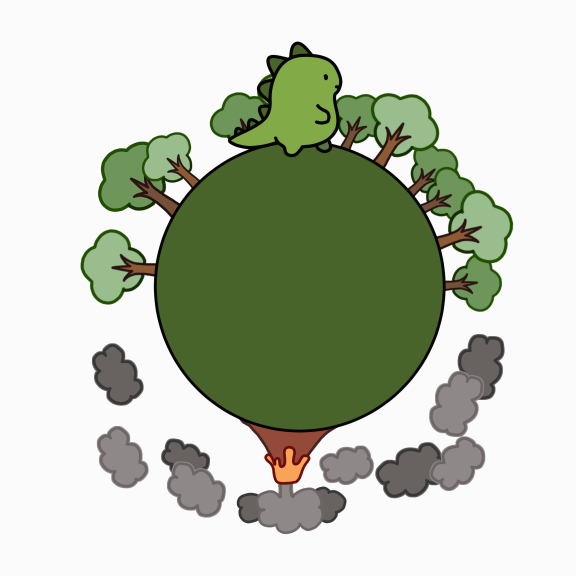
이미지 분류 AI model 개발 프로젝트

기간	2023.02.20 ~ 2023.04.07
기술스택	FastAPI Flutter Jupyter Notebook Python Pytorch
담당역할	AI model
프로젝트 개요	AI model을 통해 공룡을 찾고 정보를 알려주며, 그림을 그리면 해당 그림과 유사한 공룡을 추천해줍니다



✔️구현 사항

AI model

1. model 선정 이유

- 신경망
  - CNN(Convolutional Neural network) - 이미지처리에 주로 사용
  - RNN(Recurrent Neural network) - 자연어 처리에 주로 사용

- CNN 모델 비교

Model	val_Accuracy	Accuracy	Training time(sec)
DenseNet201	1.0000	1.0000	49.20
DenseNet121	1.0000	0.9964	1197.90
ResNet152V2	0.9919	1.0000	58.44
ResNet50V2	0.9919	1.0000	21.12
MobileNetV2	0.9960	0.9964	16.69
Inception	0.9798	0.9964	25.27
Xception	1.0000	0.9964	32.77
VGG16	0.9677	0.9674	46.58
MobileNetV3Large	0.5806	0.5326	18.55

- 모델 선정 기준
  - 6주 간 프로젝트 내에 만들 수 있는 제일 완성도 높은 모델
  - 고려 사항
    1. 학습 시간
    2. 정확도
    3. 모델의 구조

    ⇒ **ResNet50**

CNN의 문제점

- [논문출처] 'Deep Residual Learning for Image Recognition’
- CNN 모델의 문제점
  - 신경망이 깊어질 수록 더 정확한 예측을 할 것이라고 예상
  - CNN(AlexNet, VGGNet) 모델은 layer가 깊어질수록 기울기 소실 문제가 발생

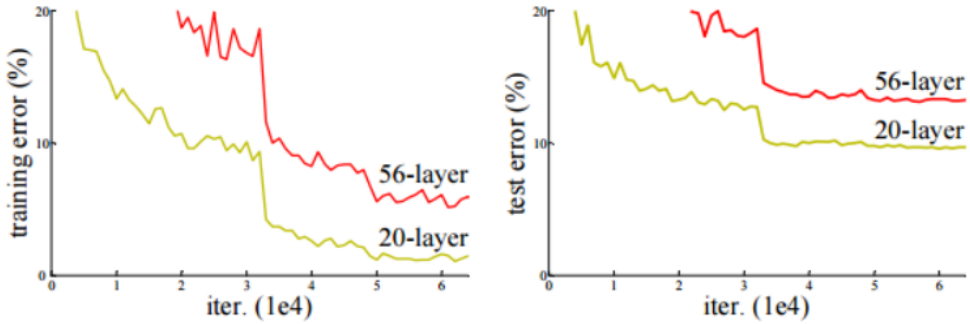


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

- 20-layer plain network가 50-layer plain network보다 더 낮은 train error와 test error
  - degradation 문제 : 기울기 소실에의해 발생
- 기울기 소실 : 신경망이 깊을 때, 작은 미분값이 여러번 곱해지면 0에 가까워지는 문제가 발생하여 가중치 갱신이 불가능해지는 상황
- 오차역전파(back propagation), 그래디언트(gradient)기반의 학습 방법은 모두 이런 문제를 가지고 있습니다. 왜냐면 오차역전파나 그래디언트 방법을 쓰게 되면 신경망의 가중치(weight)를 업데이트하게 되는데, 업데이트 하게 되는 과정에서 gradient 자체가 매우 작아지는 vanishing 문제가 생깁니다.
  - 역전파
    - 인공지능을 학습시키기 위한 일반적인 알고리즘 중 하나
    - 내가 뽑고자 하는 타겟값과 실제 모델이 계산한 output이 얼마나 차이가 나는지 구한 후 그 오차값을 다시 뒤로 전파해가면서 각 노드가 가지고 있는 변수들을 갱신하는 알고리즘

ResNet

- ResNet (Residual neural network)
  - 깊은 신경망을 훈련할 때 발생하는 그레이디언트 소실(gradient vanishing) 문제를 해결하기 위해 개발
    - 잔차를 더해 해결

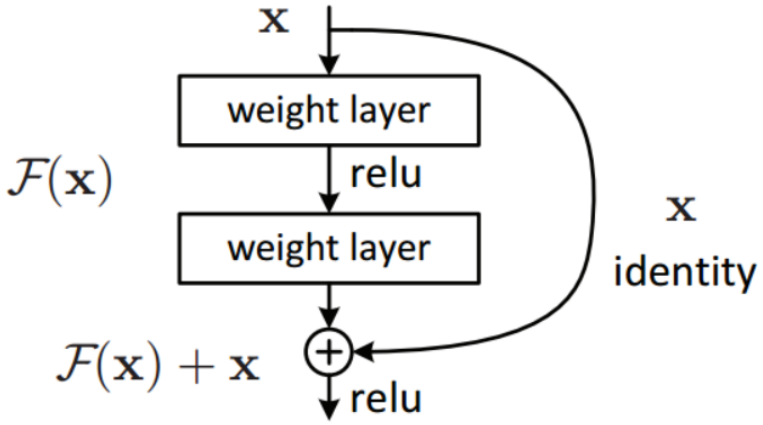


Figure 2. Residual learning: a building block.

- 입력값 x가 주어졌을 때
  - 가중치 레이어를 통과하여 얻은 F(x)와
  - 레이어를 거치지 않고 그대로 건너 뛴(identity mapping) 입력값 x(잔차)를 더해 입력값이 잊혀지는 vanishing gradient problem 해결
  - `y = F(x)+x` 로 가장 간단한 형태의 ResNet 구조

- Residual Block
  - Convolution layer
    - 이미지 처리에 사용되는 필터(filter) 기반의 연산
    - 입력 이미지와 필터를 합성곱하여 출력값을 계산하는 레이어
    - 입력 이미지의 특징을 추출
  - Batch Normalization
    - 입력값을 정규화
    - 각 레이어의 입력값의 분포를 안정화
    - 학습 과정에서 발생하는 기울기 소실(Gradient vanishing) 문제를 해결
    - Batch Normalization은 가중치 초기화를 간단하게 할 수 있으며, 학습 속도를 높일 수 있는 장점
  - ReLU(Recified Linear Unit)
    - 활성화 함수 중 하나
    - 입력값이 0 이하인 경우는 0

    - 0 이상인 경우는 입력값을 그대로 출력하는 함수
    - 비선형성을 추가하여 딥러닝 모델의 표현력을 높임

2. 손실 함수(loss function)와 최적화 알고리즘(Optimizer) 선택 이유

- 손실 함수 : 머신 러닝의 분류 모델이 얼마나 잘 수행되는지 측정하기 위해 사용되는 지표
    - 목표 : 모델의 손실을 가능한 0에 가깝게 만드는 것
    - 모델의 성능과 학습에 영향을 미치는 요소로 데이터셋과 모델에 따라 적절한 손실 함수 선택 필요

    1. 평균 제곱 오차(Mean Squared Error, MSE)
        - 회귀(regression) 문제에서 사용
        - 가장 일반적으로 사용되는 손실 함수
        - 실제 값과 예측 값 간의 제곱 오차를 평균하여 계산
        - 모델이 예측한 값이 정답과 얼마나 멀리 떨어져 있는지를 나타내는 지표로 사용
    2. **교차 엔트로피 손실(Cross Entropy Loss) << 선택**
        - **분류(classification) 문제에서 사용 ⇒ 다중 클래스 분류 문제에 권장**
        - 예측 값과 실제 값이 서로 다른 경우, 손실 값이 매우 크게 측정되고, 예측 값과 실제 값이 같은 경우에는 손실 값이 매우 작게 측정
        - 이러한 특징으로 인해 교차 엔트로피 손실은 모델이 정확하게 예측하는 방향으로 학습되도록 도움
    3. 이진 교차 엔트로피 손실(Binary Cross Entropy Loss)
        - 이진 분류 문제에서 사용
        - 예측 값이 1일 때와 0일 때의 손실 값을 계산하여 평균을 내는 방식으로 동작합니다.

- 교차 엔트로피 손실
    - 딥 러닝에서 교차 엔트로피 손실(Cross Entropy Loss)은 모델이 예측한 확률 분포와 실제 라벨의 분포 간의 차이를 계산하여 모델의 예측이 얼마나 정확한지를 측정하는 손실 함수입니다.
    - 예측에 대해 가능한 모든 값이 저장(다중클래스에 사용되는 이유)

        - ex) 동전 던지기에서 특정 값(odd)을 찾는 경우 해당 정보가 0.5와 0.5(앞면과 뒷면)에 저장
        - 반면Binary Cross Entropy Loss는 하나의 값만 저장
            - 즉, 0.5만 저장하고 다른 0.5는 다른 문제에서 가정
            - 첫 번째 확률이 0.7이면 다른 하나는 0.3이라고 가정
    - log는 자연상수를 밑으로 하는 로그
    - 모델이 1이라고 정확한 예측을 하는 경우 손실함수의 값은 0이 되는 구조

    - 분류 문제에서 사용되며, 다중 클래스 분류와 이진 분류에 모두 적용
        - 다중 클래스 분류에서는 softmax 함수를 사용하여 각 클래스의 확률을 계산
        - 이진 분류에서는 시그모이드 함수를 사용하여 이진 분류 문제에서 확률을 계산
    - 교차 엔트로피 손실 함수의 수식

$$H(p, q) = -\sum_{y} p(y) \log q(y)$$

        - 여기서 p는 실제 라벨의 분포를 나타내며, q는 모델이 예측한 확률 분포
            - 모델이 예측한 확률 분포 q와 실제 라벨의 분포 p 사이의 차이를 계산
            - 이 값이 작을수록 모델의 예측이 정확하다는 것을 의미

- 최적화 : 손실함수를 최소화하는 파라미터(Weight)를 찾는 과정
    - 목표 : 손실함수 최소화
    - 경사하강법
        - 경사 하강법은 전체 데이터셋을 사용하여 손실 함수(Loss Function)의 기울기(Gradient)를 계산
        - 이를 사용하여 파라미터를 업데이트
        - 대규모 데이터셋에서는 전체 데이터셋을 사용하여 기울기를 계산하는 것이 시간과 메모리 측면에서 매우 비효율적

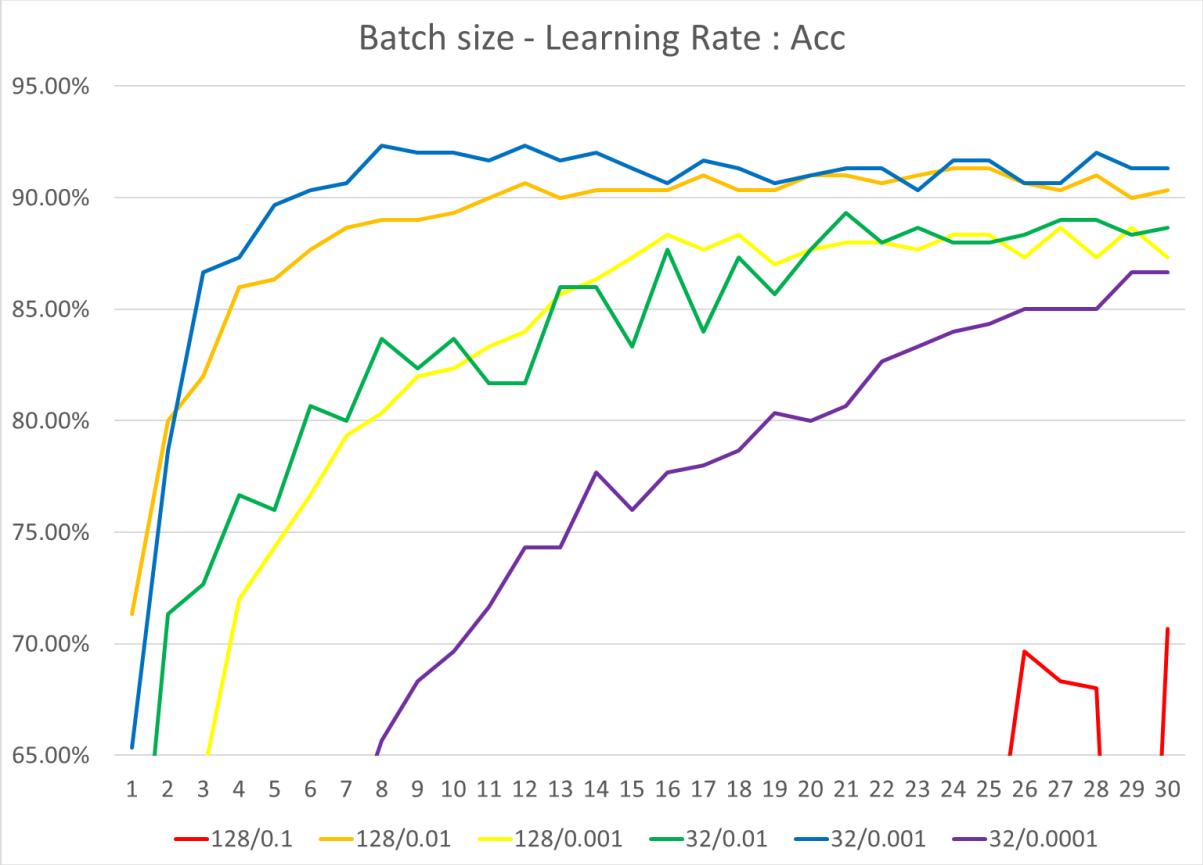
1. **SGD(Stochastic Gradient Descent) ⇒ 대규모 데이터를 학습하고 하이퍼 파라미터의 의미 파악을 위해 선택**
    - 경사하강법(Gradient Descent)의 한 종류
        - 미니배치(mini-batch)라고 불리는 작은 데이터셋을 무작위로 선택하여 기울기를 계산
        - 이를 사용하여 파라미터를 업데이트
        - 반복적으로 수행하여 전체 데이터셋에 대한 손실 함수의 최솟값 찾기
    - 장점
        - 미니 배치를 사용하기 때문에 계산 비용이 낮고, 대규모 데이터셋에서도 적용 가능
        - SGD는 지역 최솟값(local minimum)에 빠지는 것을 피할 가능성이 높음
    - 단점
        - 무작위로 선택한 미니배치에 대한 기울기는 전체 데이터셋의 기울기와 다르기 때문에, SGD는 수렴 속도가 느림
2. Adam(Adaptive Moment Estimation)
    - 경사 하강법(Gradient Descent)의 한 종류
        - SGD(Stochastic Gradient Descent)와 비교하여 빠른 수렴과 더 나은 성능
        - 학습률을 자동으로 조절 가능
    - 장점
        - Adam은 SGD보다 빠르게 수렴하며, 학습률을 직접 설정하지 않아도 되기 때문에 사용이 간편
        - 그래디언트에 따라 학습률과 모멘텀 등 하이퍼파라미터를 자동으로 조절하여 안정적으로 학습
    - 단점
        - Adam은 모든 그래디언트를 메모리에 저장해야 하기 때문에, 메모리 사용량이 높음
        - Adam은 수식이 복잡하기 때문에, SGD보다 계산량이 많음

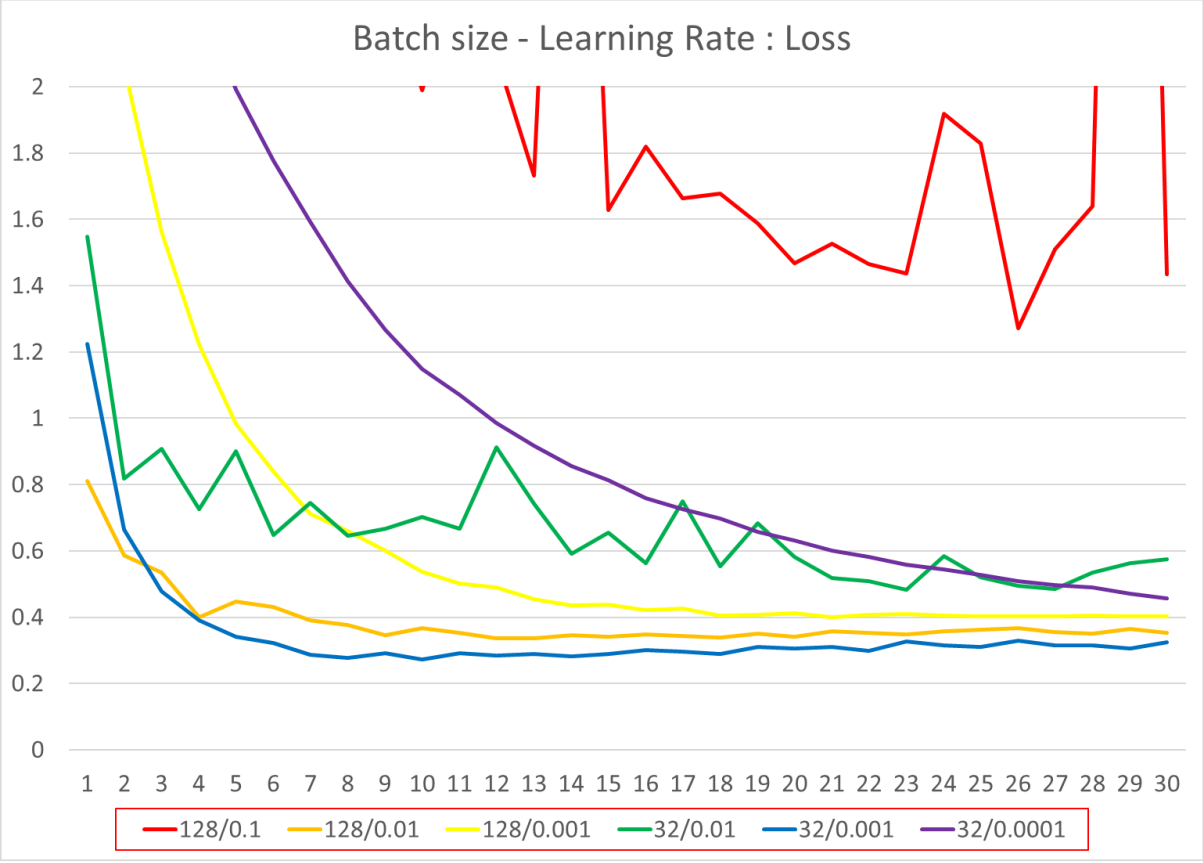
3. 하이퍼 파라미터 선정

- 모델의 성능에 큰 영향
- 적절한 하이퍼파라미터 값은 모델 성능을 최대로 끌어올리기 위해 매우 중요
- 배치크기(batch_size)
    - 배치 크기란 모델이 한 번의 학습에 사용하는 데이터의 양입니다. 작은 배치 크기는 학습 속도가 빠르지만, 모델 파라미터가 덜 안정적이며 노이즈가 많은 경향이 있습니다. 큰 배치 크기는 학습 속도가 느리지만, 모델 파라미터가 더 안정적이며 더 작은 노이즈를 가지는 경향이 있습니다.
- 학습률(learning rate)
    - 학습률은 모델이 각 반복에서 얼마나 많은 파라미터를 업데이트할지 결정합니다. 학습률이 너무 크면 학습 과정에서 발산할 수 있습니다. 학습률이 너무 작으면 학습 속도가 매우 느려지며 최적점까지 수렴하는 데에 시간이 매우 오래 걸릴 수 있습니다.
- 에포크(epoch)
    - 에포크는 모델이 학습 데이터 세트 전체를 한 번 학습하는 것을 의미합니다. 에포크가 많을수록 모델은 학습 데이터를 더 많이 볼 수 있지만, 오버피팅(overfitting)이 발생할 수 있습니다. 반면, 에포크가 적으면 모델이 학습 데이터를 적게 보게 되어 모델 성능이 제한될 수 있습니다. 적절한 에포크 수를 찾는 것은 모델의 성능을 최대화하기 위해 매우 중요합니다.

⇒ **그리드 탐색(Grid Search): 사용 가능한 하이퍼파라미터 조합을 모두 시도해보고 가장 좋은 조합을 선택**

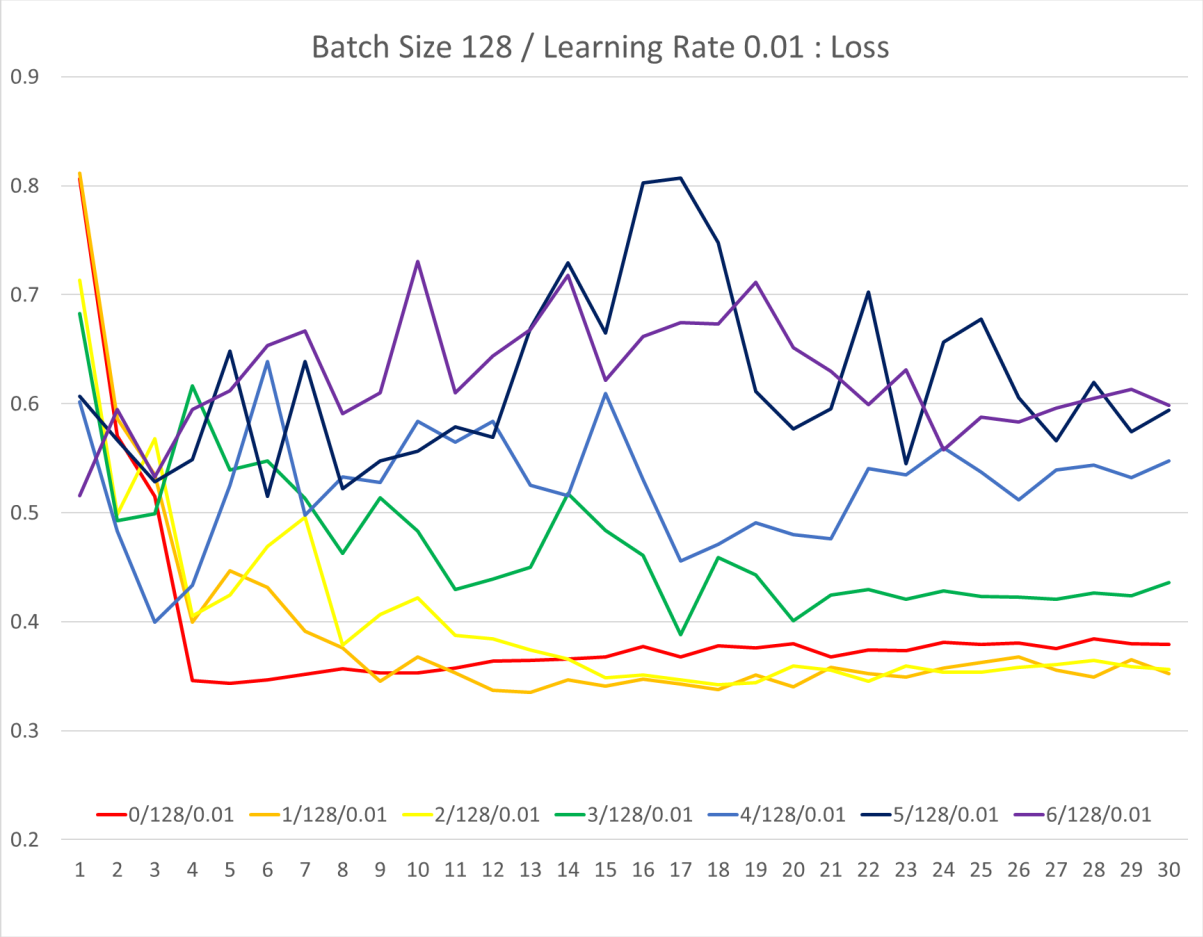
1. 배치크기 학습률 선택

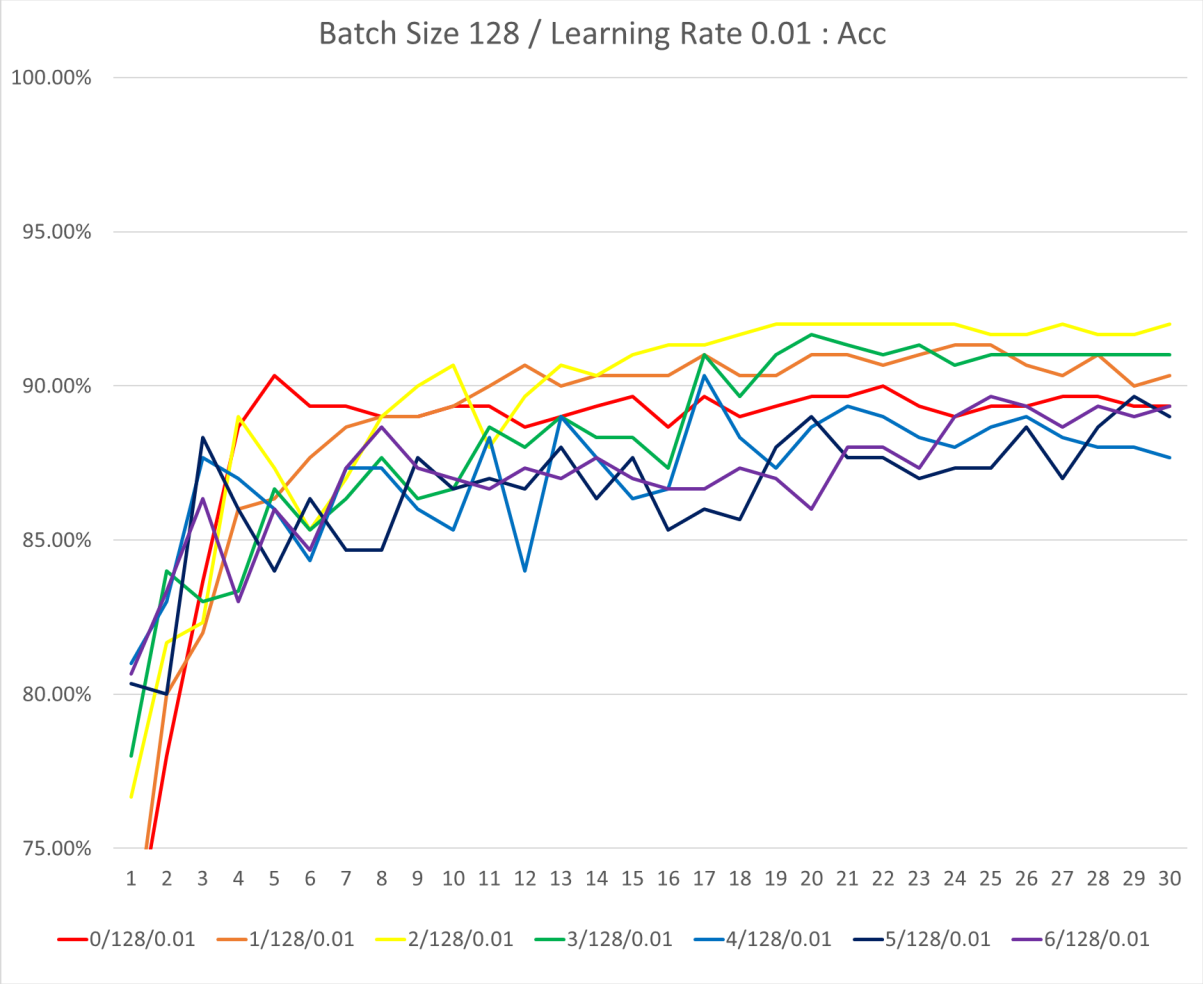

Batch size - Learning Rate : Acc
95.00%
90.00%
85.00%
80.00%
75.00%
70.00%
65.00%
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
128/0.1
128/0.01
128/0.001
32/0.01
32/0.001
32/0.0001


Batch size - Learning Rate : Loss
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
128/0.1
128/0.01
128/0.001
32/0.01
32/0.001
32/0.0001

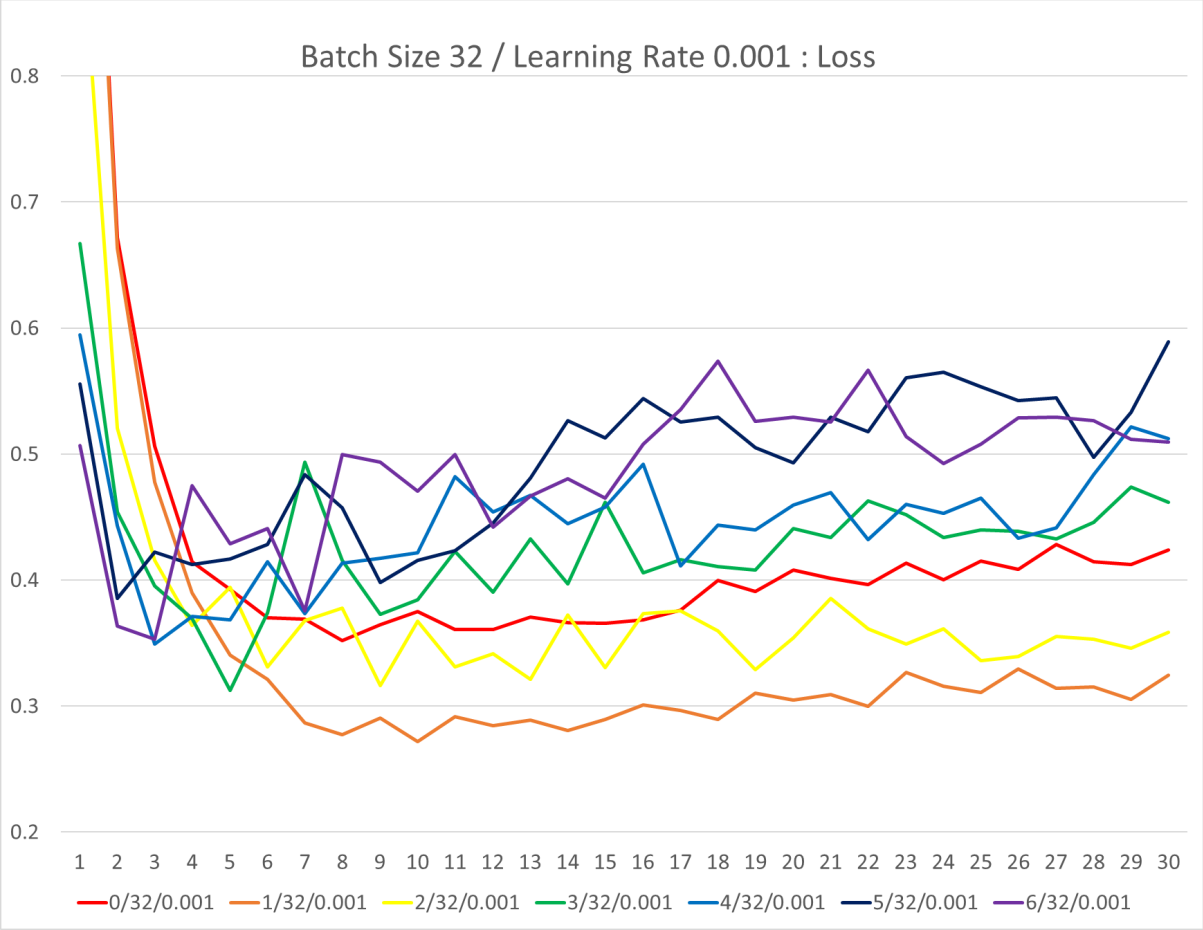
- 정확도 가장 좋은 조합
    - 10 epoch
    1. 배치사이즈 32 학습률 0.001
    2. 배치사이즈 128 학습률 0.01
- 손실율 가장 좋은 조합
    - 10 epoch
    1. 배치사이즈 32 학습률 0.001
    2. 배치사이즈 128 학습률 0.01

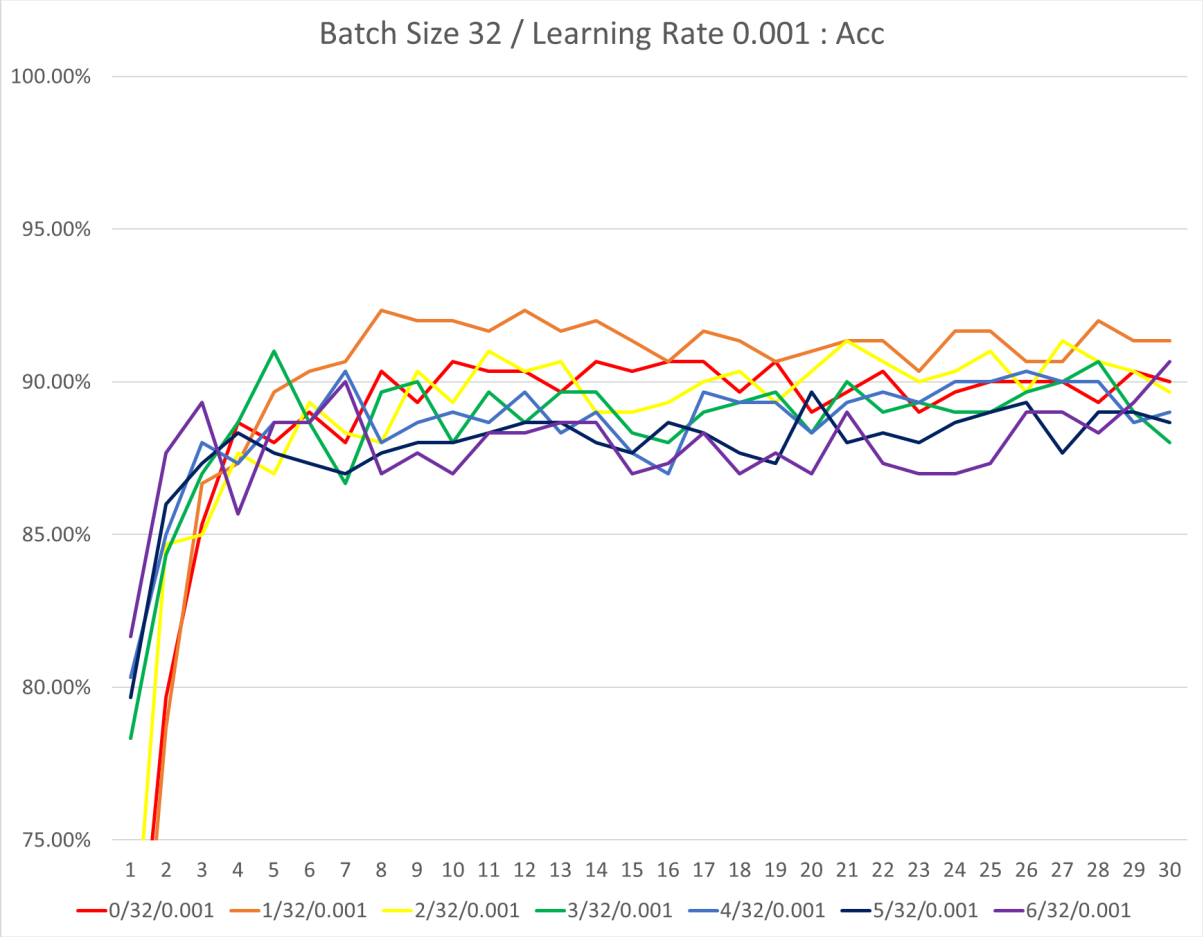
2. augmentation의 효율 확인
    - augmentation / batch_size / learning rate
        - augmentation
            - 0 : 증강 X
            - 1 : 좌우 증강(x2)
            - 2 : 사진 촬영 시 일어날 상황 고려한 증강(x2) 후 좌우 증강(x2)
            - 3 : 사진 촬영 시 일어날 상황 고려한 증강(x4) 후 좌우 증강(x2)
            - 4 : 사진 촬영 시 일어날 상황 고려한 증강(x6) 후 좌우 증강(x2)
            - 5 : 사진 촬영 시 일어날 상황 고려한 증강(x8) 후 좌우 증강(x2)
            - 6 : 사진 촬영 시 일어날 상황 고려한 증강(x10) 후 좌우 증강(x2)


Batch Size 128 / Learning Rate 0.01 : Loss
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0/128/0.01
1/128/0.01
2/128/0.01
3/128/0.01
4/128/0.01
5/128/0.01
6/128/0.01
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30



- 배치사이즈 128학습률 0.01일 때 가장 좋은 증강 횟수
  1. 1 / 128 / 0.01
  2. 2 / 128 / 0.01


Batch Size 32 / Learning Rate 0.001 : Loss
0.8
0.7
0.6
0.5
0.4
0.3
0.2
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
0/32/0.001
1/32/0.001
2/32/0.001
3/32/0.001
4/32/0.001
5/32/0.001
6/32/0.001



- 배치사이즈 32학습률 0.001일 때 가장 좋은 증강 횟수
  1. 1/32/0.001
  2. 2/32/0.001

3. 최적모델 구하기

최적모델 구하기

- augmentation / batch_size / learning rate 별로 학습 시켰을 때 가장 낮은 손실 값

	1/128/0.01	1/32/0.001	2/32/0.001	2/128/0.01
1	0.26	0.322	0.2921	0.4165
2	0.3137	0.3221	0.3074	0.344
3	0.3108	0.3645	0.315	0.3898
4	0.3166	0.3285	0.3043	0.3845
5	0.3444	0.3472	0.319	0.3868

6	0.3361	0.3094	0.3369	0.3931
7	0.3176	0.2926	0.3128	0.3171
8	0.3559	0.2871	0.3014	0.371
9	0.2983	0.3502	0.355	
10	0.2983	0.3502	0.355	
11	0.3344	0.3466	0.3238	
12	0.3432	0.3475	0.2986	
13	0.3428	0.3334	0.3426	
14	0.2614	0.3285	0.3025	
15	0.3383	0.3193	0.2952	
16	0.33	0.3447	0.3237	
17	0.3461	0.3248	0.3195	
18	0.3413	0.3108	0.3394	
19	0.3254	0.3363	0.3229	
20	0.3221	0.3253	0.3309	
21	0.3431	0.3433	0.3491	
22	0.3557	0.343	0.3538	
23	0.3396	0.3214	0.3253	
24	0.3193	0.3258	0.3235	
25	0.3209	0.321	0.3034	
26	0.3114	0.3384	0.3531	
27	0.2723	0.3437	0.3108	
28	0.3114	0.3384	0.3531	
29	0.3828	0.3215	0.356	
30	0.3339	0.322	0.3114	
	0.32423667	0.33031667	0.32458333	0.37535

- 평균 loss : 0.3386
- 공룡 사진 분류 최적 모델
  - augmentation 2
  - batch_size 128
  - learning_rate 0.01 * 10 epoch 0.005 * 1 epoch

- Cross Entropy Loss
- optimizer SGD
- [Result] Loss_ 0.2491 Acc_ 93.0000% Top3 Acc_ 99.0000%

4. 모델의 평가(F1 score)

- 정밀도
  - 모델이 예측한 양성 클래스 샘플 중에서 실제로 양성 클래스인 비율
- 재현율
  - 실제 양성 클래스 샘플 중에서 모델이 양성 클래스로 예측한 비율
- 정밀도와 재현율이 모두 높을수록 높은 값
- F1 score
  - 이진 분류 문제에서 모델의 성능을 측정
  - 정밀도(precision)와 재현율(recall)의 조화 평균값
  - F1 score = 2 x (precision x recall) / (precision + recall)
- macro F1 score
  - 다중 분류 문제에서 모델의 성능을 측정하는 평가 지표
  - 각 클래스마다 TP(true positive), FP(false positive), FN(false negative)을 계산
    - 해당 클래스의 정밀도(precision)과 재현율(recall)을 계산
    - 각 클래스마다 계산된 F1 score을 모두 더한 후 클래스의 개수로 나누어 평균값
  - macro F1 score는 각 클래스마다 동등한 가중치를 부여하여 계산하기 때문에, 클래스의 불균형성(imbalanced data)에 민감하지 않습니다.
    - 크래스의 크기가 서로 다른 경우 유용한 평가지표
- micro F1 score
  - 다중 분류 문제에서 모델의 성능을 측정하는 평가 지표
  - 모든 클래스의 TP(true positive), FP(false positive), FN(false negative)의 총합을 이용하여 정밀도(precision)과 재현율(recall)을 계산

- micro F1 score는 클래스 간의 구분 없이 모든 예측값과 실제값을 하나의 통합된 결과로 계산하기 때문에, 클래스의 크기가 서로 매우 다른 경우에도 불균형성을 고려
- 클래스의 크기가 서로 매우 다른 경우나, 분류해야 하는 클래스의 수가 많은 경우에 유용한 평가 지표

- **macro F1 score : 0.9284700662503867**
- **micro F1 score : 0.9284700662503866**

- AI 학습 시 손실율과 정확도 중 AI 모델의 성능에 더 중요한 요소는?
  - 손실율과 정확도는 모두 AI 모델의 성능을 평가하는 중요한 지표입니다. 그러나 어떤 지표가 더 중요한지는 문제의 성격에 따라 다를 수 있습니다.
  - 이진분류모델
    - 정확도는 중요한 지표
    - ex) 스팸 메일 분류 문제를 생각해보면, 정확도는 스팸 메일을 식별하는 데 매우 중요
    - 이 경우에는 AI 모델의 예측이 얼마나 정확하냐가 가장 중요한 지표
  - 이미지분류모델
    - 손실율이 더 중요한 지표
    - 이미지 분류 문제에서는 AI 모델이 예측한 클래스와 실제 클래스가 얼마나 다른지를 평가하는 손실 함수를 사용하여 모델을 학습
    - 손실율이 작을수록 모델이 예측을 더 잘 수행할 가능성이 높음
  - 어떤 지표가 더 중요한지는 문제의 성격과 목적에 따라 다르므로, 상황에 맞게 적절한 지표를 선택하는 것이 중요

  ⇒ 실수를 최소화 하려면 손실율 고려

  ⇒ 정확성 극대화 하려면 정확도 고려