**Data Creation**

One of the most under rated things in deep learning is data creation. Data creation takes lots of time and there are certain tips which needs to be documented.

I was able to classify 6 distinct objects using ResNet-50. It took me 100 epochs of batch size 32. Since I was using laptop it took me around 36 hours to classify 730 images. I initially took only 110 images.

The following steps were taken:

* Labeling of Image Data
* Cropping
* Data Augmentation (Creation of more images by performing image transforms)
* Input Pipeline (Input tensor to feed in the network)
* Shuffling both Labels and Data tensors both in same sequence.

**Labeling all Images:**

We first need to separate all objects of same class in different folders and rename each image in class in sequence of natural numbers, to take benefit of iterations while creating input pipeline.

So each image might have an address like: Source/class\_ (number)/image\_ (number).jpg

**Cropping**

To create image data for deep learning, image must contain only one object, not group of objects, for that we must crop images. Using OpenCv throughout.

* Iterate all image in loop, load them one by one.
* Resize each image so that it can fir in screen (my choice 640, 480).
* Use cv2.SetMouseCallBack to draw rectangle around object. <https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_mouse_handling/py_mouse_handling.html>
* Use keys to show select and display selected ROI in different window and if you are satisfied by cropping choose another key to save in desired location.

**Data Augmentation:**

This is basically creating more data using cropping, shearing, rotating and color shifting. I found Image augmentation package ImgAug , it is good but you will need to understand the package rather than blindly copy code. The link:

<https://github.com/aleju/imgaug>

The second reliable source is using tensor flow. The functions are well described on tensor flow site: <https://www.tensorflow.org/api_docs/python/tf/image>

But if you are newbie and tensor flow makes little sense you can follow this blog spot which has clearly descripted how to use functions: <https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9>

**Input Pipeline:**

Input pipeline means Data and Labels in two different folders. First of all get address of all images in a list (python) using loop and do the same for labels. Use OpenCv to retrieve all images into numpy array:

data = [ ]

for images in data:

data.append(cv2.imread(image))

X\_train = np.asarray(data)

X\_train=X\_train/255

Do the same for labels store it into a Numpy array with two columns.

**Shuffling:**

Without shuffling I got accuracy of 42 percent and after shuffling I got accuracy of 93 percent. Shuffling matters. Here is how you shuffle two arrays in similar sequence.

**>>> import numpy as np**

**>>> x = np.arange(10)**

**>>> y = np.arange(9, -1, -1)**

**>>> x**

**array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])**

**>>> y**

**array([9, 8, 7, 6, 5, 4, 3, 2, 1, 0])**

**>>> s = np.arange(x.shape[0])**

**>>> np.random.shuffle(s)**

**>>> s**

**array([9, 3, 5, 2, 6, 0, 8, 1, 4, 7])**

**>>> x[s]**

**array([9, 3, 5, 2, 6, 0, 8, 1, 4, 7])**

**>>> y[s]**

**array([0, 6, 4, 7, 3, 9, 1, 8, 5, 2])**

Here ‘s’ is of common dimension to both data and labels so it’s taken as index. It is nothing but number image datasets.

Source : <https://play.pixelblaster.ro/blog/2017/01/20/how-to-shuffle-two-arrays-to-the-same-order/>

**Saving Model:**

It is described well how to save architecture and model weights at <https://jovianlin.io/saving-loading-keras-models/>