
Keras 딥러닝 라이브러리를 이용한 영상[인쇄체 숫자] 분류

숫자 이미지를 입력 받아서 인식하는 문제

◎ 패턴인식(Recognition)과 패턴분류(Classification) : Class의 한정 유무에 따른 **관점의 차이**

- 분류 : 대상의 범위(class)가 한정되어 있을 때, 어떤 부류에 속하는지 아는 것
예: [사람, 개, 고양이, 자전거] 중 어떤것인지 찾는 것
[0, 1, 2, ..., 9] 10개의 입력 이미지 중 어떤것에 속하는 지 찾는 것
- 인식 : 특징(feature)을 이용하여 대상이 무엇(누구)인지 아는 것
예: 숫자인식 - 숫자의 범위(대상, class)가 한정되어 있지 않을 경우
얼굴인식 - 영상에 있는 얼굴이 누구인지 맞추는 것

※ 그러나, 분류와 인식은 관점의 차이에서 오는 구별이다. 예를 들어,
얼굴인식도 용의자 몇 명중에 하나인지 구분하는 것으로 보면, **인식이자 분류**이다

◎ 지능적인 존재가 하는 많은 일 중의 하나가 분류 혹은 인식이므로 분류를 잘 하는 것이 의미가 크다
영상분류의 응용은 다음을 포함하여 무수히 많다

- 글자 분류(인식) - 차량 번호판 인식, 외국어 메뉴판 번역, 문서 자동인식/번역, 각종 안내표지판 인식
- 영상 분류 - 용의자 얼굴인식, 방공(아군기.적군기 구분), PCBA부품/회로인쇄상태/부품장착불량 분류

※ CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart)
or HIP (Human Interactive Proof)
Asirra (Animal Species Image Recognition for Restricting Access)

1. anaconda 설치 (<https://repo.continuum.io/archive/>) <-- 이미 설치한 경우는 필요 없음

PATH설정 체크박스 체크

설치후에 Command Prompt에서 conda --version 실행해서 설치 확인

윈도우즈 시작메뉴-Anaconda Prompt 실행후

conda update conda 로 conda 최신버전으로 업데이트

conda --version

(* 특정 가상환경 제거 : conda env remove -n 가상환경명)

conda info --envs 로 가상환경 목록 확인

conda list -n 가상환경명 : 가상환경아래에 설치된 패키지 목록 출력

2. 프로젝트 디렉토리+가상환경 classify 만들기 (관리자 권한으로 Command Prompt 실행한 후)

d:

d:\W>md classify_num

d:\W>cd classify_num

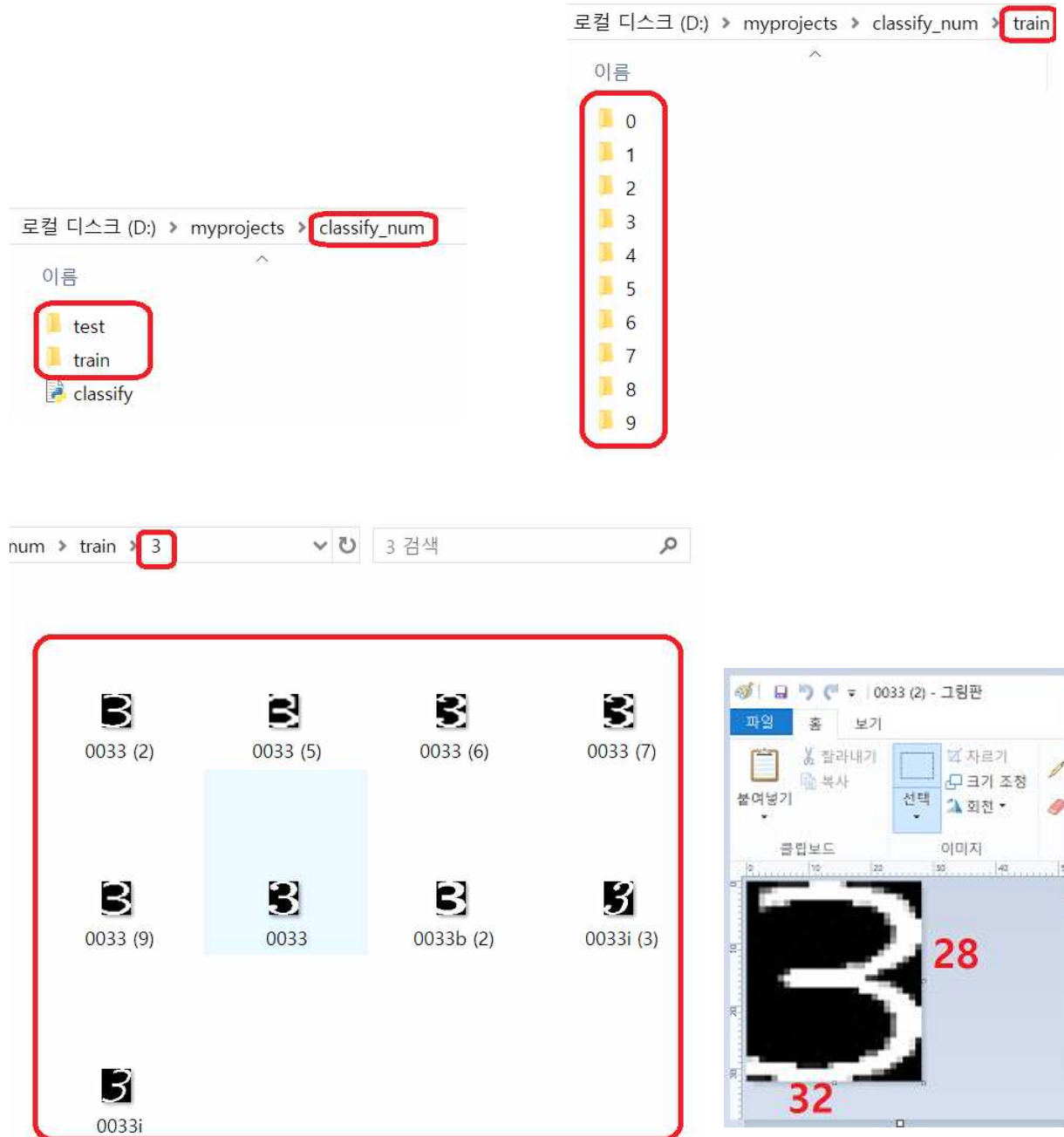
'classify' 가상환경 생성 및 필요 패키지 동시 설치

d:\Wclassify_num> conda create -n classify_num tensorflow-gpu keras pillow pydot matplotlib
opencv

d:\Wclassify_num> conda activate classify_num

3. 데이터 폴더 생성 및 데이터 준비 (탐색기 이용)

▶ train(학습용), test(테스트용) 데이터 폴더 생성



4. 강의 사이트(ailab.silla.ac.kr/lec/ai)에서 classify_num.py 소스파일과 숫자 데이터 다운로드

5. python classify_num.py 실행

6. 실제 응용하기 위해서는.....

- ▶ 문서/문자열에서 문자 분리가 선행 되어야 함 (어디가 문자인지, 붙은문자 분리.....)
- ▶ 한글/한자/영어/숫자/특수기호 등이 혼재 된 상황에서 수천가지 문자를 분류해야 함
- ▶ 인쇄체의 경우에도 다양한 폰트, Italic체, bold체, 글자크기 등 변형이 많음
- ▶ 필기체의 경우에는 경우의 수가 훨씬 많음
 - 학습데이터 수집의 어려움
- ▶ 때때로, 구별이 모호한 문자에 대한 문맥정보의 사용이 필요함

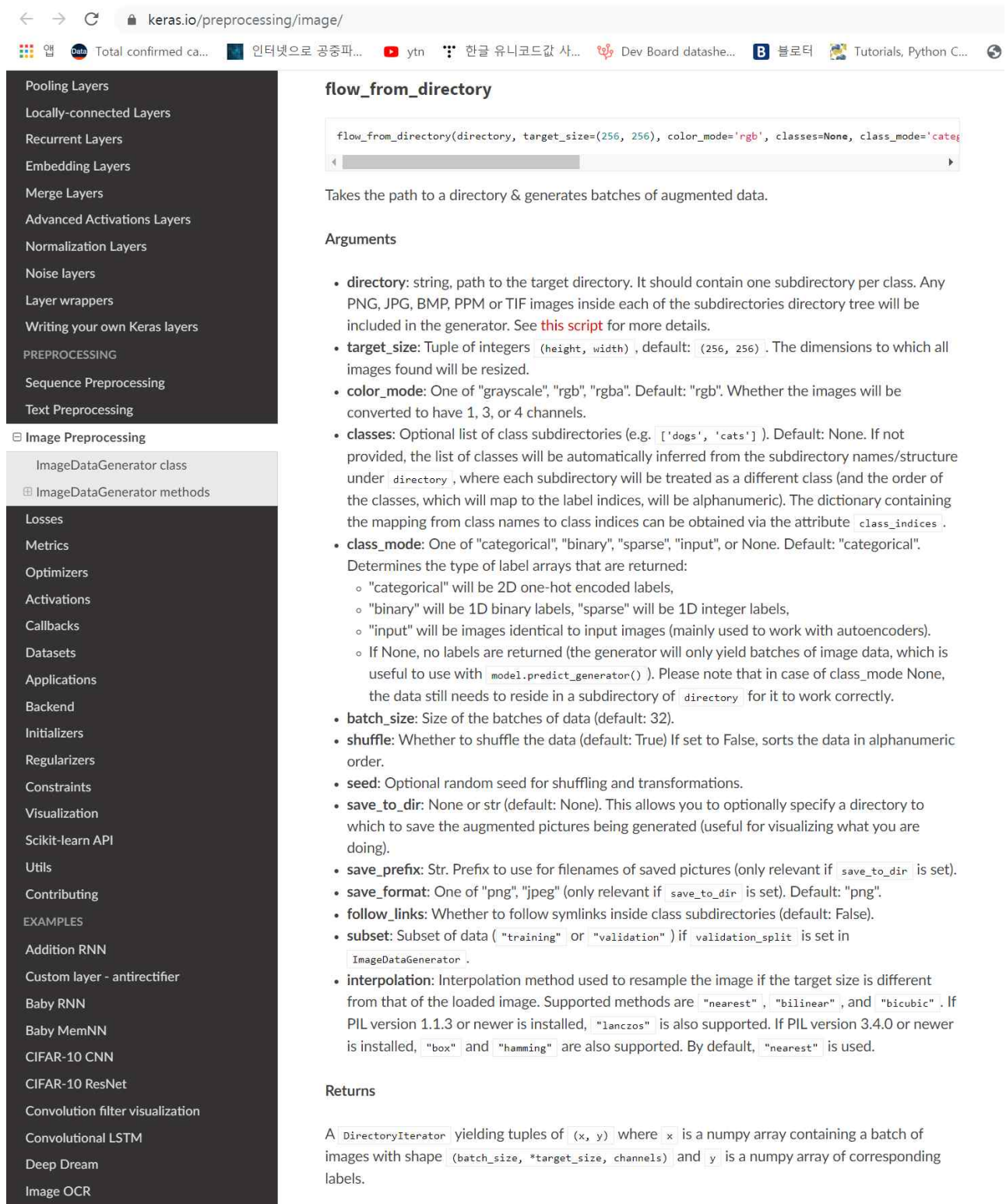
7. 아직까지는 문자인식 수준이 사람에 미치지 못하지만, 조만간 역전될 것으로 예상

■ 자율실습 과제 (제출하지는 않습니다)

8. 그림판 프로그램을 이용하여, 0~9 사이의 임의의 숫자 이미지 10개를 만들어서 test 폴더에 저장한 후 분류(인식)이 어느정도 되는지확인해 보세요.

딥러닝 관련 메소드 학습 방법 (<http://keras.io>에서 검색)

▶ 예: flow_from_directory()



The screenshot shows the Keras.io website with the navigation menu on the left and the documentation for the `flow_from_directory` method on the right.

Navigation Menu:

- Pooling Layers
- Locally-connected Layers
- Recurrent Layers
- Embedding Layers
- Merge Layers
- Advanced Activations Layers
- Normalization Layers
- Noise layers
- Layer wrappers
- Writing your own Keras layers
- PREPROCESSING
 - Sequence Preprocessing
 - Text Preprocessing
- Image Preprocessing
 - ImageDataGenerator class
 - ImageDataGenerator methods
- Losses
- Metrics
- Optimizers
- Activations
- Callbacks
- Datasets
- Applications
- Backend
- Initializers
- Regularizers
- Constraints
- Visualization
- Scikit-learn API
- Utils
- Contributing
- EXAMPLES
 - Addition RNN
 - Custom layer - antirectifier
 - Baby RNN
 - Baby MemNN
 - CIFAR-10 CNN
 - CIFAR-10 ResNet
 - Convolution filter visualization
 - Convolutional LSTM
 - Deep Dream
 - Image OCR

flow_from_directory

```
flow_from_directory(directory, target_size=(256, 256), color_mode='rgb', classes=None, class_mode='categorical', batch_size=32, shuffle=True, seed=None, save_to_dir=None, save_prefix='', save_format='png', follow_links=False, subset=None, interpolation='nearest')
```

Takes the path to a directory & generates batches of augmented data.

Arguments

- directory:** string, path to the target directory. It should contain one subdirectory per class. Any PNG, JPG, BMP, PPM or TIF images inside each of the subdirectories directory tree will be included in the generator. See [this script](#) for more details.
- target_size:** Tuple of integers (height, width), default: (256, 256). The dimensions to which all images found will be resized.
- color_mode:** One of "grayscale", "rgb", "rgba". Default: "rgb". Whether the images will be converted to have 1, 3, or 4 channels.
- classes:** Optional list of class subdirectories (e.g. ['dogs', 'cats']). Default: None. If not provided, the list of classes will be automatically inferred from the subdirectory names/structure under `directory`, where each subdirectory will be treated as a different class (and the order of the classes, which will map to the label indices, will be alphanumeric). The dictionary containing the mapping from class names to class indices can be obtained via the attribute `class_indices`.
- class_mode:** One of "categorical", "binary", "sparse", "input", or None. Default: "categorical". Determines the type of label arrays that are returned:
 - "categorical" will be 2D one-hot encoded labels,
 - "binary" will be 1D binary labels, "sparse" will be 1D integer labels,
 - "input" will be images identical to input images (mainly used to work with autoencoders).
 - If None, no labels are returned (the generator will only yield batches of image data, which is useful to use with `model.predict_generator()`). Please note that in case of `class_mode None`, the data still needs to reside in a subdirectory of `directory` for it to work correctly.
- batch_size:** Size of the batches of data (default: 32).
- shuffle:** Whether to shuffle the data (default: True) If set to False, sorts the data in alphanumeric order.
- seed:** Optional random seed for shuffling and transformations.
- save_to_dir:** None or str (default: None). This allows you to optionally specify a directory to which to save the augmented pictures being generated (useful for visualizing what you are doing).
- save_prefix:** Str. Prefix to use for filenames of saved pictures (only relevant if `save_to_dir` is set).
- save_format:** One of "png", "jpeg" (only relevant if `save_to_dir` is set). Default: "png".
- follow_links:** Whether to follow symlinks inside class subdirectories (default: False).
- subset:** Subset of data ("training" Or "validation") if `validation_split` is set in `ImageDataGenerator`.
- interpolation:** Interpolation method used to resample the image if the target size is different from that of the loaded image. Supported methods are "nearest", "bilinear", and "bicubic". If PIL version 1.1.3 or newer is installed, "lanczos" is also supported. If PIL version 3.4.0 or newer is installed, "box" and "hamming" are also supported. By default, "nearest" is used.

Returns

A `DirectoryIterator` yielding tuples of (x, y) where `x` is a numpy array containing a batch of images with shape (batch_size, *target_size, channels) and `y` is a numpy array of corresponding labels.