Smart Media Journal / Vol.10, No.1 / ISSN:2287-1322
https://dx.doi.org/10.30693/SMJ.2021.10.1.32

VGGNet을 활용한 석재분류 인공지능 알고리즘 구현

(Implementation of the Stone Classification with AI Algorithm Based on VGGNet Neural Networks)

최경남*
(Kyung Nam Choi)

요 약

사진 이미지에서의 딥러닝 학습을 통한 이미지 분류는 지난 수년간 매우 활발한 연구 분야로 자리하고 있다. 본 논문에서는 국내산 석재 이미지로부터 딥러닝 학습을 통해 자동으로 석재를 판별하는 방법을 제안한다. 제안된 방법은 300x300픽셀의 황등석, 고흥석, 포천석의 사진 이미지들을 파이썬의 해시 라이브러리를 이용하여 석재별 중복된 이미지를 검사하고, 검사 결과로 해시값이 같은 중복된 이미지를 제거하여 석재별 딥러닝 학습이미지를 만드는 데이터 전처리 과정을 수행한다. 또한 미리 학습된 모델인 VGGNet을 활용하기 위해 학습된 이미지 사이즈인 224x224픽셀로 석재별 이미지들의 사이즈를 재조정하고, 학습데이터와 학습을 위한 검증데이터의 비율을 80% 대 20%로 나누어 딥러닝 학습을 수행한다. 딥러닝 학습을 수행한 후 손실 함수 그래프와 정확도 그래프를 출력하고 세 종류의 석재 이미지에 대해 딥러닝 학습 모델의 예측 결과를 출력하였다.

■ 중심어 : 딥러닝 ; 미리 학습된 모델 ; 데이터 전처리 ; 케라스 ; 파이썬

Abstract

Image classification through deep learning on the image from photographs has been a very active research field for the past several years. In this paper, we propose a method of automatically discriminating stone images from domestic source through deep learning, which is to use Python's hash library to scan 300x300 pixel photo images of granites such as Hwangdeungseok, Goheungseok, and Pocheonseok, performing data preprocessing to create learning images by examining duplicate images for each stone, removing duplicate images with the same hash value as a result of the inspection, and deep learning by stone. In addition, to utilize VGGNet, the size of the images for each stone is resized to 224x224 pixels, learned in VGG16 where the ratio of training and verification data for learning is 80% versus 20%. After training of deep learning, the loss function graph and the accuracy graph were generated, and the prediction results of the deep learning model were output for the three kinds of stone images.

■ keywords : DeepLearning ; VGG16 ; Data Preprocessing ; Keras ; Python

Ⅰ. 서 론

오늘날 인공지능 딥러닝 하드웨어 및 소프트웨어의 발달로 이미지를 인식하여 처리하는 딥러닝 영상처리는 다양한 영상처리 분야에서 매우 일반화되었으며, GPU 및 Tensorflow (Keras포함) 환경에서의 사용자들은 보다 손쉬운 방법으로 다양한 인공지능 알고리즘을 수행할 수 있게 되었다.

머신러닝 또는 기계학습은 인공 지능의 한 분야로, 컴퓨터가 학습할 수 있도록 하는 알고리즘과 기술을 개발하는 분야를 말한다. 가령, 기계학습을 통해서 수신한 이메일이 스팸인지 아닌지를 구분할 수 있도록 훈련할 수 있다. 기계의 핵심은 표현과 일반화(generalization)에 있다. 표현이란 데이터의 평가이며, 일반화란 아직 알 수 없는 데이터에 대한 처리이다. 이는 전산학습 이론 분야이기도 하며, 다양한 기계 학습의 응용이 존재한다. 영상 및 문자 인식은 이를 이용한 가장 잘 알려진 사례이다 [1].

* 정회원, 원광대학교 SW중심대학사업단 부교수
접수일자 : 2021년 02월 01일
수정일자 : 2021년 03월 05일
게재확정일 : 2021년 03월 12일
교신저자 : 최경남 e-mail : knchoi@nate.com

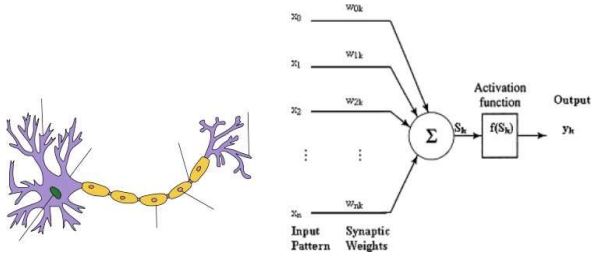


그림 1. Deep Learning의 신경망 개념도

딥러닝이란 여러 비선형 변환기법의 조합을 통해 높은 수준의 추상화(abstractions, 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 작업)를 시도하는 기계학습 알고리즘의 집합으로 정의 할 수 있다. 요약하면 사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야이다. 간략하게 그림으로 표현하면 그림 1과 같다.

대표적인 딥러닝 알고리즘으로는 영상처리에 주로 사용되는 합성곱 신경망(Convolutional Neural Network, CNN)과 연속적인 데이터 처리에 사용되는 순환 신경망(Recurrent Neural Network, RNN) 등이 있다[2].

이에 본 논문에서는 이러한 딥러닝 알고리즘을 근간으로 전라북도 익산의 대표석재인 황등석을 가공하여 파는 전라북도 익산에 위치해 있는 석재 가공회사의 요청에 의해 석재 가공회사가 석재의 종류를 석재를 납품하는 회사의 말만 믿고 정확히 황등석인지에 대한 검증도 없이(육안으로는 햇빛 반사등의 외부적인 환경영향으로 석재의 모양만 보고는 황등석과 비슷한 모양의 석재를 구별하기가 어려움) 석재를 납품받아 사용하는 현실적인 문제점을 개선하기 위해 딥러닝 기반 컴퓨터 비전 모델의 시대를 열었던 AlexNet(2012년)의 출현이후 영상처리의 대세로 자리잡은 딥러닝 영상처리 알고리즘을 통해 고해상도 카메라의 석재 이미지를 학습하여 석재의 종류를 자동으로 판별할 수 있는 석재분류 인공지능 알고리즘을 제안하게 되었다. 이를 석재가공회사에서 도입하면 납품받는 석재의 종류가 정확히 어느 종류인지 판별하게 되어 저가의 중국산이 고가의 국내산으로 둔갑하여 납품되어 사용되는 경제적인 폐해를 미연에 방지할 수 있다.

또한 본 논문에서는 인공지능 모델의 학습시간과 성능을 획기적으로 개선한 Neural Network Architecture로 ImageNet 영상 데이터 베이스를 기반으로 한 2014년 ILSVRC (ImageNet Large Scale Visual Recognition Challenge)에서 2위를 차지한 Oxford 대학교의 VGGNet을 사용하였다[3]. VGGNet은 1위를 차지한 구글의 GoogLeNet(inception)보다 구조적인 측면에서 훨신 간단한 구조를 가지고 있고 이해가 쉬으며 변형을 시켜가면서 테스트 하기에 용이하여 GoogLeNet보다 더 많이 사용되는 편이다.

(a)황등석 (b)포천석 ©고흥석

그림 2. 석재 사진 샘플

II. 본 론

1. 데이터 전처리

인공지능 모델 구현에 있어서 학습데이터를 만드는 일이 전체 일의 약 70%를 차지한다. 이에 본 연구에서도 학습데이터를 만들기 위해 석재사진집에서 황등석 사진 이미지 여러장을 카메라 각도를 달리하여 6,500장을 만들었으며, 똑같은 방법으로 고흥석 이미지 630장과 포천석 이미지 1,100장을 만들었다. 원본 이미지가 카메라 각도만 달리하여 만든 이미지여서 중복된 이미지가 많아, 중복된 이미지를 걸러내기 위해 해시함수를 사용하였다.

카메라로 찍은 300x300픽셀의 황등석, 고흥석, 포천석 사진 이미지(그림2)들의 이미지 중복성을 검사하기 위해 이미지를 해시 값으로 변환하는 함수의 파이썬 코드는 아래와 같이 구현하였다[8].

```
def get_hash_val(img):
    md5 = hashlib.md5()
    md5.update(img.tobytes())
    return md5.hexdigest()
```

해시를 이용한 석재별 중복된 이미지를 제거하는 함수의 파이썬 코드는 아래와 같이 구현하였다.

```
def check_duplicated(root_path):
    file_list = sorted(os.listdir(root_path))
    files = []
    hashes = []
    for file in tqdm(file_list):
        file_path = os.path.join(root_path, file)
        img = cv.imread(file_path)
        if type(img) == type(None): continue
        hash_val = get_hash_val(img)
        if hash_val in hashes: continue
        hashes.append(hash_val)
        files.append(file)
    return files, hashes
```

석재별 중복된 이미지를 제거한 학습데이터를 new_data 폴더의 각 석재별 폴더에 저장하였다. 새로 저장된 황등석 이미지

791개, 포천석 이미지는 136개, 고흥석 이미지 83개를 학습데이터로 사용하였다. 본 연구의 주제인 황등석 판별을 위해 황등석 이미지만 학습데이터로 사용하지 않고 포천석 및 고흥석 이미지를 황등석 학습데이터에 추가한 이유는 석재별 카테고리 분류가 아닌 황등석이냐 아니냐를 판단하는 바이너리 이진 분류 문제(황등석이 아닌 이미지도 학습해야 하므로)로 접근했기 때문이며 이에 대한 파이썬 코드는 아래와 같이 구현하였다.

```
if __name__ == "__main__":
    raw_data_root = "./data"
    new_data_root = "./new_data"
    stone_list = [stone for stone
    in os.listdir(raw_data_root)]
    for stone in stone_list:
        print(f"About {stone} data")
        src_stone_path=os.path.join(raw_data_root,
        stone)
        dst_stone_path=os.path.join(new_data_root,
        stone)
        os.makedirs(dst_stone_path, exist_ok=True)
        files,hashes=check_duplicated(src_stone_path)
        for file in files:
            src_path=os.path.join(src_stone_path, file)
            dst_path=os.path.join(dst_stone_path, file)
            shutil.copy(src_path, dst_path)
```

2. VGGNet의 구조

VGG-16 모델은 ImageNet Challenge에서 Top-5 테스트 정확도를 92.7% 달성하면서 2014년 컴퓨터 비전을 위한 딥러닝 관련 대표적 연구 중 하나로 자리매김하였다.

VGG 모델은 딥러닝 기반 컴퓨터 비전 모델의 시대를 열었던 AlexNet(2012)의 8-layers 모델보다 깊이가 2배 이상 깊은 네트워크의 학습에 성공했으며, 이를 통해 ImageNet

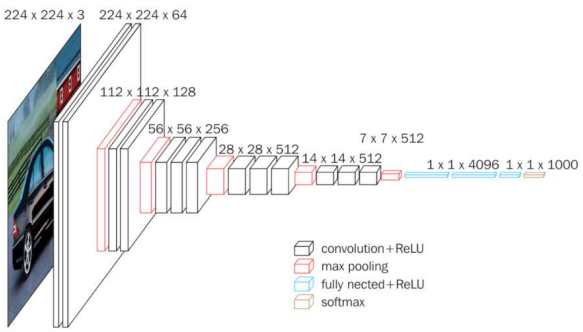


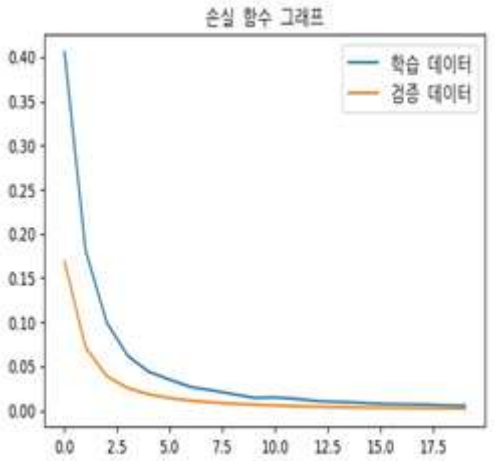
그림 3. VGG16 구조

Challenge에서 AlexNet의 오차율을 절반(16.4 > 7.3)으로 줄였다. VGG 모델이 16 - 19 레이어에 달하는 깊은 신경망을 학습할 수 있었던 것은 모든 합성곱 레이어에서 3x3 필터를 사용했기 때문이다(그림 3).

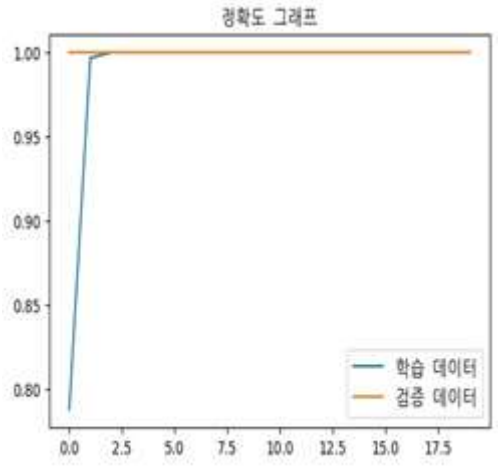
VGG 모델 이전에 Convolutional Network를 활용하여 이미지 분류에서 좋은 성과를 보였던 모델들은 비교적 큰 Receptive Field를 갖는 11x11필터나 7x7 필터를 포함한다. 그러나 VGG 모델은 오직 3x3 크기의 작은 필터만 사용했음에도 이미지 분류 정확도를 비약적으로 개선시켰다. 이 부분에서 좋은 통찰을 얻을 수 있다. VGG 연구팀의 실험 결과를 통해 네트워크의 깊이가 깊어질수록 이미지 분류 정확도가 높아지는 것을 확인할 수 있었다. 실험에서 네트워크의 깊이를 최대 19 레이어(VGG-19)까지만 사용한 이유는 해당 실험의 데이터에서는 분류 오차율이 VGG-19에서 수렴했기 때문이다. 학습 데이터 세트가 충분히 많다면 더 깊은 모델이 더 유용할 수도 있다[12].

기존의 연구에서와 마찬가지로 본 연구에서도 모델 학습(Training) 시 입력 이미지의 크기는 VGGNet의 학습에 사용된 이미지 사이즈인 224x224픽셀로 재조정하였다.

본 논문에서 제안한 방법은 VGG16과 VGG19를 다 사용할 수 있게 파이썬의 딕셔너리로 정의하여 구현하였고 학습에 사용될 학습데이터셋의 갯수가 많지 않은 관계로 레이어가 16개인 VGG16을 사용하였으며, 본 연구가 기존의 하이퍼 파라미터와 레이어를 조정해 가며 구현하는 인공지능 모델 구현기법과 달리 미리 학습되어 최적화된 VGG16 weight을 사용하여 하이퍼 파라미터와 레이어를 조정해 가는 번거로움 없이 인공지능 모델을 구현할 수 있다는 점에서 시사하는 바가 크다 할 것이다.



(a)



(b)

그림 4. 결과 그래프: (a)손실함수, (b)정확도

3. 딥러닝 학습

본 논문에서 사용되는 Keras는 파이썬으로 구현된 쉽고 간결한 딥러닝 라이브러리로 딥러닝 비전문가라도 각자 분야에서 손쉽게 딥러닝 모델을 개발하고 활용할 수 있도록 Keras는 직관적인 API를 제공하며, Tensorflow2.0부터는 기본적으로 Tensorflow에 내장되어 있다[10].

전체 석재 이미지 데이터에서 80%는 학습에 사용하는 데이터용으로 20%는 학습된 모델의 성능을 검증하여 하이퍼 파라미터(Hyper Parameter)를 조정하는 지표로 활용되는 검증데이터용으로 나누었다.

Keras에서 제공하는 ImageDataGeneraor를 이용하여 간단하게 학습용 데이터와 검증용 데이터를 로딩하였다[15]. 이를 구현한 파이썬 코드는 아래와 같다.

```
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
train_generator = ImageDataGenerator(rescale=1/255.)
valid_generator = ImageDataGenerator(rescale=1/255.)
IMG_SIZE = 224
train_loader=train_generator.flow_from_directory(train
,target_size=(IMG_SIZE,IMG_SIZE),batch_size=32,
    class_mode='binary',shuffle=True, )
```

그리고 Keras의 콜백함수인 ModelCheckpoint 함수를 사용하여 학습한 모델의 weight값을 *.h5 형식의 파일로 저장하였다.

```
filename = f'/stone/vgg16stone.h5'
checkpoint=ModelCheckpoint(filename,monitor='val_los
s',verbose=1,save_best_only=True, mode='auto')
```

그 다음으로 Loss(손실)는 대표석재인 황등석을 구별하는 이진분류 문제로 정의하였기에 binary cross entropy를 사용하였고 optimizer로는 Adam을 사용하여 학습을 진행하였다.

```
model.compile(loss='binary_crossentropy',optimizer='a
dam', metrics=['acc'])
history=model.fit(train_loader,epochs=20,
batch_size=32,validation_data=valid_loader,callbacks=[
checkpoint])
```

학습을 수행한 후 손실 및 정확도 그래프를 그림 4와 같이 출력하였다. 가로축은 epoch 20회를 수행한 횟수를 나타내고, 세로축은 손실률과 정합률을 나타낸다. 손실률은 epoch 횟수가 증가함에 따라 낮아지는 것을 볼 수 있는데, 이는 학습에 대한 손실률이 줄어들고 있어 학습이 잘 되고 있음을 나타내며, 정확도는 epoch 2회부터 100%에 이르는 것을 볼 수 있다. 이에 대한 코드는 아래와 같다.

```
path = './NanumGothicCoding.ttf'
fontprop = fm.FontProperties(fname=path, size=12)
plt.figure(figsize=(14, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['학습 데이터','검증 데이터'],loc=0,
prop=fontprop)
plt.title("손실 함수 그래프", fontproperties=fontprop)
plt.subplot(1, 2, 2)
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.legend(['학습 데이터', '검증 데이터'],loc=0,
prop=fontprop)
plt.title("정확도 그래프", fontproperties=fontprop)
os.makedirs('./figures', exist_ok=True)
plt.savefig(f"./figures/{model_name}_training_graph.jpg")
```

표 1. 황등석일 확률 예측치

석재 종류	황등석으로 인식할 확률(%)
황등석	93.07
포천석	21.73
고흥석	59.32

4. 딥러닝 추론

checkpoint 함수로 저장한 weight을 불러와서 세 종류의 석재 이미지에 대해서 예측을 하였고 그 결과는 표 1과 같고,

이 결과를 얻은 윈도우즈 버전 코드는 아래와 같다.

```
def preference(model):
    model = models.load_model('./vgg16stone.h5')
    path = 'C:/Stone/'
    os.chdir(path)
    file_list = os.listdir(path)
    file_list_jpg=[file for file in file_list if
    file.endswith(".jpg")]
    path = './NanumGothicCoding.ttf'
    fontprop=fm.FontProperties(fname=path,
    size=12)
    SIZE=224
    image0=cv2.imread(file_list_jpg[0])
    image0=cv2.cvtColor(image0,
    cv2.COLOR_BGR2RGB)
    image0=cv2.resize(image0, (SIZE, SIZE))/255.
    image1=cv2.imread(file_list_jpg[1])
    image1=cv2.cvtColor(image1,
    cv2.COLOR_BGR2RGB)
    image1=cv2.resize(image1, (SIZE, SIZE))/255.
    image2=cv2.imread(file_list_jpg[2])
    image2=cv2.cvtColor(image2,
    cv2.COLOR_BGR2RGB)
    image2=cv2.resize(image2, (SIZE, SIZE))/255.
    pred = model.predict(image0[np.newaxis])[0, 0]
    plt.subplot(1, 3, 1)
    plt.imshow(image0)
    plt.xticks([])
    plt.yticks([])
    plt.title(f"황등석일 확률: {pred*100:.4f}%",
    fontproperties=fontprop)
    pred1 = model.predict(image1[np.newaxis])[0, 0]
    plt.subplot(1, 3, 2)
    plt.imshow(image1)
    plt.xticks([])
    plt.yticks([])
    plt.title(f"황등석일 확률: {pred1*100:.4f}%",
    fontproperties=fontprop)
    pred2 = model.predict(image2[np.newaxis])[0, 0]
    plt.subplot(1, 3, 3)
    plt.imshow(image2)
    plt.xticks([])
    plt.yticks([])
    plt.title(f"황등석일 확률: {pred2*100:.4f}%",
    fontproperties=fontprop)
    plt.savefig(f"./figures/{model_name}_training_
    predict.jpg")
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

추가적으로 우분투 리눅스 환경의 구글 코랩 버전 소스는 아래와 같다.

```
net = models.load_model('./vgg16stone.h5')
def upload_inference(model):
    def inference(filepath):
        path = './NanumGothicCoding.ttf'
        fontprop = fm.FontProperties(fname=path,
        size=12)
        SIZE=224
        img = cv.imread(filepath)
        img = cv.cvtColor(img, cv.COLOR_BGR2RGB)
        img = cv.resize(img, (SIZE, SIZE))/255.
        pred = model.predict(img[np.newaxis])[0, 0]
        plt.imshow(img)
        plt.axis('off')
        plt.title(f"황등석일 확률: {pred*100:.4f}%",
        fontproperties=fontprop)
        plt.show()
    uploaded = files.upload()
    for fn in uploaded.keys():
        print('User uploaded file "{name}" with
        length {length} bytes'.format(name=fn,
        length=len(uploaded[fn])))
    filelist = list(uploaded.keys())
    interact(inference, filepath=filelist)
```

표 1을 보면 모델이 황등석 이미지를 황등석일 확률로 93%를 예측하였고 포천석 이미지를 황등석일 확률로 21%를 예측하였으며, 황등석 이미지와 유사한 고흥석 이미지를 황등석일 확률로 59%를 예측하여 여러 종류의 석재 이미지에서 황등석을 구별하는 것으로 나타났다.

본 논문에서는 윈도우즈 환경에서 구현하여 편의상 3개의 이미지에 대해서만 결과를 보여 줬지만 구글의 코랩환경에서는 files.upload()함수를 이용하여 더 많은 이미지를 불러와서 예측결과를 볼 수 있다. 윈도우즈 환경에서는 구글 클라우드인 코랩(Colab)에서 돌아가는 files.upload() 함수를 사용할 수 없어 이와 비슷한 기능을 사용하려면 웹 프레임워크인 파이썬의 Django, Flask, FastAPI를 활용하여 구현할 수 있는데 이는

추가적인 코드 구현이 필요한 관계로 본 논문에서는 간략하게 대표적으로 3개의 이미지에 대해서만 윈도우즈 환경에서 출력하는 것으로 구현하였다.

구글 코랩 환경은 GPU를 제공하고 개발툴을 따로 설치할 필요없이 웹상에서 코딩을 하고 바로 코딩 결과를 볼 수 있어 편리하기는 하나 구글 드라이브에 마운트 하여 작업해야 하는 관계로 자기만의 인공지능 모델을 개발하여 배포하는데는 어려움이 따른다. 이에 본 논문에서는 향후 상업화를 위한 자기만의 배포 서비스를 위해 구글 코랩 환경에서 구현한 코드 보다는 윈도우즈 환경에서 구현한 코드를 중심으로 설명하였다.

III. 결 론

본 논문에서는 Pretrained 모델의 하나인 VGGNet(계층의 깊이가 깊어질수록 계층 단계별 학습시 Back propagation에 의한 weight의 업데이트 시간이 길어짐으로 인해 학습이 잘 되지 않을 수 있는 문제를 중간 중간 계층 단계를 건너 뛰어 학습하는 방법을 통해 해결한 전훈련 모델) 활용하여 국내산 석재 이미지를 판별하는 딥러닝 모델을 케라스와 파이썬 언어를 사용하여 구현하였다.

본 논문에서 구현한 딥러닝 모델은 전라북도 익산의 대표석재인 황등석 이미지를 주로 학습하여 다양한 석재 이미지에서 황등석을 판별할 수 있게 구현하여 구현한 모델이 황등석 이미지를 황등석일 확률로 93%를 예측하여 인공지능 서비스 상용화 기준인 정확도 90%를 넘어 석재가공회사에서 사용해도 될 만큼의 결과를 도출 하였다. 또한 본 논문에서 구현한 모델은 석재 이미지를 대상으로 하였으나 석재 이미지가 아닌 다른 종류의 이미지에도 적용할 수 있도록 구현되어 있으며, 다른 전훈련(Pretrained) 모델인 InceptionV3 등 다양한 전훈련 모델로도 구현해 볼 수 있게 코딩되어 있으며, 석재별 학습을 통한 석재별 카테고리컬 분류로 확장하여 구현해 볼 수 있도록 구조화되어 있다.

본 논문의 전체적인 과정을 파이썬 코드 구현 위주로 서술한 것은 요즈음 인공지능 모델 구현에 대한 관심이 지대하고 케라스의 등장과 더불어 전공자가 아닌 일반인들도 인공지능 모델을 구현할 수 있는 토대가 마련되어 인공지능 모델을 이론적으로 이해하는 것이 아닌 실제적인 인공지능 모델을 개발하고 연구하여 실제 산업현장에 적용하는데 있어서 조금이나마 도움이 되고자 함이다.

추후로는 구현한 딥러닝 모델의 배포를 위해 파이썬의 Django, Flask, FastAPI와 같은 웹 프레임워크를 사용해서 웹상에서 딥러닝 모델 추론 결과를 보여줄 수 있는 코딩 구현이 추가적으로 필요할 것으로 생각된다.

REFERENCES

[1] 전소연, 박종화, 윤상병, 김영수, 이용성, 전지혜, “딥러닝 기반 영상 분석 알고리즘을 이용한 실시간 작업자 안전관리 시스템 개발,” *스마트미디어저널*, 제9권 제3호, 25-30쪽, 2020년 9월

[2] 노순국, “인공지능 기반 구글넷 딥러닝과 IoT를 이용한 의류 분류,” *스마트미디어저널*, 제9권 제3호, 41-45쪽, 2020년 9월

[3] 이병우, 이우창, 채승완, 김동현, 이충권, “딥러닝 기반 이미지 특징 추출 모델을 이용한 유사 디자인 검출에 대한 연구,” *스마트미디어저널*, 제9권 제4호, 162-169쪽, 2020년 12월

[4] 오렐리앙 제롱, 박해선, “핸즈온 머신러닝: 사이킷런, 케라스, 텐서플로2를 활용한 머신러닝,” *한빛미디어*, 2020년 5월

[5] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, “Densely Connected Convolutional Networks,” *Computer Vision and Pattern Recognition*, Aug. 2017.

[6] Karen Simon and Andrew Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” *Conference paper at ICLR*, 2015.

[7] Francois Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *Computer Vision and Pattern Recognition*, Apr. 2017.

[8] 박응용, “Do it! 점프 투 파이썬,” *이지스퍼블리싱*, 2019년 6월

[9] Github Source(2020), http://github.com/Finfra/AI_Vision (accessed Dec., 2020).

[10] keras 공식 한글 문서: https://keras.io/ko/ (accessed Dec., 2020).

[11] Convolutional neural network의 이해, http://deeplearning.stanford.edu/tutorial/supervised/FeatureExtractionUsingConvolution/(accessed Dec. 2020).

[12] NNA(Neural Network Architecture) 소개 및 특성 비교: http://blog.naver.com/PostView.nhn?blogId=laonple&logNo=220654387455(accessed Dec. 2020).

[13] Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, “Densely Connected Convolutional Networks,” *Computer Vision and Pattern Recognition*, Aug. 2017.

[14] Francois Chollet, “Xception: Deep Learning with Depthwise Separable Convolutions,” *Computer Vision and Pattern Recognition*, Apr. 2017.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional

neural networks," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1106 - 1114, 2012.

저 자 소 개

최경남(정회원)

1991년 충남대학교 전자공학과 학사 졸업.
1994년 충남대학교 전자공학과 석사 졸업.
2019년 원광대학교 SW중심대학사업단 부교수.
2020년 군산대학교 건설기계공학과 박사 수료.
<주관심분야 : 인공지능, Robot Process Automation, 스마트 팩토리>