Kaggle - Dacon 과 하는 머신러닝
Team Project

정준상
송원영
장원
전우철

목차

1.주제 선정, 선정 이유

주제 선정

Kaggle에 있는 Digit Recognizer과 A-Z Handwritten Alphabets 데이터셋을 활용하여 입력된 손글씨에 대한 알맞은 index(대문자 A-Z, 숫자 0-9)를 부여할 수 있는 모델 구현 및 평가

주제 선정 이유

강의 시간에 MNIST – fashion 데이터셋을 이용하여 옷의 종류를 분류하는 모델을 구현했던 실습을 해본 이후 직관적으로 보았을 때 수치가 아닌 이미지 데이터를 처리하는 알고리즘에 대해 관심이 생겼다. 이에 대해 공부해보던 중 DNN에 convolution/pooling layer를 적용시킨 CNN이라는 모델에 대해 알게 되었고 인간처럼 이미지 판별을 각 부분에 대한 특징점으로 한다는 점이 신기했고, 이 모델을 사용하여 응용하는 사례를 찾아보게 되었다. 그 중 가장 간단한 것으로 MNIST 데이터셋의 0부터 9, A부터 Z까지 있는 손글씨 데이터셋과 이에 대한 competition이 kaggle에 있었고 이를 활용하여 우리는 임의의 손글씨 이미지가 입력되었을 때 이것을 숫자와 영어 대문자로 분류하고 인덱싱할 수 있는 모델을 만들어보고자 하였다. 이에 대한 활용은 크게 두가지로 기대되는데, 첫째는 스마트폰이나 패드에서 전자 펜으로 손글씨를 입력했을 때 글자로 바꿔주는 어플리케이션이고 두번째는 아직 전산화되지 않은 이전 문서(ex. 고대 문서)들을 전산화하는 과정에서의 활용이다. 전자의 경우는 예견 시간이, 후자의 경우는 정확도가 더 중요한 요소일 것이라고 추측되고 따라서 우리는 이 두가지를 모두 고려하여 다양한 모델을 구현해보고 파라미터를 조절해보고자 한다.

또한 이미지는 색이 포함될 수도 있는데, 이 경우 convolution layer가 2차원 픽셀 수치 + RGB 값을 포함하게 되어 3차원 Tensor가 된다. 우리는 모델과 파라미터를 변경해가며 많은 시도를 해보고 최적의 모델을 찾는 것에 집중하고자 running time을 줄이기 위해 흑백인 데이터를 선정하였다. 또한 벤치마킹 데이터셋인 MNIST 데이터를 사용하였으므로 평가 지표를 통한 모델의 평가가 타 모델과의 비교에 있어서 합리적일 것이라 기대한다.

2. 평가지표 기준 제시

1. 작성한 코드가 정상적으로 작동하여 실제 손 글씨를 인식하고 무엇인지(숫자, 알파벳) 예측할 수 있는가?

2. F1 score가 얼마나 높은가?

TP: True Positive

TN: True Negative

FP: False Positive

FN: False Negative

정밀도= $FP/(TP + FP)$

재현율= $TP/(FN + TP)$

F1 score (조화평균)= 2(정밀도 × 재현율)/(정밀도 + 재현율)

3, 손 글씨를 인식하고 예측하는데 걸리는 시간이 얼마나 걸리는가?

4. F1 score와 데이터 인식, 예측 시간을 사용자의 요구에 맞춰 유동적으로 조절할 수 있는가?

5. kaggle에 직접 작성한 코드를 올려 코멘트를 받는다.

3. 사용 데이터 분석

1)Introduction of Datasets

(1)

Competition : Digit recognizer - 손글씨의 10개의 문자(0~9)를 보고 무슨 수인지 예측하는 모델을 만드는 Competition이다.

URL: https://www.kaggle.com/competitions/digit-recognizer

Data overview :

데이터 파일 구성 : train_data, test_data

데이터 저장형식 : csv

데이터 파일구조분석 : (데이터프레임을 기준으로 설명)

각 파일의 Index 개수는 42,000개, 28,000개이다. 각 데이터가 가지는 샘플의 개수와 동일하다.

각 파일의 Columns 개수는 785개와 784개이다. 손글씨 데이터는 28×28개의 pixels로 구성되어 있고, 각 픽셀은 0~255까지의 int형 데이터로 밝기를 나타낸다. 따라서 784개의 columns이 각 픽셀의 밝기의 정보를 가지고 있다. 단, train_data을 사용할 때는 학습을 목적으로 하기에 각 샘플이 어떤 수임을 나타내는 정보를 나타내는 것을 하나 더 가지고 있어야하므로 'label'이라는 이름의 columns을 추가로 가진다. 이것은 0~9의 데이터를 가지며 숫자 그대로의 의미를 가진다.

이때 column name은 pixel0, pixel1,pixel2, … 로 구성되어 있다.

(2)

Data : A to Z Handwritten Alphabets Dataset in csv

URL :

https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format

Data overview :

데이터 파일 구성 A_Z Handwritten Data

데이터 저장형식 : csv

데이터 파일구조분석 :

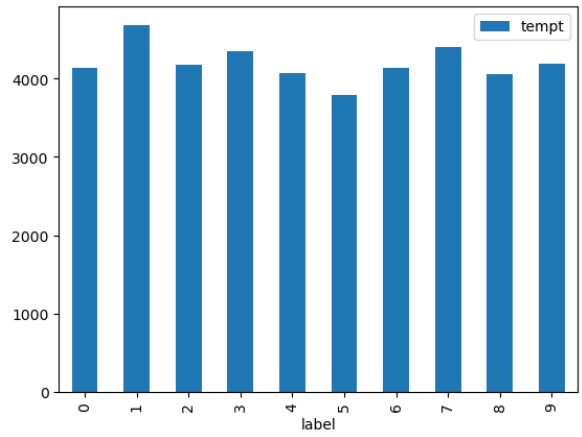
이 파일의 Index와 Columns의 개수는 각 372,450개, 785개이다.

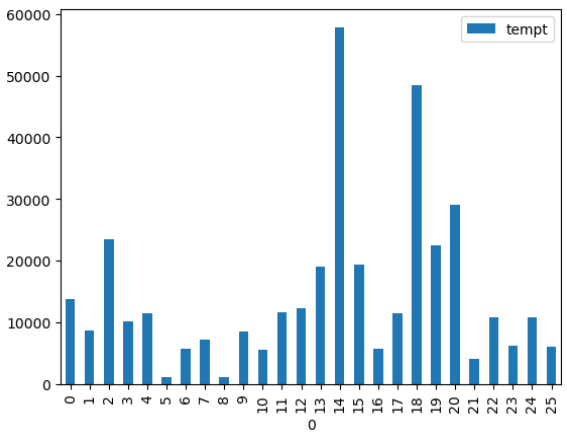
먼저, Columns의 경우 pixel의 개수 784개와 label의 역할을 하는 Column 1개로 구성되어 있음을 알 수 있다. 단, 데이터 탐색 결과 Column name에 대한 전처리는 필요하다.

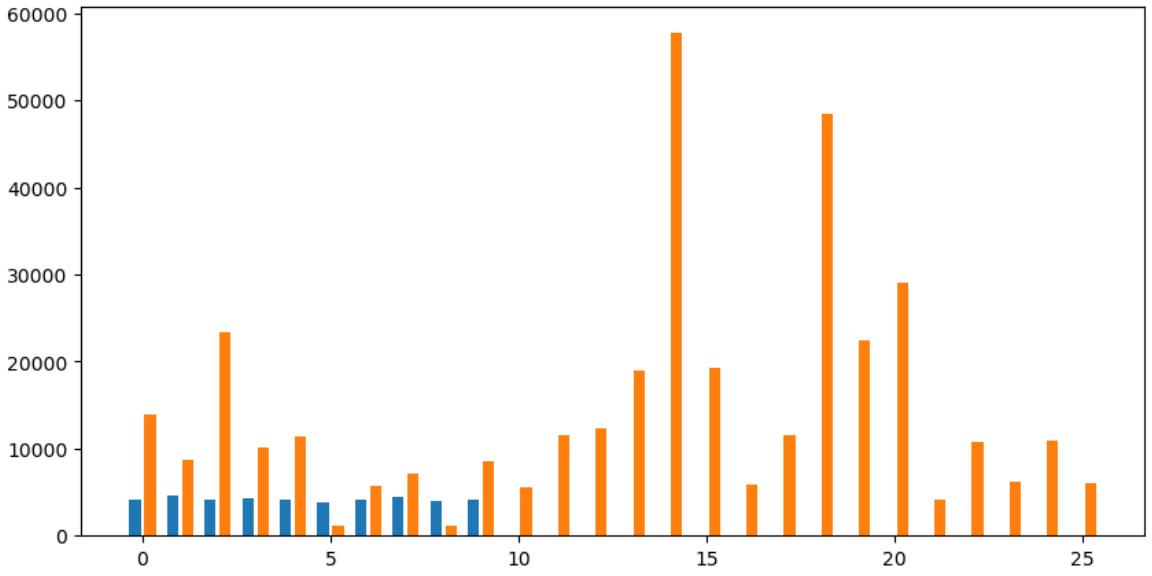
2) Data Exploration

각각의 데이터를 plot을 하는 것은 의미가 있다. 우리가 구현할 모델을 학습시킬 데이터가 한 특정값이 부각된다면 올바르지 않은 예측을 할 확률이 올라간다. 다음의 그림은 차례대로 다음과 같다 : 1) 0부터 9까지의 손글씨에 대한 sample 개수, 2) A to Z sample의 개수인데, 0은 A와 대응된다. 3) 두 Dateset의 label별 개수 비교이다.

Alphabets에 대한 sample이 다소 균일하지 않으며, 특히 둘을 모두 놓고 비교하면 Sample의 개수 차이가 많이 나서 이 데이터를 그대로 사용할 경우 모델이 적절히 학습하지 못하고 올바르지 않은 예측을 할 것이라고 예상할 수 있다. 따라서 Keras의 이미지 데이터 전처리 package ImageDataGenerator를 사용하여 데이터의 개수를 균일하게 조정하는 과정을 진행한다.







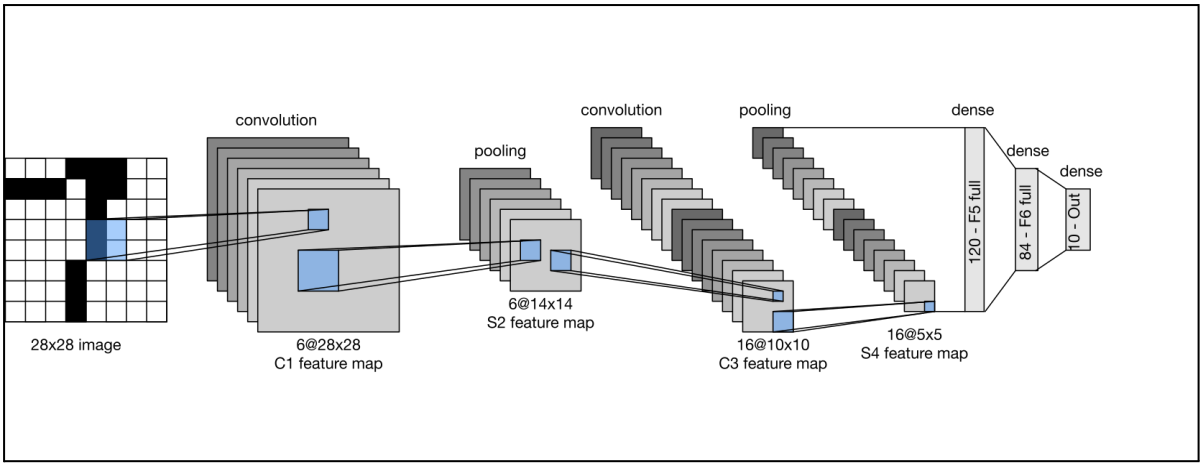
4. 사용 알고리즘 제시

우리는 딥러닝 구조 중, CNN (Convolution Neural Network) 를 이용한다. CNN 은 크게 다음 3 가지 구조로 나눠진다.

- Convolution layer
- Pooling layer
- Fully connected layer

1. 전체적인 구조 및 간단한 설명

아래의 그림은 CNN 중 유명한 LeNet-5 CNN Architecture 이다. 그림의 가장 왼쪽은 학습시키고자 하는 이미지 데이터가 있고, 가장 오른쪽에는 이미지를 통해 결과를 예측하는 부분이 있다. 이 두 부분 사이에 존재하는 층이 Convolution layer, Pooling layer, Fully connected layer (dense layer) 이다.



2. Convolution layer

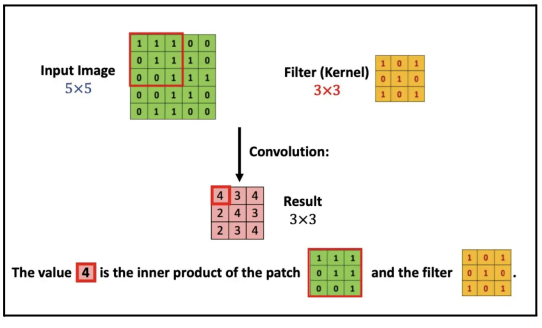
Convolution layer 를 알아보기 전에, 먼저 convolution 이 무엇인지 알아보자. Convolution 이란 “합성", 또는 “합성 곱" 이란 뜻을 가지고 있다. 다음과 같은 두 행렬을 생각해보자.

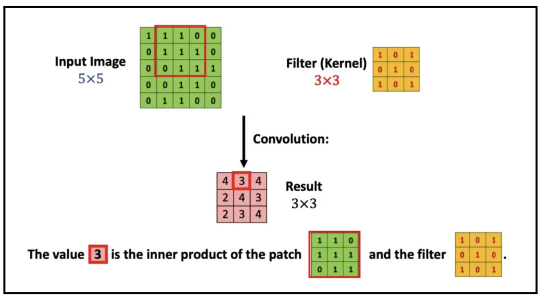
- $\mathbf{A} = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix}$ and $\mathbf{B} = \begin{bmatrix} 5 & 7 \\ 6 & 8 \end{bmatrix}$.
- Inner product:

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_i \sum_j a_{ij} b_{ij} = 70.$$

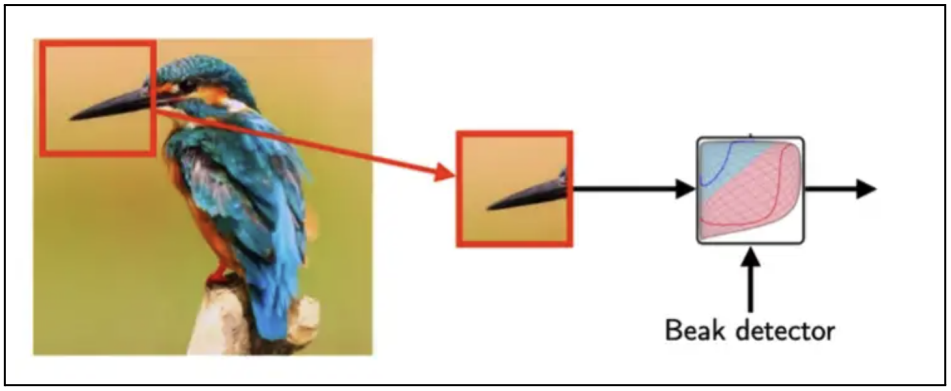
위처럼 두 행렬이 있을 때 행렬간 합성곱은 행렬간 내적한 값과 같다. 그런데 이와 비슷한 과정이 convolution layer 에서 일어난다.

우리의 이미지 데이터가 각 pixel 의 밝기 또는 색상을 나타내는 행렬로 주어졌다 생각하자. 그러면 CNN 모델에서 Filter 라 하는 어떤 특수한 행렬로 이미지 데이터를 분석할 수 있다.





위 그림처럼 이미지 데이터가 있을 때, CNN 모델은 Filter (또는 kernel) 를 통해 “Feature Map” (Result) 을 얻는다. Feature Map 이란, 이미지 데이터가 있을 때 “이미지에 대한 어떤 특징을 뽑아놓은 행렬" 이라 말할 수 있다.

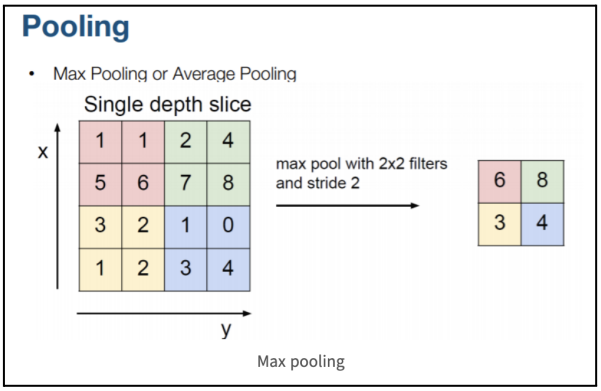


결국 CNN 의 Convolution layer 는 “이미지 데이터의 특징을 정리하는 층" 인 것이다.
CNN 모델을 통해 학습시키는 경우, “몇가지 특징을 뽑아낼지" (Filter 의 개수), “어느 크기 단위로 뽑아낼지" (Filter 의 크기, kernel_size), “Filter 의 step size” (stride) 등에 따라 모델의 복잡성이 달라진다. 또한 Convolution layer 를 여러개 배치시켜 모델의 복잡성을 조절할 수 있다.

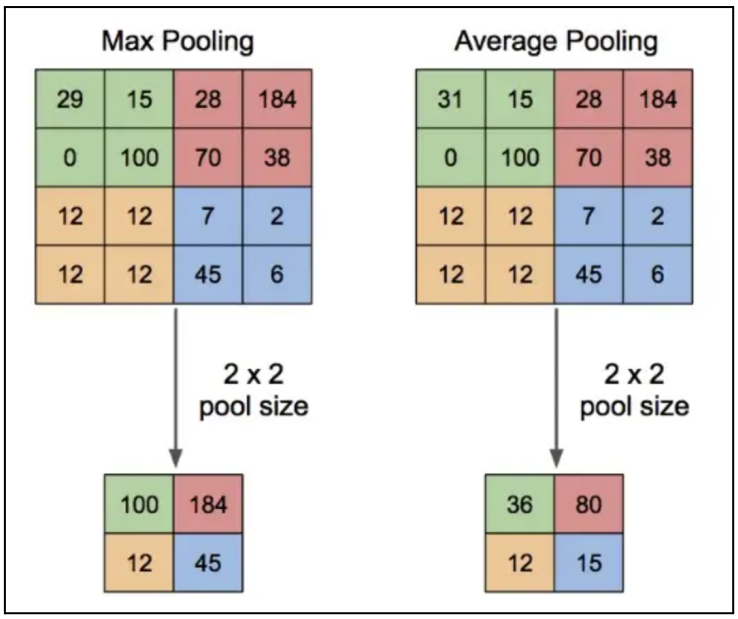
추가적으로 Filter 의 행렬값은 모델이 학습하며 달라지며, 모델의 비선형성 (non-linearity) 를 높이기 위해 Dropout 방법, relu 와 같은 활성함수가 쓰일 수 있다.

3. Pooling layer

앞서 Convolution layer 를 통해 이미지 데이터에서 특징 (Feature Map) 을 뽑아낼 수 있었다. Pooling layer 는 이 Feature Map 을 한번 단순화 시켜주는 과정이다.
Convolution layer 에서 Feature Map 은 Filter 의 개수만큼 새로 생성된다. 예를들어 Filter 가 10 개라고 생각해보자. 우리의 이미지 데이터는 “사진 1장” 이었다. 하지만 생성된 Feature Map 은 “10장" 이다. 하나의 데이터에서 너무 많은 데이터가 파생된 것이다.
이를 조절하기 위한 과정이 Pooling 이다. Pooling 이란 쉽게말해 “Feature Map 을 한단계 단순화 시켜주는 작업" 이다. 다음과 같은 Feature Map 을 생각해보자.



Feature Map 은 Pooling layer 를 통해 한단계 “압축" 된다. 주어진 Pooling size 에 맞춰 Feature Map 을 나눈 다음, 주어진 방식 (Max or Average, etc …) 에 따라 계산되어 더 작은 Feature Map 으로 변환된다.
대표적인 Pooling 방법으로 Max Pooling, Average Pooling 이 있다. 두 방법은 쉽게말해, “나눠진 구역 중, 최댓값으로 변환하냐, 또는 각 값들의 평균값으로 변환하냐” 이다.

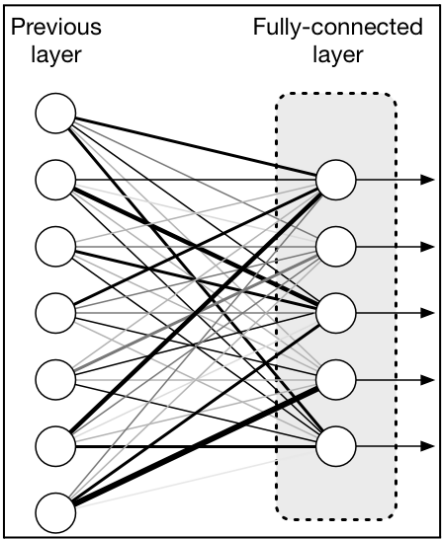


결국 Pooling layer 는 “Feature Map 을 한단계 정리해주는 층" 이라 할 수 있다.

추가적으로 Convolution layer 와 Pooling layer 의 개수를 조절해 모델의 복잡성을 결정지을 수 있고, 대채로 층 개수가 많아질수록 복잡해진다.

4. Fully connected layer

앞서 Convolution layer 와 Pooling layer 를 통해, 이미지 데이터의 특징을 추출, 정리할 수 있다. Fully connected layer 는 이렇게 정리한 특징을 “고전적인 딥러닝 방식" 을 통해 학습하는 층이다.



FC (Fully Connected layer) 는 이전 층의 퍼셉트론 (Perceptron) 들 과 다음 층의 퍼셉트론들이 모두 연결되어 있는 층이다. 각 퍼셉트론을 잇는 선은 가중치 (weight) 가 존재하여, 어느 자극을 “환산시켜" 다음 층으로 전달시킨다.

FC 를 통해 모델이 학습하는 과정을 쉽게 말해보자면, “주어진 가중치들로 먼저 예측을 내놓고, 답에 가깝도록 가중치와 편향치 (bias) 를 조절" 하는 방식으로 이루어진다.

우리가 사용하는 CNN 의 FC 의 경우, 결국 이미지를 분류하는 것이기 때문에, FC 마지막 층은 이전의 자극을 "정리" 해주는 활성화 함수와 10 개의 퍼셉트론으로 만들어진다.

추가적으로 FC 내부의 활성화 함수는 데이터의 특징에 따라 적절한 함수를 사용해야 한다. (데이터와 함수에 따라 기울기 소실 (Gradient vanishing), 기울기 폭주 (Gradient exploding) 이 일어날 수 있음.)

5. Summary
    a. CNN architecture
        i. Convolution layer
        ii. Pooling layer
        iii. Fully connected layer

    b. Convolution layer
        i. Input data → Feature Map
            1. Padding
            2. Convolution
            3. Filter
            4. Filter size
        ii. Feature Map
            1. Activation function

    c. Pooling layer
        i. Feature Map → smaller Feature Map
            1. Pooling size
            2. Pooling method
                a. Max Pooling
                b. Average Pooling
        ii. Flatten

    d. Fully connected layer
        i. Using ANN (Artificial Neural Network)
            1. Perceptron
            2. Weight and Bias
        ii. Proper activation function (+ optimizer etc…)
            1. Gradient vanishing
            2. Gradient exploding
        iii. Proper number of Perceptron and activation function to classify input data
            1. Digit recognizer : 10 Perceptrons
            2. Handwritten alphabet : 26 Perceptrons

6. Other keywords
    - Padding
        - 'same', 'valid', 'casual'
    - Flatten
    - Model optimizer
        - adam

- Proper activation function
    - relu, sigmoid, softmax