"""Module containing wrapper class to allow numpy arrays to work with twml functions"""

import ctypes as ct

from absl import logging

from libtwml import CLIB

import numpy as np

\_NP\_TO\_TWML\_TYPE = {

'float32': ct.c\_int(1),

'float64': ct.c\_int(2),

'int32': ct.c\_int(3),

'int64': ct.c\_int(4),

'int8': ct.c\_int(5),

'uint8': ct.c\_int(6),

}

class Array(object):

"""

Wrapper class to allow numpy arrays to work with twml functions.

"""

def \_\_init\_\_(self, array):

"""

Wraps numpy array and creates a handle that can be passed to C functions from libtwml.

array: Numpy array

"""

if not isinstance(array, np.ndarray):

raise TypeError("Input must be a numpy array")

try:

ttype = \_NP\_TO\_TWML\_TYPE[array.dtype.name]

except KeyError as err:

logging.error("Unsupported numpy type")

raise err

handle = ct.c\_void\_p(0)

ndim = ct.c\_int(array.ndim)

dims = array.ctypes.get\_shape()

isize = array.dtype.itemsize

strides\_t = ct.c\_size\_t \* array.ndim

strides = strides\_t(\*[n // isize for n in array.strides])

err = CLIB.twml\_tensor\_create(ct.pointer(handle),

array.ctypes.get\_as\_parameter(),

ndim, dims, strides, ttype)

if err != 1000:

raise RuntimeError("Error from libtwml")

# Store the numpy array to ensure it isn't deleted before self

self.\_array = array

self.\_handle = handle

self.\_type = ttype

@property

def handle(self):

"""

Return the twml handle

"""

return self.\_handle

@property

def shape(self):

"""

Return the shape

"""

return self.\_array.shape

@property

def ndim(self):

"""

Return the shape

"""

return self.\_array.ndim

@property

def array(self):

"""

Return the numpy array

"""

return self.\_array

@property

def dtype(self):

"""

Return numpy dtype

"""

return self.\_array.dtype

def \_\_del\_\_(self):

"""

Delete the handle

"""

CLIB.twml\_tensor\_delete(self.\_handle)