

Can AI Augment Data?

- 데이터 증폭을 위한 이미지 생성 모델의 활용 가치 -

1 소개 및 개요

AI 모델링에서의 데이터 수집과 전처리 과정은 중요하다. 학습 자체가 데이터를 기반으로 진행되기 때문에 데이터 수집과 전처리 과정이 결과값에 영향을 미친다. 데이터 양이 적을 경우, 일반화 능력이 떨어지고 이는 좋은 결과를 도출해내지 못한다. 또한, 이미지 데이터의 화질이나 노이즈 문제 등 데이터의 퀄리티가 떨어질 경우, **data uncertainty**가 존재하게 된다. 이는 본론에서 설명하도록 하겠다.

그 중 수집 과정의 경우, 시간적·경제적 비용 문제가 존재한다. 이 문제를 해결하기 위해 우리는 기존처럼 이미지를 수집하는 것이 아닌 텍스트를 이용하여 이미지 데이터셋을 설정한다.

본 보고서에서는 **DALL-E** 모델을 사용해서 이미지를 생성하여 데이터를 증폭하였다. 이 연구를 통해서 도출된 결과의 가치 판단과 기존처럼 이미지를 수집하는 것과의 차이점을 확인하고자 한다. 이때, 데이터셋 생성을 AI가 해주기 때문에 **noise** 발생을 가정하여 진행하였다. 따라서 모델은 기존 **ANN**의 **architecture**가 아닌 데이터 **noise**에 강한 **BNN(Bayesian Neural Network)**을 사용하고자 한다.

2 관련 지식

2.1 DALL-E

DALL-E는 자연어 처리와 컴퓨터 비전을 결합하여 텍스트에서 이미지를 생성할 수 있는 AI 모델로, 어떤 간단한 텍스트를 입력하면 이미지를 만들어주는 모델이다.

2.1.1 데이터셋

DALL-E의 학습에 사용된 데이터는 250,000,000개의 이미지-텍스트 쌍이다. 3가지 데이터셋을 모아서 학습한 것으로 알려져 있다. 첫 번째는 구글에서 공개한 **Conceptual Caption** 데이터셋이다. 약 3,000,000개의 이미지-텍스트 데이터로 구성되어 있습니다. 두 번째는 YFCC100M으로 99,200,000개의 이미지, 800,000개의 동영상으로 이루어진 데이터셋이다. 마지막으로 위키피디아의 이미지와 이미지에 대한 캡션을 데이터셋으로 사용하였다.

2.1.2 과정

Stage1

이미지를 픽셀 단위로 모델에 넣어 학습하게 된다면 1차원으로 바꿔 진행하여야 하는데 이는 길이가 약 200,000가 되므로 학습이 제대로 안될 뿐더러 메모리에 올라가지 않는다. DALL-E는 이러한 문제를 해결하기 위해 VQ-VAE(Vector Quantized Variational AutoEncoder)를 사용하였다. VQ-VAE는 입력 받은 이미지를 CNN 모델에 넣어 32 * 32 피쳐맵을 뽑는다. 뽑힌 피쳐맵을 바탕으로 Embedding Space를 생성한다. 그 후, 이 Embedding Space를 통해 다시 디코딩하여 이미지를 얻는다. 결과적으로 이 과정을 거치게 되면 256 * 256의 이미지가 Embedding Space에 있는 벡터들로 표현된 32 * 32 이미지 토큰으로 변환된다. 해당 과정을 거치게 되면 공간적인 해상도가 약 8배 정도 줄어든다. 이로 인해 물체의 테두리, 질감, 얇은 선 등 일부 정보는 왜곡되거나 손실될 수 있다.

이러한 손실을 최소화하기 위해 8192라는 큰 **vocabulary size**를 사용함으로써 정보의 손실을 최소화하였다.

Stage2

이후 이미지 캡션을 BPE 인코딩을 통해 최대 256개의 토큰으로 만든 뒤, 아까 만들어 두었던 이미지 토큰과 합치고 Transformer Decoder에 넣어 학습을 진행하면 끝이다.

2.2 CNN

Convolutional Neural Network의 약자로 일반 Deep Neural Network에서 이미지나 영상과 같은 데이터를 처리할 때 발생하는 문제점들을 보완한 방법이다.

2.2.1 CNN을 사용하는 이유

일반 DNN의 문제점에서부터 출발한다. DNN은 기본적으로 1차원 형태의 데이터를 사용한다. 그러므로 입력값이 2차원 형태인 이미지가 되는 경우, 이것을 평탄화시켜서 한 줄 데이터로 만들어야 한다. 하지만 이 때 이미지의 정보가 손실나고 추상화 과정 없이 바로 연산 과정으로 넘어가 버리기 때문에 학습시간과 능률의 효율성이 저하된다.

이러한 문제점을 해결하는 것이 CNN이다. CNN은 평탄화를 시키지 않고 계층을 빌드업한다. CNN은 이미지 전체보다는 부분, 즉 주변 픽셀들의 연관성을 본다.

2.2.2 Convolution Layer

Convolution은 마치 입력 데이터에 도장을 찍어 유용한 특성만 드러나게 하는 것으로 비유한다. CNN에서는 완전 연결 신경망과 달리 뉴런을 filter나 kernel이라고 부르는데 실제 여러 분야에서 사용되는 방법으로 특히 이미지 분류 작업에서 좋은 성능을 보여준다. convolution layer의 뉴런에 있는 가중치 개수는 사용자가 정할 수 있고, 이를 하이퍼파라미터라고 한다.

Convolution의 장점은 2차원 입력에도 적용할 수 있다는 것이다. 입력 데이터와 convolution 연산을 수행하게 되는 행렬을 kernel이라고 부른다. convolution 계산을 통해 얻은 출력을 feature map이라고 부른다. convolution에도 bias를 적용할 수 있다.

현재 위치의 출력 데이터는 인접한 픽셀에 Convolution Filter를 곱해서 얻어진다.

$$\begin{array}{|c|c|c|c|} \hline 0 & 1 & 7 & 5 \\ \hline 5 & 5 & 6 & 6 \\ \hline 5 & 3 & 3 & 0 \\ \hline 1 & 1 & 1 & 2 \\ \hline \end{array} \odot \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 1 & 2 & 0 \\ \hline 3 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|} \hline 40 & 32 \\ \hline 26 & \\ \hline \end{array}$$
$$\begin{array}{|c|c|c|c|} \hline 0 & 1 & 7 & 5 \\ \hline 5 & 5 & 6 & 6 \\ \hline 5 & 3 & 3 & 0 \\ \hline 1 & 1 & 1 & 2 \\ \hline \end{array} \odot \begin{array}{|c|c|c|} \hline 1 & 0 & 1 \\ \hline 1 & 2 & 0 \\ \hline 3 & 0 & 1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline 40 & 32 \\ \hline 26 & 25 \\ \hline \end{array}$$

Convolution Layer를 거치면서 이미지의 크기는 점점 작아지게 되고 이미지의 가장자리에 위치한 픽셀들의 정보는 점점 사라지게 된다. 이러한 손실을 막기 위한 방법이 패딩(Padding)이다. 패딩은 입력 데이터 주변을 특정값으로 채우는 것을 의미한다. 입력과 특성 맵의 크기를 동일하게 만들기 위해 입력 주위에 0으로 패딩하는 것을 제로 패딩이라고 하며 CNN에서는 주로 제로 패딩을 사용한다.

2.2.3 Pooling Layer

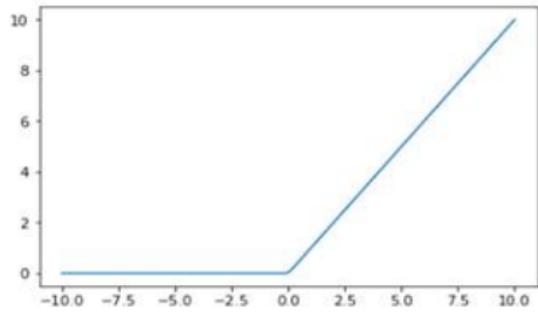
Pooling Layer은 Convolution Layer의 출력 데이터를 받아서 출력 데이터의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용된다. Pooling Layer는 Convolution Layer와 다르게 학습 대상 파라미터가 없고 Pooling Layer를 통과하면 행렬의 크기가 감소된다. Pooling Layer에는 Max Pooling, Min Pooling, Average Pooling이 있다. 그 중 CNN이 신경세포와 유사한 방식을 취하기 때문에 Max Pooling 기법이 많이 사용된다. Max pooling은 특징의 값이 큰 값이 다른 특징들을 대표한다는 개념을 기반으로 하고 있다.

2.3 Rectified Linear Unit(ReLU)

활성화 함수는 각각의 뉴런들에 들어오는 입력신호의 합을 출력신호로 변환시키는

함수이다. 활성화 함수에는 선형 함수와 비선형 함수가 있다.

ReLU 함수는 최근에 신경회로망에서 가장 많이 쓰이는 활성화 함수이다. 다음은 ReLU 함수의 수식이다.

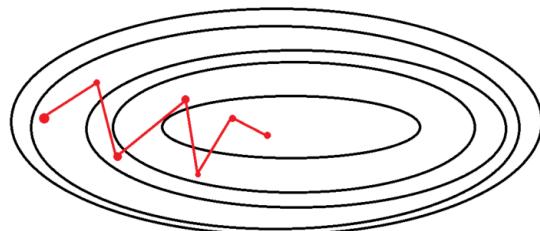


ReLU 함수는 Fully Connected Layer에서 뿐만 아니라 Convolution Layer에서 사용된다. Convolution Layer에서 각각의 픽셀들을 뉴런들로 볼 때 이 뉴런들을 ReLU라는 함수를 통과 시켜준다. ReLU 함수의 장점은 Gradient Vanishing 현상이 없다는 것과 계산이 매우 효율적이며 수렴하는 속도가 Sigmoid 함수에 비해 약 6배 정도 빠르다는 것이다. 하지만 중심 값이 0이 아니라는 점이다. 그리고 음의 값을 항상 0으로 만들기 때문에 음의 값을 갖는 뉴런은 업데이트가 안 될 가능성도 있다. 하지만 칼라 이미지는 거의 양의 값을 가지기 때문에 이 단점이 큰 문제가 되지는 않는다. 또, 지금까지의 활성화함수의 단점인 파라미터 수치들이 업데이트가 안 된다는 장점을 공통으로 가지고 있다. 이를 보완하기 위해 Leaky ReLU라 하여 ReLU의 변형 함수와 마찬가지로 ReLU에서 변형된 ELU라는 가장 최근의 활성화 함수가 있다.

2.4 SGD(확률적 경사 하강법)

전체 훈련 데이터셋을 대상으로 학습하는 것은 한정된 리소스를 가지고 있는 우리의 분석 환경에서 매우 비효율적이며, 파라미터 업데이트 수가 적다는 것은 랜덤하게 뽑힌 시작 위치의 가중치 수도 적으므로, Local minimum현상이 발생할 확률도 높다는 것이다. 그래서 나온 방법이 학습데이터셋에서 무작위로

한계의 샘플 데이터 셋을 추출하고, 그 샘플에 대해서만 기울기를 계산하는 확률적 경사 하강법이다. 샘플 데이터 셋에 대해서만 경사를 계산하므로, 매 반복에서 다뤄야 할 데이터 수가 매우 적어, 학습 속도가 매우 빠르다. 또한, 메모리 소모량이 매우 낮으며, 매우 큰 훈련 데이터 셋이라 할지라도 학습이 가능하다. 그러나, 무작위로 추출된 샘플에 대해서 경사를 구하므로, 불안정하게 움직인다.



손실 함수가 최솟값에 다다를 때까지 위아래로 움직이다보니, 학습이 진행되다 보면, 최적해에 매우 근접하게 움직이긴 하겠으나, 최적해에 정확히 도달하지 못할 가능성이 있다. 그러나, 이렇게 위아래로 움직이므로, 지역 최솟값에 빠진다 할지라도, 지역 최솟값에서 쉽게 빠져나올 수 있으며, 그로 인해 전역 최솟값을 찾을 가능성이 BGD에 비해 더 높다. 즉, 확률적 경사 하강법은 속도가 매우 빠르고 메모리를 적게 먹는다는 장점이 있으나, 경사를 구할 때, 무작위성을 띄므로 지역 최솟값에서 탈출하기 쉬우나, 전역 최솟값에 다다르기 힘들다는 단점을 가지고 있다.

2.5 Cross-entropy

Cross-entropy는 예측 모형은 실제 분포인 q 를 모르고, 모델링을 하여 p 분포를 예측하고자 하는 것이다. 예측 모델링을 통해 구한 분포를 $p(x)$ 라고 해보자. 실제 분포인 q 를 예측하는 p 분포를 만들었을 때, 이 때 Cross-entropy는 아래와 같이 정의된다.

$$H_p(q) = - \sum_{i=1}^n q(x_i) \log p(x_i)$$

크로스 엔트로피는 두 확률 분포의 차이를 구하기 위해서 사용된다. 딥러닝에서는 실제 데이터의 확률 분포와, 학습된 모델이 계산한 확률 분포의 차이를 구하는데 사용된다. q 와 p 가 모두 들어가서 크로스 엔트로피라고 한다.

머신러닝을 통한 예측 모형에서 훈련 데이터에서는 실제 분포인 q 를 알 수 있기 때문에 Cross-entropy를 계산할 수 있다. 즉, 훈련 데이터를 사용한 예측 모형에서 Cross-entropy는 실제 값과 예측 값의 차이를 계산하는데 사용할 수 있다는 것이다.

2.6 Calibration

◎ Calibration이란?

Deep Learning에서의 Performance는 일반적으로 정확도(Accuracy)를 의미한다. model performance 평가지표에는 accuracy, precision, recall, f1-score 등이 있는데, 그 중 정확도는 예측값이 참값에 근접한 정도, 쉽게 말하자면, 예측한 것이 얼마나 맞았는지 또는 데이터가 얼마나 잘 분류되었는지를 뜻한다. 현대 Neural Network은 과거보다 정확도가 향상되었지만, Calibration이 좋지 않다는 점을 문제로 삼았다. 그렇다면 Calibration은 무엇일까?

Calibration이란 모형의 출력값이 실제 Confidence(Calibrated Confidence)를 반영하도록 하는 것이다. 일반적으로 deep learning model은 overconfident하다. Overfitting은 overconfident 하다. (overfitting - overconfident) 모형의 출력이 실제 confidence를 반영한다면, Confidence와 accuracy가 같아야 한다.

Calibration은 왜 중요할까? 의사결정 프로세스에서 모든 판단을 deep learning에게 맡기는 식으로 의사결정이

이루어지지 않는다. Confidence가 낮은 경우에만 사람이 개입할 수 있는 의사 결정이 가능하게 하기 위해서는 모형의 Confidence 확인이 중요하고 이 Confidence를 Calibrate해야만 값에 신뢰성이 있다고 말할 수 있다.

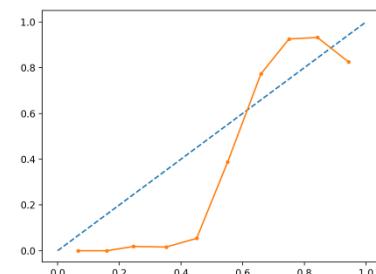
◎ Calibration Measure

모형의 Calibration이 잘 되었는지를 확인할 수 있는 measure는 크게 3가지가 있다.

1) Reliability Diagram

$$acc(B_m) = \frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i)$$

Reliability diagram은 expected accuracy와 observed accuracy를 각각 x,y축으로 하여 그린 그래프이다. expected accuracy를 기준으로 bin을 쪼개서 각 bin에서 observed accuracy를 구하는 방법이다.



1) Expected Calibration Error (ECE)

$$ECE = \sum_{m=1}^M \frac{|B_m|}{n} |acc(B_m) - conf(B_m)|$$

ECE는 confidence와 실제 accuracy의 차이의 기댓값으로 연속형 변수에서 구할 수 없기 때문에, binning을 통해 위와 같이 approximation되는 값이다. M개의 bin에 대해 각 bin마다의

accuracy와 confidence 차이를 weighted sum 한 값이다.

예측하기 때문에 모델이 Overconfident를 갖게 된다. 이는 test set에 대한 예측에서도 나타나는 현상이다.

2) Negative log likelihood(NLL)

$$L = - \sum_{i=1}^n \log(\hat{\pi}(y_i | x_i))$$

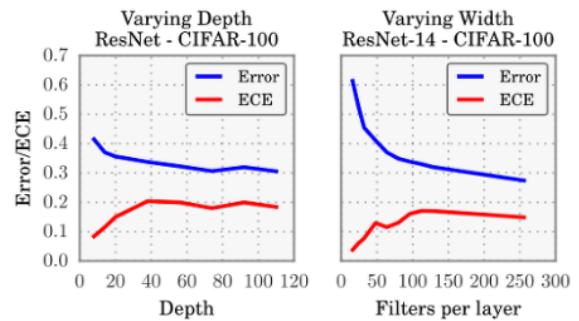
NLL은 통계 모형의 quality를 평가하는데 표준적으로 많이 쓰이는 measure이며, calibration을 어느 정도 반영하고 있다고 볼 수 있다.

◎ Observing Miscalibration

현대 Deep Learning Model들은 매우 많은 layer를 갖고 있습니다. 수많은 layer마다 수백개의 Convolution filter가 사용되는데, 이는 training set의 특징을 더욱 잘 학습하며, generalization이 더 좋다는 것을 최근 연구에서 보여주고 있다. 모델의 깊이와 넓이를 증가시키는 것이 Classification error를 줄여줄 수 있지만, Model Calibration에는 부정적인 영향을 미치게 된다.

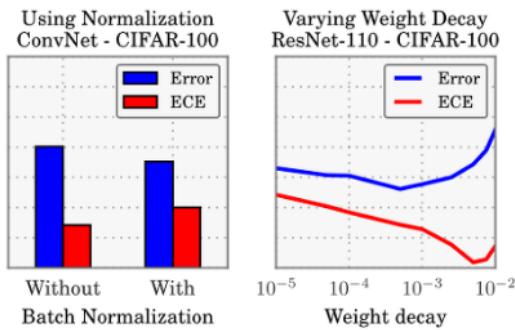
그림에서 좌측 첫 번째 그래프는 ResNet에 대하여 64개의 filter를 고정으로 하고, Model Depth를 올리면서 error 와 ECE 변화를 그린 그래프인데, depth가 증가할수록 error는 줄어들지만, ECE가 증가하는 것을 볼 수 있다. 좌측에서 두 번째 그래프는 depth를 고정으로 하고, filter 수를 증가시키면서 error와 ECE를 확인한 그래프인데, 이 역시 filter 수가 증가할수록 error는 줄어들지만, ECE가 증가하는 것을 볼 수 있다.

Model capacity가 클수록 ECE는 증가하는 것을 위의 관찰을 통해 확인할 수 있었다. 이 이유는 모델이 training set에 대한 loss를 최소화하는 방향으로 학습되고, loss를 줄이기 위해 confidence를 높여 1에 가까운 값을



최근 연구들은, Normalization 기술이 ResNets 과 DenseNets 과 같은 깊은 구조의 모델을 학습 시킬 수 있는 방법이라고 얘기하였다. Batch Normalization(BN) 방법은 Distribution Shifts(test set이 train set 과 다른 distribution을 따르는 경우)을 최소화해서 딥러닝의 optimization 및 regularization 을 향상시키는 방법이다. BN이 모델의 최종 예측에 정확히 어떤 예측을 주는지는 알 수 없지만, 그래프를 보면 BN을 적용한 모델이 그렇지 않은 경우보다 ECE값이 높음을 확인할 수 있다.

Weight decay(가중치 감쇠)는 학습된 모델의 복잡도를 줄이기 위한 방법으로, 학습 중 Loss function의 weight가 커질 경우에 페널티 항목을 둘으로써 weight 가 너무 큰 값을 가지지 않게 하는 방법이다. 최근 연구에서는, BN의 regularization 효과 때문에 L2 regularization을 사용한 모델 학습이 일반화에 더 좋다고 제안하고 있다. 그래프에서는 less weight decay 와 함께 모델을 학습시키는 것이 calibration에 부정적인 영향을 준다는 것을 볼 수 있다.



2.7 Uncertainty

◎ Uncertainty 란?

최신 딥러닝 모델들은 모델이 커지면서 예측 성능에서는 좋은 모습을 보이지만 모델의 설명가능성이 떨어지고 해석하기 어려운 딥러닝 모델의 출력값을 얼마나 신뢰할 수 있는지 알기가 어렵다. 또한, 데이터가 적을 경우, **Noise**가 내재된 경우, **unseen data**가 입력으로 들어오는 경우 등에는 정확한 예측을 하지 못하게 되는 경우가 발생한다. 기존 모델들처럼 단순 입력을 통해 출력을 만들어내는 모델들은 믿을 수 있는지가 확실하지 않기 때문에 Explainable AI의 중요성이 커지고 있다. 따라서, 딥러닝 모델을 구축할 때, 예측 결과에 대해 얼마나 믿을 수 있는지를 수치화한 불확실성(Uncertainty)를 고려할 수 있는 모델을 만드는 것이 제안되었다. 이러한 모델이 입력을 받았을 때, 출력값(확률 모델일 경우 출력값은 예측 확률)과 더불어 uncertainty도 출력할 수 있게 한다. 의사 결정 프로세스에서 모든 판단을 온전히 딥러닝에게 맡기는 식으로 이루어지게 해선 안되므로, uncertainty가 특정 threshold를 넘은 경우에만 출력값에 대한 별도의 경고가 이루어질 수 있게 한다면, 자율 주행, 의료, 금융 시스템과 같이 의사결정이 특히나 중요한 시스템에서의 사고를 예방할 수 있다.

◎ Uncertainty의 유형

1) Epistemic uncertainty(model uncertainty)

- 주어진 데이터를 가장 잘 설명하는 최상의 모델 매개변수 및 모델구조의 불확실성
- 더 많은 데이터가 학습된다면 줄일 수 있음(reducible uncertainty).
- 학습데이터가 부족하여 학습되지 않은 상태를 식별할 수 있다는 점에서 중요함.
- 높은 불확실성 모델은 추가적인 학습이 필요할 가능성이 높다는 의미로, 안전이 중요한 문제상황에서 높게 발생하는 경우 모델을 신뢰할 수 없음.

2) Aleatoric uncertainty(data uncertainty)

- 데이터에 내재된 노이즈로 인한 불확실성
- 더 많은 데이터가 학습되더라도 줄일 수 없음(irreducible uncertainty).
- 측정 정밀도를 높이면 줄일 수 있음.
- 실제 상황에서와 같이 일부 데이터의 **Noise**가 많이 있는 경우에 중요함.
- **Noise**가 큰 데이터에 대해 학습 과정에서 제약을 부여할 수 있으므로, 예측 성능 안정화에 기여할 수 있음.

◎ Out-of-Distribution

Unknown class data에 대해 현대의 딥러닝 모델들은 'i do not know'라고 말할 수 없다. Out-of-distribution test data는 한번도 학습에 사용하지 못한 유형의 데이터가 테스트 시 사용되는 경우를 말한다. 예컨대, 강아지 품종 예측 모델에서 고양이 사진을 테스트 데이터 입력으로 주고 예측을 요청하는 경우가 있다. Uncertainty는 Out-of-distribution detection에 적용 가능하기 때문에 중요하다고 말할 수 있다.

2.8 BNN



테슬라의 자율 주행 시스템은 카메라에 굉장히 의존적이다. 그렇기에 카메라가 무엇을 어떻게 인식하느냐가 중요하다. 사진 1은 테슬라의 사고 사례를 보여준 사진이다. 컨테이너의 옆부분을 하늘로 오인식하여 발생한 사고이다. 모델이 명확히 판단할 수 없는 물체에 대한 처리가 중요하게 되었다. 이런 점을 해결하기 위해 기존의 딥러닝에서의 고정된 가중치 값이 아닌 가중치가 특정 확률분포를 띠는 형태가 나오게 되었다. 그것이 바로 BNN(Bayesian Neural Network)이다. BNN에 들어가 이전에 간략하게 베이즈 정리를 살펴보자.

◎ 베이즈 정리

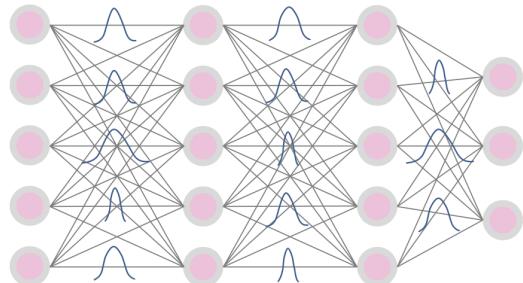
이전의 경험과 현재의 증거를 토대로 어떤 사건의 확률을 추론하는 이론이다. 확률 $P(A|B)$ 를 알고 있을 때, 관계가 정반대인 확률 $P(B|A)$ 를 계산하기 위해 등장했다.

1) $P(A)$: A의 사전 확률(evidence)

2) $P(B)$: B의 사전 확률(prior probability)

3) $P(A|B)$: 사건 B가 주어졌을 때, A의 조건부 확률(likelihood)

4) $P(B|A)$: 사건 A라는 증거에 대한 사후 확률(posterior probability)



Classification 문제에서 기존 ANN의 문제점으로 꼽히는 것 중 하나가 Overconfidence이다. 예를 들어, 학습 과정 때 존재하지 않았던 class의 input이 들어온다면 이를 해결하기 위해, 기준 확률값을 잡고 예측 class의 softmax값이 기준 확률값을 넘지 않으면 Overconfidence하지 않다고 볼 수 있지 않을까? 하지만 softmax값 같은 경우 uncertainty하더라도 높은 값이 나오는 경우가 존재한다. 그러므로 softmax로 uncertainty를 규제하는 것은 좋지 않은 방법이다. 다음 방법이 사진2 와 같이 예측에 대한 uncertainty를 추정하는 것이다. 예측값에 대한 분산의 정보를 활용하여 그 분포로 추정 할 수 있다. 그렇다면 어떤 방법으로 분포를 추정할까?

◎ parameter 분포 추정

Posterior

$$p(W|X, Y) = \frac{p(Y|X, W)p(W)}{p(Y|X)}$$

Evidence

$$p(W|X, Y) = \int p(Y|X, W)p(W)dw$$

Evidence를 계산함에 있어서 방대한 적분 계산량을 요구하기 때문에 Posterior를 이와 비슷한 임의의 분포로 가정한다.

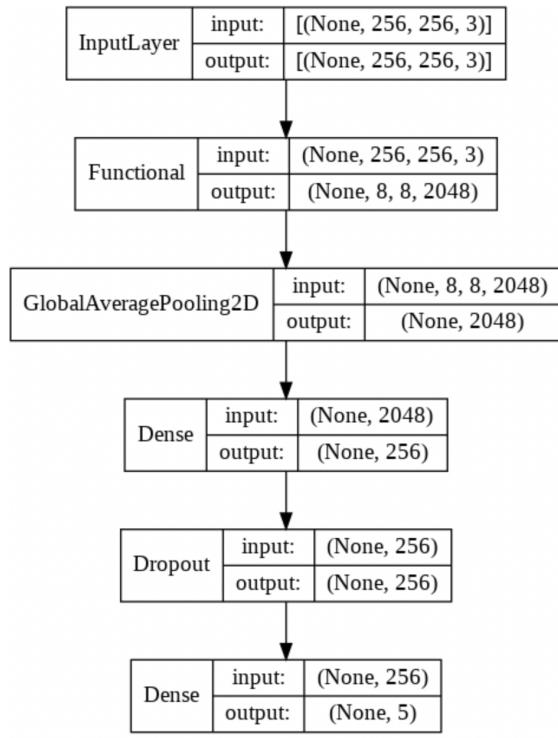
$q_{\theta}(W)$: Variational distribution

$$q_{\theta}(W)^* = \operatorname{argmin} KL(q_{\theta}(W) || p(W|X, Y))$$

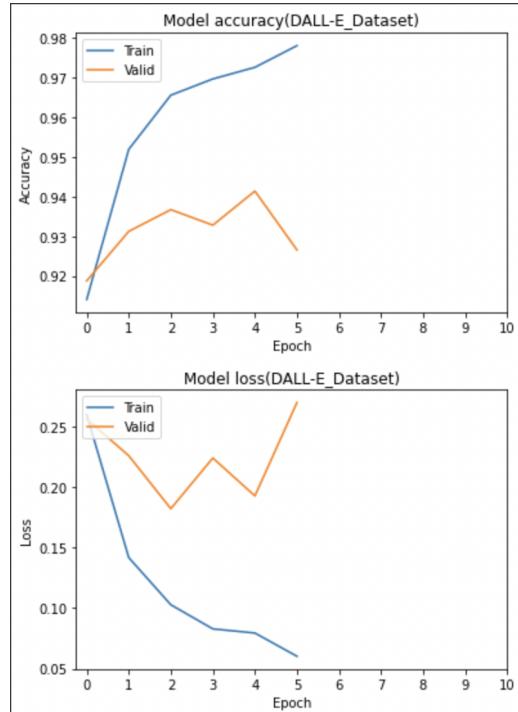
3 실험

3.1 실험 1

학습 데이터으로는 DALL-E에서 생성한 데이터 셋으로 모델을 학습을 시켰다.



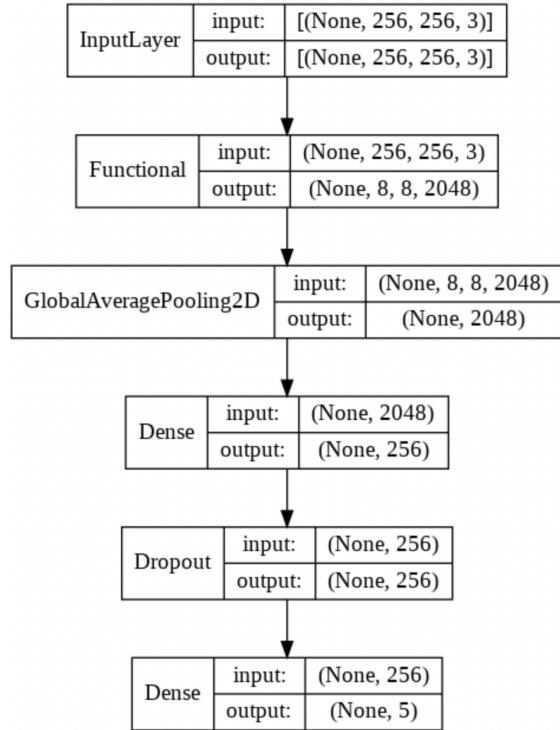
위 사진은 사용된 모델의 구조이다.
pre-trained model을 이용하였다. 모델의 feature 추출 부분을 ResNet152-V2를 freeze하여 사용하였다. 다음 사진은 모델 훈련 과정을 시작화한 사진이다.



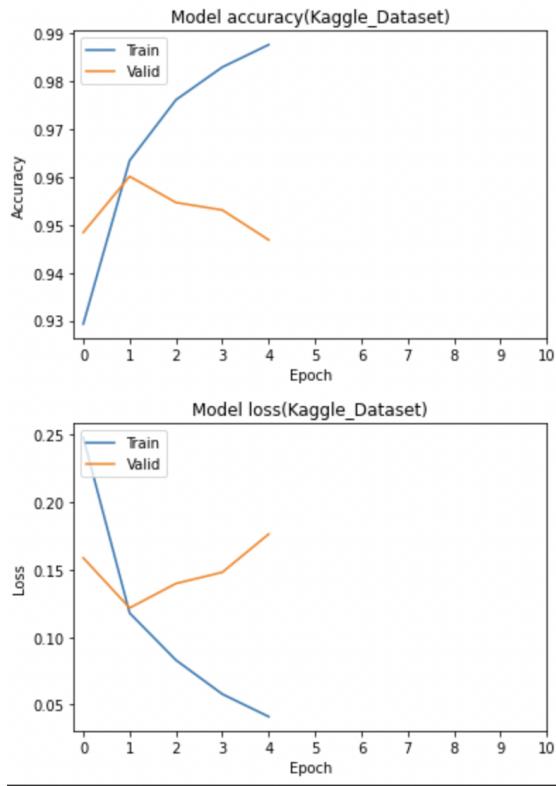
테스트 데이터에 대한 정확도는 91.062498%로 나왔다.

3.2 실험 2

학습 데이터으로는 Kaggle에서 가져온 데이터 셋으로 모델을 학습을 시켰다.

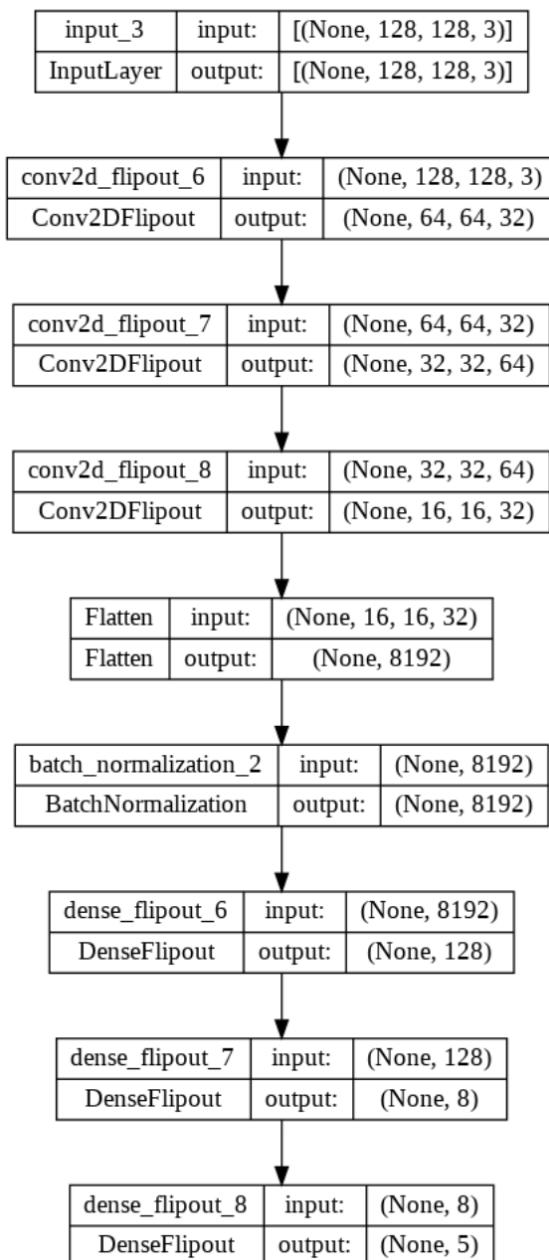


위 사진은 사용된 모델의 구조이다.
pre-trained model을 이용하였다. 모델의
feature 추출 부분은 ResNet152-V2를
freeze하여 사용하였다. 다음 사진은 모델
훈련과정을 시각화한 사진이다.

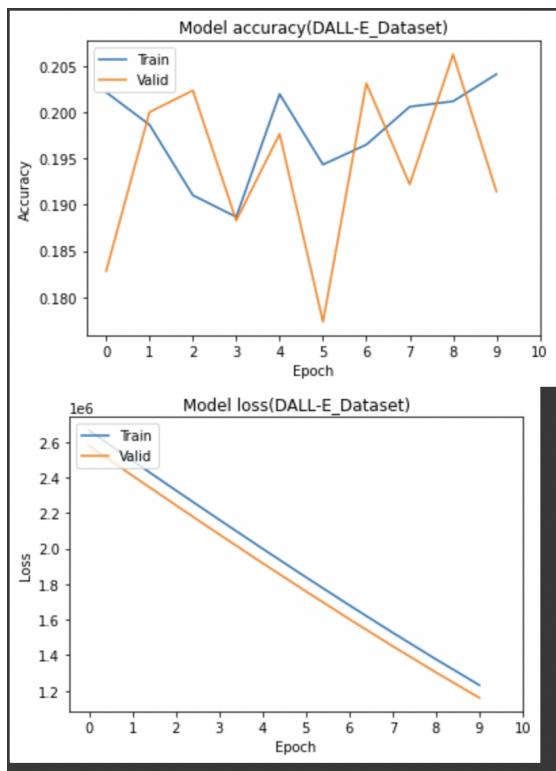


테스트 데이터에 대한 정확도는
94.921875%로 나왔다.

3.3 실험3
학습 데이터으로는 DALL-E에서 생성한
데이터 셋으로 모델을 학습을 시켰다.



위 사진은 사용된 모델의 구조이다. 실험
1,2와 다르게 분포를 고려한 BNN를
사용하였다. 다음 사진은 모델 훈련
과정을 시각화한 사진이다.



테스트 데이터에 대한 정확도는 19.562501%로 나왔다.

3.4 실험 결과

실험 1,2,3의 성능평가는 동일한 테스트 데이터로 진행하였다. 실험 1,2의 성능은 크게 나지 않았다. 즉, DALL-E 데이터는 데이터로써의 활용가치가 있다. 하지만 보다 복잡한 데이터가 요구될 때, DALL-E가 질 높은 데이터를 생성해 줄거라는 것을 신뢰 할 수 없다. 데이터의 노이즈가 발생 할 우려가 있다는 것이다. 이를 고려한 것이 BNN을 사용한 실험 3이다. 하지만 좋은 성능을 내지는 못하였다.

4 결론

본 보고서에서 설계한 CNN은 이미 많은 이미지 검출 및 이미지 인식에 사용되고 있다. BNN의 경우 모델의 설명력을 높이기 위한 모델로 Uncertainty 정량화에 대한 방법을 개선시키기 위해 Deep Ensemble이나 Non-Bayesian Approach 등의 방법론 역시 추가되고 있다. 우리는 이러한 모델들을 활용해 인공지능(DALL-E)이 생성한 데이터가

유의미하다는 것을 모델 결과 보고서를 통해 확인하였다. 앞으로 이미지 학습에서의 데이터 증강에 있어, 좋은 영향을 끼칠 수 있을 것으로 보인다.

앞서 말했듯이, 질 높은 데이터를 인공지능으로 생성했을 시, 그 생성된 데이터를 100% 신뢰할 수는 없다. 즉, 데이터의 noise가 있다는 것을 염두해 두어야 한다. 이를 해결하기 위해, BNN을 사용하였지만, 예상한 결과를 얻지는 못하였다. 해당 모델을 조금 더 보완해 볼 것이며, 더 나아가 data noise에 강력한 여러 가지 방법들을 모색해 보려고 한다.

5 참고문헌

Guo, C., Pleiss, G., Sun, Y., & Weinberger, K. Q. (2017, July). On calibration of modern neural networks. In *International conference on machine learning* (pp. 1321-1330). PMLR.

LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

Jospin, L. V., Laga, H., Boussaid, F., Buntine, W., & Bennamoun, M. (2022). Hands-on Bayesian neural networks—A tutorial for deep learning users. *IEEE Computational Intelligence Magazine*, 17(2), 29-48.

Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., ... & Sutskever, I. (2021, July). Zero-shot text-to-image generation. In *International Conference on Machine Learning* (pp. 8821-8831). PMLR.

Thomas, M. T. C. A. J., & Joy, A. T. (2006). *Elements of information theory*. Wiley-Interscience.

이현수, 「이미지 컨텐츠 검색을 위한 CNN의 구조 A Structure of Convolutional Neural Networks for Image Contents Search」, 중앙대학교 대학원 석사학위논문, 2018.