CIDeR-ML
General meeting

21 November 2023

Ryo Matsumoto (TokyoTech)

Homework at the last meeting in Kashiwa

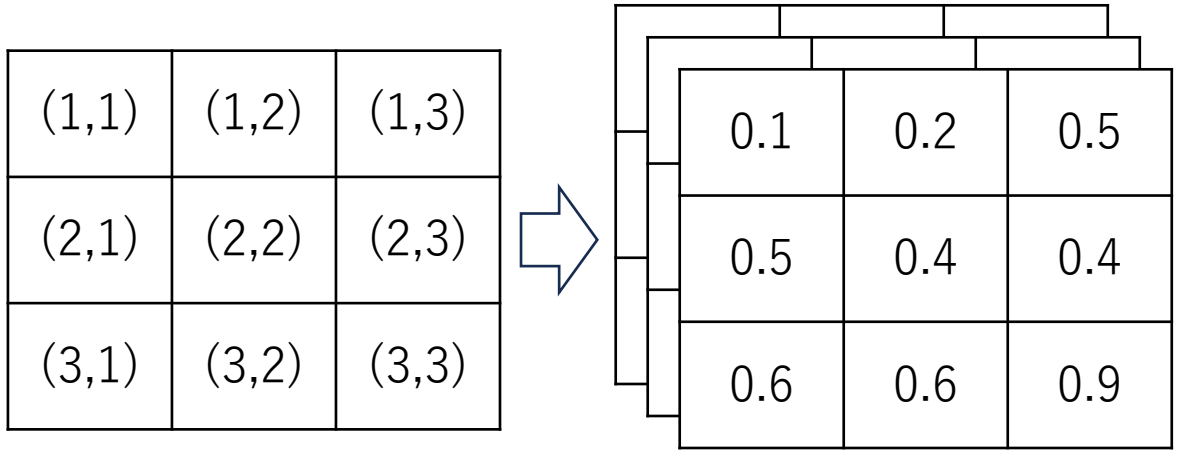
- We worked on the example of SIREN at the last meeting

(1,1)	(1,2)	(1,3)
(2,1)	(2,2)	(2,3)
(3,1)	(3,2)	(3,3)

0.1	0.2	0.5
0.5	0.4	0.4
0.6	0.6	0.9

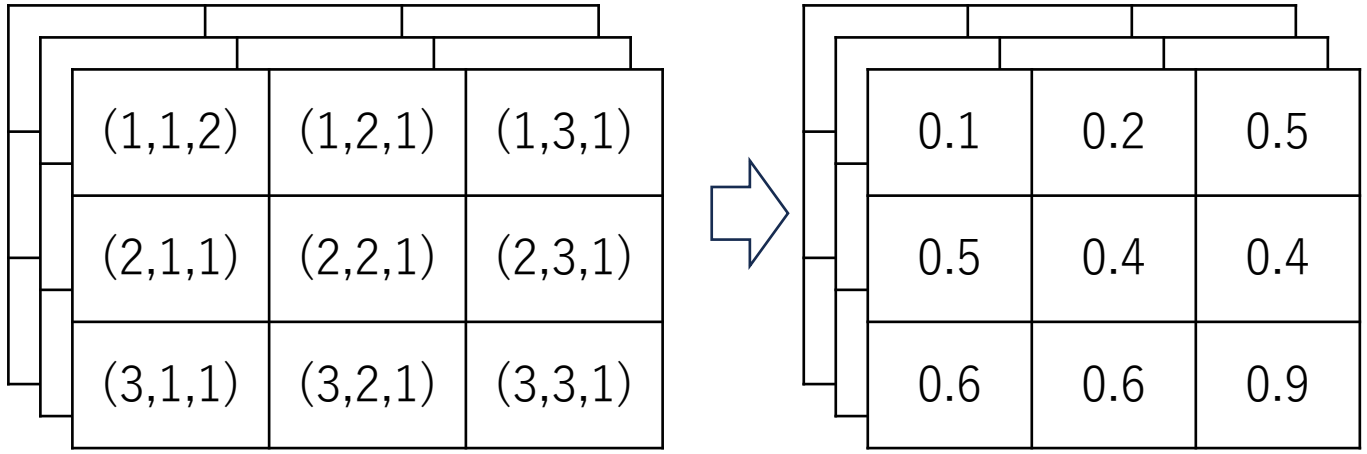
- Remaining homework was to train SIREN by a color image you have.
- Patrick finished the homework as shown in Slack.
- I also finished the homework independently.
- There was difference in the implementation.

Implementation by Patrick



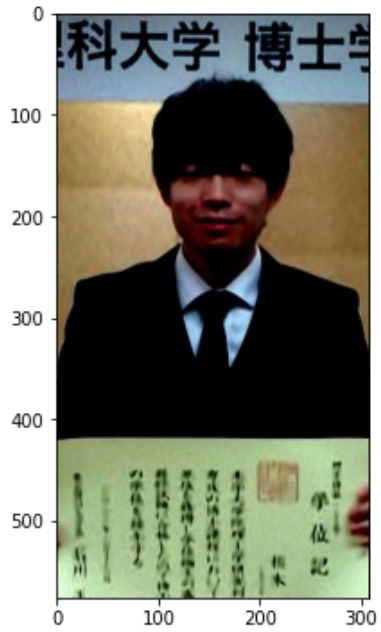
- Patrick got the RGB values from pixel coordinates
  - 3 output values for 1 input components

Implementation by Ryo

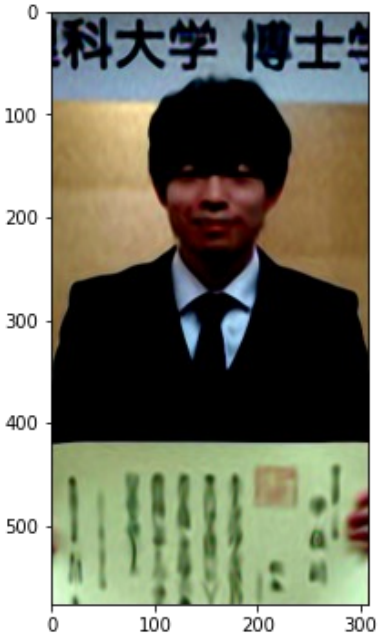


- Ryo got the RGB values from from pixel coordinates with RGB dimension
  - 1 output values for 1 input components

Implementation by Patrick



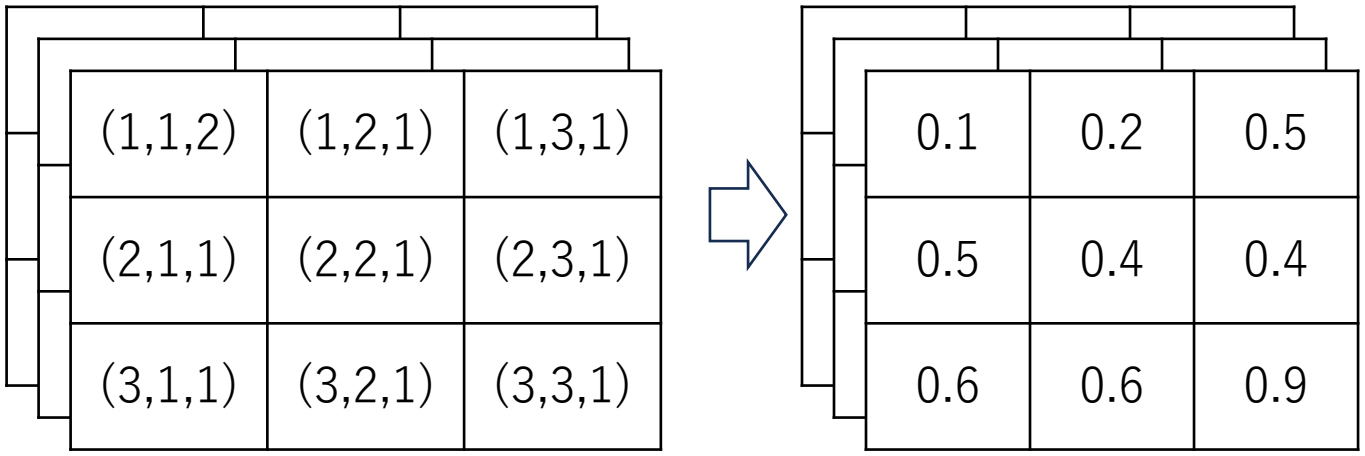
Implementation by Ryo



True

- Implementation by Patrick seems to be working better (since the input variable space is smaller than the one by Ryo?)

Plan



- I think this implementation can be applied directly to the indirect light table by extending input dimension
  - Normalized input 6 variables → indirect light distributions

Backup

```python
def get_mgrid(sidelen, dim=2):
    '''Generates a flattened grid of (x,y,...) coordinates in a range of -1 to 1.
    sidelen: int
    dim: int'''
    tensors = tuple(dim * [torch.linspace(-1, 1, steps=sidelen)])
    mgrid = torch.stack(torch.meshgrid(*tensors), dim=-1)
    mgrid = mgrid.reshape(-1, dim)
    return mgrid
```

```python
def get_mgrid(sidelen, dim=2):
    '''Generates a flattened grid of (x,y,...) coordinates in a range of -1 to 1.
    sidelen: int
    dim: int'''
    #tensors = tuple(dim * [torch.linspace(-1, 1, steps=sidelen)])
    tensors = tuple(torch.linspace(-1, 1, steps=size) for size in sidelen)
    mgrid = torch.stack(torch.meshgrid(*tensors), dim=-1)
    mgrid = mgrid.reshape(-1, dim)
    return mgrid
```

```python
def get_mgrid(sidelen, dim=2):
    '''Generates a flattened grid of (x,y,...) coordinates in a range of -1 to 1.
    sidelen: int
    dim: int'''
    tensors = tuple( [torch.linspace(-1, 1, steps=step) for step in sidelen])
    #tensors = tuple( [torch.linspace(-1, 1, steps=step) for step in [sidelen[0],sidelen[1]]])
    mgrid = torch.stack(torch.meshgrid(*tensors), dim=-1)
    mgrid = mgrid.reshape(-1, dim)
    return mgrid
```

```python
class ImageFitting(Dataset):
    def __init__(self, sidelength):
        super().__init__()
        img = get_cameraman_tensor(sidelength)
        self.pixels = img.permute(1, 2, 0).view(-1, 1)
        self.coords = get_mgrid(sidelength, 2)

    def __len__(self):
        return 1

    def __getitem__(self, idx):
        if idx > 0: raise IndexError

        return self.coords, self.pixels
```

```python
class ImageFitting(Dataset):
    def __init__(self, sidelength):
        super().__init__()
        img = get_cameraman_tensor(sidelength)
        #self.pixels = img.permute(1, 2, 0).reshape(-1, 1)
        self.pixels = img.permute(1, 2, 0).reshape(-1, 3)
        self.coords = get_mgrid(sidelength, 2)

    def __len__(self):
        return 1

    def __getitem__(self, idx):
        if idx > 0: raise IndexError

        return self.coords, self.pixels
```

```
img_siren = Siren(in_features=2, out_features=1, hidden_features=256,
                  hidden_layers=3, outermost_linear=True)
```

```
img_siren = Siren(in_features=2, out_features=3, hidden_features=256,
                  hidden_layers=3, outermost_linear=True)
```

```
img_siren = Siren(in_features=3, out_features=1, hidden_features=256,
```