**ImageDataGenerator class**

keras.preprocessing.image.ImageDataGenerator(featurewise\_center=**False**, samplewise\_center=**False**, featurewise\_std\_normalization=**False**, samplewise\_std\_normalization=**False**, zca\_whitening=**False**, zca\_epsilon=1e-06, rotation\_range=0, width\_shift\_range=0.0, height\_shift\_range=0.0, brightness\_range=**None**, shear\_range=0.0, zoom\_range=0.0, channel\_shift\_range=0.0, fill\_mode='nearest', cval=0.0, horizontal\_flip=**False**, vertical\_flip=**False**, rescale=**None**, preprocessing\_function=**None**, data\_format='channels\_last', validation\_split=0.0, interpolation\_order=1, dtype='float32')

Generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches).

**Arguments**

* **featurewise\_center**: Boolean. Set input mean to 0 over the dataset, feature-wise.
* **samplewise\_center**: Boolean. Set each sample mean to 0.
* **featurewise\_std\_normalization**: Boolean. Divide inputs by std of the dataset, feature-wise.
* **samplewise\_std\_normalization**: Boolean. Divide each input by its std.
* **zca\_epsilon**: epsilon for ZCA whitening. Default is 1e-6.
* **zca\_whitening**: Boolean. Apply ZCA whitening.
* **rotation\_range**: Int. Degree range for random rotations.
* **width\_shift\_range**: Float, 1-D array-like or int
  + float: fraction of total width, if < 1, or pixels if >= 1.
  + 1-D array-like: random elements from the array.
  + int: integer number of pixels from interval (-width\_shift\_range, +width\_shift\_range)
  + With width\_shift\_range=2 possible values are integers [-1, 0, +1], same as with width\_shift\_range=[-1, 0, +1], while with width\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).
* **height\_shift\_range**: Float, 1-D array-like or int
  + float: fraction of total height, if < 1, or pixels if >= 1.
  + 1-D array-like: random elements from the array.
  + int: integer number of pixels from interval (-height\_shift\_range, +height\_shift\_range)
  + With height\_shift\_range=2 possible values are integers [-1, 0, +1], same as with height\_shift\_range=[-1, 0, +1], while with height\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).
* **brightness\_range**: Tuple or list of two floats. Range for picking a brightness shift value from.
* **shear\_range**: Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)
* **zoom\_range**: Float or [lower, upper]. Range for random zoom. If a float, [lower, upper] = [1-zoom\_range, 1+zoom\_range].
* **channel\_shift\_range**: Float. Range for random channel shifts.
* **fill\_mode**: One of {"constant", "nearest", "reflect" or "wrap"}. Default is 'nearest'. Points outside the boundaries of the input are filled according to the given mode:
  + 'constant': kkkkkkkk|abcd|kkkkkkkk (cval=k)
  + 'nearest': aaaaaaaa|abcd|dddddddd
  + 'reflect': abcddcba|abcd|dcbaabcd
  + 'wrap': abcdabcd|abcd|abcdabcd
* **cval**: Float or Int. Value used for points outside the boundaries when fill\_mode = "constant".
* **horizontal\_flip**: Boolean. Randomly flip inputs horizontally.
* **vertical\_flip**: Boolean. Randomly flip inputs vertically.
* **rescale**: rescaling factor. Defaults to None. If None or 0, no rescaling is applied, otherwise we multiply the data by the value provided (after applying all other transformations).
* **preprocessing\_function**: function that will be applied on each input. The function will run after the image is resized and augmented. The function should take one argument: one image (Numpy tensor with rank 3), and should output a Numpy tensor with the same shape.
* **data\_format**: Image data format, either "channels\_first" or "channels\_last". "channels\_last" mode means that the images should have shape (samples, height, width, channels), "channels\_first" mode means that the images should have shape (samples, channels, height, width). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".
* **validation\_split**: Float. Fraction of images reserved for validation (strictly between 0 and 1).
* **dtype**: Dtype to use for the generated arrays.

**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](https://gist.github.com/fchollet/0830affa1f7f19fd47b06d4cf89ed44d) 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.