|  |
| --- |
| **Auto Intelligence Car**   * **자율 주행 영상 처리 시스템-** |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| |  |  | | --- | --- | | ○ 제정일자 : | 2018년 12월 26일 | | ○ 문서버전 : | Ver 1.0 | | ○ 팀명 : | AIC | |  |  | |  |  |  |

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | logo |

1. **프로젝트 기획 & 설계**

**C. 사용 기술**

**1) 분석 모델: CNN, VGG16, ResNet50**

**2) 방법론: Train 데이터 전처리 작업을 수행하고 5-Fold 교차 검증을 한다. 4/5의 데이터를 기반으로 학습을 수행하고 1/5 데이터로 모델을 평가한다. 모델 학습 후 테스트 데이터 예측을 시행한다.**

**3) tools: Spyder(python 3.6)**

**4) 사용 언어: Python**

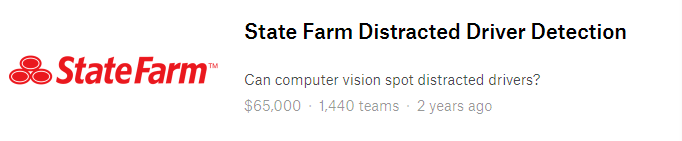
1. **프로젝트 내용**

**미국의 보험 및 금융 회사인 스테이트 팜에서 주최한 운전자의 이미지를 기반으로 운전자의 상태를 분류하는 머신러닝 경진대회 데이터를 바탕으로 이미지 분류 모델 개발을 목표로 한다.**

1. **프로젝트 배경지식/기술/알고리즘**
2. **Data**

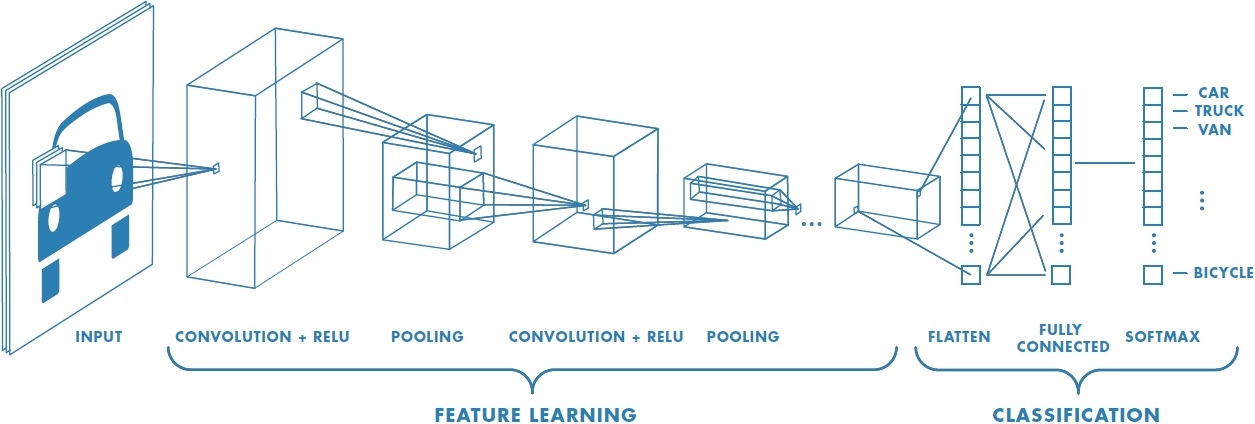
**미국의 보험 및 금융 회사인 스테이팜에서 제공해주는 운전자 이미지를 data로 한다.**

**이 데이터는 조석 위치에서 운전자 전체를 담고 있는 이미지 데이터로 운전자의 운전 상태를 10개 중 1개로 분류해야 한다. 10개의 상태는 다음과 같다. 1) 안전 운전 상태, 2) 오른손으로 문자 상태, 3) 오른손으로 전화 상태, 4) 왼손으로 문자 상태, 5) 왼손으로 전화 상태, 6) 라디오 조작 상태, 7) 음료수 섭취 상태, 8) 뒷자석에 손 뻗기 상태, 9) 얼굴/머리 만지기 상태, 10) 조수석과 대화 상태 들로 나뉘어 진다. Test 데이터는 총 79,726 개의 이미지로 구성되어 있으며, Train 데이터는 22,424개로 구성되어 있다.**



1. **CNN(Convolutional Nenural Network)**

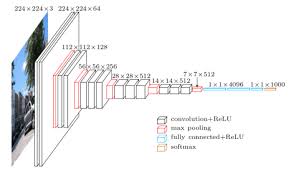
**컨볼루션 신경망으로 인간의 시션경 구조를 모방해 만들어진 인공신경망 알고리즘의 한 종류이다. CNN은 이미지, 텍스트, 사운드, 비디오의 특징을 식별하고 분류하는데 사용된다. 특히 이미지에서 객체, 얼굴, 장면을 인식하기 위해 패턴을 찾는데 유용한 알고리즘이다. CNN은 학습 가능한 가중치(Weight)와 바이어스(bias)로 구성되어 있다. 각 뉴런은 입력을 받아 내적 연산을 한 뒤 선택에 따라 비선형 연산을 한다.**



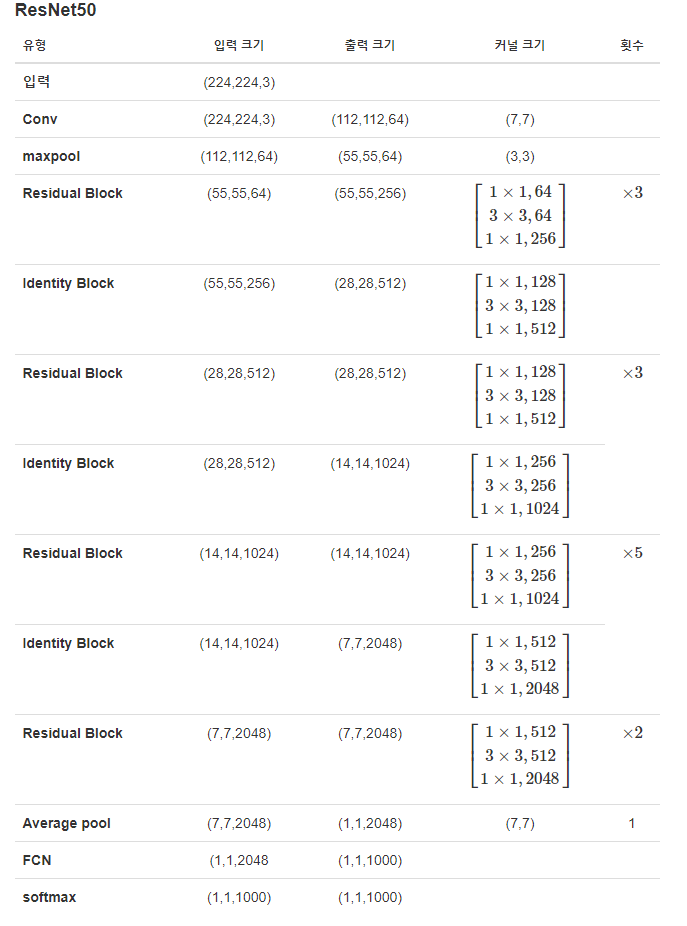
1. **GPU(Graphics Processing Unit)**

**그래픽 처리장치는 메모리를 빠르게 처리하고 바꾸어 화면으로 출력할 프레임 버퍼 안의 영상 생성을 가속하도록 설계된 전자회로이다. GPU는 컴퓨터 그래픽과 영상처리를 매우 효과적으로 처리하며, 고도의 병행 구주는 큰 덩어리의 영상 데이터가 병행 처리되는 알고리즘에 다용도 CPU보다 능률적이다. cnn에서 다량의 이미지 데이터를 학습 할 때 GPU를 적용하면 모델을 학습시키기 위한 처리 시간을 대폭 단축할 수 있다.**

1. **VGG16**

**VGG16은 옥스포드대학교에서 만든 모델로 16개의 레이어로 구성된 모델이다. 이모델은 224 x 224 x 3 크기의 이미지 데이터를 입력값으로 받은 후, 5번의 컨볼루션 블록을 통과하여 7 x 7 x 512 크기의 임베딩으로 표현된다. 각 컨볼류션 블록에는 3 x 3 커널과 비선형적 특징을 포착하기 위한 ReLU를 총 3번 통과한 후, 이미지 안 특징을 추출하는 MaxPooling 과정을 통과한다. 컨볼루션 블록을 통과하여 생성된 7 x 7 x 512 에는 이미지에 대한 정보가 압축되어 있다. 이 정보를 2번의 전결합층(Fully Connected Layer)을 통과하고, 최종적으로 문제에 알맞은 클래스에 크기에 softmax함수를 통과하여 클래스별 확률값을 가진다.**

1. **ResNet50**

**ResNet은 Activation을 직접적으로 변환하는 weight를 학습하는게 아니라 input과 output의 차이인 residual을 학습시키는 방식이다. VGG는 Convolutiondmf 통해 Tensor(Feature maps)을 변환 시켜왔지만 ResNet은 input에 추가한 값을 컨벌루션으로 학습시킨다. ResNet은 CNN의 깊이가 깊어질수록 학습이 어렵고, 성능이 하락했던 것을 해결하였다.**

1. **Keras**

**케라스는 파이썬으로 작성된 오픈 소스 신경망 라이브러리이다. Keras는 MXNet, Deeplearning4j, Tensorflow 등 위에서 수행할 수 있다. 딥 러닝과의 빠른 실험을 가능케 하도록 설계되었으며 최소한의 모듈 방식의 확장 가능성에 초점을 둔다. 다층 퍼셉트론 신경망 모델, 컨볼루션 신경망 모델, 순환 신경망 모델, 조합 모델 등을 구성할 수 있다.**

1. **OpenCV**

**OpenCV는 컴퓨터 비전 라이브러리 중 하나로 실시간 이미지 프로세싱에 중점을 둔 라이브러리이다. 조금이라도 영상처리가 들어간다면 필수적으로 사용하게 되는 라이브러리이다. OpenCV는 C/C++ 으로 개발 되었으며 Python, Java 에서도 지원한다.**

1. **Pandas**

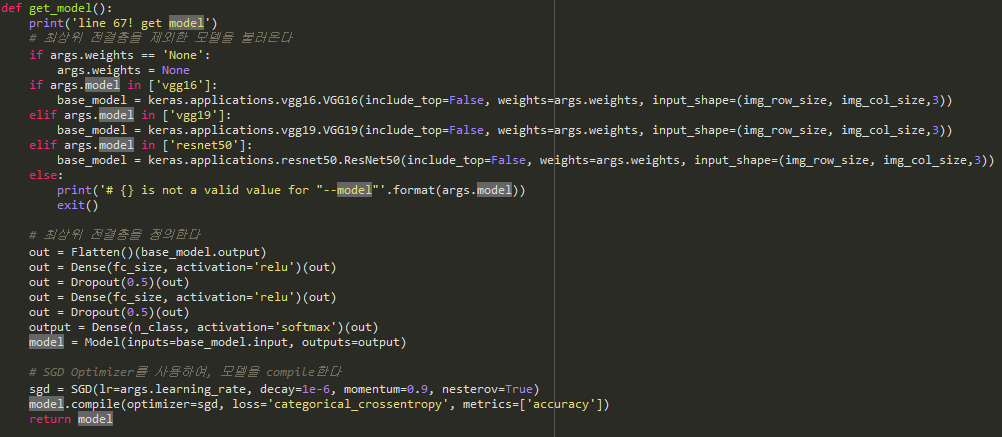
**Pandas 패키지는 데이터 분석을 할 때 가장 많이 쓰이는 패키지이다. 대부분의 데이터는 table의 형태로 나타낼 수 있는데 Pandas 패키지에서는 이러한 표 데이터를 다루기 위한 Series 와 DataFrame 클래스를 제공한다.**

1. **NumPy**

**NumPy는 행렬이나 대규모 다차원 배열을 쉽게 처리할 수 있게 해주는 파이썬의 라이브러리이다. 이외에도 수치계산을 위해 효율적으로 구현 된 기능을 제공한다.**

**B. 핵심 구현 소스**

**1) CNN 모델 정의**

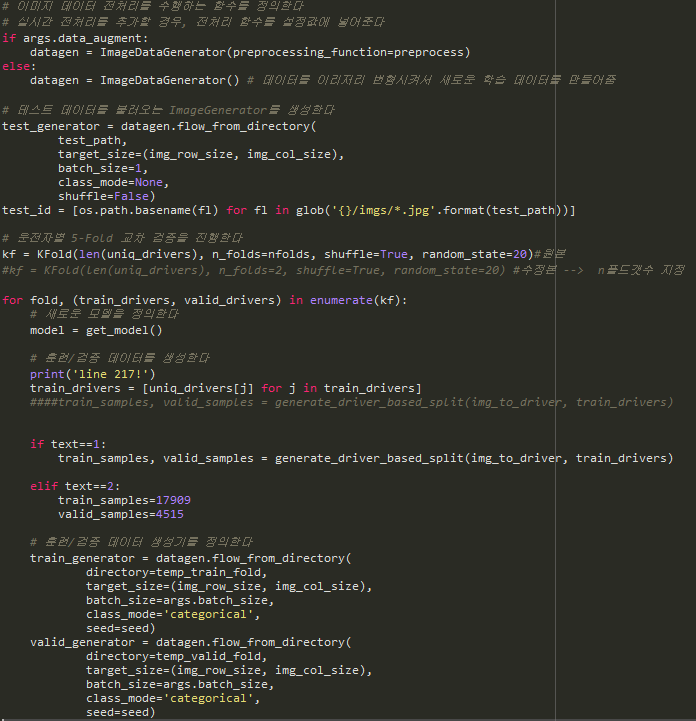


**args 에서 지정한 model 종류에 따라 if 조건문을 탄다. 만약 args 에서 지정한 model 값이 vgg16 일때 vgg16 모델을 사용한다. 또한 include\_top=False 설정을 통해 모델의 최상위 전결층을 제외한 컨볼루션 블록들만 정의한다. Weights 변수는 기학습 모델 파라미터 사용 유무이다. Input\_shape 설정을 통해 모델 입력값의 크기를 정의한다. 이번 프로젝트에서는 224 x 224 x 3으로 지정하여 사용한다.**

**이미지의 특징을 추출하는 컨볼루션 블록의 결과물 위에 클래스 분류를 위한 전결층을 정의한다. Flatten() 함수를 통해 컨볼루션 블록의 결과물을 1차원 벡터로 치환한 후, 비선형 함수인 Relu를 적용한 fc\_size크기의 전결층과 Dropout=0.5 레이어를 총 두번 정의한다. 마지막으로 output 개수 즉 10 크기의 전결층에 softmax 함수를 통과시킨다. 이 과정을 거치면 모델의 최종 결과물이 확률 값의 형태로 출력된다.**

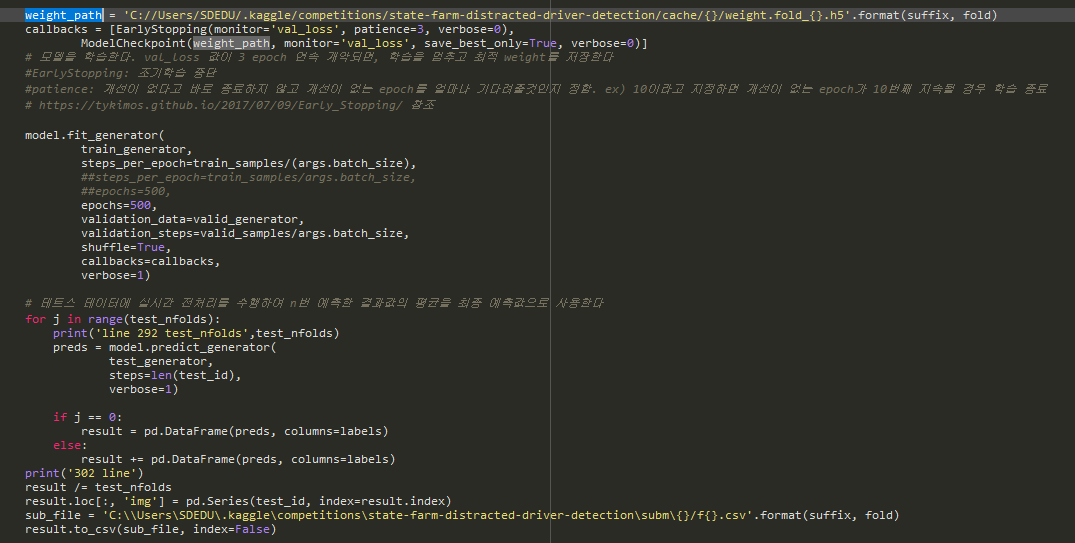
**모델 학습에는 SGD optimizer를 사용한다. 학습속도(learning\_rate)를 적절한 값으로 설정하여 model.compile()을 수행하면 keras 프레임워크에서 학습 및 예측을 할 수 있는 모델이 완성된다.**

**2) 데이터 전처리**



**ImageDataGenerator()는 특정 폴더에 위치한 훈련/검증 데이터를 실시간으로 읽어오는 데이터 생성 함수이다. .flow\_from\_directory() 함수를 통해 데이터들이 위치해 있는 폴더 경로와, 모델의 입력값으로 들어가야 할 이미지 크기, 한 번의 학습 과정에서 사용할 훈련 데이터의 크기를 정하는 batch\_size를 설정한다. 테스트 데이터는 클래스 값이 필요 없으므로 none으로 설정한다. 테스트 데이터는 예측하는 데이터의 순서가 매우 중요하기 때문에 batch\_size=1, shuffle=False로 설정한다. 훈련 데이터는 10개의 클래스로 분류가 가능하므로 class\_mode에 categorical로 설정한다. seed값은 모델 학습 재현성을 위하여 임의의 수로 고정시킨다.**

**3) 모델 학습**

 **모델 학습과정을 들어가기에 앞서 과적합을 방지하기 위하여 Early-Stopping 기법을 도입한 훈련 과정을 진행한다. Early-Stopping 기법에서는 훈련 데이터를 1epoch 학습할 때마다, 검증 데이터에 대한 다중 클래스 로그 손실 값을 계산한다. 3epoch 연속 로그 손실 값이 개악될 경우 학습을 멈추고 해당 모델 파라미터를 저장한다. 결과적으로 일반화 성능이 가장 좋은 지점에서의 모델 파라미터를 저장한다. 교차 검증에는 Fold StratifiedKFold 기법을 사용한다. 훈련 데이터를 랜덤으로 5등분하여, 45의 데이터를 기반으로 학습을 수행하고, 1/5의 데이터를 기반으로 모델을 평가한다. 평가한 결과를 csv파일로 저장한다.**

1. **프로젝트 환경 설치**
2. **프로젝트 시연 순서/동작 매뉴얼**
3. **VGG16**
4. **VGG16 + 운전자별**
5. **VGG16 + 운전자별 + Imagenet 적용**
6. **VGG16 + 운전자별 + Imagenet 적용 + 어그멘테이션**
7. **시행착오/ 어려웠던 부분들 극복방안**
8. **ZeroDivisionError**

ZeroDivisionError: integer division or modulo by zero

**원인: train 이미지들을 train/valid 로 나눠서 복사하는 과정 중 발생, 복사 명령어인 cp가 윈도우에서 실행되지 않았다.**

**해결 방안: cp 를 copy로 수정 후 해결했다.**

1. **FileNotFoundError**

FileNotFoundError: [Errno 2] No such file or directory:

**원인: 경로명중 \f 가 포함되있으면 ASCII 코드 문자 의미로 읽혀 \f 가 0x0c 로 경로명이 바뀌었다.**

**해결 방안: \f 를 /\로 수정 후 정상 실행 확인했다.**

1. **Train/valid로 나눈 폴더에 저장되있는 이미지들이 삭제되어 에러나는 경우**

**원인: 런타임 단축을 위해 기존의 train이미지를 train/valid로 나누어 복사하는 과정을 생략하고 기존에 존재했던 복사한 이미지로 대체하려고 했을 때 오류 발생했다.**

**해결 방안: 기존의 코드에서 shutil.rmtree 부분 주석처리, 결과적으로 약 30분 시간 절약 가능하다.**

1. **매우 긴 런타임 시간**

**원인: 이미지 파일을 학습하는 과정이 상상을 초월하는 시간이 걸림. 학원 PC사양 기준 총 예상 실행 시간은 약 72시간이 걸린다.**

**해결 방안: 팀원의 개인 PC로 GPU를 사용하여 진행 , 결과적으로 수행 시간 대량 감소되었다.**

1. **현재 한계점 및 향후 개선방안**
2. **한계점: 이미지 데이터를 학습하는 시간이 학원 PC사양 기준 총 예상 실행 시간은 약 72시간이 걸린다.**

**개선 방안: 기존 CPU대신 GPU를 사용함으로서 50배 가까이 시간 단축이 가능할 것으로 보인다.**

1. **한계점: Test 데이터에 비해 Train 데이터 부족으로 인한 과적합 발생 가능성이 높다.**

**개선 방안: 앙상블 결과물을 기반으로 준 지도학습을 수행하기 위한 훈련 데이터를 추가한다.**

**참고사이트**

[**http://aikorea.org/cs231n/convolutional-networks/**](http://aikorea.org/cs231n/convolutional-networks/)

[**https://namu.wiki/w/%EC%9D%B8%EA%B3%B5%EC%8B%A0%EA%B2%BD%EB%A7%9D**](https://namu.wiki/w/%EC%9D%B8%EA%B3%B5%EC%8B%A0%EA%B2%BD%EB%A7%9D)

[**https://kr.mathworks.com/solutions/deep-learning/convolutional-neural-network.html**](https://kr.mathworks.com/solutions/deep-learning/convolutional-neural-network.html)

[**https://ko.wikipedia.org/wiki/%EA%B7%B8%EB%9E%98%ED%94%BD\_%EC%B2%98%EB%A6%AC\_%EC%9E%A5%EC%B9%98**](https://ko.wikipedia.org/wiki/%EA%B7%B8%EB%9E%98%ED%94%BD_%EC%B2%98%EB%A6%AC_%EC%9E%A5%EC%B9%98)

**http://blog.creation.net/mxnet-part-5-vgc16-resnet152**

[**http://tmmse.xyz/2016/10/15/lstm-resnet/**](http://tmmse.xyz/2016/10/15/lstm-resnet/)

[**https://datascienceschool.net/view-notebook/958022040c544257aa7ba88643d6c032/**](https://datascienceschool.net/view-notebook/958022040c544257aa7ba88643d6c032/)

[**https://ko.wikipedia.org/wiki/%EC%BC%80%EB%9D%BC%EC%8A%A4**](https://ko.wikipedia.org/wiki/%EC%BC%80%EB%9D%BC%EC%8A%A4)

[**https://namu.wiki/w/OpenCV**](https://namu.wiki/w/OpenCV)

[**https://datascienceschool.net/view-notebook/ee0a5679dd574b94b55193690992f850/**](https://datascienceschool.net/view-notebook/ee0a5679dd574b94b55193690992f850/)

**https://ko.wikipedia.org/wiki/NumPy**