```python
import os
import numpy as np
import PIL.Image as Image
import torch
from torch.utils import data
from torchvision import transforms
import pdb
import random
import sys
import matplotlib.pyplot as plt


class ImageNetDetCls(data.Dataset):
    def __init__(self, root,
                 source_transform=None):
        super(ImageNetDetCls, self).__init__()
        self.root = root
        self.s_transform = source_transform
        txts = os.listdir(os.path.join(root, 'data', 'det_lists'))
        txts = filter(lambda x: x.startswith('train_pos') or x.startswith('train_part'), txts)
        file2lbl = {}
        for txt in txts:
            files = open(os.path.join(root, 'data', 'det_lists', txt)).readlines()
            for f in files:
                f = f.strip('\n')+'.JPEG'
                if f in file2lbl:
                    file2lbl[f] += [int(txt.split('.')[0].split('_')[-1])]
                else:
                    file2lbl[f] = [int(txt.split('.')[0].split('_')[-1])]
        self.file2lbl = file2lbl.items()

    def __len__(self):
        return len(self.file2lbl)

    def __getitem__(self, index):
        # load image
        img_file, lbl = self.file2lbl[index]
        img = Image.open(os.path.join(self.root, 'images', img_file)).convert('RGB')
        if self.s_transform is not None:
            img = self.s_transform(img)
        onehot = np.zeros(200)
        lbl = np.array(lbl)-1
        onehot[lbl] = 1
        onehot = torch.from_numpy(onehot).float()
        return img, onehot


if __name__ == "__main__":
    sb = ImageNetDetCls('../../data/datasets/ILSVRC2014_devkit')
    img, gt = sb.__getitem__(0)
    pdb.set_trace()
```