CNN 신경망 기반
숫자 인식 알고리즘

30128 조준범

목차

개요 / 동기

인공신경망 (ANN)을 기반으로 한 딥러닝 시장이 커지는 가운데, 그중에서도 각광 받고 있는 분야 중 하나가 CNN (Convolutional Neural Network)이다. CNN은 주로 이미지 학습, 영상처리를 위해 사용되며, 2차원 데이터의 입력과 훈련이 용이하기 때문에 객체 인식이나 컴퓨터 비전 등에서 주로 사용된다.

CNN을 포함한 딥러닝 기술이 사회 여러 분야에 적용되고 있으며 스스로 데이터간의 패턴을 찾아내고 특성을 분석한다는 것이 매력적으로 다가와 이에 대해 탐구하기로 결정하였다. 직접 알고리즘을 개발해보며 관련 개념 공부 및 딥러닝의 활용성을 체감해보고, 앞으로 나 스스로가 어떤 자세로 나아가야 할지 마음의 틀을 구성하고자 한다.

해당 탐구 활동은 CNN 신경망 기반 숫자 인식 알고리즘 개발을 주제로 숫자 이미지를 수집하고, 케라스 라이브러리를 활용하여 CNN을 포함한 딥러닝 층을 쌓고 학습을 진행하여 결론을 도출하는 순서로 진행된다.

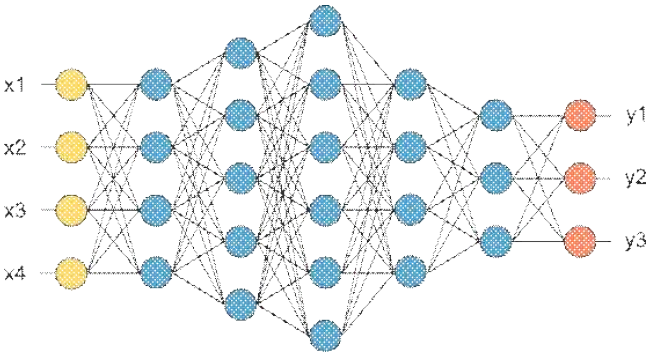
활동 중간에 층의 구조, 이미지 변형 등 여러 변수를 바꾸어가며 학습 모델을 생성한 후 가장 정확도가 높은 모델을 선정하기 위한 실험을 진행하며, 최종 모델로 숫자를 인식했을 때 옳지 않은 인식에 대한 분석을 진행하여 정확도를 높이기 위한 방법을 구상한다.

탐구 절차 및 계획

7월 18일 ~ 7월 30일	케라스 기반 머신러닝 공부
7월 30일 ~ 8월 8일	케라스 기반 CNN 신경망 공부
8월 2일 ~ 8월 8일	숫자 이미지 데이터 수집
8월 6일 ~ 8월 8일	CNN 신경망 알고리즘 개발
8월 7일 ~ 8월 8일	데이터 정리 및 전처리
8월 8일	학습 및 정확도 판단
8월 9일	결론 도출

CNN 신경망 원리 탐구

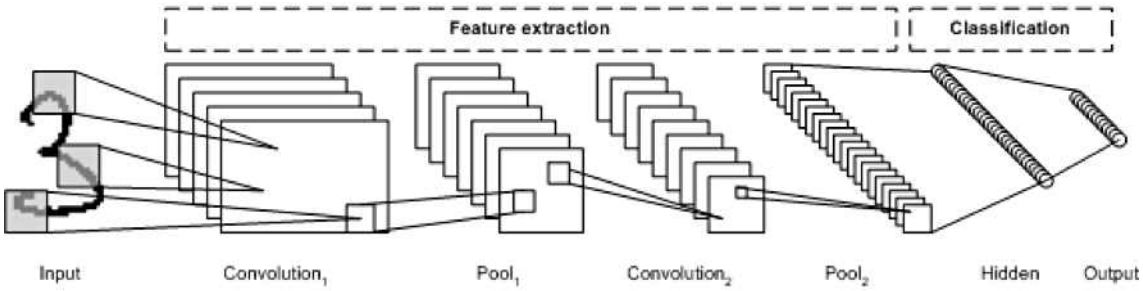
CNN 등장 이전의 이미지 인식은 3차원 (x, y, RGB 채널) 형태의 이미지를 1차원 형태로 변환시킨 후 FC (fully-connected) 레이어 방식으로 학습을 진행하였다.



입력, 은닉, 출력층으로 나뉘어 각 유닛끼리 정보를 주고받으며 학습을 진행한다. 일반적인 신경망의 형태이다.

FC 레이어는 1차원 데이터만 받을 수 있기 때문에, 3차원 데이터를 평탄화해서 입력해야 한다. 하지만 차원 변환 과정에서 3차원 데이터의 공간적 정보가 소실된다는 문제가 발생한다. 이 때문에 신경망이 특징을 추출하고 학습하는 과정이 비효율적이고 정확도를 높이는데 한계가 존재한다.

이런 단점을 보완하고 **이미지의 공간정보를 유지**한 채 학습을 이어나가는 모델이 CNN이다. CNN (Convolutional Neural Network) 의 학습 과정은 다음과 같다.



앞의 Feature extraction은 학습하는 단계 이전 이미지를 작게 자르고 각 부분의 특징을 추출하는 역할을 담당한다. 추출된 특성을 1차원으로 변환하여 FC layer에 입력하여 최종 학습을 진행한다.

1. Convolution : 이미지를 여러 부분으로 잘게 잘라 필터를 거쳐 행렬의 형태로 출력한다.
2. Padding : 입력 크기와 출력 크기를 같게 만들기 위해 유실된 외곽부분을 0으로 채운다.
3. Pooling : Convolution 단계의 출력인 Activation Map의 크기를 줄이거나
   특정 부분을 강조한다. 중요한 부분만 모아 새로운 표본을 만든다.
4. 학습 : 배열로 변환된 데이터를 FC layer를 거쳐 학습한다.

숫자 이미지 데이터 수집

학습에 필요한 이미지 데이터를 수집하는 과정이다.

물론 MNIST 데이터셋과 같이 연구를 위해 인터넷에 공개된 이미지들을 활용하면 더 좋은 결과가 나오겠지만, 실전에서는 직접 데이터를 수집, 분석, 변형하여 학습하기에 좋은 경험을 쌓고자 봉명고등학교 학생들의 손글씨 숫자 이미지 데이터를 이용하여 학습을 진행한다.

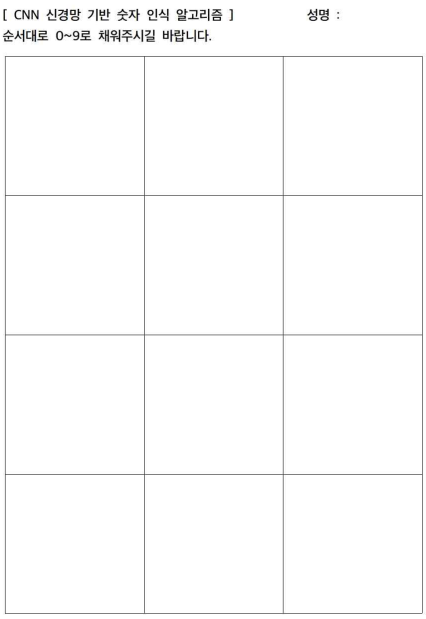
처음에는 웹사이트에 그림판 기능을 넣어 영역 내에 숫자를 그리고 저장하는 방식을 택했다. 하지만 개발해보고 나니 부드럽게 그려지지 않는 등 이대로 학습시켰다간 결코 좋은 결과를 얻을 수 없기에 이 방법은 과감히 포기하였다.

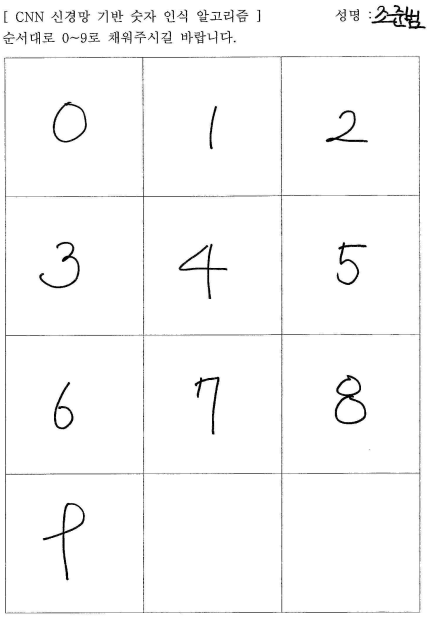
좋은 방법이 떠올랐다. 약간의 노가다가 필요하지만 그래도 처음의 방법보다는 확실하고 깔끔한 이미지들을 얻을 수 있다.

A4 용지에 칸을 나누고, 학생들이 직접 펜으로 각 칸 안에 숫자를 그려 넣으면, 이를 스캔하고 개발한 프로그램을 이용하여 각 칸에 있는 숫자들을 따로 저장하는 것이다.

우선 학생들이 숫자를 그려 넣을 틀을 제작하고 인쇄하였다. 이런 경우에는 데이터가 많을수록 학습의 정확도가 높아지므로 봉명고 학생과 교사를 대상으로 데이터를 풍부히 수집하였다.

각 숫자별로 100개, 총 1000개의 이미지 데이터를 활용하여 학습, 검증, 테스트를 진행한다.



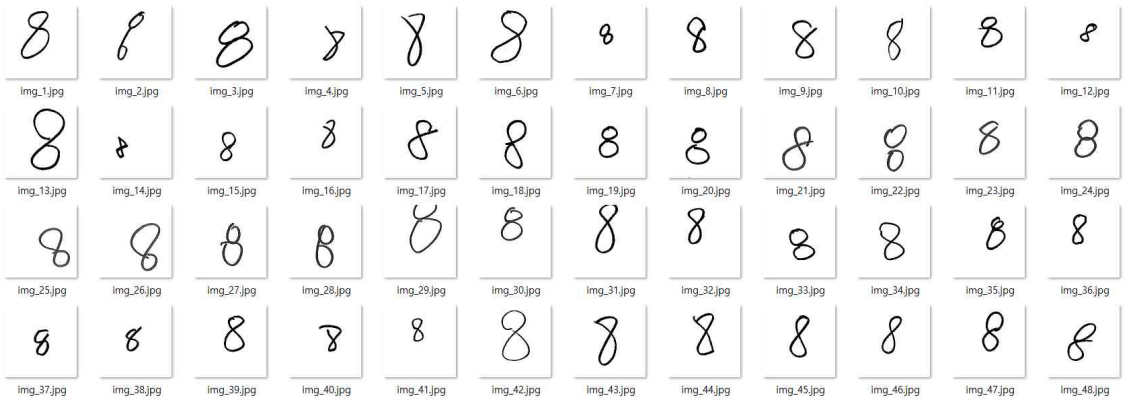


[프로그램 개발]

학생, 교사의 도움을 받아 손글씨 데이터를 수집하였으니, 이를 스캔하면 자동으로 각 항목별 (숫자별) 이미지를 저장하는 프로그램이 필요하다.

```
import cv2

for i in range(10):
    img = cv2.imread('scan/img_'+str(i+1)+'.jpg')
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img_resize = cv2.resize(gray, (int(img.shape[1] * 0.5), int(img.shape[0] * 0.5)))

    for k in range(4):
        for j in range(3):
            if(3*k + j == 10):
                break
        img_crop = img_resize[116+(168*k):284+(168*k), 21+(168*j):189+(168*j)]
        img_crop_2 = img_crop[10:158, 10:158]
        cv2.imwrite('images/'+str(3*k+j)+'/img_'+str(i*2+1)+'.jpg', img_crop_2)

    for k in range(4):
        for j in range(3):
            if(3*k + j == 10):
                break
        img_crop = img_resize[116+(168*k):284+(168*k), 618+(168*j):786+(168*j)]
        img_crop_2 = img_crop[10:158, 10:158]
        cv2.imwrite('images/'+str(3*k+j)+'/img_'+str(i*2+2)+'.jpg', img_crop_2)
```

파이썬 언어와 openCv 라이브러리를 이용하여 이미지 저장 프로그램을 개발했다.
컴퓨터가 0부터 9까지 총 10개의 칸을 가위질하고, 각 칸의 숫자를 이미지 형태로 정해진 폴더에 저장하는 것이라 이해하면 편하다.



데이터 수집은 모두 끝났다.
이제 본격적으로 CNN 신경망 알고리즘을 개발한다.

알고리즘 개발 및 학습

[데이터 분류]

인공지능의 학습을 위해서는 데이터를 필요에 맞게 세분화시켜야한다.
크게 학습(Train), 검증(Validation), 테스트(Test) 데이터로 나뉘는데,
총 1000개의 이미지 중 700개, 200개, 100개를 학습, 검증, 테스트용으로 이용한다.

검증과 테스트 데이터의 차이점은 검증 데이터는 학습에 직접적으로 관여하고, 테스트 데이터는 학습이 끝난 모델에 대한 정확도를 평가하는 용도이다.

학습 데이터를 이용하여 학습을 진행하고 검증 데이터로 학습의 정도를 평가하여 학습에 관여하는 변수인 가중치(weigh)를 조절해 다시 학습 데이터로 학습을 진행하는 과정이 반복되는데, 이를 몇 번 반복하는가를 에포크(epoch)라고 한다.

[모델 구조 설정]

이미지의 특성을 추출하는 CNN 부분과 학습을 진행하는 FC layer 부분으로 나누어 신경망 층을 생성한다. 케라스 라이브러리를 사용하면 다양한 층을 쉽게 쌓을 수 있다.

```
from keras import models
from keras import layers

model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), padding='valid', activation='relu', input_shape=(108,
108, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), padding='valid', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), padding='valid', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(256, (3, 3), padding='valid', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dropout(0.5))
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

4가지 모델을 생성하여 가장 성능이 좋은 모델을 채택하였다.
과대적합을 예방하기 위하여 Dropout 층을 추가하였고 층의 개수를 제한하였다.

[데이터 전처리]

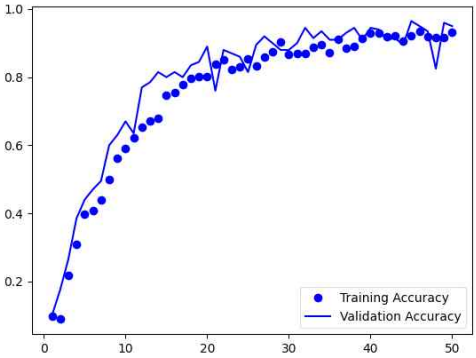
학습을 진행하기 위해 108x108 크기의 이미지의 RGB 값을 0~1 사이로 조정하는 전처리 단계이다. 추가로 좌우 최대 25° 회전하도록 설정하는 등 학습, 검증 데이터의 수를 늘렸다.

```
from keras.preprocessing.image import ImageDataGenerator

train_dir = 'train/img_train/'
valid_dir = 'train/img_valid'

train_datagen = ImageDataGenerator(rescale = 1./255, rotation_range = 25,
width_shift_range=0.2, height_shift_range=0.2)
valid_datagen = ImageDataGenerator(rescale = 1./255, rotation_range = 25,
width_shift_range=0.2, height_shift_range=0.2)

train_generator = train_datagen.flow_from_directory(train_dir, target_size = (108, 108),
batch_size = 20, class_mode = 'categorical')
valid_generator = valid_datagen.flow_from_directory(valid_dir, target_size = (108, 108),
batch_size = 20, class_mode = 'categorical')
```

[학습 및 훈련 및 검증 정확도 시각화]

학습 데이터와 검증 데이터를 통해 학습과 검증을 반복하여 최종 모델을 생성한다.
matplotlib 라이브러리를 사용하여 정확도를 시각화하여 학습의 정도를 파악할 수 있다.

```
history = model.fit_generator(train_generator, steps_per_epoch = 35, epochs = 50,
validation_data = valid_generator, validation_steps = 10)
model.save('model_final.h5')

import matplotlib.pyplot as plt

accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(accuracy) + 1)

plt.plot(epochs, accuracy, 'bo', label = 'Training Accuracy')
plt.plot(epochs, val_accuracy, 'b', label = 'Validation Accuracy')
plt.legend()
plt.show()
```



점 - 훈련 데이터에 대한 정확도
선 - 검증 데이터에 대한 정확도

학습이 진행될수록 두 데이터에 대한 정확도가 높아짐에 따라 성능이 향상되고 있음을 알 수 있다. 과대적합과 과소적합이 반복되지만 차이가 심하지 않아 추가 파라미터 조정 없이 활용한다.

정확도 판단 및 문제 분석

[최종 정확도 판단]

학습에 전혀 관여하지 않은 테스트 데이터를 모델에 입력 및 인식하여 최종 정확도를 판단한다. 테스트 데이터의 개수는 각 숫자별 10개, 총 100개이다.

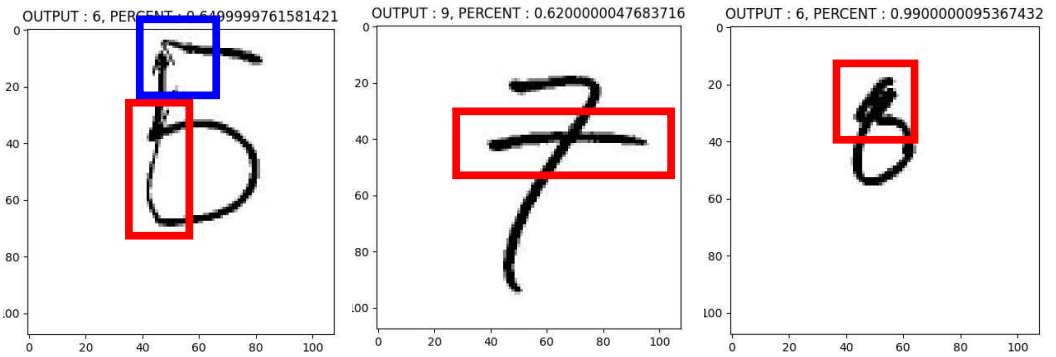
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from keras.models import load_model

def img_transform(img):
    image = cv2.imread(img)
    return (image/255)

test = []
name = []
for i in range(10):
    for j in range(91, 99):
        dir = 'train/img_test/img_'+str(j)+'_'+str(i)+'.jpg'
test.append(img_transform(dir))
name.append('img_'+str(j)+'_'+str(i)+'.jpg')

test = np.array(test)
model = load_model('model_final.h5')
prediction = model.predict_proba(test)

success = 0
for i in range(len(test)):
    output = prediction[i]
    percent = round(output[np.argmax(output)], 2)
    number = np.argmax(output)
    if(int(i/10) == number):
        result = 'true'
        success += 1
    else:
        result = 'wrong'
    plt.imshow(test[i])
    plt.title('OUTPUT : {0}, PERCENT : {1}'.format(number, round(percent, 2)))
    plt.show()
    print('{0} : {1} -> {2}'.format(name[i], number, result))

    print("정확도: {0}".format((success/len(test)*100)))
```

학습된 모델과 테스트할 이미지를 불러와 각 이미지에 대한 인식을 진행하였다.
100개의 이미지로 테스트를 진행하였을 때 **최종 정확도는 97.0%** 이다.

predict_proba() 함수는 각 클래스별 인식 확률을 출력하는데,
틀린 인식을 했을 경우 해당 이미지와 인식된 값과 그때의 확률을 출력하도록 하였다.

[결과 분석]

최종 인식의 정확도는 97.0% 이다.
틀린 인식을 시각화(이미지, 확률을 출력)하여 문제에 대한 분석을 진행한다.



답 : 5, 인식 : 6 (64%)	답 : 7, 인식 : 9 (62%)	답 : 8, 인식 : 6 (99%)

100개의 테스트 데이터 중 위의 3개의 이미지에 대한 인식이 옳지 않았다.

첫 번째 이미지의 경우,
5의 아래 **끝 부분에서 실수로 그어진 선**으로 인해 6으로 인식한 것으로 보인다.
또한, **왼쪽 상단의 경우 5는 각져있지만, 6은 둥글게 처리되어 있는 것**을 컴퓨터가 제대로 학습하지 못한 것도 인식 실패의 원인이 될 가능성이 존재한다. 데이터의 수를 늘리거나 CNN 층의 변수값을 조정하여 숫자의 특성을 더 정확하게 추출할 수 있도록 해야한다.

두 번째 이미지의 경우,
중간의 가로로 긴 선 때문에 9로 인식한 것으로 보인다.
학습 및 검증 데이터를 살펴보니, 7 또는 7 형태의 데이터만 존재하였고 위의 형태와 같은 7은 존재하지 않아 학습 자체가 불가능했다.

다시 말해, 다양한 종류의 7의 형태를 헤아리지 못한 것이다. 다른 숫자들도 이를 고려하여 데이터 수집 과정에서 더욱 많은 양을 모을 수 있도록 하여 자연스럽게 더 많은 형태를 고려할 수 있도록 해야한다.

세 번째 이미지의 경우,
8의 상단 부분을 원이 아닌 하나의 선으로 인식하여 6으로 인식한 것으로 보인다.
두 번째의 경우와 같은 이유로 인식에 실패하였으나, **8의 해당 부분**이 두꺼운 형태의 데이터를 추가로 수집하고 학습하면 해결될 것으로 보인다.

결론 도출

[딥러닝의 발전 방향]

2000년대에 들어서야 현실로 다가온 딥러닝 기술은 무궁무진한 활용성, 엄청난 성능과 미래가치를 바탕으로 무한한 상승 가도를 달리고 있다. 이는 IT 분야만의 이야기가 아니다.

의학, 농업, 생명 등 우리 삶에 직간접적으로 영향을 끼치는 많은 분야들과 딥러닝 기술을 접목시키려는 시도가 계속되고 있으며 이미 상당한 성과를 내고 있는 사례들 또한 풍부하다.

최근 세계적 IT 기업이 구글이 딥러닝 기술을 활용해 놀라운 결과를 발표했다. 인공지능 '알파폴드'가 유전정보와 단백질 구조 사이의 연관 패턴을 분석하여 단백질 구조를 분석했다는 소식인데, 비싼 장비를 이용하여 실제로 알아낸 구조와 유사한 결과를 도출했다.

위의 사례처럼 인간의 연산 속도를 뛰어넘어 데이터를 분석하여 규칙을 찾아내는 인공지능 산업은 아직까지 해결하지 못한 수 많은 문제들을 해결하는데 큰 역할이 될 것이며 우리가 상상하지 못한 미래를 만들어 낼 잠재성이 풍부하다.

앞으로 기업의 미래는 기업의 인공지능 기술력에 따라 좌우된다는 말은 과언이 아닐 것이다. 문제를 해결하는 데에 있어 효과적이고 합리적인 방법을 찾아낼 수 있을 뿐만 아니라 재정적 효율성을 상승시킬 수 있어 새로운 사업을 시작하는데 경제적 부담을 줄일 수 있기 때문이다.

기업은 인공지능을 개발하는 기술뿐만 아니라 데이터를 수집하고 가공하는 능력 또한 지녀야 하며 문제 해결에 있어 더 좋은 성능을 내는 모델을 개발하기 위해 끊임없이 노력해야 한다.

하지만 인공지능 기술이 발전함에 따라 윤리적인 문제와 맞닥뜨리는 것은 결코 피할 수가 없다. 데이터 수집 과정에서부터 활용까지 사람의 개입이 불가피하기 때문에 결국에는 사람의 주관이 반영될 수 밖에 없다.

절대적으로 완벽하고 공정한 인공지능은 존재하지 않는다. 하지만 사회가 원하고 지키려는 가치와 공감대는 뚜렷하다. '완전히 공정한 인공지능을 개발해야겠다' 라고 생각하기보다는 인권과 자유 등 현 사회에서 중요시되는 사회적 가치가 무엇인지 파악하고 이들을 최대한 균형 있게 반영하여 끊임없이 검토하며 나아가야 하는 자세가 필요하다.

[최종 결론]

직접 데이터를 수집하고, 정리하고, 알고리즘을 개발하여 학습과 추가 실험까지의 길은 결코 쉬운 모험이 아니었다. 하지만 딥러닝이 무엇인지 자세히 알아보았고, 앞으로 딥러닝을 중심으로 사회가 어떻게 변화할지 예상해볼 수 있는 충분히 가치 있는 활동이었다고 생각한다.

총 4개의 모델을 생성하여 테스트하였다. 서로 층의 개수와 형태가 다르거나, 규제를 가했거나, 유닛의 수를 변경했느냐 등의 차이였는데, 결국에는 데이터를 변형하여 수가 많게 늘린 것이 정확도가 높았다.

틀린 인식을 분석할 때도 결국 데이터의 수를 늘려야 해결이 가능하였는데, 해당 탐구를 통해 정보화 시대에서의 데이터의 중요성을 다시금 체감할 수 있었다.

프로그램을 개발하는데에 있어, 인간관계 등 살아가는데에 있어 많은 정보를 수집하되 무분별하지 않은지, 윤리적이고 도덕적으로 모으고 있는지 반복 점검하며 효율적이고 실용적인 프로그램을 만들고, 주어진 상황에 따라 모두의 이득을 고려한 적절한 판단을 하고자 한다.

새로운 것을 즐길 수 있고, 받아들일 준비가 된 사람만이 빠르게 변하는 세상에 적응할 수 있다. 인공지능 등 여러 기술들이 빠르게 개발, 발전되고 있고 이러한 기술들에 의존할 수 밖에 없는 오늘날 감정을 지닌 산물인 우리가 인공지능 사회에 있어서 무엇을 할 수 있는지 끊임없이 고민하고 실천하는 것이 우리가 새로운 사회를 받아들이는 방식이 될 것이다.

후퇴가 무조건적으로 나쁜 것은 아니다. 다만 충분히 역량이 됨에도 불구하고 도전을 하지 않는 것은 출발선에서 가만히 있는 것이나 마찬가지이다. 실패도 곧 도전의 일부분임을 기억하고 어둠 속에서 빛을 찾아 한 발짝 더 내딛어 보자.

참고 문헌 및 출처

[개념 관련]

http://theyoonicon.com/10-convolutional-neural-networkscnn-%ED%95%99%EC%8A%B5%ED%95%98%EA%B8%B0/

https://yeo0.github.io/data/2018/10/05/CNN-%ED%8A%B9%EC%A7%95-%EC%B6%94%EC%B6%9C-%EA%B3%BC%EC%A0%95/

[이미지]

https://miro.medium.com/max/1108/1*Mw6LKUG8AWQhG73H1caT8w.png

https://taewanmerepo.github.io/2018/01/cnn/head.png

[프로그램 개발]

케라스 창시자에게 배우는 딥러닝 (길벗 출판사)

파이썬과 OpenCV를 이용한 컴퓨터 비전 학습 (에이콘 출판사)

https://twinw.tistory.com/252

https://tykimos.github.io/2017/06/10/Model_Save_Load/

https://acdongpgm.tistory.com/169

[딥러닝의 활용 사례와 윤리]

https://www.youtube.com/watch?v=rbwHyiD_djg

http://www.biospectator.com/view/news_view.php?varAtcId=13782

https://horizon.kias.re.kr/17815/

SPECIAL THANKS TO....

이석화 선생님
차태호 선생님
류민아 선생님
서아람 선생님
최선우 선생님
이지혜 선생님
김민정 선생님
이민희 선생님
이승주 선생님
이은희 선생님

신용옥
김민섭
차정현
김성락
박성환
김송언
주낙운
홍성민
오세훈
정택영
강형택
이예찬
김지수
고호성
윤석준
이성률
이의범
조수길
김태환
신승민
안상현
원대람
윤수영
김영헌
오율리
김종현
임지영
박성재
김동민
박서진
정현규
성하민
심용구

조소연
어지연
김민
정문희
박규리
박지윤
김은정
이지수