```
import struct as st
import numpy as np
import pickle as pkl


class Parser(object):
    def __init__(self, dataset, type):
        self.type = type  # dataset type
        self.dataset = dataset  # dataset

    def parse(self):
        self.images_set = []
        self.labels_set = []
        if (
            self.type == "MNIST"
        ):  # dataset follows a dictionary construct based on images and labels
            for k, v in self.dataset.items():
                if "images" in k:
                    with open(v, "rb") as f:
                        magic, num_imgs, num_rows, num_cols = st.unpack(
                            ">IIII", f.read(16)
                        )
                        self.images_set.append(
                            np.fromfile(f, dtype=np.dtype(np.ubyte))
                            .newbyteorder(">")
                            .reshape(num_imgs, num_rows, num_cols, 1)
                        )
                elif "labels" in k:
                    with open(v, "rb") as f:
                        magic, num_items = st.unpack(">II", f.read(8))
                        self.labels_set.append(
                            np.fromfile(f, dtype=np.dtype(np.uint8)).newbyteorde
r(">")
                        )
        elif "CIFAR" in self.type:
            # dataset follows a dictionary construct based on training and test
            # https://mattpetersen.github.io/load-cifar10-with-numpy
            labels = b"fine_labels" if ("100" in self.type) else b"labels"
            for k, v in self.dataset.items():
                batch_dict = self.__unpickle(self.dataset[k])
                self.images_set.append(
                    np.transpose(
                        batch_dict[b"data"].reshape(
                            len(batch_dict[b"data"]), 3, 32, 32
                        ),
                        (0, 2, 3, 1),
                    )
                    / 255.0
                )
                self.labels_set.append(batch_dict[labels])

        return (
            (
                np.concatenate([*self.images_set[:-1]]),
                np.array(self.labels_set[:-1]).flatten(),
            ),
            (self.images_set[-1], np.array(self.labels_set[-1]).flatten()),
        )

    # https://www.cs.toronto.edu/~kriz/cifar.html
    def __unpickle(self, file):
        with open(file, "rb") as fo:
```

```
            dict = pkl.load(fo, encoding="bytes")
        return dict
```