Deep Learning

For Image Classification

Yeojin-Lee

이미지 분류를 위한 딥러닝 모델

이여진

Abstract

This work develops a Convolutional Neural Network (CNN), a deep learning model for solving image classification problems. Through this, I propose artificial neural network structures and methods, and present results and conclusions through class activation map (CAM) visualization. In this work, 50,000 images were loaded from Cifar10 dataset and used to train the network. Then, In this project, I use these CIFAR-10 datasets to train CNN models and measure their accuracy. Using this, I aim to test different parameter values on each layer of the CNN model, and to implement the most optimal CNN model by applying various techniques.

요 약

본 연구는 이미지 분류 (image classification) 문제를 해결하기 위한 deep learning model인 CNN(Convolutional Neural Network) 모델을 개발한다. 이를 통해 인공신경망 구조 및 방법을 제안하며 CAM (class activation map) 가시화를 통한 결과 및 결론을 제시한다. 본 연구에서는 Cifar10 dataset에서 50,000장의 이미지를 로드 하여 네트워크를 훈련시키는데 사용하였다. 본 프로젝트에서는 이러한 CIFAR-10 데이터 셋을 이용하여 CNN 모델을 학습시키고 정확도를 측정한다. 이를 이용해 CNN모델의 각각의 Layer에서 여러 가지 매개 변수 값을 테스트해보고, 다양한 technique을 적용하여 가장 최적의 CNN 모델을 구현하는 것을 목표로 한다.

Key word : CNN, Deep Learning, Image, Cifar10, Layer

키워드 : CNN, 딥러닝, 사진, Cifar10, 층

Ⅰ. 서론

본 연구는 이미지 분류 (image classification) 문제를 해결하기 위한 deep learning model인 CNN(Convolutional Neural Network) 모델을 개발한다. 이를 통해 인공신경망 구조 및 방법을 제안하며 CAM (class activation map) 가시화를 통한 결과 및 결론을 제시한다.

[1] Fully Connected Layer 만으로 구성된 인공 신경망의 입력 데이터는 1차원(배열) 형태로 한정되는데, 한 장의 컬러 사진은 3차원 데이터이며 배치 모드에 사용되는 여러 장의 사진은 4차원 데이터이다. 사진 데이터로 전연결 (FC, Fully Connected) 신경망을 학습시켜야 할 경우에, 3차원 사진 데이터를 1차원으로 평면화 시켜야 한다. 사진 데이터를 평면화 시키는 과정에서 공간 정보가 손실될 수밖에 없다. 결과적으로 이미지 공간 정보 유실로 인한 정보 부족으로 인공 신경망이 특징을 추출 및 학습이 비효율적이고 정확도를 높이는데 한계가 존재한다. 따라서 이미지의 공간 정보를 유지한 상태로 학습이 가능한 모델인 CNN(Convolutional Neural Network)을 통하여 보다 효율적이며, 정확도가 높도록 한다. CNN(Convolutional Neural Network)은 기존 Fully Connected Neural Network와 비교하면 레이어의 입출력 데이터의 형상 유지할 수 있으며 이미지의 공간 정보를 유지하면서 인접 이미지와의 특징을 효과적으로 인식하도록 할 수 있다. 또한 복수의 필터로 이미지의 특징 추출 및 학습이 가능하고 Pooling layer을 통해 추출한 이미지의 특징을 모으고 강화하도록 한다. 더불어 필터를 공유 파라미터로 사용하기 때문에, 일반 인공 신경망과 비교하여 학습 파라미터가 매우 적다는 차별점이 있다.

이에 본 연구는 CNN 모델을 직접 구현하여 Layer에서 여러 가지 매개 변수 값을 테스트해보고, 다양한 technique을 적용하여 가장 최적의 CNN 모델을 구현하는 것을 목표로 하였다. 본 논문의 구성은 다음과 같다. 제2장에서는 CNN 모델의 구조와 본 연구에서 구현한 실제 CNN 모델의 구성을 알아본다. 제3장에서는 최적의 CNN 모델을 찾기 위하여 여러 가지 연구 변인들을 실험하는 과정을 설명하고, 제4장에서는 그 결과와 최종 CNN 모델을 소개한다. 제5장에서는 결론 및 향후 연구 방향을 이야기한다.

Ⅱ. 모델 구조

2.1 관련 기술

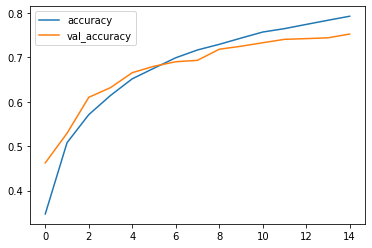
본 연구의 핵심 모델인 CNN은 Convolution Layer, Pooling Layer, Fully-connected Layer 단계로 이루어진 구조다. Convolution Layer에서는 filter를 이용해 특징을 추출하고, Pooling Layer에서는 추출된 특징을 Sub-Sampling한다. Convolution과 Pooling을 반복해 데이터량을 줄이고 인위적으로 왜곡을 생성해 분류가 일어나도록 신경망을 생성한다. 본 프로젝트에서 네트워크를 훈련시키기 위해 사용한 데이터 셋은 Citar10 이다. CIFAR-10 데이터 셋은 픽셀 이미지를 비행기, 자동차, 새, 고양이, 사슴, 개, 개구리, 말, 배, 트럭의 총 10개의 라벨로 분류하는 것이 목표이다. 각 이미지는 RGB 값으로 구성된 32\*32 크기의 픽셀 이미지이며, 10개의 클래스로 라벨링이 이루어져 있다. 카테고리별로 6,000개의 이미지가 존재하며 50,000개의 학습 데이터, 10,000개의 테스트 데이터로 구성되어 있다. 본 프로젝트에서는 이러한 CIFAR-10 데이터 셋을 이용하여 CNN 모델을 학습시키고 정확도를 측정하였다.

2.2 CNN Model 구현

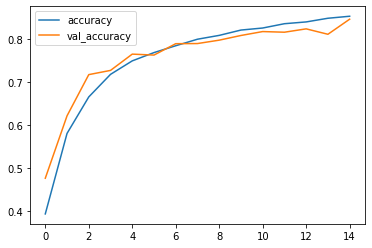
연구를 위해 CNN 모델의 각 층을 구현하고 학습시켜 테스트를 진행하기 위하여 Tensorflow, keras에서 제공하는 모듈과 class, function을 사용하도록 하였다. 구현된 CNN모델은 ConvolutionNET Class를 통해서 CNN 모델을 구현하였으며, Convolution – Convolution – Pooling Layer 구조를 반복적으로 쌓아서 데이터의 특징을 추출하였다. 이렇게 추출된 특징 값을 기존의 신경망에 적용하여 분류하기 위한 Fully Connected Layer를 구성한 뒤, Softmax Function과 Dropout을 통해 최종 CNN 모델을 구현하였다.

[2] Convolution은 합성곱 연산은 두 함수 f, g 가운데 하나의 함수를 반전(reverse), 전이(shift)시킨 다음, 다른 하나의 함수와 곱한 결과를 적분하는 것을 의미한다. 이러한 첫 번째 Convolution Layer에서는 filter을 이용해 이미지 특징(feature)을 계산하고, 그 계산 결과를 activation function을 이용해 정리함으로써 특징을 추출하도록 하였다. 본 연구에서는 Keras가 제공하는 Conv2D Function을 이용하여 Convolution layer을 구현하였다. 여기서 [3] filter(필터)란 이미지의 특징을 찾아내기 위한 공용 파라미터이다. Filter를 Kernel이라고 하기도 하며, CNN에서 Filter와 Kernel은 같은 의미이다. 필터는 일반적으로 (4, 4)이나 (3, 3)과 같은 정사각 행렬로 정의된다. CNN에서 학습의 대상은 필터 파라미터이다. Conv2D 함수의 Argument는 strides, padding, activation으로 이루어져 있다. [4] stride는 지정된 간격으로 필터를 순회하는 간격을 이야기한다. 또한, Convolution 레이어에서 Filter와 Stride에 작용으로 Feature Map 크기는 입력데이터 보다 작은데, Convolution 레이어의 출력 데이터가 줄어드는 것을 방지하기 위하여 [5] padding(패딩)이라는 Argument를 적용하였다. 패딩은 입력 데이터의 외각에 지정된 픽셀만큼 특정 값으로 채워 넣는 것을 의미하며, 보통 패딩 값으로 0으로 채워 넣는데 Conv2D 함수에는 ‘same’으로 값을 설정한다. 마지막으로 activation(활성화 함수)로 딥러닝에서 사용되는 Relu 활성 함수를 사용하였다. [6] 활성화 함수는 Transfer Function으로부터 전달받은 값을 출력할 때 일정 기준에 따라 출력값을 변화시키는 비선형 함수이다. 이러한 활성화 함수를 통해 출력 값을 0 ~ 1 사이의 실수 값으로 정규화하여 확률적 개념으로 사용 가능하다. 대표적인 활성화 함수로는 Sigmod, tanh, ReLU 등이 있다. 본 연구에서는 0보다 작은 값이 나온 경우 0을 반환하고, 0보다 큰 값이 나온 경우 그 값을 그대로 반환하는 ReLU 활성화 함수를 사용하여 Convolution Layer을 구현하도록 하였다.

그 다음으로 Pooling Layer을 하기 전에, Batch Normalization을 위해 Tensorflow에서 제공하는 BatchNormalization Function을 사용하였다. 이는 mean output을 0, output standard deviation은 1에 가깝게 유지하도록 하여 딥러닝의 고질적인 문제인 vanishing과 exploding gradient 문제를 방지하도록 하였다. Batch Normalization을 적용할 것인지에 대한 결정을 위해 본 연구에서 간단한 Ablation study를 진행하였다. Deep Learning에서는 앞서 말했듯이 vanishing과 exploding gradient 문제가 자주 발생하는데, 그 이유는 Layer 수가 적은 경우 그 문제가 심각하지 않지만 Layer 수가 많아지면 이러한 문제가 누적되어 나타나기 때문에 이러한 문제가 성능에 영향을 끼칠 수가 있다는 것이다. 본 연구에서는 이러한 문제에 대한 해결책으로 ReLU(Rectifier Linear Unit)을 활성함수로 사용하여 문제가 완화되기는 하였지만 본질적인 해결책이 아니었다. 따라서 추가적으로 Convolution Layer을 구현한 다음에 스칼라 Feature들을 독립적으로 정규화 하는 방식인, 즉 각각의 Feature들의 Mean 및 Variance를 0과 1로 정규화를 하는 Batch Normalization을 적용하여 이러한 vanishing, exploding gradient 문제를 해결하도록 하였다. 본 Ablation Study에서는 Batch Normalization의 유무에 따른 Validation Accuracy 측정 experiment를 진행하였다. 가설로써 Batch Normalization을 적용하면 Validation Accuracy가 더 높을 것이라 가정하였다. 각각의 변인 환경으로는 layer 수 6, activation은 ReLU, Optimizer은 SGD, Pooling Function으로는 MaxPooling, Batch Size 32, 정규화 여부로는 None으로 설정하여 epoch = 15로 적용 후 Validation Accuracy는 다음과 같다. Batch Normalization을 적용하지 않았을 때는 약 80.31%, 적용한 후에는 약 85.96%로 가설과 같이 Batch Normalization을 적용하였을 때 더 높은 Validation Accuracy를 측정할 있다.

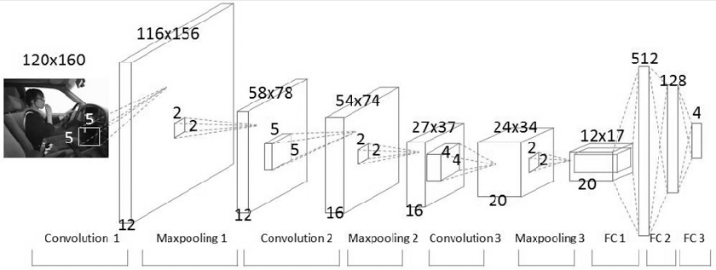


**그림1. Batch Normalization 적용 전 Validation Accuracy**

****

**그림2. Batch Normalization 적용 후 Validation Accuracy**

그 다음으로 Pooling Layer을 구현하였다. [7] Pooling Layer란 컨볼류션 레이어의 출력 데이터를 입력으로 받아서 출력 데이터(Activation Map)의 크기를 줄이거나 특정 데이터를 강조하는 용도로 사용된다. Pooling Layer를 처리하는 방법으로는 Max Pooling과 Average Pooning, Min Pooling이 있다. 정사각 행렬의 특정 영역 안에 값의 최댓값을 모으거나 특정 영역의 평균을 구하는 방식으로 동작한다. Pooling 방식에 따른 정확도 측정 과정에 대한 내용은 제3장에서 다룰 예정이다. 본 연구에서는 Convolution Layer의 계산 결과, 즉 feature Map을 sub-sampling 하는 Pooling을 하기 위해서 Keras에서 제공하는 MaxPooling Function을 사용하였다. MaxPooling Function의 Argument는 각각 pool size, strides로 이루어져 있다. 일반적으로 Pooing 크기와 Stride를 같은 크기로 설정하여 모든 원소가 한 번씩 처리되도록 값을 2로 설정하였다.

CNN은 Convolution Layer와 Max Pooling 레이어를 반복적으로 쌓는 특징 추출(Feature Extraction) 부분과 Fully Connected Layer를 구성하고 마지막 출력층에 Softmax를 적용한 분류 부분으로 나뉜다. 다음은 CNN 구조를 시각화 한 것이다.

**그림3. CNN 구조[8]**

[9] Fully Connected Layer는 한 계층의 모든 뉴런을 다른 계층의 모든 뉴런에 연결하도록 한다. 기존의 [다층 퍼셉트론](https://en.wikipedia.org/wiki/Multi-layer_perceptron) 신경망 (MLP)과 동일하며, 평면화 된 행렬은 완전히 연결된 계층을 통과하여 이미지를 분류하도록 하는 층이라고 할 수 있다. 본 연구에서 구현한 Fully Connected Layer은 Flatten Fuction, Dropout Function, Dense Function, Softmax Layer 단계로 구성되어 있다. [10] Flatten Layer는 CNN의 데이터 타입을 Fully Connected Neural Network의 형태로 변경하는 레이어이다. Flatten 레이어에는 파라미터가 존재하지 않고, 입력 데이터의 Shape 변경만 수행한다. 또한 Activation 역할을 하는 Softmax Layer에서는 추출된 특징 값을 확률을 모방한 일정한 값으로 변환하기 위하여 사용하도록 하였다. Dense Function에서는 dimensionality of the output space의 값을 각각 1024, 128, 10으로 설정하였고 여기에서도 Activation으로 ReLU를 적용하였다.

본 연구에서는 네트워크를 훈련시키기 위하여 Dropout Function과 Optimizer을 적용하였다. 네트워크를 훈련시키는 과정의 핵심은 optimizer이다. [11] Optimizer은 Deep Learning, 즉 CNN에서 optimizer는 학습속도를 빠르고 안정적이게 하기 위해 사용된다. 본 연구에서 optimizer는 SGD, Adam, RMSprop를 적용하였으며 이 중 성능이 더 좋은 Adam을 최종적으로 선택하였다. 세 가지 optimizer의 성능 평가는 제3장에서 다시 확인하도록 한다.

또한 과적합(Overfitting)을 방지하기 위하여 Dropout Function을 사용하였다. Dropout은 학습 시, 지나치게 학습 데이터에 집중하는 것을 위한 Regularization 방법 중 하나이다. 이는 네트워크의 일부를 생략하고 학습을 진행하여, 생략되는 뉴런의 조합만큼 지수함수적으로 다양한 모델을 학습시킨다. 이를 통해 모델 결합(Model Combination)을 하게 되어 학습의 성능을 개선할 수 있게 된다.

Ⅲ. CNN 성능 최적화 실험

앞서 초기에 구현한 모델에서 여러 요인들을 조작한 뒤 Validation Accuracy를 비교하는 실험을 진행하였다. 연구 변인으로는 Convolution layer 수, Activation function 종류, Optimizer 종류, Pooling Function 종류, Batch Size로 설정하였으며, 각각의 정확도를 비교해보도록 하였다.

3.1 통제 변인

본 연구는 Google Colab Notebooks와 Jupyter Notebook 6.1.4에서 진행되었다. 통제 변인으로는 Layer 수 6, Activation Function ReLU, Optimizer SGD, Pooling Function으로 MaxPooling, Batch Size 32, Normalization 여부는 None으로 설정 후 epoch = 15로 적용하여 각각의 종속 변인을 실험하였다. 실험의 결과로 보여지는 각각의 테이블 값은 Validation Accuracy이며, 첫 번째 Validation Accuracy는 fit generator 함수를 적용하였을 때와 적용하지 않았을 때를 나타낸다. Fit generator은 Keras에서 제공하는 모델의 과적합을 방지하고 모델의 일반화 능력을 높이기 위해 데이터 증대를 수행하여 모델을 훈련시키는 함수이다.

3.2 Layer 수에 따른 Accuracy

통제 변인 환경에서 layer 수를 각각 4, 6, 8, 11으로 설정하여 성능을 테스트해보았다. 다음은 각각의 layer 수에 따른 Validation Accuracy이다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **4** | **6** | **8** | **11** |
| 1 | 75.95% / 72.22% | 76.01 / 74.34 | 70.58 / 75.77 | 65.94 / 63.13 |

**표1. Layer 수에 따른 Accuracy**

예상(가설)과 다르게 Layer수가 많을수록 성능이 개선되는 것이 아닌 Layer 6 또는 8에서 가장 높은 정확도가 도출됨을 알 수 있었다.

3.3 Activation에 따른 n Accuracy

통제 변인 환경에서 Activation Function ReLU와 tanh의 성능을 테스트하였다. 다음은 활성화 함수에 따른 3번의 실험으로 도출된 Validation Accuracy이다.

|  |  |  |
| --- | --- | --- |
|  | **ReLu** | **tanh** |
| 1 | 76.01% / 74.34% | 77.34% / 76.21% |
| 2 | 72.52% / 72.93% | 77.40% / 75.94% |
| 3 | 74.82% / 80.31% | 73.15% / 76.39% |
| 평균 | 74.45% / 75.86% | 75.96% / 76.18% |

**표2. Activation에 따른 Accuracy**

실험 결과 ReLU와 tanh의 정확도 차이가 그렇게 크지 않다고 판단되어, 최종 모델을 선정하기 위해 제4장에서 Activation을 다시 한 번 조작하여 실험을 진행하도록 하였다.

3.4 Optimizer에 따른 Accuracy

통제 변인 환경에서 Optimizer을 종속 변인으로 설정하고 SGD, Adam, RMSprop의 성능을 테스트하였다. Adam은 파라미터 당 개별 학습 속도를 계산한다. 가장 빠른 최적화 알고리즘 중 하나이며, RMSProp은 데이터가 노이즈를 일으키는 경우 잘 작동한다는 특징을 가지고 있다. 또한 Adamax은 희소 데이터 세트에서 잘 작동하는 Adam Optimizer의 한 형태이고, SGD를 optimizer로 설정하면 개별 데이터를 사용하여 실시간 학습이 가능하다는 장점이 있다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **SGD** | **Adam** | **RMSProp** |
| 1 | 74.82% / 80.31% | 82.44% / 83.09% | 69.68% / 69.1% |
| 2 | 72.52% / 72.93% | 82.02% / 84.65% | 67.43% / 69.08% |
| 3 | 76.01% / 74.34% | 80.89% / 81.05% | 68.21% / 68.97% |
| 평균 | 74.45% / 75.86% | 81.78% / 82.93% | 68.44% / 69.05% |

**표3. Optimizer에 따른 Accuracy**

각각의 Optimizer 당 3번씩 테스트를 진행하여 Accuracy에 따른 평균값을 내본 결과 Adam 방식이 최적화에 가장 성능이 뛰어나다는 결과를 얻을 수 있었다.

3.5. Pooling 방식에 따른 Accuracy

통제 변인 환경에서 Pooling을 종속 변인으로 설정하여 성능을 테스트하였다. 성능 평가 실험에 사용된 MaxPooling은 Feature Map을 M x N 크기로 잘라낸 후 그 안에서 가장 큰 값을 뽑아내는 방법이다. Average Pooling은 잘라낸 feature Map에서 존재하는 파라미터 값만 뽑아내는 방식이다. 다음은 이러한 Pooling 방식에 따른 3번의 실험으로 도출된 Validation Accuracy이다.

|  |  |  |
| --- | --- | --- |
|  | **MaxPooling** | **AveragePooling** |
| 1 | 74.82% / 80.31% | 69.1% / 78.4% |
| 2 | 72.52% / 72.93% | 69.3% / 75.87% |
| 3 | 76.01% / 74.34% | 69.85% / 77.15% |
| 평균 | 74.45% / 75.86% | 69.41% / 77.14% |

**표4. Pooling 방식에 따른 Accuracy**

실험 결과 본 CNN 모델에는 MaxPooling 방식을 사용하는 것이 더 좋은 성능을 도출한다는 점을 알 수 있었다.

3.6 Batch Size에 따른 Accuracy

통제 변인 환경에서 Batch Size를 종속 변인으로 설정하여 성능을 테스트하였다. 다음은 각각의 Batch Size 32, 100, 300, 500으로 설정된 CNN 모델의 정확도이다.

|  |  |  |  |
| --- | --- | --- | --- |
|  | 32 | 300 | 500 |
| 1 | 74.82% / 80.31% | 64.70 / 62.11 | 71.83 / 56.62 |
| 2 | 72.52% / 72.93% | 72.19 / 60.45 | 69.93 / 58.01 |
| 3 | 76.01% / 74.34% | 64.48 / 62.21 | 70.94 / 55.36 |
| 평균 | 74.45% / 75.86% | 69.41% / 61.14% | 69.83 / 56.04 |

**표5. Batch Size에 따른 Accuracy**

본 실험에서는 Batch Size가 32 일 때 Validation Accuracy가 가장 높음을 알 수 있었다.

Ⅳ. 결과 및 최종 CNN 모델

제3장에서 진행했던 여러 변인들을 조작한 후 시행한 실험의 결과를 종합하여 성능이 가장 높을 것이라 예상되는 모델을 4가지로 추려 최종 모델을 결정하도록 하였다. 각각의 최종 CNN 모델의 후보는 다음과 같다.

|  |  |
| --- | --- |
| Layer 수 | 6 |
| Activation | Relu |
| Optimizer | Adam |
| Pooling Type | Max Pooling |
| Batch Size | 32 |
| Batch Normalization | O |
| Validation Accuracy | 75.02%/ 80.59% |

**표6. Model 1, epoch = 100**

|  |  |
| --- | --- |
| Layer 수 | 4 |
| Activation | tanh |
| Optimizer | Adam |
| Pooling Type | Max Pooling |
| Batch Size | 16 |
| Batch Normalization | O |
| Validation Accuracy | 73.87% / 78.29% |

**그림10. Model 2, epoch = 100**

|  |  |
| --- | --- |
| Layer 수 | 4 |
| Activation | Relu |
| Optimizer | Adam |
| Pooling Type | Max Pooling |
| Batch Size | 32 |
| Batch Normalization | O |
| Validation Accuracy | 79.45% / 85.96% |

**표7. Model 3, epoch = 100**

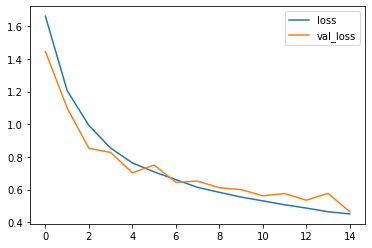
|  |  |
| --- | --- |
| Layer 수 | 6 |
| Activation | tanh |
| Optimizer | Adam |
| Pooling Type | Max Pooling |
| Batch Size | 32 |
| Batch Normalization | O |
| Validation Accuracy | 66.06% / 64.88% |

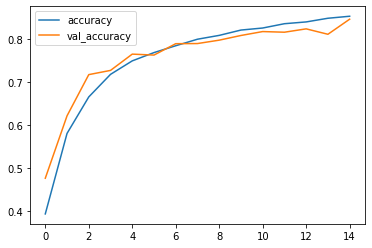
**8. Model 3, epoch = 100**

위와 같이 4가지 모델을 비교한 결과, Validation Accuracy가 가장 높은 Model 3을 최종 CNN 모델로 선정하였다.

Ⅴ. 결론

본 연구에서는 앞에서 보았던 것과 같이 CNN 모델을 구현하고 그 주요 함수나 Filter 사이즈 등을 조작하는 실험을 진행하였다. 그 결과, 제4장으로부터 Model 3의 정확도가 가장 높았기에 이를 최종 모델로 선정하였다. 최종적으로 도출된 CNN의 Validation Accuracy는 Fit generator 함수를 적용한 85.96%이다.

****

****

**그림4. 최종 CNN 모델의 Loss, Accuracy 값**

본 연구에서는 CNN 모델을 구현하고, 데이터성능을 높이기 위한 시도로 데이터를 증강하였으며, 각 단계의 주요 함수들이나 filter 사이즈 등을 바꾸어 실험해보았다. 이때 조작변인 하나씩을 기준으로만 성능을 비교했기 때문에, 두 개 이상의 변인들 사이의 연관성은 고려하지 못했다. 그리고 Data augmentation을 거친 후에 (미세한 차이지만) 오히려 accuracy가 감소한 경우도 더러 확인됐다. 또한 overfitting(과대적합) 현상으로 인해 train accuracy와 test accuracy의 차이가 존재했다. 이러한 현상을 해결하기 위한 연구가 필요하다는 결론을 내렸다. 더하여, 더 향상된 모델을 구현하기 위하여 CNN 이외에 더 다양한 구조의 모델을 구현해볼 수 있을 것이다.

**참고 문헌**

[1], [3], [4], [5], [7], [10]“CNN, Convolution Neural Network 요약”, http://taewan.kim/post/cnn/

[2] “Convolution”, https://ko.wikipedia.org/wiki/%ED%95%A9%EC%84%B1%EA%B3%B1

[6] “딥러닝 활성화 함수 종류 및 특징”, https://muzukphysics.tistory.com/entry/DL-3-%EB%94%A5%EB%9F%AC%EB%8B%9D-%ED%99%9C%EC%84%B1%ED%99%94-%ED%95%A8%EC%88%98-%EC%A2%85%EB%A5%98-%EB%B0%8F-%ED%8A%B9%EC%A7%95-Activation-Function

[8] “전형적인 CNN”, https://www.researchgate.net/figure/Architecture-of-our-unsupervised-CNN-Network-contains-three-stages-each-of-which\_283433254

[9] “Fully Connected Layer”, Wikipedia, https://en.wikipedia.org/wiki/Convolutional\_neural\_network#Fully\_connected\_layers

[11] “Part2. CNN 핵심 기술 – Dropout [1]”, http://blog.naver.com/laonple/220818841217