Data Input and Augmentation using Keras for Object Classification

Matthew Kutugata

Problem:

- Complex Nilgai cattle-tick eradication project
- Trail cam data
- Primed for computer vision solution

Object Classification

cow

nilgai

pig

3 classes

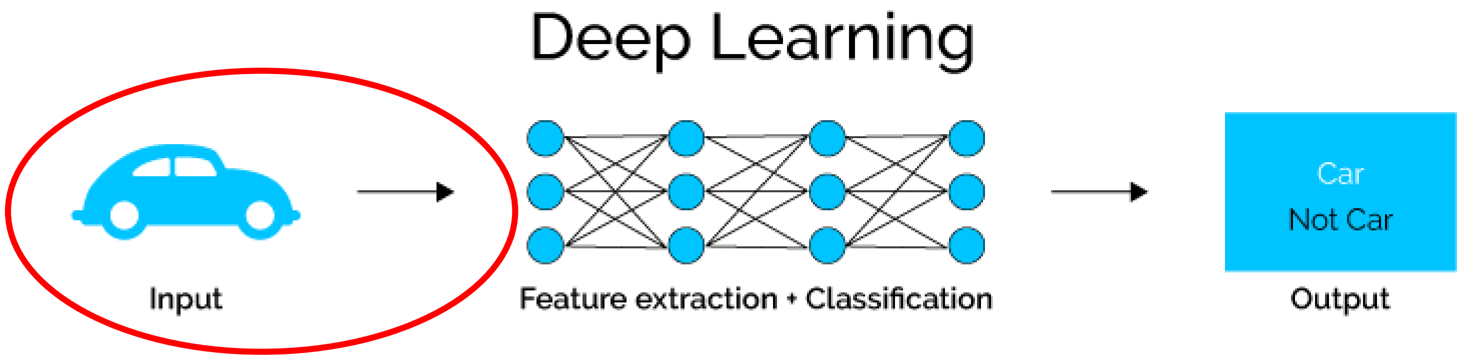
Keras and Tensorflow 2 as a “backend”

- Keras is an open-sourced neural-network library written in Python
- Provides high-level building blocks for deep learning models
- Keras does not handle low-level operations
- It relies on a specialized tensor manipulation library to do so, serving as the "backend engine" of Keras
- Allows developers to focus on high-level development

```
[1]: # example of normalizing a image dataset
     #import tensorflow as tf
     from keras.datasets import mnist
     from keras.preprocessing.image import ImageDataGenerator

     Using TensorFlow backend.
```

Why is "**Data input**" a thing?

- Unstructured data (images) and labels
- 100 – 100,000 example dataset difficult to organize
- Automatic generation of train, validation, and test sets
- Progressive loading of image data that may not fit in system memory



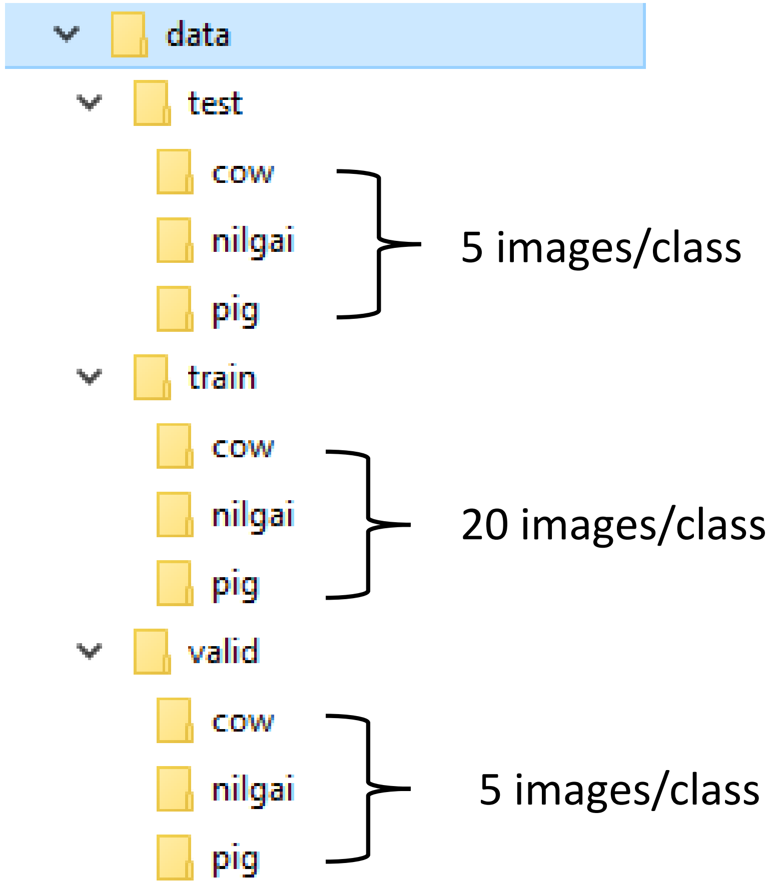
Organizing → Loading → Preprocessing

Organize train, validation, and test image datasets

Training set: examples used for learning - to fit the parameters of the classifier

Validation set: examples used to finetune the parameters of a classifier

Test set: examples used only to assess the performance of a fully-trained classifier (5 images)



— Brian Ripley, page 354, [Pattern Recognition and Neural Networks](), 1996

Loading data using `ImageDataGenerator`

```python
# Import dependencies and TF for good measure
import tensorflow as tf
from keras.datasets import mnist
from keras.preprocessing.image import ImageDataGenerator

#check tf version
tf.__version__
```

```
'2.1.0-dev20191126'
```

- data
  - test
    - cow
    - nilgai
    - pig
  - train
    - cow
    - nilgai
    - pig
  - valid
    - cow
    - nilgai
    - pig

```python
train_path = 'C:\\Users\\ncu116\\Desktop\\data\\train'
val_path = 'C:\\Users\\ncu116\\Desktop\\data\\valid'
test_path =  'C:\\Users\\ncu116\\Desktop\\data\\test'
# create data generator
datagen = ImageDataGenerator()

# prepare an iterators for each dataset arg = data directory and
#class_mode = classification type - either multiclass 'categorical' or 'binary
train_it = datagen.flow_from_directory(train_path, class_mode='categorical')
val_it = datagen.flow_from_directory(val_path, class_mode='categorical')
test_it = datagen.flow_from_directory(test_path, class_mode='categorical')

# confirm iterator works
batchX, batchy = train_it.next()
print('Batch shape=%s, min=%.3f, max=%.3f' % (batchX.shape, batchX.min(), batchX.max()))
```

```
Found 60 images belonging to 3 classes.
Found 15 images belonging to 3 classes.
Found 15 images belonging to 4 classes.
Batch shape=(32, 256, 256, 3), min=0.000, max=255.000
```

`ImageDataGenerator()`

- Generate batches of tensor image data with on-the-fly data augmentation. The data will be looped over (in batches).

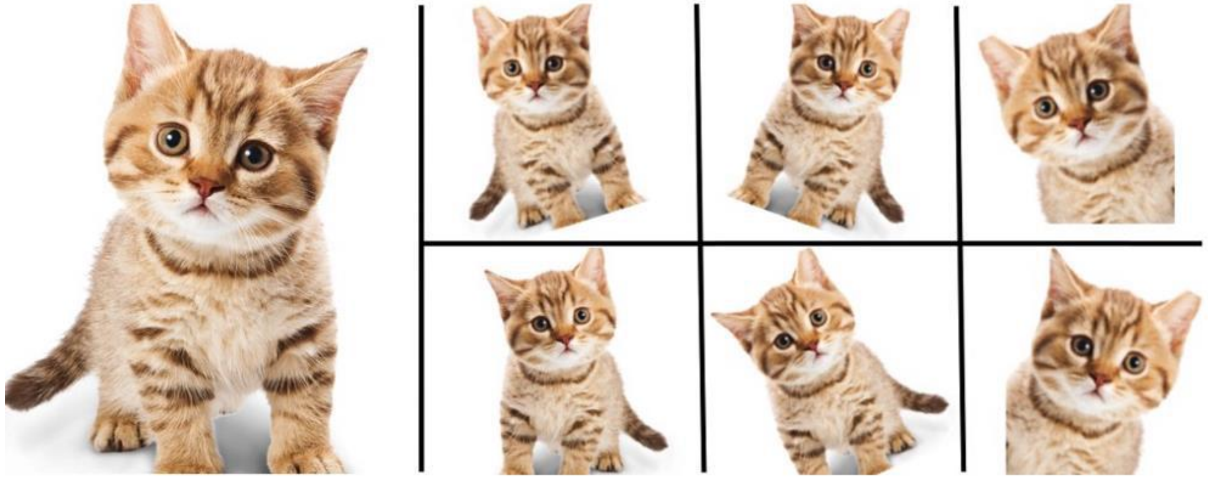
`flow_from_directory`

- Reads images from folders

What is and **Why** use Data Augmentation?

- Technique used to artificially expand size of training set
- Training on more data results in more accurate models
- Augmentation creates variations of images that can improve the ability of a model to generalize on new images
- Image augmentation in Keras is performed using

`ImageDataGenerator()` class



Data Augmentation Types

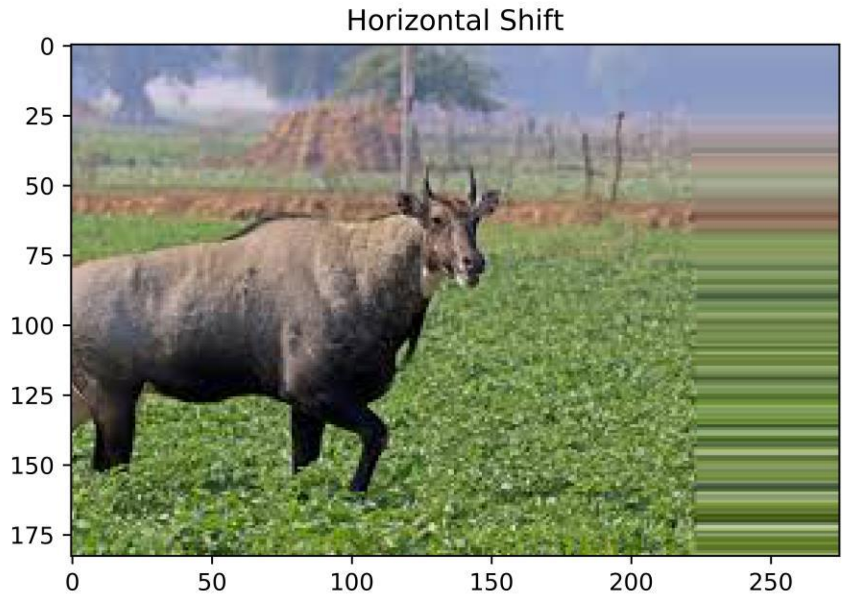
1. Horizontal/Vertical shift
2. Horizontal/Vertical flip
3. Random rotation
4. Random brightness
5. Random zoom

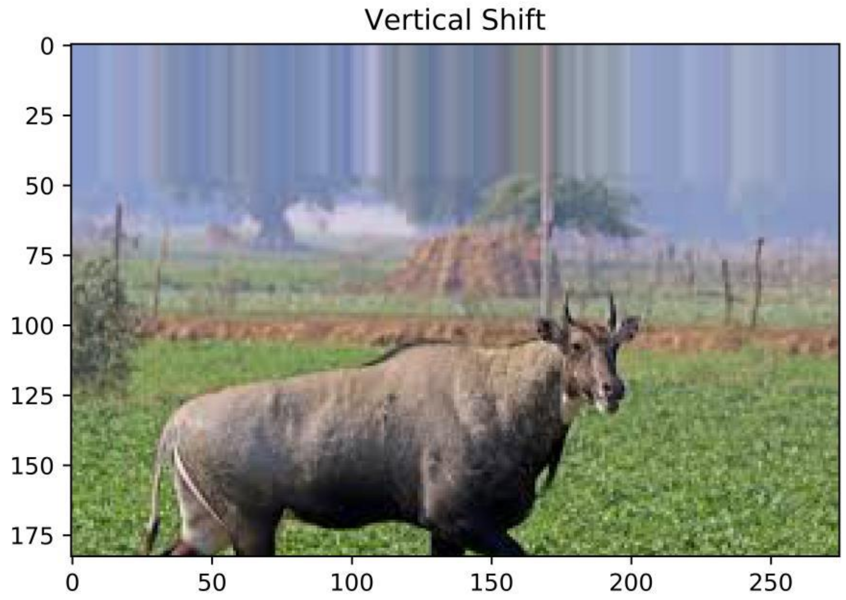
Data Augmentation Types

1. Horizontal/Vertical shift

```
datagen =
ImageDataGenerator(width_shift_range=[-50,50])
```

```python
# load the image
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai1.png'
#img = load_img('C:\\Users\\ncu116\\Desktop\\data\\chapter_09\\bird.jpg')
save_here = 'C:\\Users\\ncu116\\Desktop\\data\\test\\test_augmented'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(width_shift_range=[-50,50])

# prepare iterator and save new examples to file "save here"
it = datagen.flow(samples, batch_size=1 ) #additional "save" argument can be used
pyplot.figure(figsize=(12, 12))
for i in range(1):
    pyplot.subplot(330 + 1 + i)
    batch = it.next()
    image = batch[0].astype('uint8')
    #fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Horizontal Shift')


# show the figure

pyplot.show()
fig.savefig('hhhoz_shift.png', dpi=800)
```



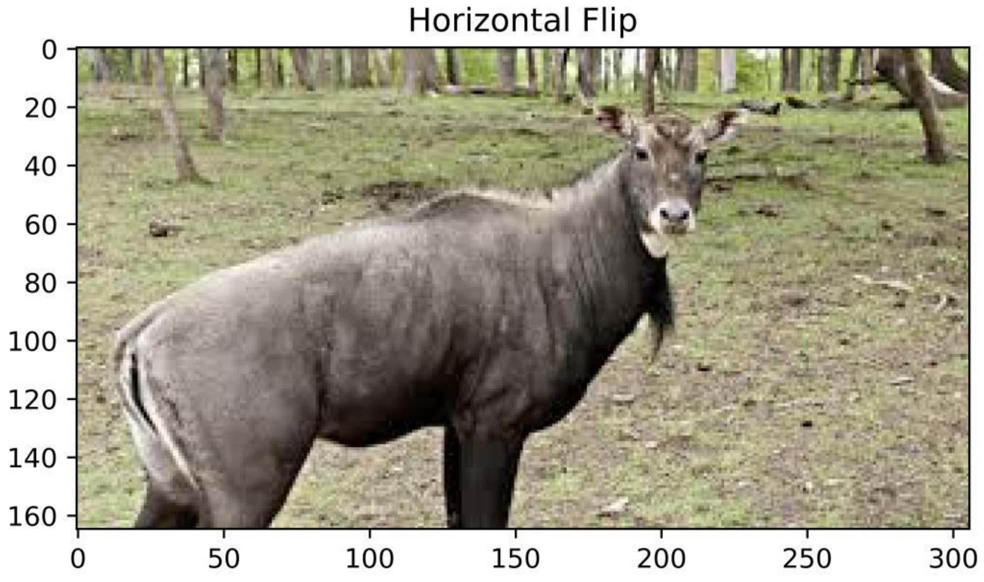


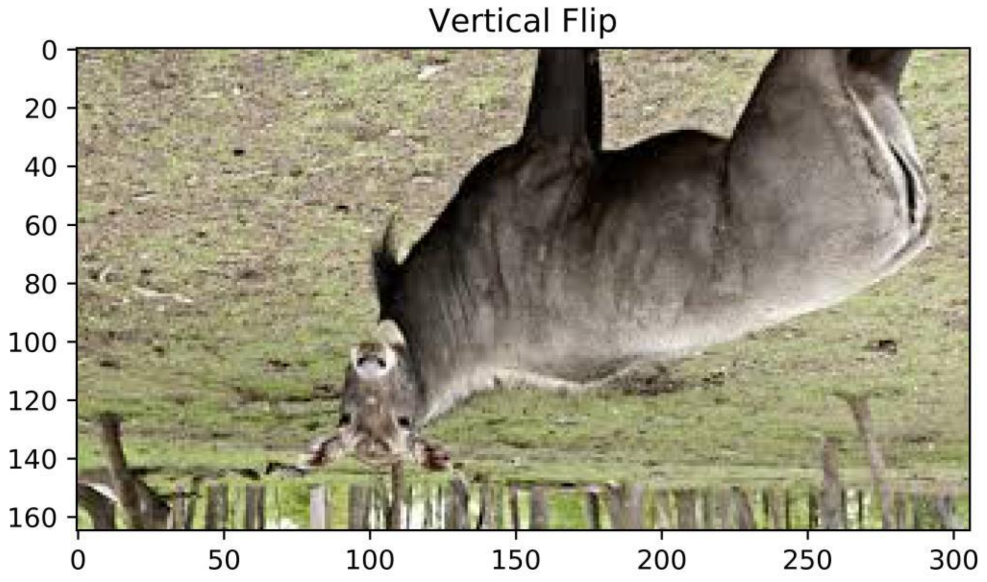
Data Augmentation Types

2. Horizontal/Vertical flip

```
datagen = ImageDataGenerator(vertical_flip=True)
```

```
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai2.png'
#img = load_img('C:\\Users\\ncu116\\Desktop\\data\\chapter_09\\bird.jpg')
#save_here = 'C:\\Users\\ncu116\\Desktop\\data\\test\\test_augmented'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(vertical_flip=True)
it = datagen.flow(samples, batch_size=1)
pyplot.figure(figsize=(19, 12))

#generate samples and plot
for i in range(1):
    #define subplot
    pyplot.subplot(330 +1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Vertical Flip')
    fig.savefig('vert_flip.png', dpi=800)
```



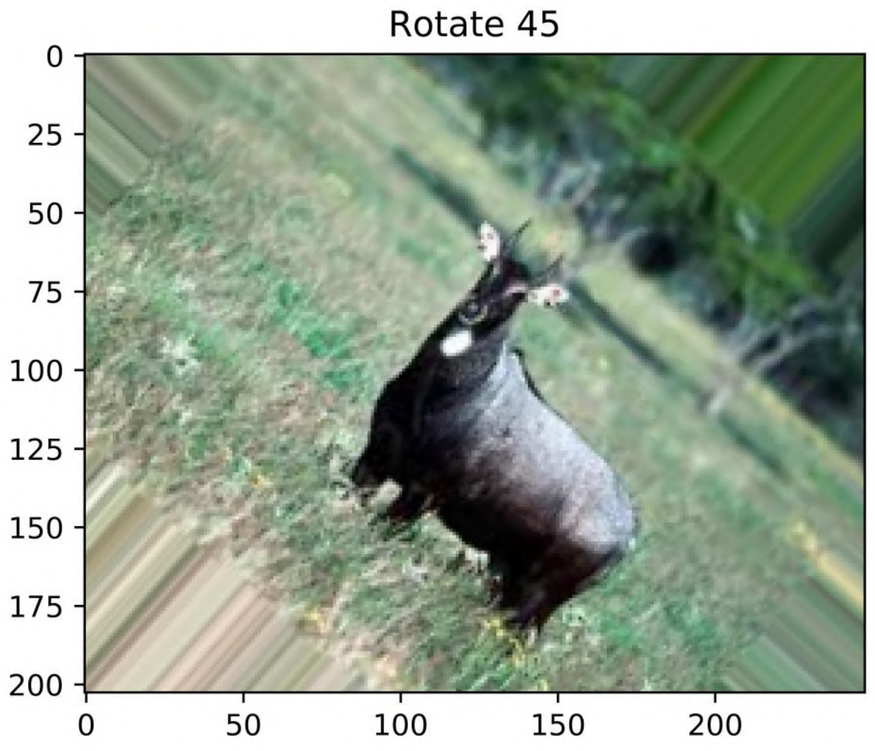


Data Augmentation Types

3. Random Rotation

```
datagen = ImageDataGenerator(rotation_range=45)
```

```
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai3.png'
#save_here = 'C:\\Users\\ncu116\\Desktop\\data\\test\\test_augmented'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(rotation_range=45)
it = datagen.flow(samples, batch_size=1)
pyplot.figure(figsize=(19, 12))

#generate samples and plot
for i in range(1):
    #define subplot
    pyplot.subplot(330 +1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Rotate 45')
    fig.savefig('Rotate_flip.png', dpi=800)
```

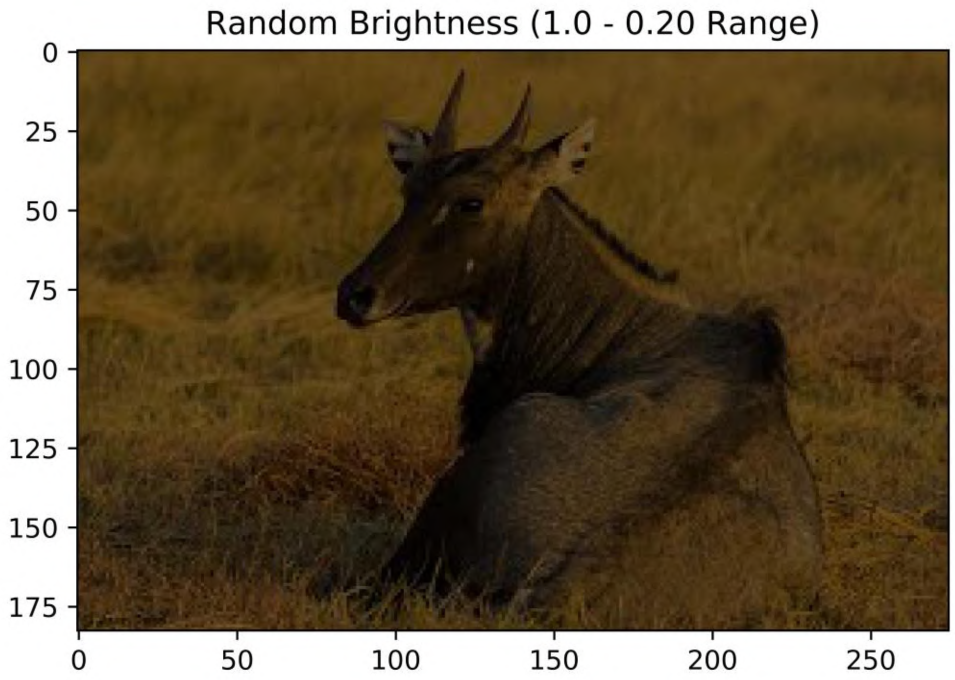


Data Augmentation Types

4. Random Brightness

```
datagen =
ImageDataGenerator(brightness_range[0.2, 1.0])
```

```python
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai4.png'
#save_here = 'C:\\Users\\ncu116\\Desktop\\data\\test\\test_augmented'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(brightness_range=[0.2, 1.0])
it = datagen.flow(samples, batch_size=1)
pyplot.figure(figsize=(19, 12))

#generate samples and plot
for i in range(1):
    #define subplot
    pyplot.subplot(330 +1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Random Brightness (1 - 20% Change)')
    fig.savefig('Rando_Bright.png', dpi=800)
```

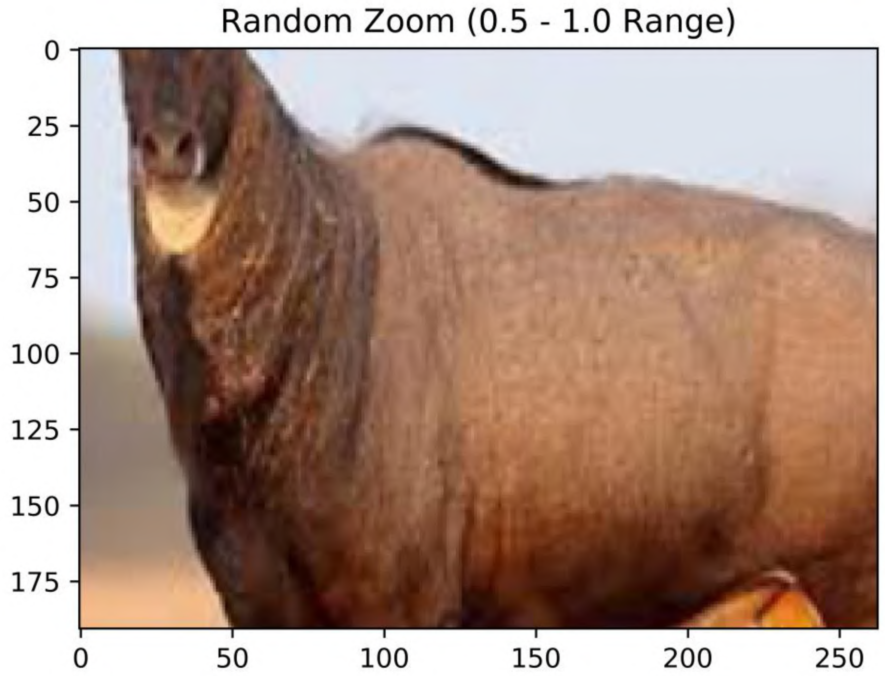


Data Augmentation Types

5. Random Zoom

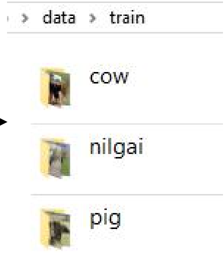
```
datagen =
ImageDataGenerator(zoom_range[0.5, 1.0])
```

```
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai5.png'
#save_here = 'C:\\Users\\ncu116\\Desktop\\data\\test\\test_augmented'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(zoom_range=[0.5, 1.0])
it = datagen.flow(samples, batch_size=1)
pyplot.figure(figsize=(19, 12))

#generate samples and plot
for i in range(1):
    #define subplot
    pyplot.subplot(330 +1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Random Zoom (0.5 - 1.0 Range)')
    fig.savefig('rando_zoom.png', dpi=800)
```



Results: Data augmentation

Save location
'train'



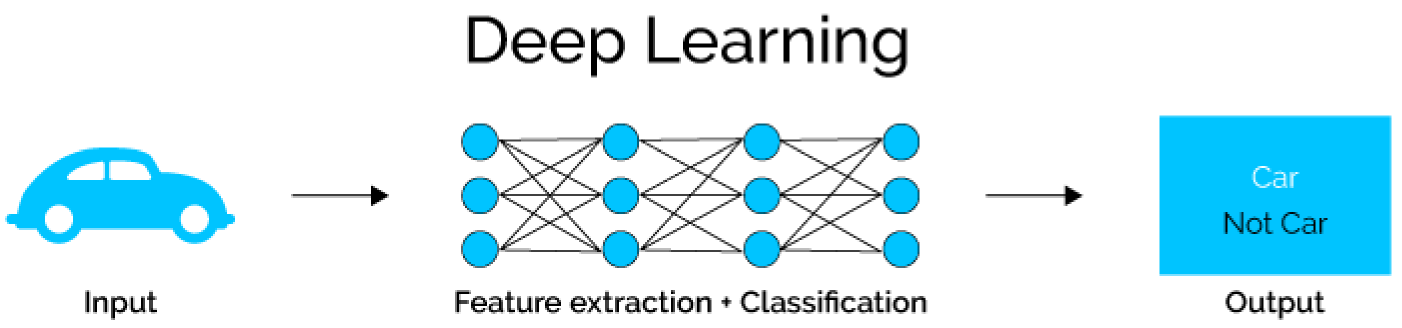

```python
single_nilgai_path = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai\\nilgai1.
#save location
save_here = 'C:\\Users\\ncu116\\Desktop\\data\\train\\nilgai'
img = load_img(single_nilgai_path)
# convert to numpy array
data = img_to_array(img)
# expand dimension to one smaple
samples = expand_dims(data, 0)
# create image data augmentation generator
datagen = ImageDataGenerator(zoom_range=[0.5, 1.0])
#batch flow form directory
it = datagen.flow(samples, batch_size=1,
                  save_to_dir=save_here,
                  save_prefix='aug',
                  save_format='png')

pyplot.figure(figsize=(19, 12))

#generate samples and plot
for i in range(1):
    #define subplot
    pyplot.subplot(330 +1 + i)
    # generate batch of images
    batch = it.next()
    # convert to unsigned integers for viewing
    image = batch[0].astype('uint8')
    fig = pyplot.figure()
    pyplot.imshow(image)
    pyplot.title('Random Zoom (0.5 - 1.0 Range)')
    fig.savefig('1rando_zoom.png', dpi=800)
```

'save' argument in `.flow` method

1 + 5 x 60 = **360** image training data saved to directory

Results and Next Step:

- Generate batches of tensor image data with real-time data augmentation. The data will be looped over (in batches)
- Image now tensor object iterable by model
- Can now be applied to train classifier



Questions?

What is a validation set?

– Training set: A set of examples used for learning, that is to fit the parameters of the classifier.

– Validation set: A set of examples used to tune the parameters of a classifier, for example to choose the number of hidden units in a neural network.

– Test set: A set of examples used only to assess the performance of a fully-specified classifier.

— Brian Ripley, page 354, Pattern Recognition and Neural Networks, 1996

What is a tensor?

- A **tensor** is a generalization of vectors and matrices to potentially higher dimensions

"Tensors are matrices of many dimensions"

— Emmanuel Caradec, Understanding Keras Tensors, 2018

- Deep learning usually involves hundreds, if not thousands, of dimensions and fields.

- Best represented by tensors since they can represent anything ranging from *zero* to *N* dimensions.

- Images can easily be broken down into a few hundred features. Therefore, tensors can be best viewed as containers that wrap and store data features in the context of machine learning and deep learning.

— CE Kan, Quick ML Concepts: Tensors, 2018