Image Augmentation for Deep Learning With Keras

by **Jason Brownlee** on June 29, 2016 in **Deep Learning**

Data preparation is required when working with neural network and deep learning models. Increasingly data augmentation is also required on more complex object recognition tasks.

In this post you will discover how to use data preparation and data augmentation with your image datasets when developing and evaluating deep learning models in Python with Keras.

After reading this post, you will know:

- About the image augmentation API provide by Keras and how to use it with your models.
- How to perform feature standardization.
- How to perform ZCA whitening of your images.
- How to augment data with random rotations, shifts and flips.
- How to save augmented image data to disk.

Let's get started.

- **Update**: The examples in this post were updated for the latest Keras API. The datagen.next() function was removed.
- **Update Oct/2016**: Updated examples for Keras 1.1.0, TensorFlow 0.10.0 and scikit-learn v0.18.
- **Update Jan/2017**: Updated examples for Keras 1.2.0 and TensorFlow 0.12.1.
- **Update Mar/2017**: Updated example for Keras 2.0.2, TensorFlow 1.0.1 and Theano 0.9.0.

Keras Image Augmentation API

Like the rest of Keras, the image augmentation API is simple and powerful.

Keras provides the ImageDataGenerator class that defines the configuration for image data preparation and augmentation. This includes capabilities such as:

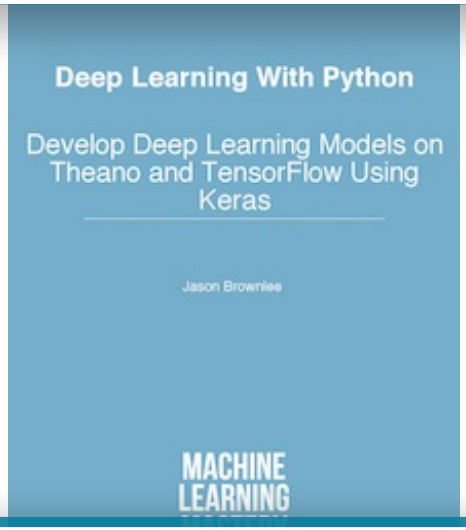
- Sample-wise standardization.
- Feature-wise standardization.
- ZCA whitening.
- Random rotation, shifts, shear and flips.
- Dimension reordering.
- Save augmented images to disk.

An augmented image generator can be created as follows:

```
1 datagen = ImageDataGenerator()
```

Rather than performing the operations on your entire image dataset in memory, the API is designed to be iterated by the deep learning model fitting process, creating augmented image data for you just-in-time. This reduces your memory overhead, but adds some additional time cost during model training.

After you have created and configured your **ImageDataGenerator**, you must fit it on your data. This will calculate any statistics required to actually perform the transforms to your image data. You can do this by calling the **fit()** function on the data generator and pass it your training dataset.



Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017

How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

```
datagen.fit(train)
```

The data generator itself is in fact an iterator, returning batches of image samples when requested. We can configure the batch size and prepare the data generator and get batches of images by calling the **flow()** function.

```
X_batch, y_batch = datagen.flow(train, train, batch_size=32)
```

Finally we can make use of the data generator. Instead of calling the **fit()** function on our model, we must call the **fit_generator()** function and pass in the data generator and the desired length of an epoch as well as the total number of epochs on which to train.

```
fit_generator(datagen, samples_per_epoch=len(train), epochs=100)
```

You can learn more about the Keras image data generator API in the Keras documentation.

Point of Comparison for Image Augmentation

Now that you know how the image augmentation API in Keras works, let's look at some examples.

We will use the MNIST handwritten digit recognition task in these examples. To begin with, let's take a look at the first 9 images in the training dataset.


Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016


Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016


Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

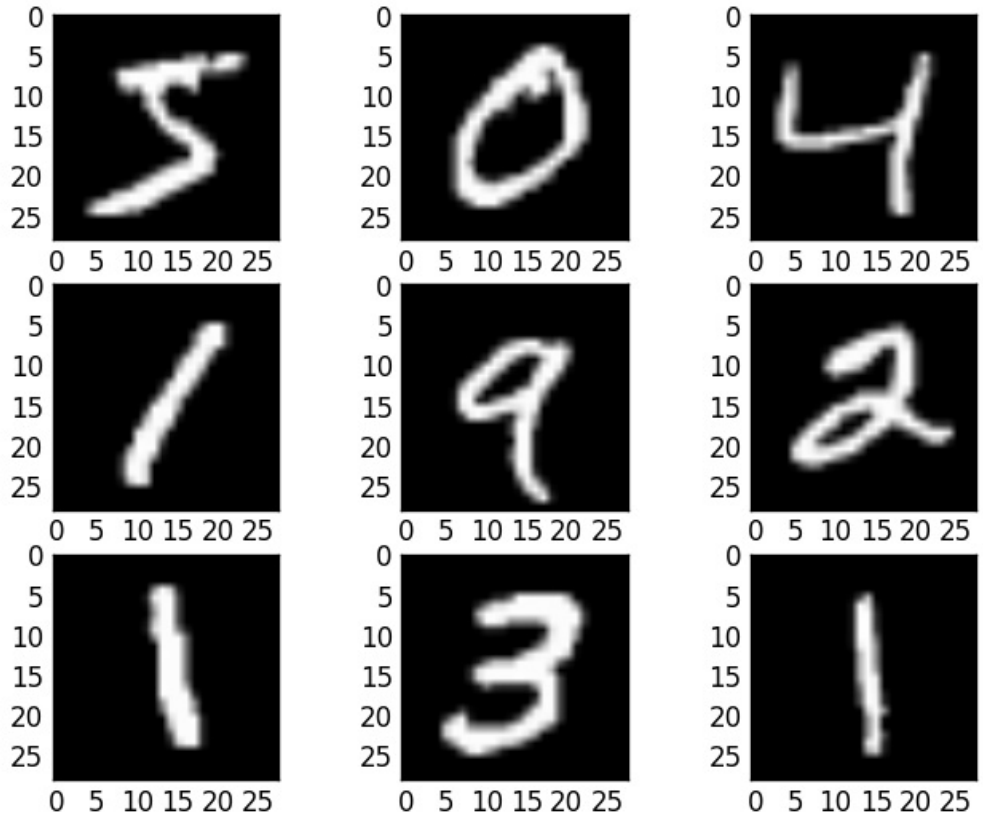

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016


Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016


How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras
AUGUST 9, 2016

```
# Plot images
from keras.datasets import mnist
from matplotlib import pyplot
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# create a grid of 3x3 images
for i in range(0, 9):
 pyplot.subplot(330 + 1 + i)
 pyplot.imshow(X_train[i], cmap=pyplot.get_cmap('gray'))
# show the plot
pyplot.show()
```

Running this example provides the following image that we can use as a point of comparison with the image preparation and augmentation in the examples below.



Example MNIST images

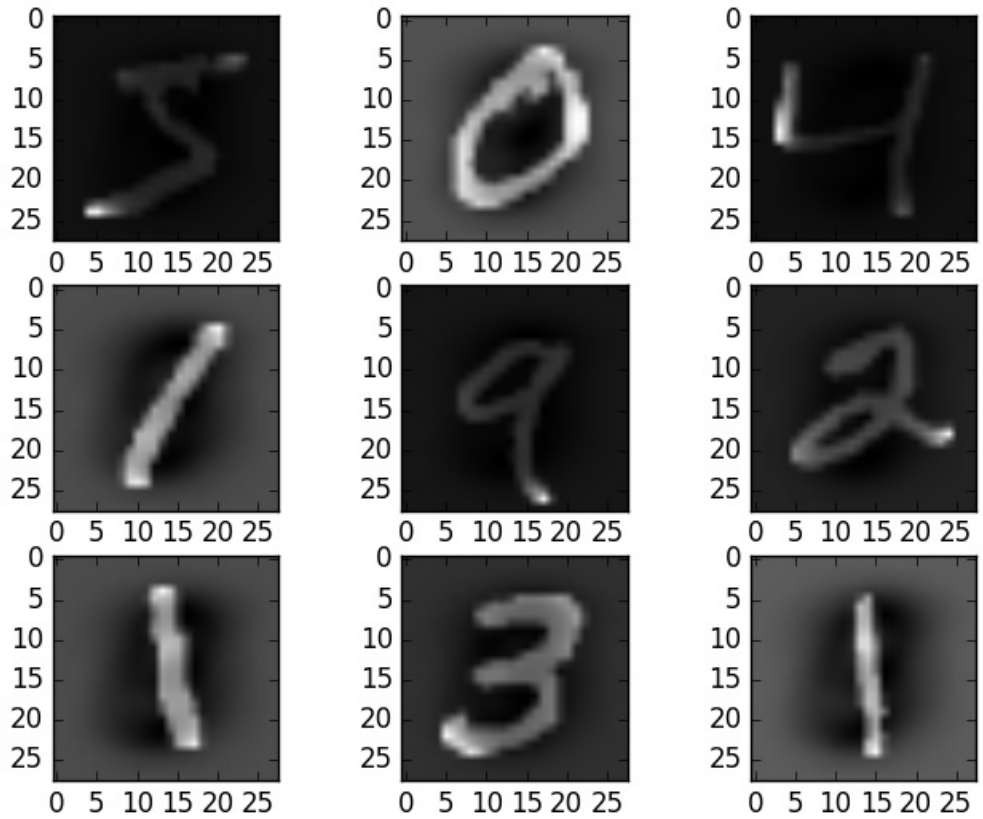
Feature Standardization

It is also possible to standardize pixel values across the entire dataset. This is called feature standardization and mirrors the type of standardization often performed for each column in a tabular dataset.

You can perform feature standardization by setting the featurewise_center and featurewise_std_normalization arguments on the ImageDataGenerator class. These are in fact set to True by default and creating an instance of ImageDataGenerator with no arguments will have the same effect.

```
# Standardize images across the dataset, mean=0, stdev=1
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(featurewise_center=True, featurewise_std_normalization=T
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running this example you can see that the effect is different, seemingly darkening and lightening different digits.



Standardized Feature MNIST Images

ZCA Whitening

A whitening transform of an image is a linear algebra operation that reduces the redundancy in the matrix of pixel images.

Less redundancy in the image is intended to better highlight the structures and features in the image to the learning algorithm.

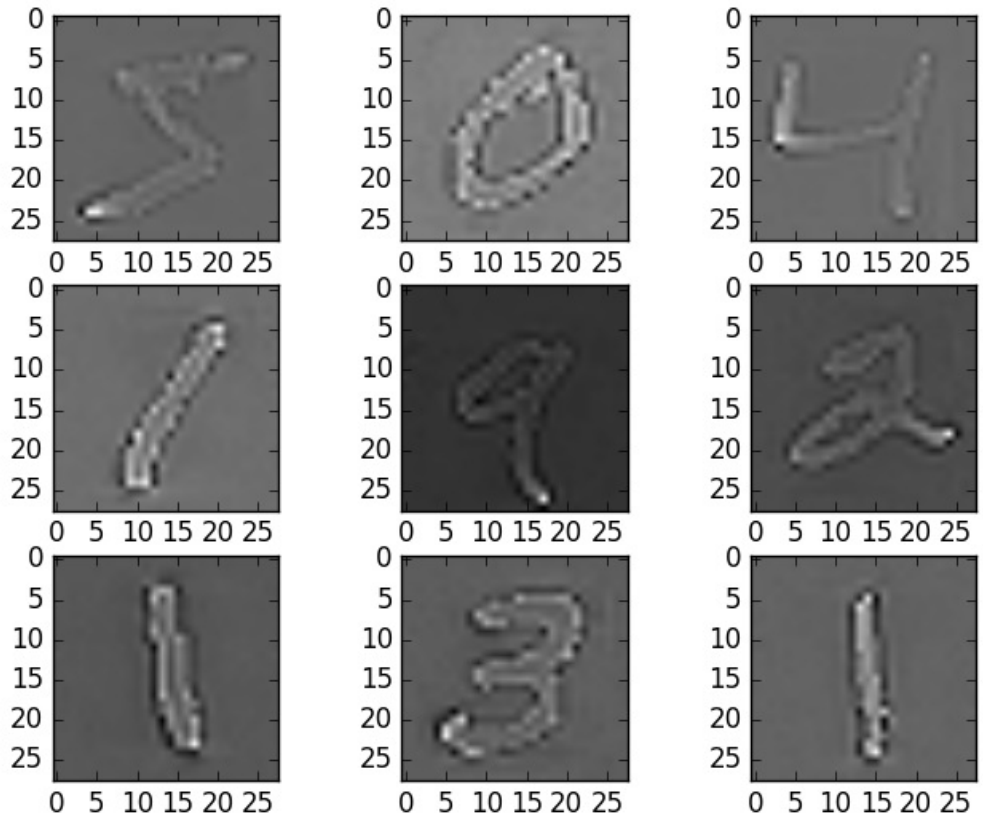
Typically, image whitening is performed using the Principal Component Analysis (PCA) technique. More recently, an alternative called ZCA (learn more in Appendix A of this tech report)

shows better results and results in transformed images that keeps all of the original dimensions and unlike PCA, resulting transformed images still look like their originals.

You can perform a ZCA whitening transform by setting the zca_whitening argument to True.

```
# ZCA whitening
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(zca_whitening=True)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running the example, you can see the same general structure in the images and how the outline of each digit has been highlighted.



ZCA Whitening MNIST Images

Random Rotations

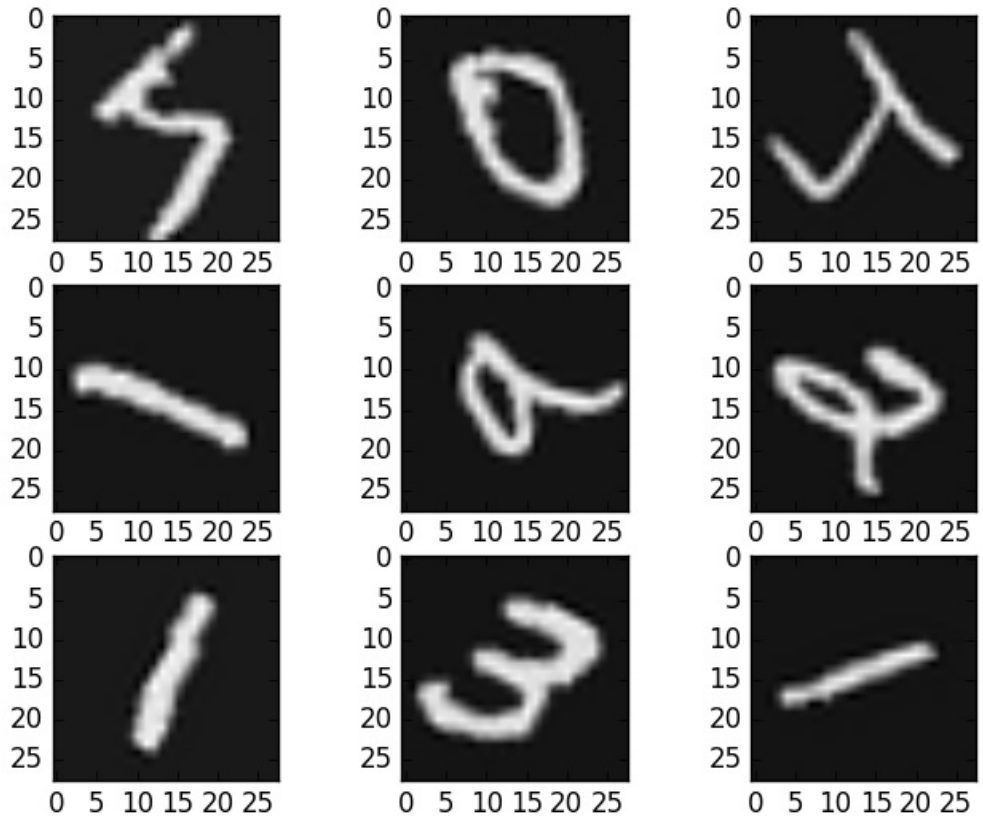
Sometimes images in your sample data may have varying and different rotations in the scene.

You can train your model to better handle rotations of images by artificially and randomly rotating images from your dataset during training.

The example below creates random rotations of the MNIST digits up to 90 degrees by setting the rotation_range argument.

```
# Random Rotations
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(rotation_range=90)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running the example, you can see that images have been rotated left and right up to a limit of 90 degrees. This is not helpful on this problem because the MNIST digits have a normalized orientation, but this transform might be of help when learning from photographs where the objects may have different orientations.



Random Rotations of MNIST Images

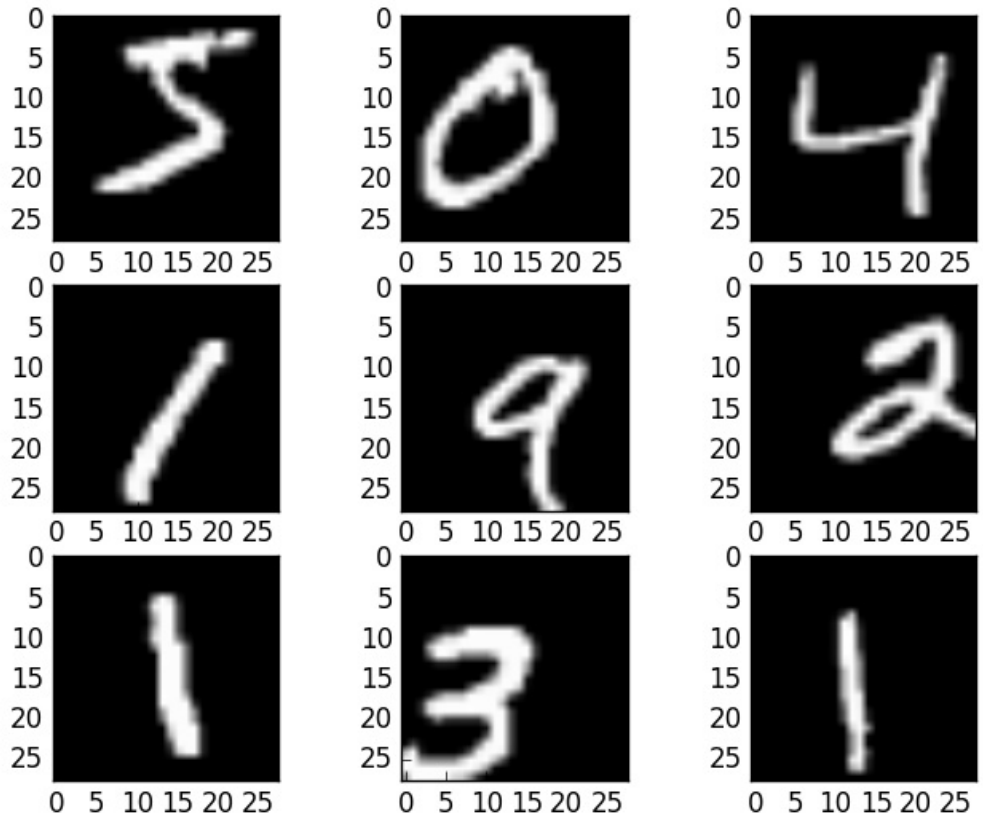
Random Shifts

Objects in your images may not be centered in the frame. They may be off-center in a variety of different ways.

You can train your deep learning network to expect and currently handle off-center objects by artificially creating shifted versions of your training data. Keras supports separate horizontal and vertical random shifting of training data by the width_shift_range and height_shift_range arguments.

```
# Random Shifts
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
shift = 0.2
datagen = ImageDataGenerator(width_shift_range=shift, height_shift_range=shift)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running this example creates shifted versions of the digits. Again, this is not required for MNIST as the handwritten digits are already centered, but you can see how this might be useful on more complex problem domains.



Random Shifted MNIST Images

Random Flips

Another augmentation to your image data that can improve performance on large and complex problems is to create random flips of images in your training data.

Keras supports random flipping along both the vertical and horizontal axes using the vertical_flip and horizontal_flip arguments.

```
# Random Flips
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(horizontal_flip=True, vertical_flip=True)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running this example you can see flipped digits. Flipping digits is not useful as they will always have the correct left and right orientation, but this may be useful for problems with photographs of objects in a scene that can have a varied orientation.

Randomly Flipped MNIST Images

Saving Augmented Images to File

The data preparation and augmentation is performed just in time by Keras.

This is efficient in terms of memory, but you may require the exact images used during training. For example, perhaps you would like to use them with a different software package later or only generate them once and use them on multiple different deep learning models or configurations.

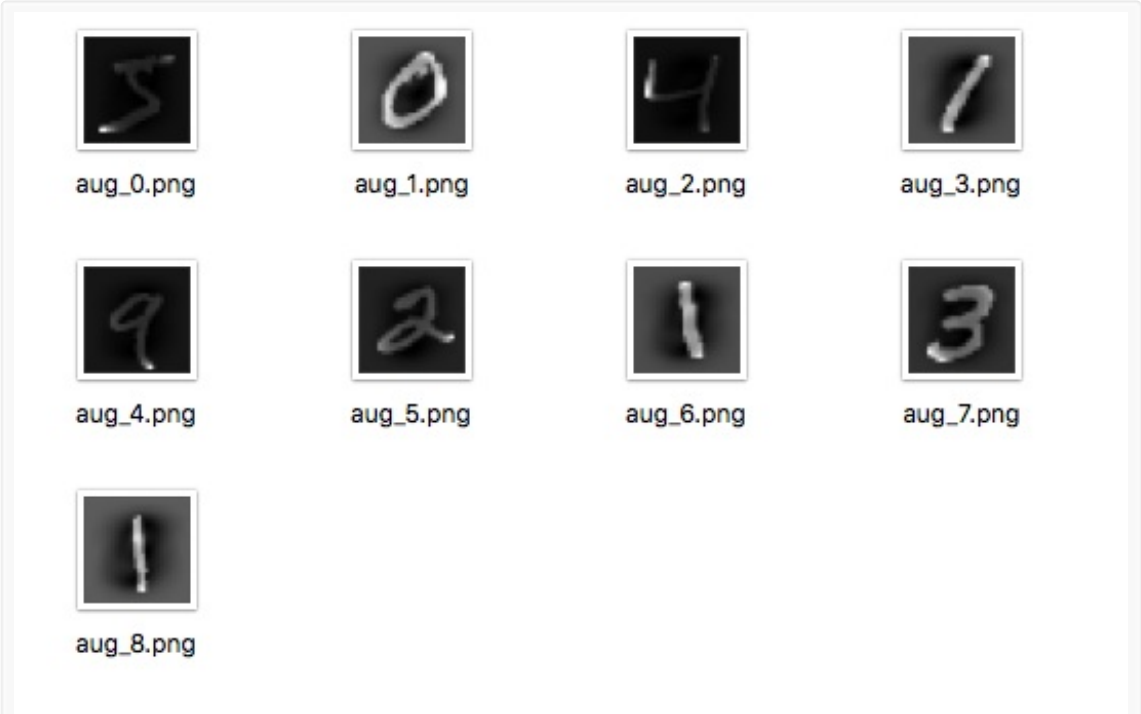
Keras allows you to save the images generated during training. The directory, filename prefix and image file type can be specified to the flow() function before training. Then, during training,

the generated images will be written to file.

The example below demonstrates this and writes 9 images to a “images” subdirectory with the prefix “aug” and the file type of PNG.

```
# Save augmented images to file
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator
from matplotlib import pyplot
import os
from keras import backend as K
K.set_image_dim_ordering('th')
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 1, 28, 28)
X_test = X_test.reshape(X_test.shape[0], 1, 28, 28)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator()
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
os.makedirs('images')
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, save_to_dir='ima
 # create a grid of 3x3 images
 for i in range(0, 9):
  pyplot.subplot(330 + 1 + i)
  pyplot.imshow(X_batch[i].reshape(28, 28), cmap=pyplot.get_cmap('gray'))
 # show the plot
 pyplot.show()
 break
```

Running the example you can see that images are only written when they are generated.



Augmented MNIST Images Saved To File

Tips For Augmenting Image Data with Keras

Image data is unique in that you can review the data and transformed copies of the data and quickly get an idea of how the model may be perceive it by your model.

Below are some times for getting the most from image data preparation and augmentation for deep learning.

- **Review Dataset**. Take some time to review your dataset in great detail. Look at the images. Take note of image preparation and augmentations that might benefit the training process of your model, such as the need to handle different shifts, rotations or flips of objects in the scene.
- **Review Augmentations**. Review sample images after the augmentation has been performed. It is one thing to intellectually know what image transforms you are using, it is a very different thing to look at examples. Review images both with individual augmentations you are using as well as the full set of augmentations you plan to use. You may see ways to

simplify or further enhance your model training process.

- **Evaluate a Suite of Transforms**. Try more than one image data preparation and augmentation scheme. Often you can be surprised by results of a data preparation scheme you did not think would be beneficial.

Summary

In this post you discovered image data preparation and augmentation.

You discovered a range of techniques that you can use easily in Python with Keras for deep learning models. You learned about:

- The ImageDataGenerator API in Keras for generating transformed images just in time.
- Sample-wise and Feature wise pixel standardization.
- The ZCA whitening transform.
- Random rotations, shifts and flips of images.
- How to save transformed images to file for later reuse.

Do you have any questions about image data augmentation or this post? Ask your questions in the comments and I will do my best to answer.

About Jason Brownlee

Jason Brownlee, Ph.D. is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.
View all posts by Jason Brownlee →

‹ Standard Machine Learning Datasets To Practice in Weka

How to Better Understand Your Machine Learning Data in Weka ›

95 Responses to *Image Augmentation for Deep Learning With Keras*

Andy August 2, 2016 at 7:34 am #

REPLY

Interesting tutorial.

I'm working through the step to standardize images across the dataset and run into the following error:

AttributeError Traceback (most recent call last)
in ()
18 datagen.flow(X_train, y_train, batch_size=9)
19 # retrieve one batch of images
—> 20 X_batch, y_batch = datagen.next()
21 # create a grid of 3×3 images
22 for i in range(0, 9):

AttributeError: 'ImageDataGenerator' object has no attribute 'next'

I have checked the Keras documentation and see no mention of a next attribute.

Perhaps I'm missing something.

Thanks for the great tutorials!

Jason Brownlee August 2, 2016 at 8:21 am #

REPLY

Yep, the API has changed. See:
https://keras.io/preprocessing/image/

I will update all of the examples ASAP.

UPDATE: I have updated all examples in this post to use the new API. Let me know if you have any problems at all.

Andy August 3, 2016 at 9:18 am #

REPLY

Works like a charm! Thanks

Jason Brownlee August 3, 2016 at 9:35 am #

REPLY

Glad to hear it Andy.

narayan August 9, 2016 at 6:38 pm #

REPLY

for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
File “/usr/local/lib/python2.7/dist-packages/keras/preprocessing/image.py”, line 475, in next
x = self.image_data_generator.random_transform(x.astype(‘float32’))
File “/usr/local/lib/python2.7/dist-packages/keras/preprocessing/image.py”, line 346, in random_transform
fill_mode=self.fill_mode, cval=self.cval)
File “/usr/local/lib/python2.7/dist-packages/keras/preprocessing/image.py”, line 109, in apply_transform
x = np.stack(channel_images, axis=0)
AttributeError: ‘module’ object has no attribute ‘stack’

how to solve this error …?

Jason Brownlee August 15, 2016 at 11:13 am #

REPLY

I have not seen an error like that before. Perhaps there is a problem with your environment?

Consider re-installing Theano and/or Keras.

narayan August 26, 2016 at 9:02 pm #

REPLY

i solved this error by updating numpy version ….previously it 1.8.0..now 1.11.1..it means it should be more than 1.9.0

Jason Brownlee August 27, 2016 at 11:33 am #

REPLY

Great, glad to here it narayan.

narayan August 26, 2016 at 9:05 pm #

REPLY

Now i have question that how to decide value for this parameter So that i can get good testing accuracy ..i have training dataset with 110 category with 32000 images ..

featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=0.,
width_shift_range=0.,
height_shift_range=0.,
shear_range=0.,
zoom_range=0.,
channel_shift_range=0.,
fill_mode='nearest',
cval=0.,
horizontal_flip=False,
vertical_flip=False,

rescale=None,
dim_ordering=K.image_dim_ordering()

Waiting for your positive reply…

Jason Brownlee August 27, 2016 at 11:34 am #

REPLY

My advice is to try a suite of different configurations and see what works best on your problem.

Walid Ahmed November 9, 2016 at 2:08 am #

REPLY

Thanks a lot.
all worked fine except the last code to save images to file, I got the following exception

Walids-MacBook-Pro:DataAugmentation walidahmed$ python augment_save_to_file.py
Using TensorFlow backend.
Traceback (most recent call last):
File “augment_save_to_file.py”, line 20, in
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9, save_to_dir=’images’, save_prefix=’aug’, save_format=’png’):
File “/usr/local/lib/python2.7/site-packages/keras/preprocessing/image.py”, line 490, in next
img = array_to_img(batch_x[i], self.dim_ordering, scale=True)
File “/usr/local/lib/python2.7/site-packages/keras/preprocessing/image.py”, line 140, in array_to_img
raise Exception(‘Unsupported channel number: ‘, x.shape[2])
Exception: (‘Unsupported channel number: ‘, 28)

Any advice?
thanks again

Jason Brownlee November 9, 2016 at 9:52 am #

REPLY

Double check your version of Keras is 1.1.0 and TensorFlow is 0.10.

Sudesh November 11, 2016 at 9:37 pm #

REPLY

Hello Jason,

Thanks a lot for your tutorial. It is helping me in many ways.

I had question on mask image or target Y for training image X
Can i also transform Y along with X. Helps in the case of training for segmentation

Sudesh November 15, 2016 at 5:25 am #

REPLY

I managed to do it.

```
datagen =
ImageDataGenerator(shear_range=0.02,dim_ordering=K._image_dim_ordering,rotation
_range=5,width_shift_range=0.05,
height_shift_range=0.05,zoom_range=0.3,fill_mode='constant', cval=0)

for samples in range(0,100):
seed = rd.randint(low=10,high=100000)
for imags_batch in
datagen.flow(imgs_train,batch_size=batch_size,save_to_dir='augmented',save_prefix='a
ug',seed=seed,save_format='tif'):
print('-')
break
for imgs_mask_batch in datagen.flow(imgs_mask_train, batch_size=batch_size,
save_to_dir='augmented',seed=seed, save_prefix='mask_aug',save_format='tif'):
print('|')
break
```

```
print((samples+1)*batch_size)
```

Addie November 29, 2016 at 6:01 am #

REPLY

This is great stuff but I wonder if you could provide an example like this with an RGB image with three channels? I am getting some really buggy results personally with this ImageGenerator.

Jason Brownlee November 29, 2016 at 8:55 am #

REPLY

Great suggestion, thanks Addie.

Lucas December 24, 2016 at 9:02 am #

REPLY

I wonder what `channel_shift_range` is about. The doc says “shift range for each channels”, but what does this actually mean? Is it adding a random value to each channel or doing something else?

Jason Brownlee December 26, 2016 at 7:37 am #

REPLY

I have not used this one yet, sorry Lucas.

You could try experimenting with it or dive into the source to see what it’s all about.

Indra December 26, 2016 at 5:30 pm #

REPLY

Hi,

Thanks for the post. I've one question i.e., we do feature standardization in the training set, so while testing, we need those standardized values to apply on testing images ?

Jason Brownlee December 27, 2016 at 5:22 am #

REPLY

Yes Indra, any transforms like standardization performed on the data prior to modeling will also need to be performed on new data when testing or making predictions.

In the case of standardization, we need to keep track of means and standard deviations.

Dan March 11, 2017 at 11:01 pm #

REPLY

Thanks again Jason. Why do we subplot 330+1+i? Thanks

Jason Brownlee March 12, 2017 at 8:24 am #

REPLY

This is matplotlab syntax.

The 33 creates a grid of 3×3 images. The number after that (1-9) indicates the position in that grid to place the next image (left to right, top to bottom ordering).

I hope that helps.

Vineeth March 13, 2017 at 7:52 pm #

REPLY

How do I save the augmented images into a directory with a class label prefix or even better into a subdirectory of class name?

Jason Brownlee March 14, 2017 at 8:15 am #

REPLY

Great question Vineeth,

You can specify any directory and filename prefix you like in the call to flow()

Richa March 21, 2017 at 10:45 pm #

REPLY

can we augment data of a particular class. I mean images of a class which are less, to deal with the class imbalance problem.

Jason Brownlee March 22, 2017 at 8:06 am #

REPLY

Great idea.

Yes, but you may need to prepare the data for each class separately.

Lebron March 26, 2017 at 4:07 pm #

REPLY

Hi Jason,

Thanks for your post!

I have a question: Does this apply to image data with RGBXYZ for each pixel?

Each of my input image is of six channels including RGB and XYZ (world coordinate), which was acquired from an organized point cloud by PCL(Point Cloud Library). I am wondering whether there is a correct way to do data augmentation for my images.

I think ImageDataGenerator might be correct only for RGB images? Because when you shift/rotate/flip the RGB image, it means camera movement indeed, and the XYZ coordinates should be changed as well.

Thanks.

Jason Brownlee March 27, 2017 at 7:52 am #

REPLY

Hi Lebron, I believe this specific API is intended for 3d pixel data. You might be able to devise your own similar domain-specific transforms for you own data.

Lebron March 27, 2017 at 3:51 pm #

REPLY

Thanks Jason!

To confirm, do you mean image with RGB only by “3d pixel data”? And if I have more channels, I have to do all the augmentation by myself, rather than using Keras API?

Jason Brownlee March 28, 2017 at 8:21 am #

REPLY

Yes, I believe that to be the case, but I could be wrong.

Brian April 16, 2017 at 9:35 am #

REPLY

When I use zoom_range of 0.2 and inspect the output images, it seems to zoom h and v axes independently. However I want to have a small amount of zoom variation while preserving the aspect ratio of the images.

Also, when I specify a rotation_range, the rotated images have aliasing artefacts. Is there any way to specify rotations with antialiasing?

Jason Brownlee April 17, 2017 at 5:06 am #

REPLY

I'm not sure off hand.

Do you think these concerns will affect the skill of the model?

Brian April 19, 2017 at 11:15 am #

REPLY

Thanks Jason,

Aspect ratio of the image is important in a facial recognition setting. Antialiasing of rotated images I'm not so sure about, but as they are small images (244 x 244) it doesn't make sense to degrade them further.

I can modify my own copy of the Keras code to maintain the aspect ratio of the zoom and should be able to substitute PIL's rotation function, which does antialiasing, for the one used in Keras.

Keep up the good work, your writing has really helped me get up to speed with Keras quickly

Jason Brownlee April 20, 2017 at 9:21 am #

REPLY

Very nice Brian.

Let me know how you go.

Joe April 23, 2017 at 4:27 pm #

REPLY

Hi Brian.

The transformations in ImageGenerator are applied using

[scipy.ndimage.interpolation.affine_transform](https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.ndimage.interpolation.affine_transform.html), with “order” (the order of spline used for interpolation) set to zero.

Change this to one for linear interpolation or higher for higher orders.

Wuchi May 4, 2017 at 5:38 pm #

REPLY

Hi Jason,

Thank you for your post! Very clear!
I am trying to use ImageDataGenerator now. But if I want to apply feature standardization to unseen data in the future, I need to save the ImageDataGenerator to disk, right? Any suggestion to do it? Thanks a lot.

Jason Brownlee May 5, 2017 at 7:29 am #

REPLY

That is correct, or you can standardize manually and just save the coefficients used.

RogerLo May 17, 2017 at 2:23 pm #

REPLY

Hi Jason

I using Keras 2.x ‘tf’ seeting.
Why I can’t using
X_batch, y_batch = datagen.flow(train, train, batch_size=32)
For example :

from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator

```
# load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# reshape to be [samples][pixels][width][height]
X_train = X_train.reshape(X_train.shape[0], 28, 28,1)
X_test = X_test.reshape(X_test.shape[0], 28, 28,1)
# convert from int to float
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
# define data preparation
datagen = ImageDataGenerator(featurewise_center=True,
featurewise_std_normalization=True)
# fit parameters from data
datagen.fit(X_train)
# configure batch size and retrieve one batch of images
X_batch, y_batch = datagen.flow(X_train, y_train, batch_size=9)
```

Can you tell me why?
Thanks!

Jason Brownlee May 18, 2017 at 8:28 am #

REPLY

What error do you get exactly?

RogerLo May 19, 2017 at 4:26 pm #

REPLY

Hi, Hason

The error message is :
too many values to unpack (expected 2)

Jason Brownlee May 20, 2017 at 5:35 am #

REPLY

I'm sorry I have not seen this error before, I do not have any good suggestions.

N1k31t4 November 24, 2017 at 7:53 am #

REPLY

load data
(X_train, y_train), (X_test, y_test) = mnist.load_data()

I think this should be done without the brackets around the train and test pairs:
load data
X_train, y_train, X_test, y_test = mnist.load_data()

It is returning four things, but you are only accepting the two tuples.

Fahad June 16, 2017 at 12:25 am #

REPLY

Hi Jason,

I have training data of the shape (2000,4,100,100) which means 2000 samples of the size 100×100 with 4 channels and dtype= uint8, stored as '.npy' file. Can I use Image Augmentation technique on such data?

Jason Brownlee June 16, 2017 at 8:03 am #

REPLY

You may, try it and see.

Umberto June 23, 2017 at 7:00 pm #

REPLY

Hi Jason,
Since I used the fit_generator method instead of fit(), I need to use evaluate_generator in order to correctly evaluate the model or not? Is the same for predict_generator? I'm a little confused.

Matthew Hancock June 25, 2017 at 1:03 am #

REPLY

Hi Jason,

I have a quick question about the image augmentation. I am attempting to greatly increase the size of my training data set using data augmentation in order to increase my segmentation accuracy. Does the image generator feed multiple augmentations of the same image to the model or does it just return a single augmented version instead of the original? There seems to be no way to modify the number of augmented images the Image Data Generator actually returns.

Matthew Hancock June 25, 2017 at 1:06 am #

REPLY

Never mind, I found my answer in the Keras documentation.

Jason Brownlee June 25, 2017 at 6:03 am #

REPLY

Glad to hear it.

Jason Brownlee June 25, 2017 at 6:02 am #

REPLY

Great question.

From the doc: “The data will be looped over (in batches) indefinitely.”
https://keras.io/preprocessing/image/

Bojan March 24, 2018 at 8:48 pm #

REPLY

Also, if for example I have multiple options set in the data augmentation generator. Will it create a lot of different combinations of the data? For example:
– original data;
– shifted data;
– rotated data;
– noisy data;
– shifted + rotated data;
– shifted + noisy data;
– shifted + rotated + noisy data, etc.

Or will it only create one set of all the transforms created together, i.e:
– shifted + rotated + noisy data only;

If it is the latter, do you have any advice as to how should we combine different output results? Append them maybe in a list or something?

All the best,
a very good tutorial

Jason Brownlee March 25, 2018 at 6:28 am #

REPLY

It applies all of the specified transforms in creating the augmented data.

Alice July 4, 2017 at 7:30 pm #

REPLY

Hello Jason,
I made the exercice with your book which I find just great!!!
The problem is: it applies on randomly choosen images instead of doing it on the same ones from the “Point of comparison” sub-chapter. And always different samples.
How could I solve this?
I must say I don’t understand how it comes the “i” applies on the pyplot.subplot and on the X-batch[].
Thank you!!
Alice

Jason Brownlee July 6, 2017 at 10:14 am #

REPLY

Think of the augmented images as randomly modified versions of your training dataset. You have a new dataset with lots of variations of the data you provided. You do not need to tie them back to the original examples.

Or perhaps I misunderstand your question?

Nathan July 25, 2017 at 6:39 pm #

REPLY

I think the problem of Alice is the same as mine, the data that are plotted after each modification are never the same, which is difficult to make a comparison because they change everytime.

For example :
-the first plot gives me : 5 6 3, 0 1 9, 2 3 1
– after the ZCA whitening i have : 2 3 8, 3 2 5, 0 1 7

Jason Brownlee July 26, 2017 at 7:48 am #

REPLY

Yes, by design, the augmentation will create different augmented

versions of the images each time it is called.

This is what we want, so the model does a better job of generalizing.

What is the problem exactly, could you help me to understand please?

Antoine Simon July 25, 2017 at 10:59 pm #

REPLY

Hello Jason,

I have the same problem as Alice. I think that what she was saying was that the pictures that she plot after random modifications are never the same

It looks like the 9 pictures that are plotted are chosen randomly everytime.

It would be nice if you could answer me on this problem,

Thank you !

Jason Brownlee July 26, 2017 at 7:54 am #

REPLY

Yes, this is by design. This is exactly what we want from image augmentation.

john Landler July 25, 2017 at 5:45 am #

REPLY

Hi,

When I run the above script, I get this error:
Using TensorFlow backend.
C:\Users\sacheu\AppData\Local\Programs\Python\Python35\lib\site-packages\keras\preprocessing\image.py:653: UserWarning: Expected input to be images (as Numpy array) following the data format convention “channels_first” (channels on axis 1),

i.e. expected either 1, 3 or 4 channels on axis 1. However, it was passed an array with shape (60000, 1, 28, 28) (1 channels).
‘ (‘ + str(x.shape[self.channel_axis]) + ‘ channels).’)

can you please tell me how to fix it?
i think i have the latest version of the libraries. And I am using python 3.5.

Thank you.

Jason Brownlee July 25, 2017 at 9:49 am #

REPLY

You could try changing the order of the channels in code or in the Keras configuration file.

For example, in code:

```
from keras import backend as K
K.set_image_dim_ordering('th')
```

Or if this is the cause, comment it out.

Tom November 11, 2017 at 2:10 am #

REPLY

I have the same problem. Nothing works.

The message is a warning, and I still get the output images, but i.e. Feature Standardization is black and white, not gray scaled. So I suppose it is not working?

When I try to comment it out or change the order from ‘th’ to ‘tf’ – it completely brakes. The message is: … (28 channels)

I am new and any comments are welcome.

Jason Brownlee November 11, 2017 at 9:23 am #

REPLY

Perhaps double check you have the latest version of Keras installed?
2.0.8 or 2.0.9?

Tom November 13, 2017 at 9:04 pm #

(C:\ProgramData\Anaconda3)
C:\ProgramData\Anaconda3\etc\conda\activate.d>set
“KERAS_BACKEND=theano”

(C:\ProgramData\Anaconda3) C:\Users\Tom>conda install -c conda-forge
keras Fetching package metadata ……
Solving package specifications: .

All requested packages already installed.
packages in environment at C:\ProgramData\Anaconda3:
#
keras 2.0.6 py36_0 conda-forge

It looks like it is up to date. But..
I used this tutorial: https://machinelearningmastery.com/setup-python-environment-machine-learning-deep-learning-anaconda/ and typing:

import keras
print(‘keras: %s’ % keras.__version__)

gives me:
Using Theano backend.
keras: 2.0.6

but according to tutorial it should be:
Using TensorFlow backend.
keras: 1.2.1

Jason Brownlee November 14, 2017 at 10:09 am #

You can change the backend used by Keras in the ~/.keras/keras.json configuration file.

Tom November 13, 2017 at 9:18 pm #

Ok, at the moment I can say that:

– Those warning messages are just a warning. They are in each example. I do not know how to make them disappear, but it turns out that they are harmless.

– Example Feature Standardization is black and white, not gray scaled. Maybe there is an error in script? I am to novice to spot it.

– Other examples seems to work correctly despite the warning message. (So I just saw a warning, saw different output, spotted comment with similar problem and just stopped. My bad).

– I can write set “KERAS_BACKEND=tensorflow” to change the backend (I don’t know what does it means, but never mind 🙂).

– You are awesome for making those tutorials. Thank You!

Jason Brownlee November 14, 2017 at 10:12 am #

Thanks.

Xiaojie Zhang July 28, 2017 at 2:45 pm #

REPLY

Hi, thanks for your share. When I try to use zca-whitening and feature-wised centering on bigger data, I found it’s very very hard to get enough memory to do the fit() function. As the data-set has about 10000 pictures and 224*224 pixels, even generate a

flow iterator will use full of my 16GB memory. When try to use fit() for zca-whitening,centering,normalization which the documents said have to use the fit() function, I never success. Will you give some advice for data preparation for bigger data? Thank you very much!

Jason Brownlee July 29, 2017 at 8:04 am #

REPLY

Are you able to use the flow_from_directory instead of loading it all into memory?
https://keras.io/preprocessing/image/

Willie Maddox December 3, 2017 at 2:44 am #

REPLY

I have the same exact problem with MS-COCO and NUS-WIDE datasets and I have 128GB memory. The flow_from_directory() requires your data to be in a specific directory structure.

From the Keras v2.1.2 documentation…

directory: path to the target directory. It should contain one subdirectory per class.

MS-COCO and NUS-WIDE are not structured this way natively.

Also, Xiaojie was talking specifically about the fit() function, which is called before the flow (or flow_from_directory) function.

Muneer Ahmad Dedmari August 20, 2017 at 10:02 pm #

REPLY

Hi Jason,
Thanks for this nice post. I have a quick question. I have large-dataset which I am loading to model using custom data-generator. I am using it in model.fit_generator(). Now I want to use data-augmentation. So my question is, how/where can I use keras ImageDataGenerator?

Thank you very much.

Jason Brownlee August 21, 2017 at 6:06 am #

REPLY

I believe this tutorial will help:
https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

Steve September 27, 2017 at 11:31 am #

REPLY

Hi

Shall we run both fit_transform with origin images set and with augmented ones separately ? Or shell we combine them into one. On the latter one how do we combine them ?

Thanks.
Steve

Jason Brownlee September 27, 2017 at 3:49 pm #

REPLY

Just the augmented images.

Steve September 28, 2017 at 1:54 am #

REPLY

Thanks Jason. Bought all of your ML books, love it ! Would you write one about Transfer Learning deep enough with ImageNet and few other so that we can re-use the pre-trained one for our own purpose ?

Thanks.

Jason Brownlee September 28, 2017 at 5:27 am #

REPLY

Thanks again for your support Steve.

Yes, I have a post scheduled on re-using the VGG model. It should appear on the blog soon.

Momo October 24, 2017 at 2:13 am #

REPLY

Thanks for this great post !
In the Random Shifts part, can we have control on the file names ?
Is it possible to save files as:
'aug'+original_file_name+'png' ?

Thanks.

Jason Brownlee October 24, 2017 at 5:36 am #

REPLY

Yes, you can control the filenames. Perhaps the API will make it clearer:
https://keras.io/preprocessing/image/

Abraham Ben November 3, 2017 at 5:12 pm #

REPLY

thanks for your tutorial. When I try to use :

pred = model.predict_generator(data_gen.flow_from_directory("../input/valid_img", target_size=(input_size, input_size)))

I cannot get the image filename that correspond to the predicted probability. Is there any solutions?

Jason Brownlee November 4, 2017 at 5:27 am #

REPLY

The order of predictions will match the order of files in the directory I would expect.

Abraham Ben November 4, 2017 at 3:01 pm #

REPLY

I have check it, however, this is not the case since I got a pretty low accuracy compared to the val_acc. I found someone solving it by setting the batch_size to 1 when use predict_generator and ImageDataGenerator.flow_from_directory, but this is not what I want.

Masun November 21, 2017 at 10:42 am #

REPLY

Hi Jason,

Do you have any idea about how we can apply the same idea on signals? I mean signal augmentation for Deep learning? Thank you

Jason Brownlee November 22, 2017 at 10:47 am #

REPLY

Not off hand. Consider how transforms can be used on your specific data to create new patterns. E.g. even just adding random noise is a good start.

smriti December 4, 2017 at 4:57 pm #

REPLY

Once the features have been centered, using featurewise_center=True in

keras.preprocessing.image.ImageDataGenerator(), How can I retrieve that statistics so that I use it to preprocess the images to be used for prediction/testing during evaluate_generator() etc

Jason Brownlee December 4, 2017 at 4:59 pm #

REPLY

Good question, I'm not sure off the cuff, perhaps post to Keras group or slack channel:
https://machinelearningmastery.com/get-help-with-keras/

safae January 4, 2018 at 12:50 am #

REPLY

Hi Jason,

Thank you very much for all the posts you shared, which are a very useful and help a lot.

I would like to ask if do you have an idea about implemented data augmentation algorithms for time series data(such as acceleration, AC voltage, …)

Thank you

Jason Brownlee January 4, 2018 at 8:12 am #

REPLY

Not at this stage, thanks for the suggestion.

Dimitris Mallios January 30, 2018 at 3:28 am #

REPLY

Hi Jason, i have a question, suppose we have semantic segmentation task and we want to rotate and flip both the image and the output image labels, how do we apply the transformations? Should i concatenate the image and the labels to a homogeneous array

and then apply the appropriate transformations?

Jason Brownlee January 30, 2018 at 9:55 am #

REPLY

Good question, you might want more control and apply augmentation one pair at a time.

Wafa March 10, 2018 at 10:38 am #

REPLY

Hi Jason,

Thank you for your nice post! How can I use augmentation to data that I have on my disk? I see you imported mnist and I could not realize how to change this for my purpose.

Jason Brownlee March 11, 2018 at 6:18 am #

REPLY

You can load the data via the augmentation API and use it to make an augmented version of your dataset. This augmentation could be used directly or saved to file.

Steven March 22, 2018 at 12:27 pm #

REPLY

hi Jason,

Quick question: after the images are augmented, your script saves them into smaller files. What if I want to stitch these augmented files into a single file, similar to the original mnist file so that I can use nmist.load_data() function to load them into, say a CNN engine? Would the following work?

```
nmist_new = []
for X_batch, y_batch in datagen.flow(X_train, y_train, batch_size=9):
# create a grid of 3×3 images
for i in range(0, 9):
nmist_new.append(array(X_batch, y_batch))
```

I am not sure what format should be used.

Jason Brownlee March 22, 2018 at 2:52 pm #

REPLY

Good question, I think some experimentation would be required.

Perhaps use of numpy's hstack and vstack to create a larger array from the image arrays?

Steven March 23, 2018 at 5:05 am #

REPLY

Yeah, I think need some tweaks or look into Keras' mnist_load() function how what the data format is when writing back. This link has the original data format:

http://yann.lecun.com/exdb/mnist/

Scroll down to the bottom for training and test dataset structures. They used some header information for each image. I think hstack and vstack are the way to go, but need to take care of those headers (think should be easy). I am still not sure what those "xxxx" mean. I think the files are just filled by the 28×28 small image data.

Jason Brownlee March 23, 2018 at 6:14 am #

REPLY

It might be easier to write your own progressive loading function. A heck of a lot simpler actually.

Tien April 8, 2018 at 2:30 pm #

REPLY

Dear Jason,

I used the code to generator featurewise standardized samples, but did not get the same result as shown in the web.

Tien

Jason Brownlee April 9, 2018 at 6:04 am #

REPLY

How was it different?

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT