use npyz::WriterBuilder;

use npyz::{AutoSerialize, WriteOptions};

use std::io::BufWriter;

use std::{

fs::File,

io::{self, BufRead},

};

pub fn load\_batch\_prediction\_request\_base64(file\_name: &str) -> Vec<Vec<u8>> {

let file = File::open(file\_name).expect("could not read file");

let mut result = vec![];

for (mut line\_count, line) in io::BufReader::new(file).lines().enumerate() {

line\_count += 1;

match base64::decode(line.unwrap().trim()) {

Ok(payload) => result.push(payload),

Err(err) => println!("error decoding line {file\_name}:{line\_count} - {err}"),

}

}

println!("result len: {}", result.len());

result

}

pub fn save\_to\_npy<T: npyz::Serialize + AutoSerialize>(data: &[T], save\_to: String) {

let mut writer = WriteOptions::new()

.default\_dtype()

.shape(&[data.len() as u64, 1])

.writer(BufWriter::new(File::create(save\_to).unwrap()))

.begin\_nd()

.unwrap();

writer.extend(data.to\_owned()).unwrap();

writer.finish().unwrap();

}