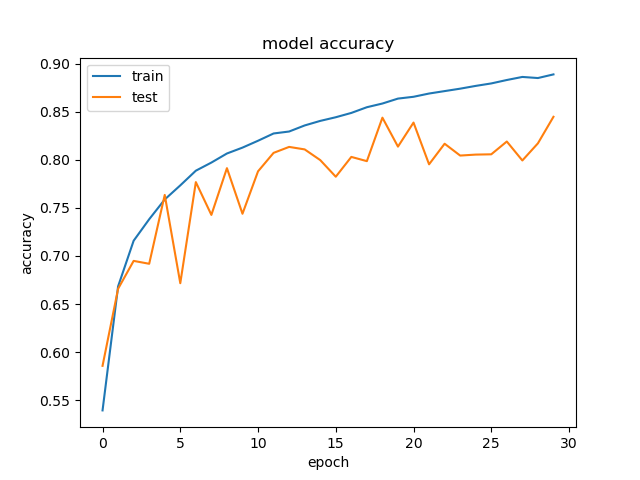
또한 Iteration 역시 존재한다. Iteration란, 1 Epoch를 마치는데 필요한 미니배치의 수를 의미한다. 예를 들어, 700개의 데이터를 100개씩 7개의 미니배치로 나눌 때, 1 Epoch를 위해서는 7-iteration이 필요하면 7번의 파라미터 업데이트가 진행됩니다.

**실험 1)**



먼저 Epoch 30, Batch Size 32일 때를 그래프로 나타냈다. Train 에 대한 accuracy(훈련 정확도)는 Epoch의 증가와 비례하나, test 에 대한 accuracy(검증 정확도)는 16 epoch에서 최고점을 찍은 후, 감소와 증가를 반복하고 있다. 이는 train 모델에서의 과적합이 test 모델에서 일반화하기 어렵다는 점을 보여주고 있다. 즉 불필요한 특징을 배운다고 볼 수 있다. 마찬가지로 Train에 대한 loss(훈련 손실)은 꾸준히 낮아지나, Epoch가 증가함에 따라 누적되는 과적합 요소가 Test에 대한 loss(검증 손실)을 높일 수 있는 요인으로 작용할 수 있다.

**실험 2)**



다음은 Epoch 30, Batch Size 50일 때의 결과를 그래프로 나타냈다. 실험 1과 두드러지게 대조되는 부분은 Train과 Test 선의 간격, 즉 차이가 커졌다는 점이다. 이 실험의 최종 결과값은 다음과 같다.

loss: 0.3219

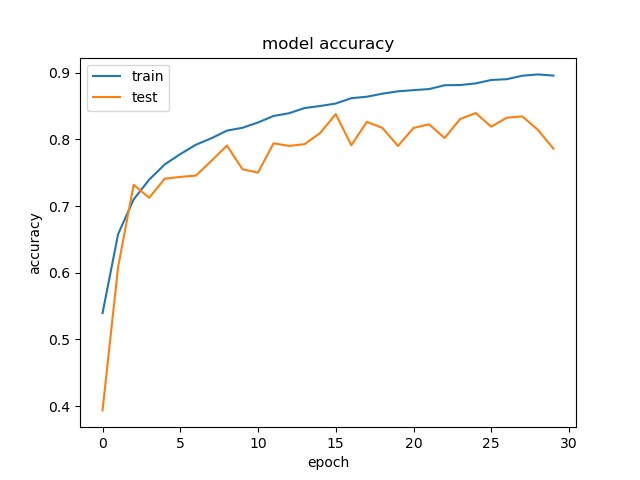
accuracy: 0.8888

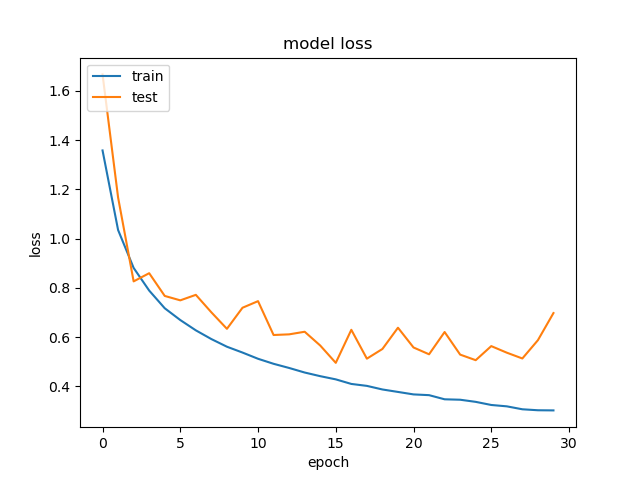
val\_loss(test): 0.4795

val\_accuracy(test): 0.8447

이렇게 검증 데이터의 손실이 높고, 정확도가 낮아진 이유는 실험2의 Batch size트레이닝 데이터를 실험 1보다 높게 설정하였기 때문이다. 이로 인해 비효율적인 리소스 사용으로 학습 시간 역시 Epoch당 평균 211초를 기록한 실험 1과 다르게 실험 2는 Epoch당 평균 282초를 기록했다.

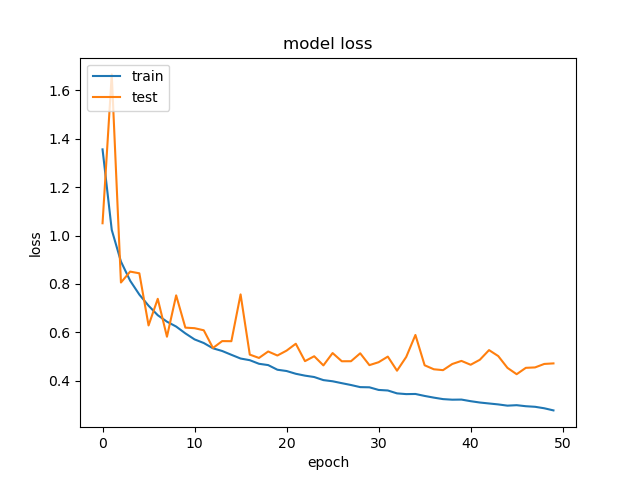
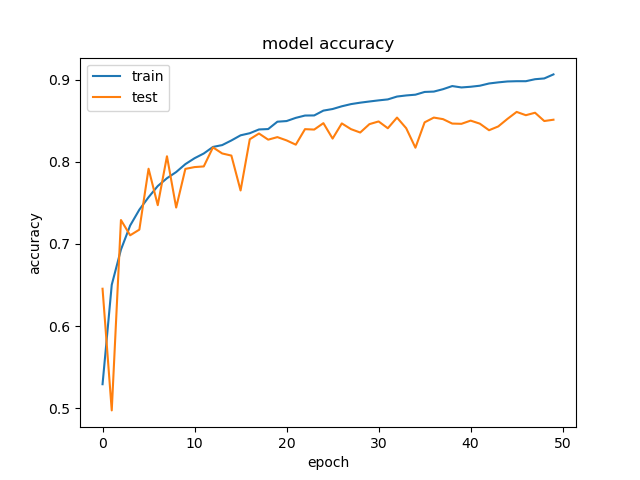
**실험 3)**





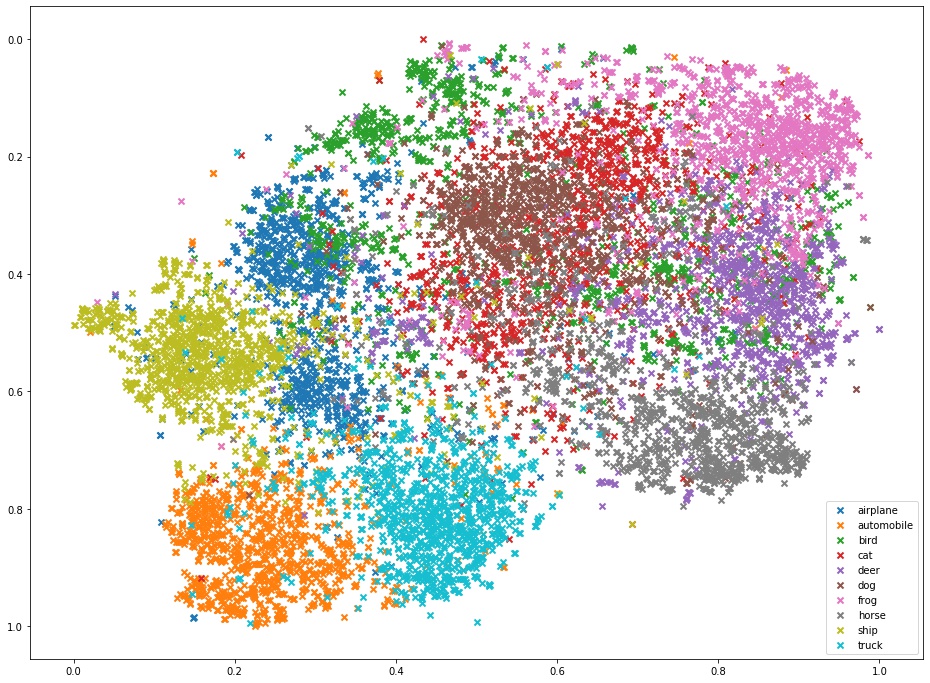
실험 3은 Epoch 30, Batch Size 128로 진행했다. 역시 실험 2와 마찬가지로 검증 데이터의 손실이 높고, 최종 정확도는 차이가 더욱 벌어졌다. Batch Size의 크기가 클수록 비효율적인 리소스 사용이 증가하였다. 즉, 작은 Batch Size의 실험 1이 generalization performance 측면에서 우수한 것으로 나타났다. 따라서 본 연구 실험은 Batch Size 32로 진행하기로 결정했다.

**실험 4)**



실험 4는 기존의 실험과 다르게 Epoch를 50으로 설정했다. Batch size는 32로 진행했다. 실험 4는 실험 1과 비교하여 다음과 같은 특징을 가진다. 먼저 실험 4 역시 Epoch가 진행됨에 따라 train accuracy (훈련 정확도)는 증가하지만 test accuracy(검증 정확도)는 계속된 증가를 보여주지 않는다는 실험 1의 가설을 뒷받침한다. 또한 실험 1과 마찬가지로, 실험 4의 loss 그래프를 통해서 train loss(훈련 손실)의 과적합 요소가 Test에 대한 loss(검증 손실)을 높일 수 있다는 점을 보여준다. 실험 1과 차이가 나는 부분은 최종값에서의 accuracy와 loss 각각의 train 모델과 test 모델이다. 두 모델 사이의 간격이 계속해서 벌어지는데 이 역시도 실험 1의 가설을 뒷받침한다. 따라서 본 실험은 Epoch의 증가가 과적합을 유발하여 일반화에 장애를 준다는 결론을 내렸다. 또한 Epoch 30으로 진행하기로 결정했다.

1. Discussion and Conclusion



텍스트이(가) 표시된 사진

자동 생성된 설명

최종적으로는 epoch = 30으로 코드를 수행하였고, epoch = 24일 때, 정확도가 0.8497로 가장 우수하게 나온 것을 확인할 수 있었다. Epoch가 30일 때보다, 24일 때의 정확도가 더 높은 이유는 네트워크가 훈련 데이터를 너무 잘 모델링하고 검증 데이터로 일반화하지 못하는 부분에 과적합이 있기 때문이다. 반면에 Epoch가 낮을 때는 모델이 학습 데이터의 기본 패턴을 학습하지 못한다. 따라서 모델의 학습을 위해서 Epoch를 적절히 조율해가며 코드를 수행할 필요가 있다는 결론에 도달했다. 다만, 본 연구는 실험을 여러 번 진행해보지 않았기 때문에, 가설에 뒷받침할 근거가 부족하다는 한계점이 있다. 또한 CAM 구조를 따로 분리하여 구현했다는 한계점 역시도 존재한다. 이를 극복하기 위해서는 다양한 실험과 여러 변수들의 설정을 조합해볼 필요가 있다. 본 연구는 Deep Learning의 CNN 구조에서 각 변수가 가지고 있는 의미와 그러한 구현 방법과 원리를 최대한 많이 활용하려고 했다.

1. References

"Convolutional Neural Network", MathWorks, accessed June 5, 2021, <https://kr.mathworks.com/discovery/convolutional-neural-network-matlab.html>

"Neural Networks", MathWorks, accessed June 5, 2021, <https://kr.mathworks.com/discovery/neural-network.html>

Yaqub, M., Jinchao, F., Zia, M. S., Arshid, K., Jia, K., Rehman, Z. U., & Mehmood, A. (2020). State-of-the-Art CNN Optimizer for Brain Tumor Segmentation in Magnetic Resonance Images. Brain sciences, 10(7), 427. <https://doi.org/10.3390/brainsci10070427>

Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). PMLR.

1. https://medium.com/@toprak.mhmt/activation-functions-for-deep-learning-13d8b9b20e [↑](#footnote-ref-1)
2. https://en.wikipedia.org/wiki/Softmax\_function [↑](#footnote-ref-2)
3. <https://www.programmersought.com/article/34203914156/>, 사진 출처 [↑](#footnote-ref-3)
4. <https://losskatsu.github.io/machine-learning/epoch-batch/#3-epoch%EC%9D%98-%EC%9D%98%EB%AF%B8>, 사진 출처 [↑](#footnote-ref-4)
5. <https://losskatsu.github.io/machine-learning/epoch-batch/#3-epoch%EC%9D%98-%EC%9D%98%EB%AF%B8>, 사진 출처 [↑](#footnote-ref-5)