[Project #2]

Deep Learning for

Image Classification

[Team Name (12팀)]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **이름** | **학번** | **학년** | **E-mail** |
| 팀장 | 김소현 | 20192812 | 3 | jeviousso@nate.com |
| 조원1 | 김승현 | 20192813 | 3 | ttrs026017@gmail.com |
| 조원2 | 양조은 | 20192847 | 3 | jxxxxharu@gmail.com |
| 조원3 | 이여진 | 20190773 | 3 | yeojinlee.dev@gamail.com |

1. **Introduction**

본 프로젝트는 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.

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 학습 시키고 정확도를 측정한다.

프로젝트의 핵심 모델인 CNN은 쉽게 말해 이미지로부터 여러가지 특징을 low level(여러 단순한 형태의 선)부터 high level(선이 모인 어떤 형태의 모양)까지 찾아내 이미지를 추론하는 딥러닝과정이다. 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. 이에 본 프로젝트는 각각의 Layer에서 여러 가지 매개 변수 값을 테스트해보고, 다양한 technique을 적용하여 가장 최적의 CNN 모델을 구현하는 것을 목표로 한다.

1. **Main Structure**

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. CNN 모델 구현을 위해 Tensorflow (Keras)에서 제공하는 함수들을 사용하였다.

<https://github.com/berkantbayraktar/CIFAR10-Classification-Tensorflow.git> 참고

**1. Convolution Layer**

첫 번째 Convolution Layer에서는 filter을 이용해 이미지 특징(feature)을 계산하고, 그 계산 결과를 activation function을 이용해 정리함으로써 특징을 추출한다.

**-Conv2D function**

**- Arguments**

(1) strides : (1,1)

(2) padding : same (입력 이미지의 가장자리에 0값을 추가)

(3) activation(활성화 함수) : relu

**2. Batch Normalization**

**- BatchNormalization function**

- mean output은 0, output standard deviation은 1에 가깝게 유지하도록 하여 vanishing, exploding gradient 문제를 방지하도록 함.

**3. Pooling Layer**

Pooling layer에서는 Convolution layer의 계산 결과(feature maps)를 pooling(sub-sampling) 한다.

**- MaxPooling function**

**- Arguments**

(1) pool\_size : (2,2)

(2) strides : 2

**4. Fully Connected Layer**

**- Flatten function**

**- Dropout function**

(1) rate : 0.5, 0.25, 0.2

**- Dense function**

(1) dimensionality of the output space : 1024, 128, 10 (분류할 class의 수)

(2) activation : relu

**- Softmax activation**

추출된 특징 값을 확률을 모방한 일정한 값으로 변환하기 위한 함수이다.

**5. Optimizer**

- Adam : 파라미터 당 개별 학습 속도를 계산한다. 가장 빠른 최적화 알고리즘 중 하나이다.

- RMSProp : 데이터가 노이즈를 일으키는 경우 잘 작동한다.

- Adamax : 희소 데이터 세트에서 잘 작동하는 Adam Optimizer의 한 형태이다.

- SGD : 개별 데이터를 사용하여 실시간 학습이 가능하다.

**6. Data Augmentation**

Data augmentation(데이터 증강)은 이미지를 여러 방법을 통해 변형(transform) 한 뒤에 네트워크의 입력 이미지로 사용하여 CNN의 성능을 높이기 위한 방법으로 쓰인다. 이미지 데이터를 학습할 때 실시간으로 data augmentation을 할 수 있도록 지원하는 Keras의 ImageDataGenerator를 사용하였다.

1. **Training the Network**

Cifar10 dataset에서 50,000장의 이미지를 로드하여 네트워크를 훈련시키는데 사용하였다. 네트워크를 훈련시키는 과정의 핵심은 optimizer이다. Deep Learning, 즉 CNN에서 optimizer는 학습속도를 빠르고 안정적이게 하기 위해 사용된다. 본 프로젝트에서 optimizer는 SGD, Adam, RMSprop를 적용하였으며 이 중 성능이 더 좋은 Adam(가정)을 최종적으로 선택하였다. 세 가지 optimizer의 성능 평가는 4번 Experimental Results에서 다시 확인하도록 한다.

또한 과적합(Overfitting)을 방지하기 위하여 Dropout을 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.

참고 : <http://blog.naver.com/laonple/220818841217>

1. **Initial Experimental Results**

초기에 구현한 모델에서 여러 요인들을 조작한 뒤 정확도를 비교하였다.

연구 변인으로 Convolution layer 수, Activation function 종류, Optimizer 종류, Pooling Function 종류, Batch Size로 설정하여 결과를 비교해보았다.

**4.1 Test Environment**

* **Google Colab Notebooks**
* **Jupyter Notebook 6.1.4**

**4.2 Experimental Process**

- 비교 기준이 되는 CNN 모델의 기본 세팅은 다음과 같다.

> Layer 수 : 6 Layers

> 활성화 함수 (Activation) : Relu

> Optimizer : SGD

> Pooling Function : MaxPooling

> Batch Size : 32

> 정규화 (Normalization) 여부 : None

> epoch : 15

- 각각의 테이블의 값은 정확도(validation\_accuracy)이며, (fit\_generate를 적용하지 않았을 때) / (fit\_generate를 적용했을 때)로 표현하였다.

**# Layer 수 - 4 / 6 / 8 / 11**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 4 | 6 | 8 | 11 |
| 1 | 75.95% / 72.22% | 76.01 / 74.34 | 70.58 / 68.77 | 65.94 / 63.13 |

**# Activation function - ReLu / tanh**

|  |  |  |
| --- | --- | --- |
|  | 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% |  |
| 평균 | 74.45% / 75.86% |  |

**# Optimizer - SGD, Adam, RMSprop**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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% |

**# Pooling - MaxPooling, AveragePooling**

|  |  |  |
| --- | --- | --- |
|  | 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% |

>> Max Pooling의 경우, Feature Map을 M x N 크기로 잘라낸 후 그 안에서 가장 큰 값을 뽑아내는 방법이며, 반면 Average Pooling은 잘라낸 Feature Map에서 존재하는 파라미터 값만 뽑아내는 방법이다. 위의 실험 결과 본 CNN 모델에는 MaxPooling 방식을 사용하는 것이 더 좋은 성능을 보였다.

참고: <https://eehoeskrap.tistory.com/281>

**# Batch Size - 32 / 100 / 300 / 500**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 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 |

**5 Ablation Studies & Final Cnn Model**

**5.1 Ablation Studies**

- Deep Learning에서는 vanishing, exploding gradient 문제가 자주 발생하는데, 그 이유는 Layer 수가 적은 경우는 그 문제가 심각하지 않지만 Layer 수가 많아지면 누적되어 나타나기 때문에 이러한 문제가 성능에 영향을 끼칠 수가 있다. 이러한 문제에 대한 해결책으로 ReLU(Rectifier Linear Unit)을 활성함수로 사용하여 문제가 완화되기는 하였지만 본질적인 해결책이 아니었다. 따라서 본 프로젝트에서는 Batch Normalization을 선택하여 이러한 vanishing, exploding gradient 문제를 해결하도록 하였다.

Batch Normalization이란 각각의 스칼라 Feature들을 독립적으로 정규화하는 방식으로 진행된다. 즉, 각각의 Feature들의 Mean 및 Variance를 0과 1로 정규화를 하는 것이다.

- 본 프로젝트에서 Batch Normalization은 Tensorflow에서 제공하는 BatchNormalization()을 사용하였으며, Conv2D() 함수 다음에 배치하도록 하였다.

- 다음은 BatchNOrmalization()의 유무에 따른 Validation Accuracy를 측정한 결과이다.

**# 기본 세팅**

> Layer 수 : 6 Layers

> 활성화 함수 (Activation) : Relu

> Optimizer : SGD

> Pooling Function : MaxPooling

> Batch Size : 32

> 정규화 (Normalization) 여부 : None

> epoch : 15

* BatchNormalization() 적용 안했을 때 : 74.82% / 80.31%
* BatchNormalization() 적용 후 : 79.45% / 85.96%

따라서 본 연구를 통해 BatchNormalization을 사용하면 정확도가 높아지며 성능이 높아진다는 것을 알 수 있었다.

**5.2 Final CNN Model**

- 위 실험을 토대로 결과를 종합하여 성능이 높을 것이라 예상되는 모델을 4가지로 추려 최종 모델을 결정하였다.

**Model 1**  > epoch : 100

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

**Model 2**  > 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% |

**Model 3**  > 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% |

**Model 4**  > 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% |

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

**6 Discussion and Conclusion**

본 프로젝트에서는 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 필요하다.

더하여, 더 향상된 모델을 구현하기 위하여 CNN 이외에 더 다양한 구조의 모델을 구현해볼 수 있을 것이다.

불태웠다...