[How to apply data augmentation]

1. create a Image data denerator
2. defines its input
3. train the model using fit\_generator

* example

# define data augmentation function

datagen = tf.keras.preprocessing.image.ImageDataGenerator(

width\_shift\_range = 0.1,

height\_shift\_range = 0.1,

horizontal\_flip = True)

# compute quantities required for featurewise normalization

# (std, mean, and principal components if ZCA whitening is applied)

datagen.fit(x\_train)

# train model by using data generator

model.fit\_generator(datagen.flow(x\_train, y\_train, batch\_size=128), # the data iterator

validation\_data = (x\_test, y\_test),

epoch=100,

workers=4,

callbacks=[tb\_cb, ck\_cb])

[ImageDataGenerator]

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=None,

validation\_split=0.0,

dtype=None)

* **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 half-open 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 half-open 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 implied 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]](https://keras.io/preprocessing/image/#flow)

<ImageDataGenerator>.flow(x,

y=None,

batch\_size=32,

shuffle=True,

sample\_weight=None,

seed=None,

save\_to\_dir=None,

save\_prefix='',

save\_format='png',

subset=None)

* return

**An Iterator yielding tuples of (x, y)** where x is a numpy array of image data (in the case of a single image input) or a list of numpy arrays (in the case with additional inputs) and y is a numpy array of corresponding labels. If 'sample\_weight' is not None, the yielded tuples are of the form (x, y, sample\_weight). If y is None, only the numpy array x is returned.

* **x**: Input data. Numpy array of rank 4 or a tuple. If tuple, the first element should contain the images and the second element another numpy array or a list of numpy arrays that gets passed to the output without any modifications. Can be used to feed the model miscellaneous data along with the images. In case of grayscale data, the channels axis of the image array should have value 1, in case of RGB data, it should have value 3, and in case of RGBA data, it should have value 4.
* **y**: Labels.
* **batch\_size**: Int (default: 32).
* **shuffle**: Boolean (default: True).
* **sample\_weight**: Sample weights.
* **seed**: Int (default: None).
* **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 (default: ''). 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".
* **subset**: Subset of data ("training" or "validation") if validation\_split is set in ImageDataGenerator.