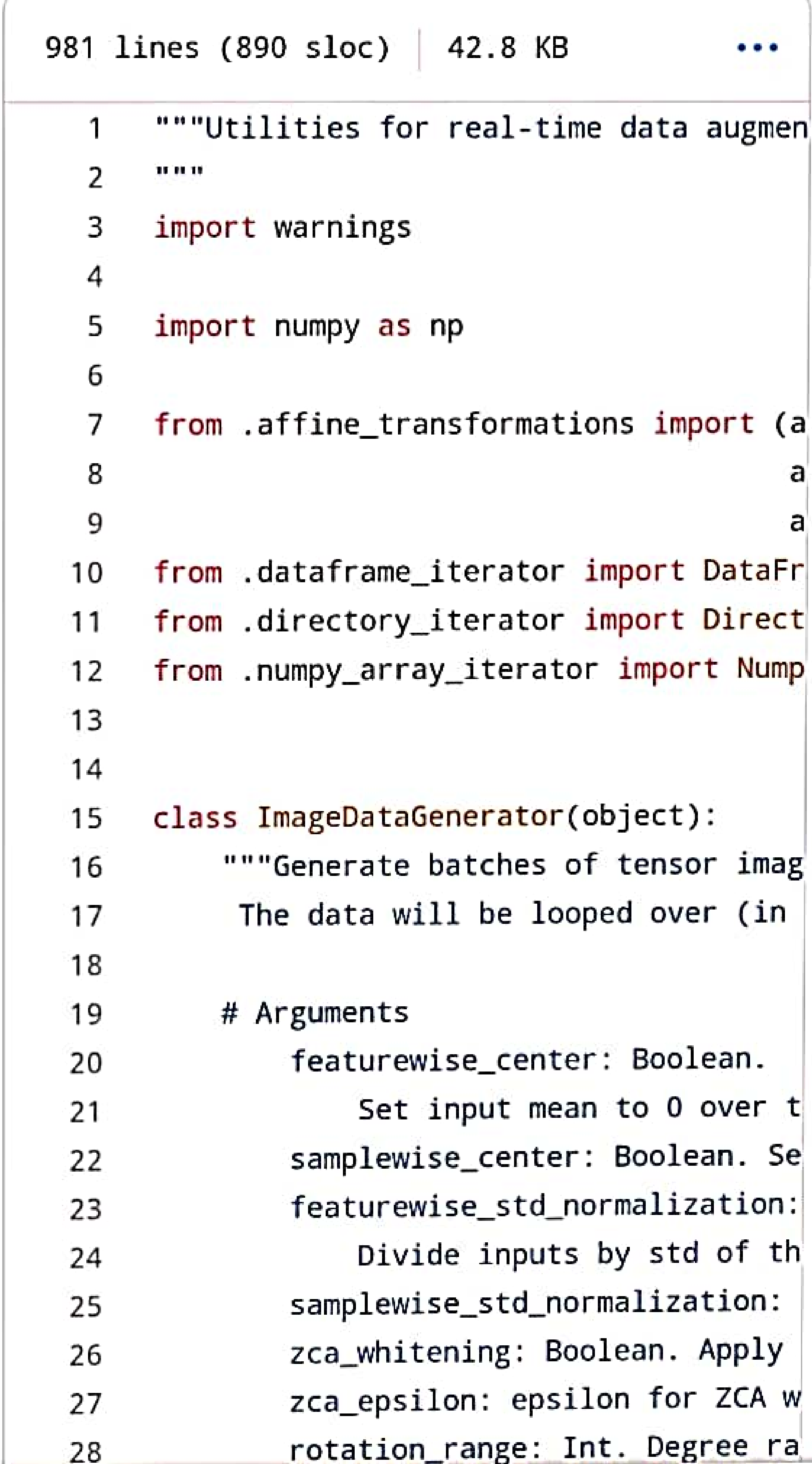


981 lines (890 sloc) | 42.8 KB

```
"""Utilities for real-time data augmen
"""
import warnings

import numpy as np

from .affine_transformations import (a
                                     a
                                     a
from .dataframe_iterator import DataFr
from .directory_iterator import Direct
from .numpy_array_iterator import Nump


class ImageDataGenerator(object):
    """Generate batches of tensor imag
     The data will be looped over (in

    # Arguments
        featurewise_center: Boolean.
            Set input mean to 0 over t
        samplewise_center: Boolean. Se
        featurewise_std_normalization:
            Divide inputs by std of th
        samplewise_std_normalization:
        zca_whitening: Boolean. Apply
        zca_epsilon: epsilon for ZCA w
        rotation_range: Int. Degree ra
        samplewise_std_normalization:
        zca_whitening: Boolean. Apply
        zca_epsilon: epsilon for ZCA w
        rotation_range: Int. Degree ra
        width_shift_range: Float, 1-D
            - float: fraction of total
            - 1-D array-like: random e
            - int: integer number of p
                `(-width_shift_range,
            - With `width_shift_range=
                are integers `[-1, 0,
                same as with `width_sh
                while with `width_shif
                in the interval `[-1.0
        height_shift_range: Float, 1-D
            - float: fraction of total
            - 1-D array-like: random e
            - int: integer number of p
                `(-height_shift_range,
            - With `height_shift_range
                are integers `[-1, 0,
                same as with `height_s
                while with `height_shi
                in the interval `[-1.0
        brightness_range: Tuple or lis
            a brightness shift value f
        shear_range: Float. Shear Inte
            (Shear angle in counter-cl
        zoom_range: Float or [lower, u
            If a float, `[lower, upper
        channel_shift_range: Float. Ra
        fill_mode: One of {"constant",
            Default is 'nearest'.
            Points outside the boundar
```

~~~
            If you never set it, then
        validation_split: Float. Fract
            (strictly between 0 and 1)
        interpolation_order: int, orde
            the spline interpolation.
        dtype: Dtype to use for the ge

    # Examples
    Example of using `.flow(x, y)`:

    ```python
    (x_train, y_train), (x_test, y_tes
    y_train = np_utils.to_categorical(
    y_test = np_utils.to_categorical(y

    datagen = ImageDataGenerator(
        featurewise_center=True,
        featurewise_std_normalization=
        rotation_range=20,
        width_shift_range=0.2,
        height_shift_range=0.2,
        horizontal_flip=True)

    # compute quantities required for
    # (std, mean, and principal compon
    datagen.fit(x_train)

    # fits the model on batches with r
    model.fit_generator(datagen.flow(x
                        steps_per_epoc

    # here's a more "manual" example
    for e in range(epochs):
        print('Epoch', e)
~~~

```
            If a float,  [lower, upper
        channel_shift_range: Float. Ra
        fill_mode: One of {"constant",
            Default is 'nearest'.
            Points outside the boundar
            according to the given mod
            - 'constant': kkkkkkkk|abc
            - 'nearest':  aaaaaaaa|abc
            - 'reflect':  abcddcba|abc
            - 'wrap':  abcdabcd|abcd|a
        cval: Float or Int.
            Value used for points outs
            when `fill_mode = "constan
        horizontal_flip: Boolean. Rand
        vertical_flip: Boolean. Random
        rescale: rescaling factor. Def
            If None or 0, no rescaling
            otherwise we multiply the
            (after applying all other
        preprocessing_function: functi
            The function will run afte
            The function should take o
            one image (NumPy tensor wi
            and should output a NumPy
        data_format: Image data format
            either "channels_first" or
            "channels_last" mode means
            `(samples, height, width,
            "channels_first" mode mean
            `(samples, channels, heigh
            It defaults to the `image_
            Keras config file at `~/.k
            If you never set it, then
        validation_split: Float. Fract
```

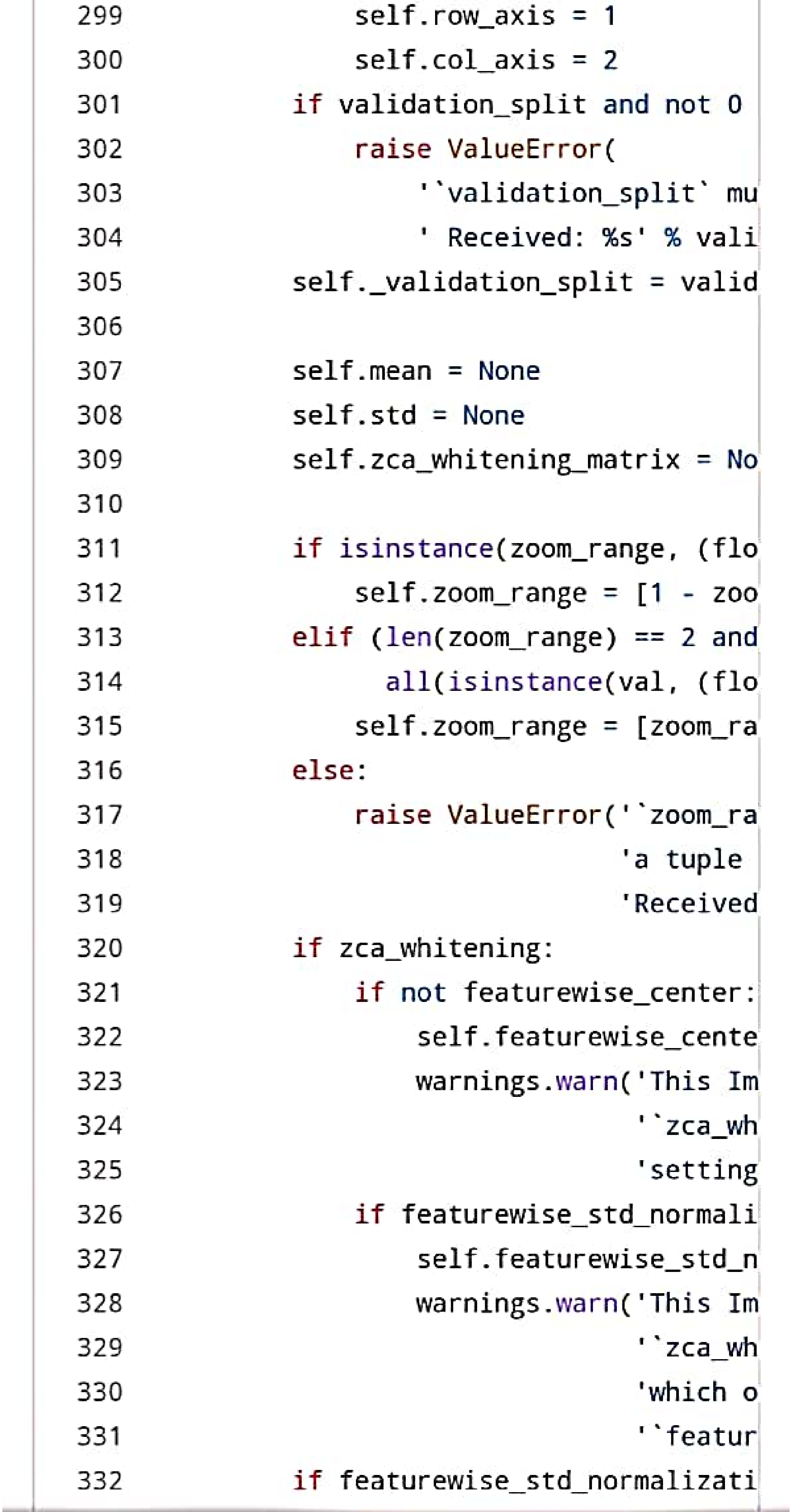
```
model.fit_generator(datagen.flow(x
                    steps_per_epoc

# here's a more "manual" example
for e in range(epochs):
    print('Epoch', e)
    batches = 0
    for x_batch, y_batch in datage
        model.fit(x_batch, y_batch
        batches += 1
        if batches >= len(x_train)
            # we need to break the
            # the generator loops
            break
```
Example of using `.flow_from_direc

```python
train_datagen = ImageDataGenerator
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True)

test_datagen = ImageDataGenerator(

train_generator = train_datagen.fl
        'data/train',
        target_size=(150, 150),
        batch_size=32,
        class_mode='binary')

validation_generator = test_datage
        'data/validation',
```

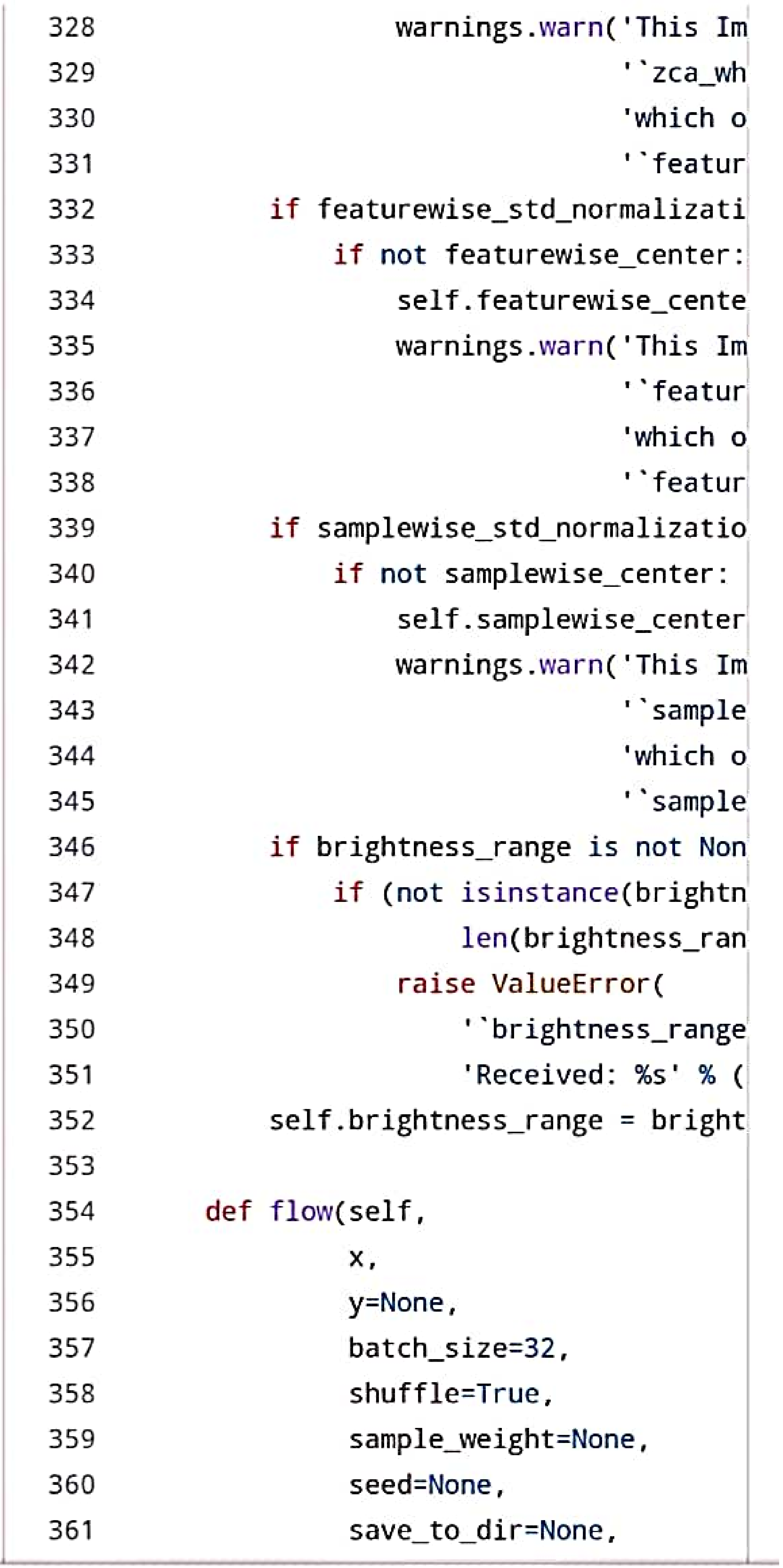
~~~
image_datagen.fit(images, augment=
mask_datagen.fit(masks, augment=Tr

image_generator = image_datagen.fl
    'data/images',
    class_mode=None,
    seed=seed)

mask_generator = mask_datagen.flow
    'data/masks',
    class_mode=None,
    seed=seed)

# combine generators into one whic
train_generator = zip(image_genera

model.fit_generator(
    train_generator,
    steps_per_epoch=2000,
    epochs=50)
```

Example of using ```.flow_from_dat


```python

train_df = pandas.read_csv("./trai
valid_df = pandas.read_csv("./vali

train_datagen = ImageDataGenerator
        rescale=1./255,
        shear_range=0.2,
        zoom_range=0.2,
~~~

```
    def __init__(self,
                 featurewise_center=Fa
                 samplewise_center=Fal
                 featurewise_std_norma
                 samplewise_std_normal
                 zca_whitening=False,
                 zca_epsilon=1e-6,
                 rotation_range=0,
                 width_shift_range=0.,
                 height_shift_range=0.
                 brightness_range=None
                 shear_range=0.,
                 zoom_range=0.,
                 channel_shift_range=0
                 fill_mode='nearest',
                 cval=0.,
                 horizontal_flip=False
                 vertical_flip=False,
                 rescale=None,
                 preprocessing_functio
                 data_format='channels
                 validation_split=0.0,
                 interpolation_order=1
                 dtype='float32'):

        self.featurewise_center = feat
        self.samplewise_center = sampl
        self.featurewise_std_normaliza
        self.samplewise_std_normalizat
        self.zca_whitening = zca_white
        self.zca_epsilon = zca_epsilon
        self.rotation_range = rotation
        self.width_shift_range = width
        self.height_shift_range = heig
    test_datagen = ImageDataGenerator(

    train_generator = train_datagen.fl
            dataframe=train_df,
            directory='data/train',
            x_col="filename",
            y_col="class",
            target_size=(150, 150),
            batch_size=32,
            class_mode='binary')

    validation_generator = test_datage
            dataframe=valid_df,
            directory='data/validation
            x_col="filename",
            y_col="class",
            target_size=(150, 150),
            batch_size=32,
            class_mode='binary')

    model.fit_generator(
            train_generator,
            steps_per_epoch=2000,
            epochs=50,
            validation_data=validation
            validation_steps=800)
    ```
    """

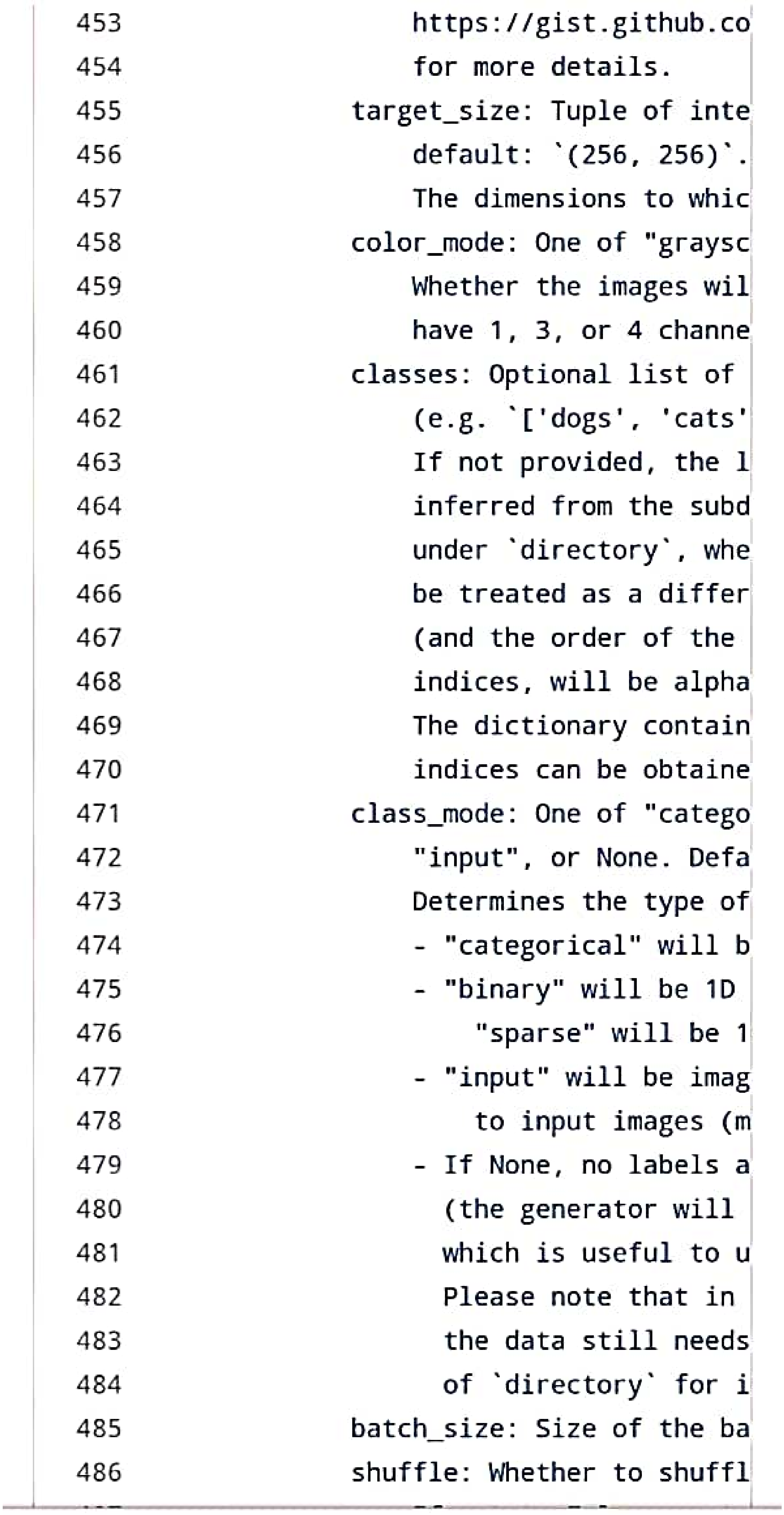
    def __init__(self,
                 featurewise_center=Fa
                 samplewise_center=Fal
                 featurewise_std_norma
```
```



```
            self.row_axis = 1
            self.col_axis = 2
        if validation_split and not 0
            raise ValueError(
                '`validation_split` mu
                ' Received: %s' % vali
        self._validation_split = valid

        self.mean = None
        self.std = None
        self.zca_whitening_matrix = No

        if isinstance(zoom_range, (flo
            self.zoom_range = [1 - zoo
        elif (len(zoom_range) == 2 and
              all(isinstance(val, (flo
            self.zoom_range = [zoom_ra
        else:
            raise ValueError('`zoom_ra
                             'a tuple
                             'Received
        if zca_whitening:
            if not featurewise_center:
                self.featurewise_cente
                warnings.warn('This Im
                              '`zca_wh
                              'setting
            if featurewise_std_normali
                self.featurewise_std_n
                warnings.warn('This Im
                              '`zca_wh
                              'which o
                              '`featur
        if featurewise_std_normalizati
```

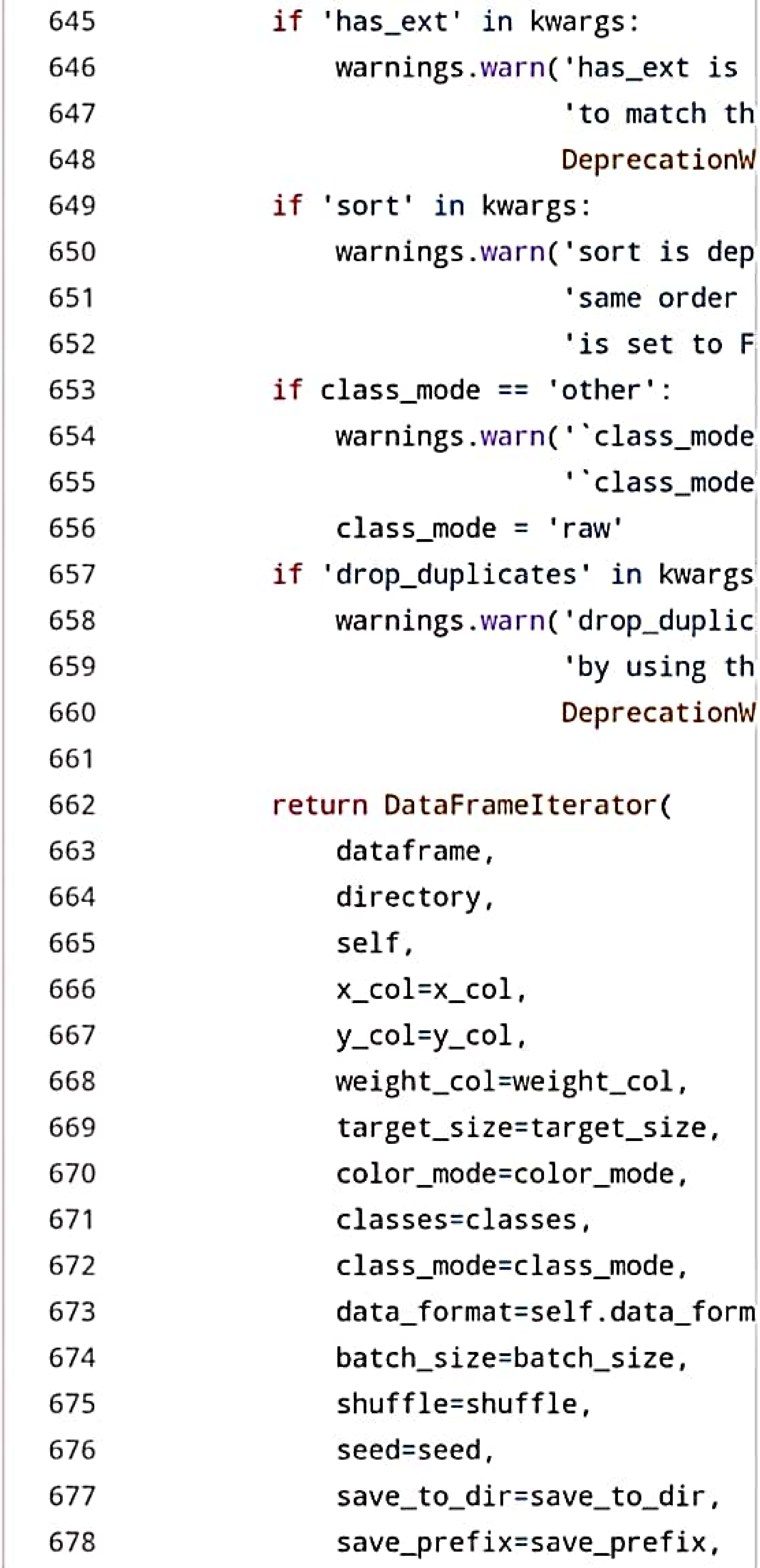


```
                warnings.warn('This Im
                              '`zca_wh
                              'which o
                              '`featur
        if featurewise_std_normalizati
            if not featurewise_center:
                self.featurewise_cente
                warnings.warn('This Im
                              '`featur
                              'which o
                              '`featur
        if samplewise_std_normalizatio
            if not samplewise_center:
                self.samplewise_center
                warnings.warn('This Im
                              '`sample
                              'which o
                              '`sample
        if brightness_range is not Non
            if (not isinstance(brightn
                    len(brightness_ran
                raise ValueError(
                    '`brightness_range
                    'Received: %s' % (
        self.brightness_range = bright

    def flow(self,
             x,
             y=None,
             batch_size=32,
             shuffle=True,
             sample_weight=None,
             seed=None,
             save_to_dir=None,
                Prefix to use for file
                (only relevant if `sav
            save_format: one of "png",
                (only relevant if `sav
            ignore_class_split: Boolea
                in number of classes i
                split (useful for non-
            subset: Subset of data (`"
                `validation_split` is

        # Returns
            An `Iterator` yielding tup
                where `x` is a NumPy a
                (in the case of a sing
                of NumPy arrays (in th
                additional inputs) and
                of corresponding label
                the yielded tuples are
                If `y` is None, only t
        """
        return NumpyArrayIterator(
            x,
            y,
            self,
            batch_size=batch_size,
            shuffle=shuffle,
            sample_weight=sample_weigh
            seed=seed,
            data_format=self.data_form
            save_to_dir=save_to_dir,
            save_prefix=save_prefix,
            save_format=save_format,
            ignore_class_split=ignore_
            subset=subset,
            save_format=save_format,
            ignore_class_split=ignore_
            subset=subset,
            dtype=self.dtype
        )

    def flow_from_directory(self,
                            directory,
                            target_siz
                            color_mode
                            classes=No
                            class_mode
                            batch_size
                            shuffle=Tr
                            seed=None,
                            save_to_di
                            save_prefi
                            save_forma
                            follow_lin
                            subset=Non
                            interpolat
                            keep_aspec
        """Takes the path to a directo

        # Arguments
            directory: string, path to
                It should contain one
                Any PNG, JPG, BMP, PPM
                inside each of the sub
                will be included in th
                See [this script](
                https://gist.github.co
                for more details.
            target_size: Tuple of inte
```



```
        https://gist.github.co
        for more details.
    target_size: Tuple of inte
        default: `(256, 256)`.
        The dimensions to whic
    color_mode: One of "graysc
        Whether the images wil
        have 1, 3, or 4 channe
    classes: Optional list of 
        (e.g. `['dogs', 'cats'
        If not provided, the l
        inferred from the subd
        under `directory`, whe
        be treated as a differ
        (and the order of the 
        indices, will be alpha
        The dictionary contain
        indices can be obtaine
    class_mode: One of "catego
        "input", or None. Defa
        Determines the type of
        - "categorical" will b
        - "binary" will be 1D 
            "sparse" will be 1
        - "input" will be imag
            to input images (m
        - If None, no labels a
          (the generator will 
          which is useful to u
          Please note that in 
          the data still needs
          of `directory` for i
    batch_size: Size of the ba
    shuffle: Whether to shuffl
```

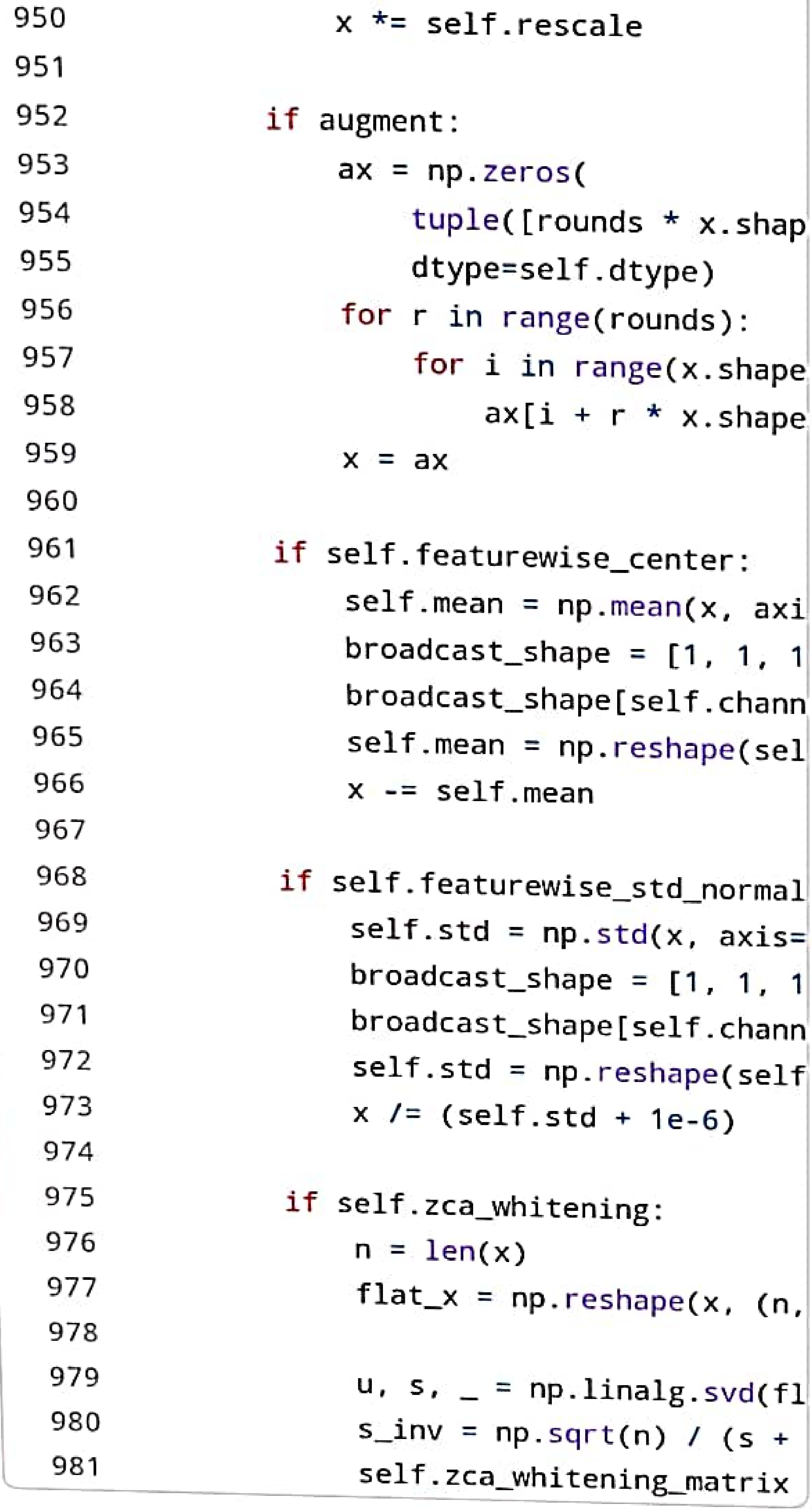
```
                            dataframe,
                            directory=
                            x_col="fil
                            y_col="cla
                            weight_col
                            target_siz
                            color_mode
                            classes=No
                            class_mode
                            batch_size
                            shuffle=Tr
                            seed=None,
                            save_to_di
                            save_prefi
                            save_forma
                            subset=Non
                            interpolat
                            validate_f
                            **kwargs):
        """Takes the dataframe and the
         and generates batches of augm

        **A simple tutorial can be fou
                                    ht


        # Arguments
            dataframe: Pandas datafram
                `directory` (or absolu
                images in a string col
                depending on the `clas
                - if `class_mode` is `
                    include the `y_col
                    Values in column c
                    or list/tuple if m
                the center with target

        # Returns
            A `DirectoryIterator` yiel
                where `x` is a NumPy a
                of images with shape `
                and `y` is a NumPy arr
        """
        return DirectoryIterator(
            directory,
            self,
            target_size=target_size,
            keep_aspect_ratio=keep_asp
            color_mode=color_mode,
            classes=classes,
            class_mode=class_mode,
            data_format=self.data_form
            batch_size=batch_size,
            shuffle=shuffle,
            seed=seed,
            save_to_dir=save_to_dir,
            save_prefix=save_prefix,
            save_format=save_format,
            follow_links=follow_links,
            subset=subset,
            interpolation=interpolatio
            dtype=self.dtype
        )

    def flow_from_dataframe(self,
                            dataframe,
                            directory=
                            x_col="fil
                            y_col="cla
                Values in column c
                or list/tuple if m
            - if `class_mode` is `
                the given `y_col`
            - if `class_mode` is `
            the columns specified
            - if `class_mode` is `
        directory: string, path tc
            data in `x_col` column
        x_col: string, column in `
            absolute paths if `dir
        y_col: string or list, col
        weight_col: string, column
            weights. Default: `Non
        target_size: tuple of inte
            The dimensions to whic
        color_mode: one of "graysc
            Whether the images wil
        classes: optional list of
            Default: None. If not
            automatically inferred
            which will map to the
            The dictionary contain
            indices can be obtaine
        class_mode: one of "binary
            "raw", sparse" or None
            Mode for yielding the
            - `"binary"`: 1D NumPy
            - `"categorical"`: 2D
                Supports multi-lab
            - `"input"`: images id
                work with autoenco
            - `"multi_output"`: li
            - `"raw"`: NumPy array
```



```
        if 'has_ext' in kwargs:
            warnings.warn('has_ext is
                          'to match th
                          DeprecationW
        if 'sort' in kwargs:
            warnings.warn('sort is dep
                          'same order
                          'is set to F
        if class_mode == 'other':
            warnings.warn('`class_mode
                          '`class_mode
            class_mode = 'raw'
        if 'drop_duplicates' in kwargs
            warnings.warn('drop_duplic
                          'by using th
                          DeprecationW

        return DataFrameIterator(
            dataframe,
            directory,
            self,
            x_col=x_col,
            y_col=y_col,
            weight_col=weight_col,
            target_size=target_size,
            color_mode=color_mode,
            classes=classes,
            class_mode=class_mode,
            data_format=self.data_form
            batch_size=batch_size,
            shuffle=shuffle,
            seed=seed,
            save_to_dir=save_to_dir,
            save_prefix=save_prefix,
            else:
                warnings.warn('This Im
                              '`featur
                              'been fi
                              'first b
        if self.featurewise_std_normal
            if self.std is not None:
                x /= (self.std + 1e-6)
            else:
                warnings.warn('This Im
                              '`featur
                              'but it
                              'been fi
                              'first b
        if self.zca_whitening:
            if self.zca_whitening_matr
                flat_x = x.reshape(-1,
                white_x = flat_x @ sel
                x = np.reshape(white_x
            else:
                warnings.warn('This Im
                              '`zca_wh
                              'been fi
                              'first b
        return x

    def get_random_transform(self, img
        """Generates random parameters

        # Arguments
            seed: Random seed.
            img_shape: Tuple of intege
                Shape of the image tha

```

```

if self.zoom_range[0] == 1 and
    zx, zy = 1, 1
else:
    zx, zy = np.random.uniform
        self.zoom_range[0],
        self.zoom_range[1],
        2)

flip_horizontal = (np.random.r
flip_vertical = (np.random.ran

channel_shift_intensity = None
if self.channel_shift_range !=
    channel_shift_intensity =


brightness = None
if self.brightness_range is no
    brightness = np.random.uni


transform_parameters = {'theta
                        'tx':
                        'ty':
                        'shear
                        'zx':
                        'zy':
                        'flip_
                        'flip_
                        'chann
                        'brigh

return transform_parameters
```

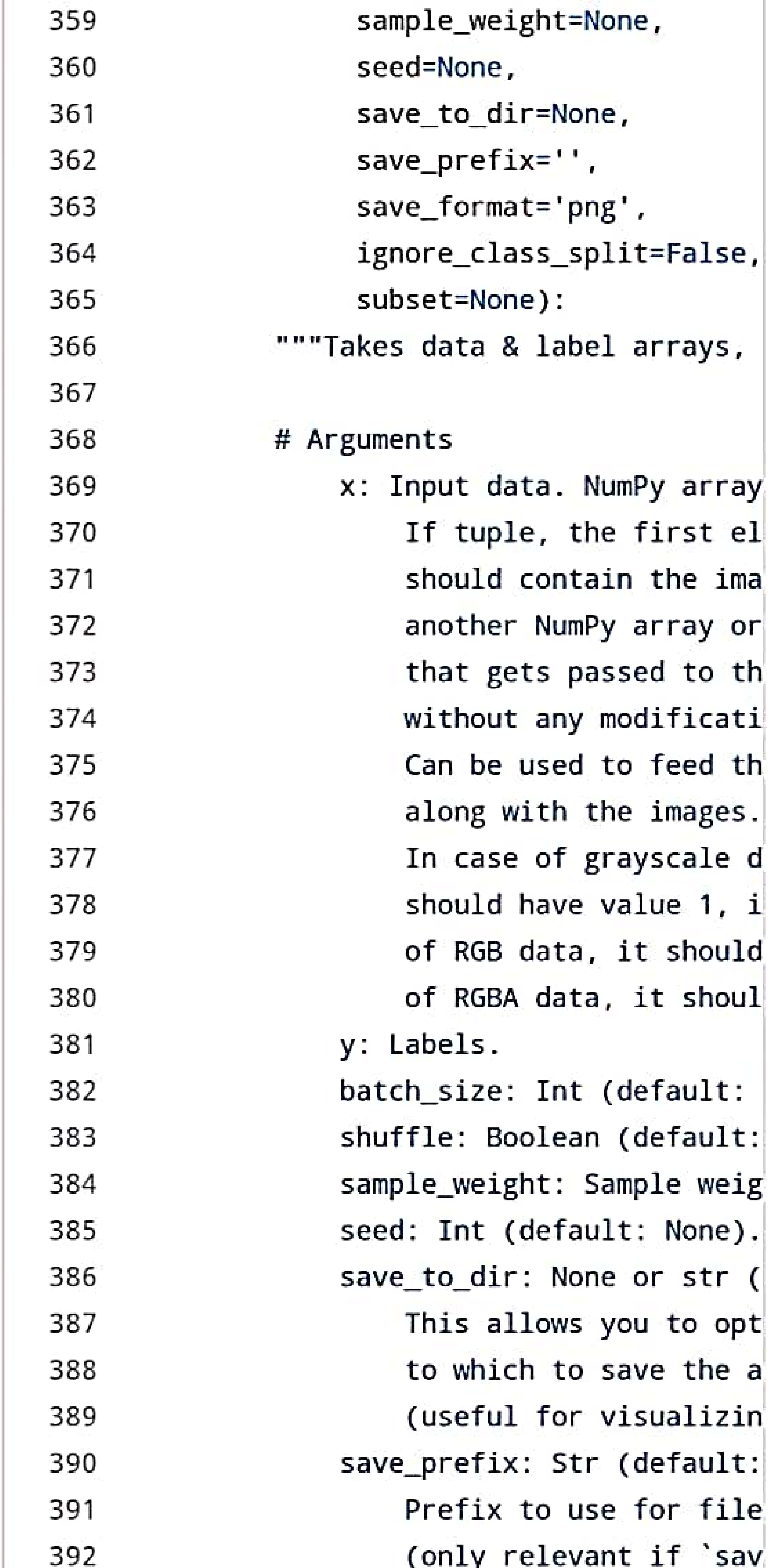
```
        Applies a transformation to

        # Arguments
            x: 3D tensor, single image
            transform_parameters: Dict
                describing the transfo
                Currently, the followi
                from the dictionary ar
                - `'theta'`: Float. Ro
                - `'tx'`: Float. Shift
                - `'ty'`: Float. Shift
                - `'shear'`: Float. Sh
                - `'zx'`: Float. Zoom
                - `'zy'`: Float. Zoom
                - `'flip_horizontal'`:
                - `'flip_vertical'`: B
                - `'channel_shift_inte
                - `'brightness'`: Floa

        # Returns
            A transformed version of t
        """
        # x is a single image, so it d
        img_row_axis = self.row_axis -
        img_col_axis = self.col_axis -
        img_channel_axis = self.channe

        x = apply_affine_transform(x,
                                   tra
                                   tra
                                   tra
                                   tra
                                   tra
                                   row
                                   col
```



```
            x *= self.rescale

        if augment:
            ax = np.zeros(
                tuple([rounds * x.shap
                dtype=self.dtype)
            for r in range(rounds):
                for i in range(x.shape
                    ax[i + r * x.shape
            x = ax

        if self.featurewise_center:
            self.mean = np.mean(x, axi
            broadcast_shape = [1, 1, 1
            broadcast_shape[self.chann
            self.mean = np.reshape(sel
            x -= self.mean

        if self.featurewise_std_normal
            self.std = np.std(x, axis=
            broadcast_shape = [1, 1, 1
            broadcast_shape[self.chann
            self.std = np.reshape(self
            x /= (self.std + 1e-6)

        if self.zca_whitening:
            n = len(x)
            flat_x = np.reshape(x, (n,

            u, s, _ = np.linalg.svd(fl
            s_inv = np.sqrt(n) / (s +
            self.zca_whitening_matrix
```

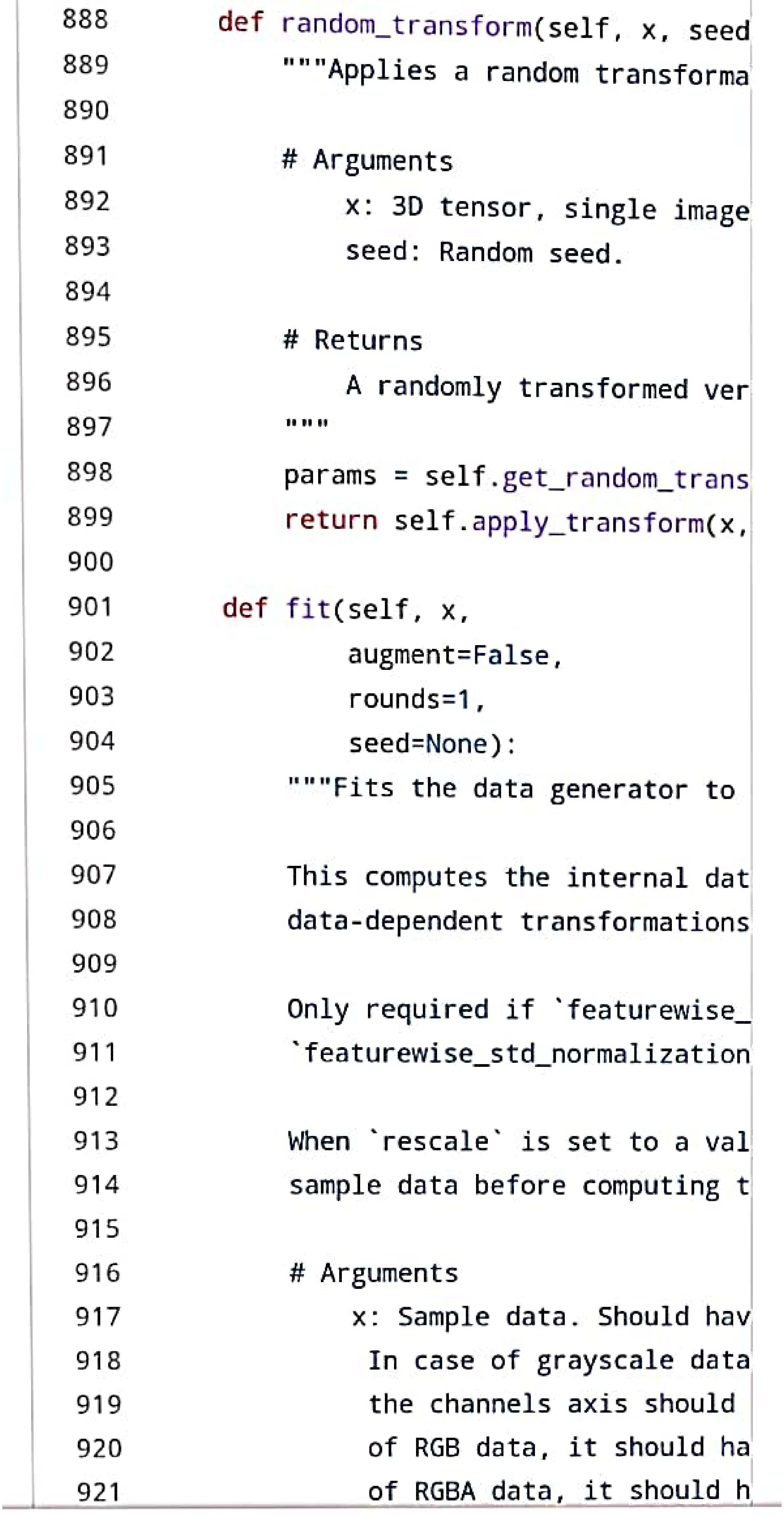
~~~

validation_generator = test_datage
        'data/validation',
        target_size=(150, 150),
        batch_size=32,
        class_mode='binary')

model.fit_generator(
        train_generator,
        steps_per_epoch=2000,
        epochs=50,
        validation_data=validation
        validation_steps=800)
```

Example of transforming images and

```python
# we create two instances with the
data_gen_args = dict(featurewise_c
                     featurewise_s
                     rotation_rang
                     width_shift_r
                     height_shift_
                     zoom_range=0.
image_datagen = ImageDataGenerator
mask_datagen = ImageDataGenerator(

# Provide the same seed and keywor
seed = 1
image_datagen.fit(images, augment=
mask_datagen.fit(masks, augment=Tr

image_generator = image_datagen.fl
    'data/images'
~~~

```
        self.zca_whitening = zca_white
        self.zca_epsilon = zca_epsilon
        self.rotation_range = rotation
        self.width_shift_range = width
        self.height_shift_range = heig
        self.shear_range = shear_range
        self.zoom_range = zoom_range
        self.channel_shift_range = cha
        self.fill_mode = fill_mode
        self.cval = cval
        self.horizontal_flip = horizon
        self.vertical_flip = vertical_
        self.rescale = rescale
        self.preprocessing_function =
        self.dtype = dtype
        self.interpolation_order = int

        if data_format not in {'channe
            raise ValueError(
                '`data_format` should
                '(channel after row an
                '`"channels_first"` (c
                'Received: %s' % data_
        self.data_format = data_format
        if data_format == 'channels_fi
            self.channel_axis = 1
            self.row_axis = 2
            self.col_axis = 3
        if data_format == 'channels_la
            self.channel_axis = 3
            self.row_axis = 1
            self.col_axis = 2
```



```
             sample_weight=None,
             seed=None,
             save_to_dir=None,
             save_prefix='',
             save_format='png',
             ignore_class_split=False,
             subset=None):
        """Takes data & label arrays,

        # Arguments
            x: Input data. NumPy array
                If tuple, the first el
                should contain the ima
                another NumPy array or
                that gets passed to th
                without any modificati
                Can be used to feed th
                along with the images.
                In case of grayscale d
                should have value 1, i
                of RGB data, it should
                of RGBA data, it shoul
            y: Labels.
            batch_size: Int (default:
            shuffle: Boolean (default:
            sample_weight: Sample weig
            seed: Int (default: None).
            save_to_dir: None or str (
                This allows you to opt
                to which to save the a
                (useful for visualizin
            save_prefix: Str (default:
                Prefix to use for file
                (only relevant if `sav
                  Please note that in
                  the data still needs
                  of `directory` for i
            batch_size: Size of the ba
            shuffle: Whether to shuffl
                If set to False, sorts
            seed: Optional random seed
            save_to_dir: None or str (
                This allows you to opt
                a directory to which t
                the augmented pictures
                (useful for visualizin
            save_prefix: Str. Prefix t
                (only relevant if `sav
            save_format: One of "png",
                (only relevant if `sav
            follow_links: Whether to f
                class subdirectories (
            subset: Subset of data (`"
                `validation_split` is
            interpolation: Interpolati
                resample the image if
                target size is differe
                Supported methods are
                and `"bicubic"`.
                If PIL version 1.1.3 o
                supported. If PIL vers
                `"box"` and `"hamming"
                By default, `"nearest"
            keep_aspect_ratio: Boolean
                size without aspect ra
                the center with target

        # Returns
            A `DirectoryIterator` yiel
```

```
                - `"sparse"`: 1D NumPy
                - `None`, no targets a
                    batches of image d
                    `model.predict_gen
            batch_size: size of the ba
            shuffle: whether to shuffl
            seed: optional random seed
            save_to_dir: None or str (
                This allows you to opt
                to which to save the a
                (useful for visualizin
            save_prefix: str. Prefix t
                (only relevant if `sav
            save_format: one of "png",
                (only relevant if `sav
            follow_links: whether to f
                (default: False).
            subset: Subset of data (`"
                `validation_split` is
            interpolation: Interpolati
                target size is differe
                Supported methods are
                If PIL version 1.1.3 o
                supported. If PIL vers
                `"hamming"` are also s
            validate_filenames: Boolea
                `x_col`. If `True`, in
                option can lead to spe
                Default: `True`.

        # Returns
            A `DataFrameIterator` yiel
            where `x` is a NumPy array
            of images with shape `(bat
            seed=seed,
            save_to_dir=save_to_dir,
            save_prefix=save_prefix,
            save_format=save_format,
            subset=subset,
            interpolation=interpolatio
            validate_filenames=validat
            dtype=self.dtype
        )

    def standardize(self, x):
        """Applies the normalization c

        `x` is changed in-place since
        to standardize images and feed
        would be created instead it wo
        If you want to apply this meth
        you can call the method creati

        standardize(np.copy(x))

        # Arguments
            x: Batch of inputs to be n

        # Returns
            The inputs, normalized.
        """
        if self.preprocessing_function
            x = self.preprocessing_fun
        if self.rescale:
            x *= self.rescale
        if self.samplewise_center:
            x -= np.mean(x, keepdims=T
        if self.samplewise_std_normali
        img_col_axis = self.col_axis -

        if seed is not None:
            np.random.seed(seed)

        if self.rotation_range:
            theta = np.random.uniform(
                -self.rotation_range,
                self.rotation_range)
        else:
            theta = 0

        if self.height_shift_range:
            try:  # 1-D array-like or
                tx = np.random.choice(
                tx *= np.random.choice
            except ValueError:  # floa
                tx = np.random.uniform

            if np.max(self.height_shif
                tx *= img_shape[img_ro
        else:
            tx = 0

        if self.width_shift_range:
            try:  # 1-D array-like or
                ty = np.random.choice(
                ty *= np.random.choice
            except ValueError:  # floa
                ty = np.random.uniform

            if np.max(self.width_shift
                ty *= img_shape[img_co
        else:
            ty = 0
```



```
    def random_transform(self, x, seed
        """Applies a random transforma

        # Arguments
            x: 3D tensor, single image
            seed: Random seed.

        # Returns
            A randomly transformed ver
        """
        params = self.get_random_trans
        return self.apply_transform(x,

    def fit(self, x,
            augment=False,
            rounds=1,
            seed=None):
        """Fits the data generator to

        This computes the internal dat
        data-dependent transformations

        Only required if `featurewise_
        `featurewise_std_normalization

        When `rescale` is set to a val
        sample data before computing t

        # Arguments
            x: Sample data. Should hav
             In case of grayscale data
             the channels axis should
             of RGB data, it should ha
             of RGBA data, it should h
             of RGB data, it should ha
             of RGBA data, it should h
            augment: Boolean (default:
                Whether to fit on rand
            rounds: Int (default: 1).
                If using data augmenta
                this is how many augme
            seed: Int (default: None).
        """
        x = np.asarray(x, dtype=self.d
        if x.ndim != 4:
            raise ValueError('Input to
                             'Got arra
        if x.shape[self.channel_axis]
            warnings.warn(
                'Expected input to be
                'following the data fo
                self.data_format + '"
                str(self.channel_axis)
                'either 1, 3 or 4 chan
                str(self.channel_axis)
                'However, it was passe
                str(x.shape) + ' (' +
                ' channels).')

        if seed is not None:
            np.random.seed(seed)

        x = np.copy(x)
        if self.rescale:
            x *= self.rescale
```