

argparse : 호출 당시 인자값을 줘서 동작을 다르게 하고 싶은 경우 사용

inpaint\_model.py의 InpaintCAModel 클래스를 가져온다.



**os.path.abspath**

특정 경로에 대한 절대 경로 획득. 만약 현재 경로가 C:\downloads\ 이면 이를 얻게 됨.

**Os.path.join**

경로를 병합하여 새로운 경로 생성. places2\_256 model을 사용하기 위한 작업이다

if \_\_name\_\_ == "\_\_main\_\_":

해당 모듈이 임포트된 경우가 아닌, 인터프리터에서 직접 실행된 경우만 if문 이하의 코드를 돌리라는 뜻.

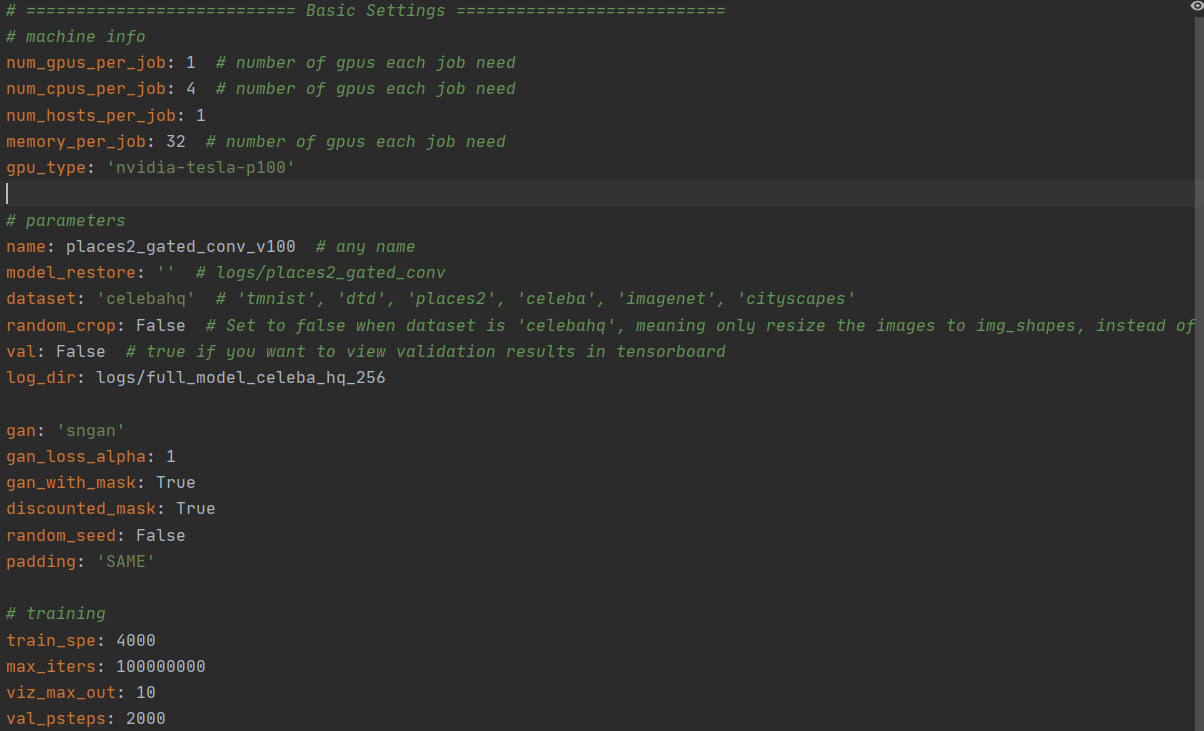
\_\_name\_\_ 은 인터프리터가 실행전에 만들어 둔 전역변수로 인터프리터가 실행하면 \_\_main\_\_ 이라는 값이 담기고, 임포트되어 실행되면 파일의 이름이 담긴다.

즉 다른 파일에서 임포트 될 때는 해당 코드가 실행되지 않게 하며, 직접 실행(cmd나 IDE)할 때 실행되게 하는 코드

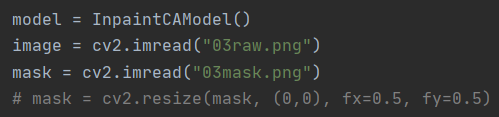


ng는 import할 때 neuralgym를 축약해서 사용하기로 했다. 뉴럴짐은 딥러닝 toolkit인데 현재는 안정성과 신속하면서 지저분한 개발(?) 때문에 제한적인 모듈로만 사용된다.

**ng.config**는 yml 파일을 불러와 설정 정보를 저장한다. yml 파일은 xml파일과 json파일의 단점을 보완하고 장점을 살려서 만들어진 설정정보 파일 형식이다.

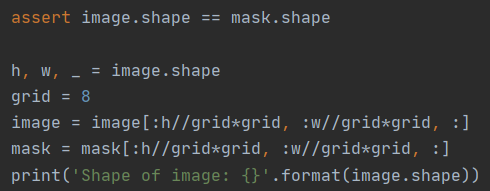


yml 파일은 다음과 같이 다양한 정보들이 저장되어 있다. 필요한 gpu나 cpu수 등등



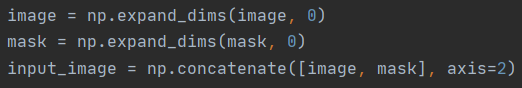
model 변수에 **InpaintCAModel()** 클래스를 불러온뒤 이미지와 마스크를 불러온다.

여기까지 정리하면, 필요한 설정들을 불러오고 사용할 러닝모델을 불러오며 필요한 이미지를 불러왔다.



**assert**는 조건이 붙는 예외처리이다. 이미지의 크기와 마스크의 크기가 다를 경우 에러를 발생시킨다

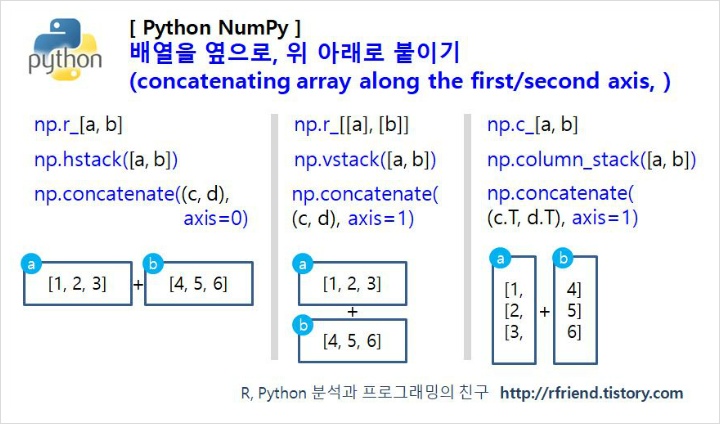
이미지의 **height, width**를 얻고 **grid = 8**로 격자 이미지 획득

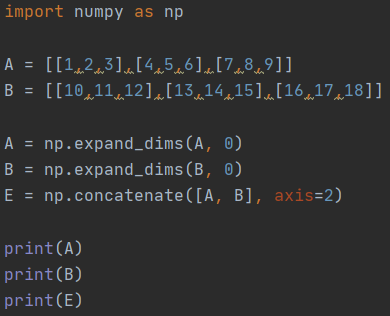


**numpy.expand\_dims** : 차원을 확장시킬 때 사용한다. 현재 이미지는 [h, w, c]로 이루어져있는데 이를 [[h,w,c]]로 한차원 높인다. image.shape는 (3,) 이었는데 (1,3) 으로 바뀌는 것

**numpy.concatenate** : 배열을 합칠 때 사용한다. (1,3) 과 (1,3)을 합치므로 (2,3)이 될 것이다.

axis는 2부터는 해당 차원을 합친다는 뜻같다. 0과 1은 다음과 같이 사용



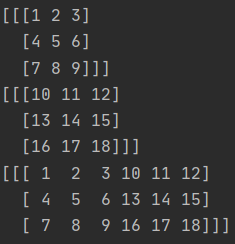


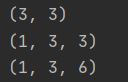
위는 이해를 돕기 위한 간단한 예

따라서 imgae = A, mask = B 라고 하면 둘을 concatenate한 결과는 E이고 이는 다음과 같음

아래와 같이 image의 RGB와 mask의 RGB가 같이 존재함을 알 수 있음.

image.shape는 (3,3) 이었지만 expand\_dims를 거치며 (1,3,3) 으로 변하고 concatenate를 거치며 (1,3,6) 으로 변했다

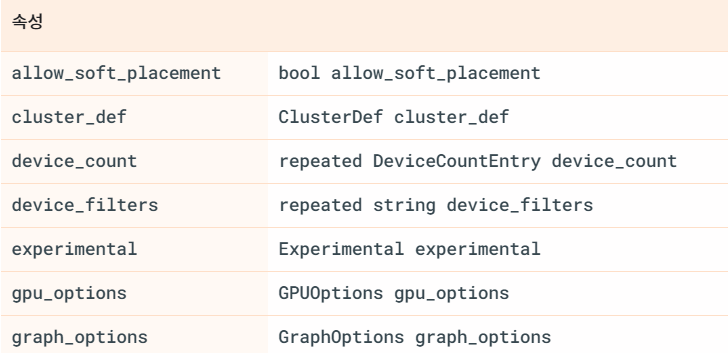




여기까지 정리하면, 불러온 이미지의 속성값을 따고, 마스크를 제작해서 특정 리스트로 차원변환 후 합쳤음.

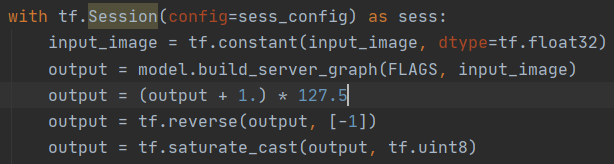


**te.configProto** : tensorflow를 위한 설정값들이 담긴다. 아래 속성 이외에도 더 많다. 주로 머신러닝을 위한 GPU 설정을 하기 위해서 사용함



**gpu\_options.allow\_growth = True :**

True 일 경우 필요할 때마다 GPU 사용률이 조절되고 False는 GPU의 모든 메모리를 점유한다]

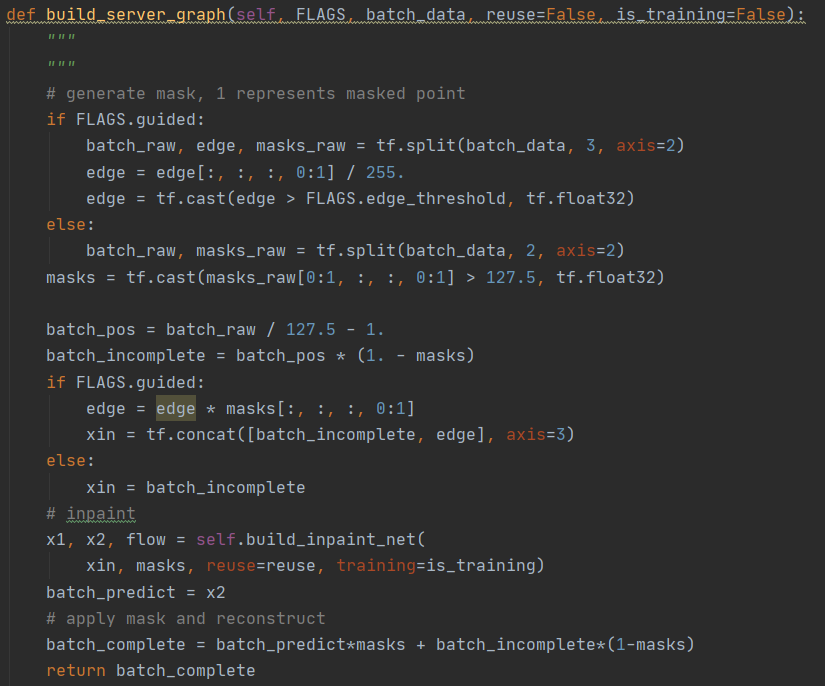


**with A as B** :

with A as B로 쓰이면 B = A() 와 같은 결과임. 다만 open() 같은 함수는 close를 꼭 해줘야하고 try catch 같은 예외처리에서는 못해줄 경우가 있는데 with는 항상 exit할 수 있도록 해줌

**tf.constant** : 수를 생성하는 함수이다. 일반 상수와 달리 머신러닝에서 이용할 상수를 생성한다.

**model.build\_server\_graph** : model = InpaintCAModel 이며 이는 inpaint\_model.py안에 있는 클래스이다. 이 클래스 안에 build\_server\_graph 함수가 있다. 이 함수로 inpaint.yml의 정보를 저장했던 FLAGS와 이미지를 인자로 전송한다



**guided = true 이면**

batch\_data(1,3,6)를 3개로 분리하되 “axis = 2” 차원에 있는 리스트를 분리하라는 뜻이다. ch\_raw, edge, mask는 (1,3,2)씩 나눠가지게 된다.

정리하면

초반에

image = [[1,2,3],[4,5,6],[7,8,9]] (3,3) grid = [[10,11,12],[13,14,15],[16,17,18]] (3,3)

이었던 것을 expand\_dims를 통해

image = [[[1,2,3],[4,5,6],[7,8,9]]] (1,3,3) grid = [[[10,11,12],[13,14,15],[16,17,18]]] (1,3,3)

그리고 concatenate를 통해

input\_image = [[[ 1 2 3 10 11 12],[ 4 5 6 13 14 15],[ 7 8 9 16 17 18]]] (1,3,6)

그리고 마지막으로 split을 통해

batch\_raw = [[[1 2] [4 5] [7 8]]] (1,3,2)

edge = [[[ 3 10] [ 6 13] [ 9 16]]] (1,3,2)

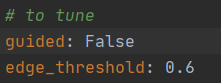
masks\_raw = [[[11 12] [14 15] [17 18]]] (1,3,2)

로 분리되게 된다. 아마 처음 image의 RGB값을 많이 가지고 있어서 batch raw이고 mask의 RGB값을 많이 가지고 있어서 masks\_raw 같다.

이후 edge의 [3 10] 과 [6 13] 만을 취하여 255(명도)로 나눈 뒤 treshhold보다 높으면 1값을,그렇지 않으면 0값을 가지게 한다

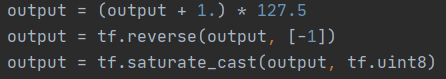
(근데 자세하게 이부분이 어떤 역할을 하는지 모르겠음)

**guided = false** 이면 유저가 따로 정한 edge\_threshold값이 없다는 뜻으로 edge를 선언하지 않는다. 실제 코드는 false



이후는 잘 모르겠음.. 좀더 실행하면서 피드백 봐야할 것 같음

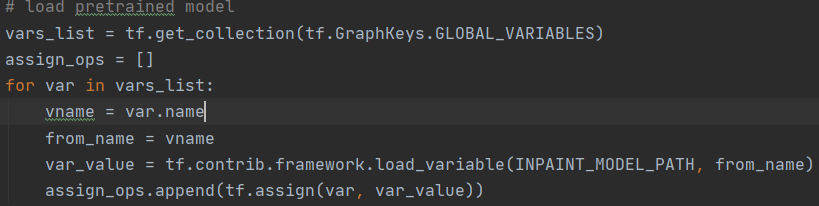
여기까지 정리하면, tensorflow의 설정값을 불러오고 gpu 속성을 만졌다. 그리고 차원확장하여 마스크와 합친 이미지를 tensorflow 변수로 선언하였고 이것과 초기 설정을 이용하여 이미지를 조작하였다.



**(output + 1) \* 127.5** : 더 알아봐야 함

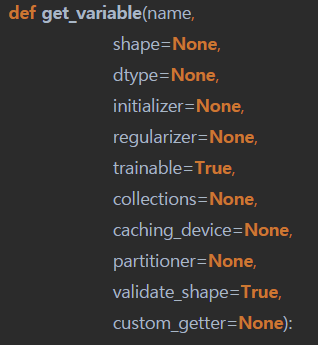
**tf.reverse** : 배열의 순서를 조작함 axis = -1 인 경우 좌우반전

**tf.saturate\_cast** : output을 tf.uint8 형태로 안전하게 캐스팅한다



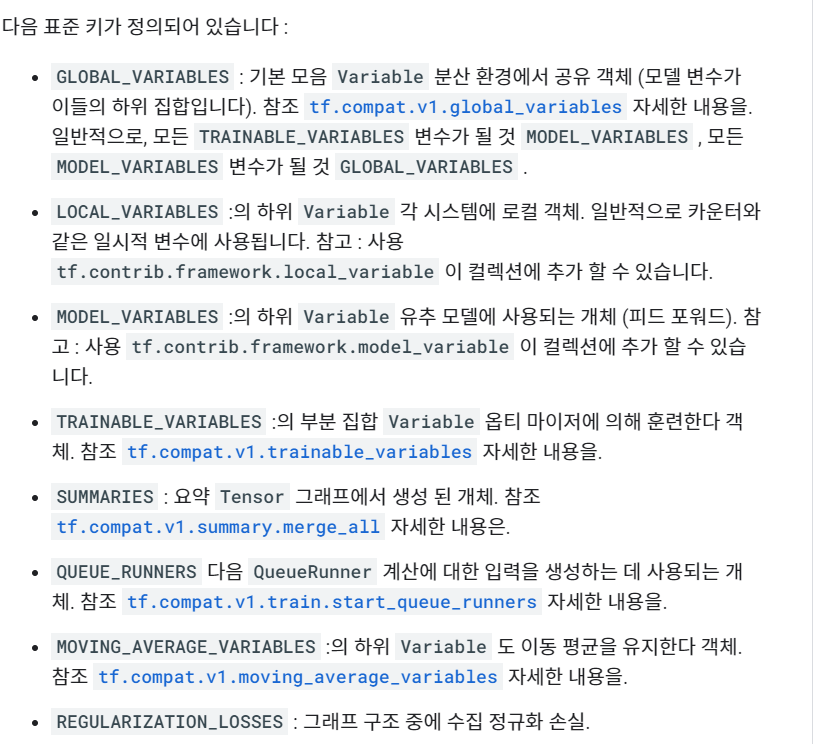
**tf.get.collection** :

get\_variables 는 다음과 같이 여러 속성들을 가진다.



name은 텐서의 name filed. shape와 dtype은 리턴할 텐서의 쉐이프와 타입. initializer는 초기화 할 때 사용하고 regularizer는 텐서에 적용될 정규화 방법론이 무엇인지에 대한 것. trainable은 학습을 통해 변수 값을 변화시킬 것인지에 대한 여부이다. collections는 변수의 소속을 의미. collections = [var1, var2] 값을 대입하면 collections이라는 소속은 var1도 되고 var2도 된다.

특정 소속 collections에 해당하는 변수들의 집합을 얻을 때 get.collection을 사용한다. 인자에는 해당 collections에 속한 key가 해당된다. 이 key에 대한 collections에 속하는 변수들을 반환받는다. 이 GraphKeys는 많이 사용되는 key로 선언되어 있다.



위와 같은 표준키들이 정해져 있다.

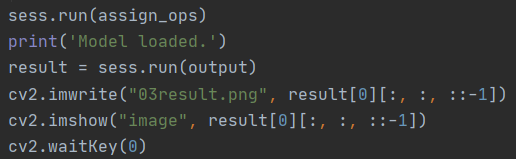
**tf.GraphKeys.GLOBAL\_VARIABLES** : 이중 global\_variables를 불러왔으며 기본 변수인 것 같음

**tf.contrib.framework.load\_variable** : 두 개의 인자를 건네며, 첫번째의 인자는 텐서의 주소, 두번째의 인자는 텐서의 이름을 의미한다. 해당 이름의 텐서를 해당 주소에서 가져온다

**tf.assign** : (var, var\_value)의 인자를 사용. var의 값을 var\_value로 설정하는 초기화 단계

이를 정리하면, 조작한 이미지의 값에 bias를 더하고 reverse 한 뒤 8비트의 int로 캐스팅 했다. 이후 global\_variables로 key의 collections에 속하는 변수들의 목록을 얻는다. (이부분은 살짝 어렵고 잘 모르겠지만, 특정 변수들의 목록을 얻기 위함이라고 생각하면 됨.)

이후, 이 변수 하나하나의 이름과 이에 해당하는 값을 inpaint\_model\_path 경로에서 불러와 초기화한다. 초기화 된 var는 assign\_ops에 담기게 된다.



**sess.run(assign\_ops)** : tensor를 실행시킴

**result** = sess.run (output) : 결과를 관찰하기 위함

**cv2.imwrite** : 해당 결과를 이미지파일로 저장