```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 ImageNetDetClsDemo(data.Dataset):
    def __init__(self, root,
                 source_transform=None):
        super(ImageNetDetClsDemo, 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()
        self.index2name = open('datasets/index2name.txt', 'r').readlines()
        self.index2name = [s.strip('\n') for s in self.index2name]

    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
        cls_names = [self.index2name[i] for i in lbl]
        cls_names = '_'.join(cls_names)
        onehot[lbl] = 1
        onehot = torch.from_numpy(onehot).float()
        return img, onehot, cls_names


def caption_collate_fn(data):
    """Creates mini-batch tensors from the list of tuples (image, caption).

    We should build custom collate_fn rather than using default collate_fn,
    because merging caption (including padding) is not supported in default.

    Args:
        data: list of tuple (image, caption).
            - image: torch tensor of shape (3, 256, 256).
            - caption: torch tensor of shape (?); variable length.

    Returns:
        images: torch tensor of shape (batch_size, 3, 256, 256).
        targets: torch tensor of shape (batch_size, padded_length).
        lengths: list; valid length for each padded caption.
    """
    # Sort a data list by caption length (descending order).
    images, labels, cls_names = zip(*data)

    # Merge images (from tuple of 3D tensor to 4D tensor).
    images = torch.stack(images, 0)
    labels = torch.stack(labels, 0)
    return images, labels, cls_names


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

```