```
import sys
import glob

import imageio
import numpy as np

class Core50Dataset:
    """Handler for Core50 dataset used in this class"""

    # Data path
    if "-s" in sys.argv:
        data_path = "/home/fagg/ml_datasets/core50/core50_128x128/"
    else:
        data_path = "/Users/willspaeth/datasets/core50/core50_128x128/"

    # Declare the various partitions
    s_inds = [1, 2, 3, 4, 5, 7, 8, 9, 10, 11]
    o_scissors = [11, 12, 13, 14, 15]
    o_mugs = [41, 42, 43, 44, 45]
    o_train = [2, 3, 4, 5]
    o_val = [1]

    def __init__(self, exp_cfg=None):
        pass

    def load_data(self):
        """Load data dictionary for training"""

        # Compile data dictionary
        data_dict = {
            "train": {"ins": [], "outs": []},
            "val": {"ins": [], "outs": []}
        }

        # Loop through images and place into appropriate partitions
        image_count = 0
        for s in self.s_inds:
            for o in self.o_scissors+self.o_mugs:
                images = self.load_images(s, o)
                image_count += len(images)

                # Scissors are positive, mugs are negative
                if o in self.o_scissors:
                    label_list = [[1, 0] for i in range(len(images))]
                else:
                    label_list = [[0, 1] for i in range(len(images))]

                # Check if the o is in the train split or val split
                if (o%10) in self.o_train:
                    data_dict["train"]["ins"] = data_dict["train"]["ins"] + images
                    data_dict["train"]["outs"] = data_dict["train"]["outs"] + label_list
                else:
                    data_dict["val"]["ins"] = data_dict["val"]["ins"] + images
                    data_dict["val"]["outs"] = data_dict["val"]["outs"] + label_list

        print("Final image count: {}".format(image_count))

        # Convert data dictionary entries into numpy arrays instead of lists
        data_dict["train"]["ins"] = np.stack(data_dict["train"]["ins"], axis=0)
        data_dict["train"]["outs"] = np.stack(data_dict["train"]["outs"], axis=0)
        data_dict["val"]["ins"] = np.stack(data_dict["val"]["ins"], axis=0)
        data_dict["val"]["outs"] = np.stack(data_dict["val"]["outs"], axis=0)

        return data_dict

    def load_images(self, s, o):
        """Load multiple images based on their s and o conditions"""
        images = []
```

```
        filenames = self._get_matching_filenames(s, o)
        for filename in filenames:
            images.append(imageio.imread(filename).astype(float))


        return images

    def get_input_size(self):
        """Gets the dimensionality of one sample"""

        return (128, 128, 3)

    def _get_matching_filenames(self, s, o):
        """Get matching filenames based on s and o conditions"""

        file_pattern = "{}s{}/o{}/C_{:02d}_{}_*0.png".format(self.data_path, s, o, s, o)
        return glob.glob(file_pattern)
```