비주얼 컴퓨팅 최신기술
기말 프로젝트

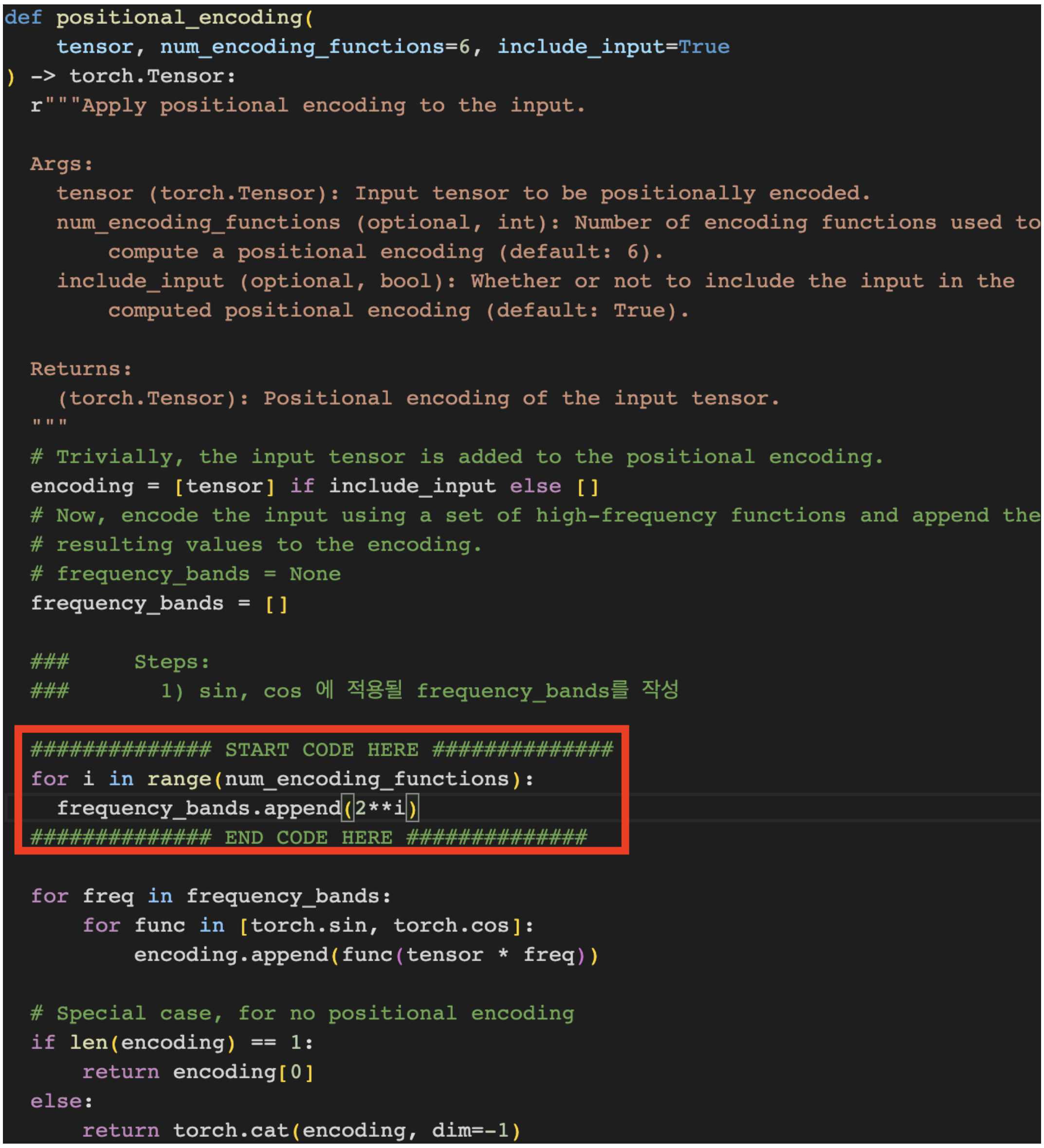
소프트웨어학부 **20195298** 박준용

목표

파이토치를 사용하여 **NeRF** 구현하기

- **NeRF** 네트워크를 직접 구현
- **skip-connection**이 **NeRF**에 미치는 영향 살펴보기

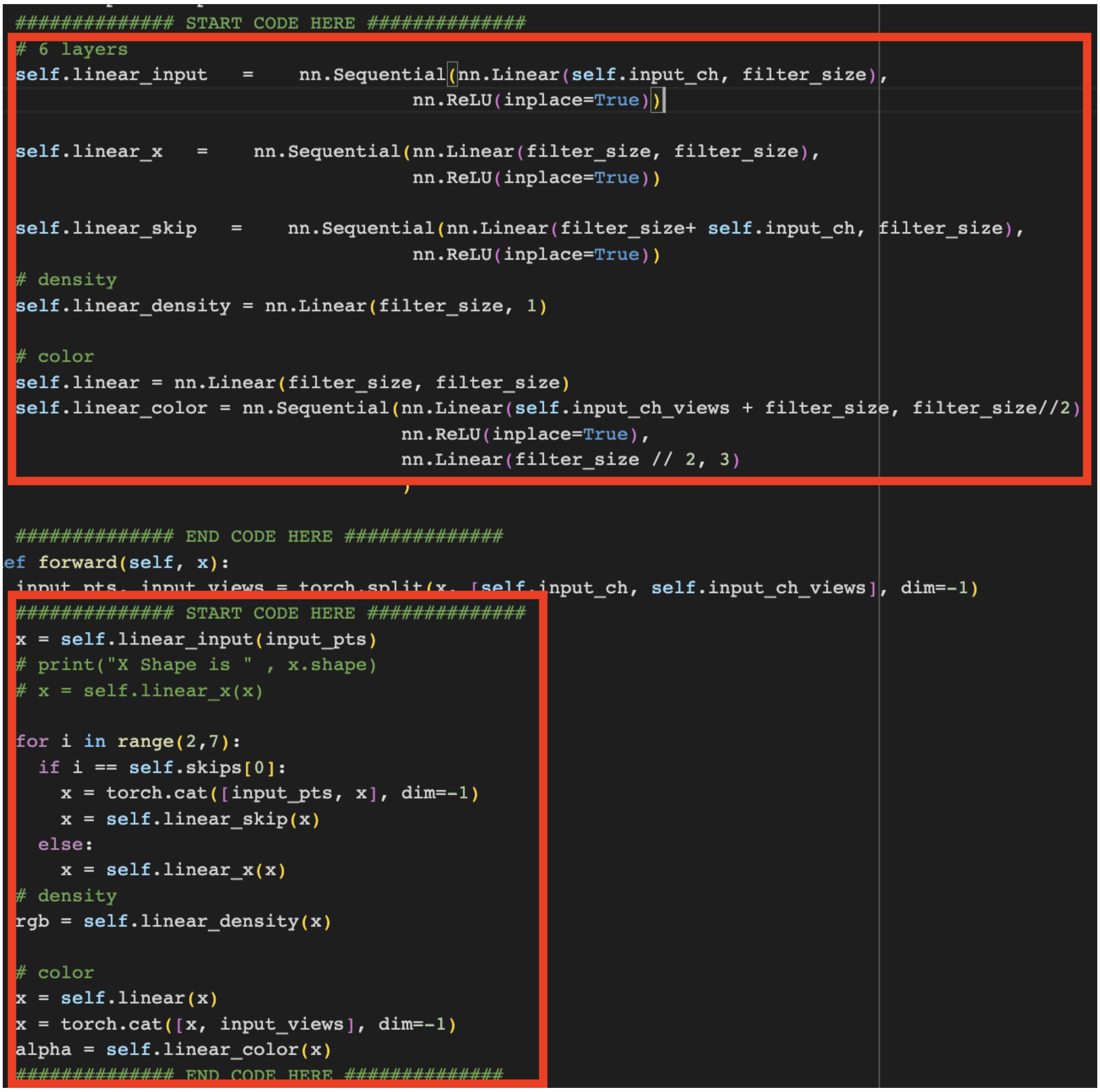
1. Get hands-on experience making a positional encoding function (Assignment 1)



```
def positional_encoding(
    tensor, num_encoding_functions=6, include_input=True
) -> torch.Tensor:
  r"""Apply positional encoding to the input.

  Args:
    tensor (torch.Tensor): Input tensor to be positionally encoded.
    num_encoding_functions (optional, int): Number of encoding functions used to
        compute a positional encoding (default: 6).
    include_input (optional, bool): Whether or not to include the input in the
        computed positional encoding (default: True).

  Returns:
    (torch.Tensor): Positional encoding of the input tensor.
  """
  # Trivially, the input tensor is added to the positional encoding.
  encoding = [tensor] if include_input else []
  # Now, encode the input using a set of high-frequency functions and append the
  # resulting values to the encoding.
  # frequency_bands = None
  frequency_bands = []

  ###     Steps:
  ###       1) sin, cos 에 적용될 frequency_bands를 작성

  ############## START CODE HERE ##############
  for i in range(num_encoding_functions):
    frequency_bands.append(2**i)
  ############## END CODE HERE ##############

  for freq in frequency_bands:
      for func in [torch.sin, torch.cos]:
          encoding.append(func(tensor * freq))

  # Special case, for no positional encoding
  if len(encoding) == 1:
      return encoding[0]
  else:
      return torch.cat(encoding, dim=-1)
```

2. **Get hands-on experience making a NeRF Network (Assignment 2)**

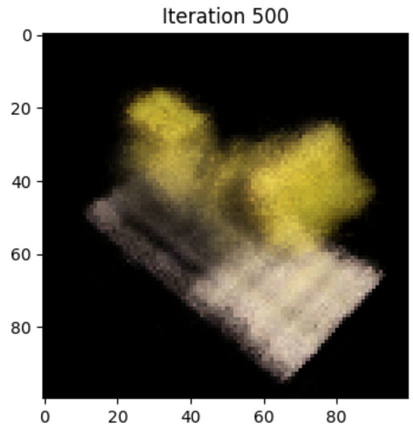
- **Build a network by stacking layers according to the figure on the left**
- **Red blocks mean output values**
- **Note that there are 6 mlp layers before the orange arrow**

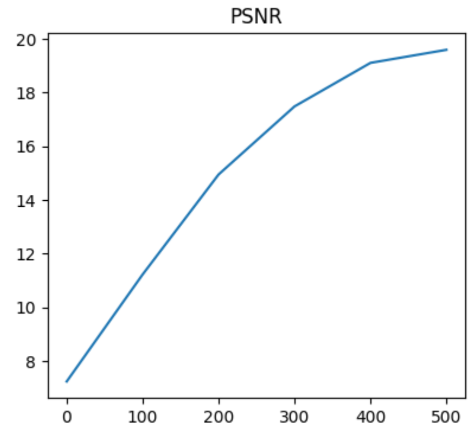


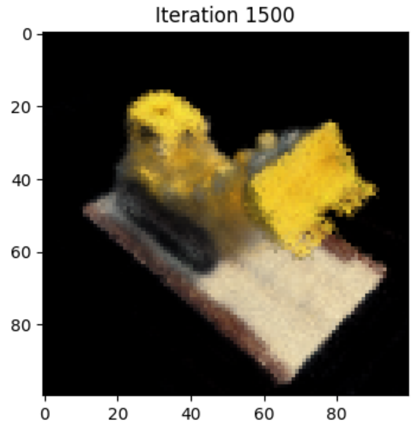
```
############## START CODE HERE ##############
# 6 layers
self.linear_input   =    nn.Sequential(nn.Linear(self.input_ch, filter_size),
                                   nn.ReLU(inplace=True))

self.linear_x   =    nn.Sequential(nn.Linear(filter_size, filter_size),
                                   nn.ReLU(inplace=True))

self.linear_skip   =    nn.Sequential(nn.Linear(filter_size+ self.input_ch, filter_size),
                                   nn.ReLU(inplace=True))
# density
self.linear_density = nn.Linear(filter_size, 1)

# color
self.linear = nn.Linear(filter_size, filter_size)
self.linear_color = nn.Sequential(nn.Linear(self.input_ch_views + filter_size, filter_size//2)
                                  nn.ReLU(inplace=True),
                                  nn.Linear(filter_size // 2, 3)
                                  )

############## END CODE HERE ##############
ef forward(self, x):
 input_pts, input_views = torch.split(x, [self.input_ch, self.input_ch_views], dim=-1)
############## START CODE HERE ##############
x = self.linear_input(input_pts)
# print("X Shape is " , x.shape)
# x = self.linear_x(x)

for i in range(2,7):
  if i == self.skips[0]:
    x = torch.cat([input_pts, x], dim=-1)
    x = self.linear_skip(x)
  else:
    x = self.linear_x(x)
# density
rgb = self.linear_density(x)

# color
x = self.linear(x)
x = torch.cat([x, input_views], dim=-1)
alpha = self.linear_color(x)
############## END CODE HERE ##############
```

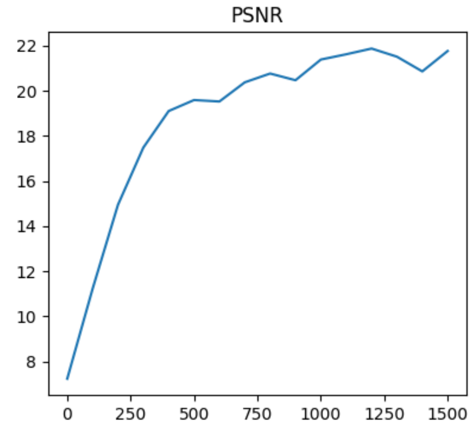
3. **Check the process in which the NeRF model is trained (Assignment 3)**

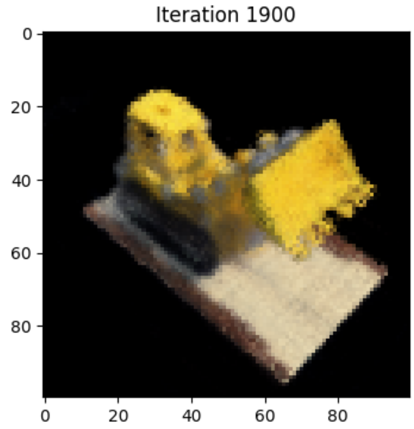
- **Take a shot at training process (500 iter, 1500 iter, 1900 iter)**

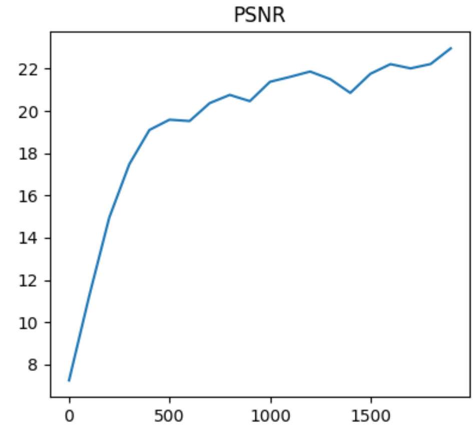

Iteration 500


PSNR


Iteration 1500

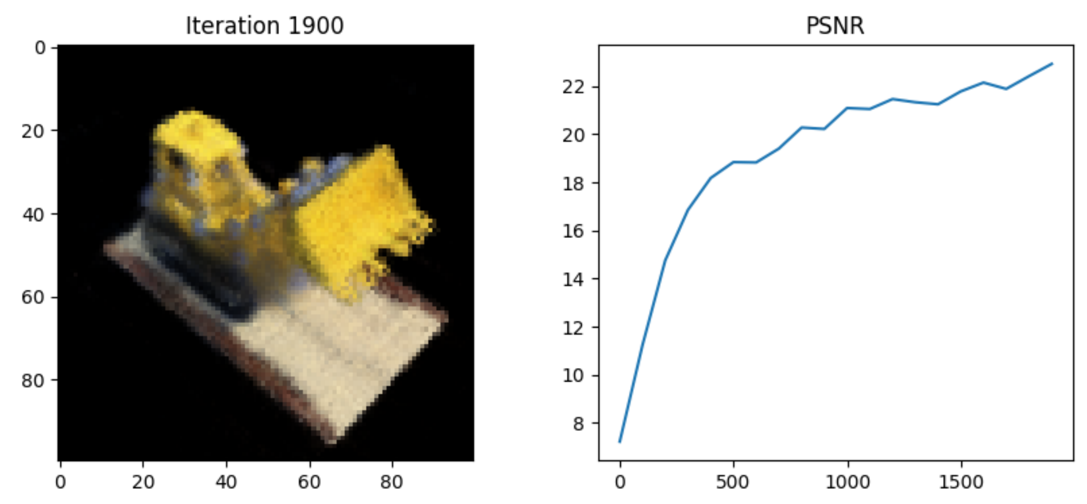

PSNR


Iteration 1900

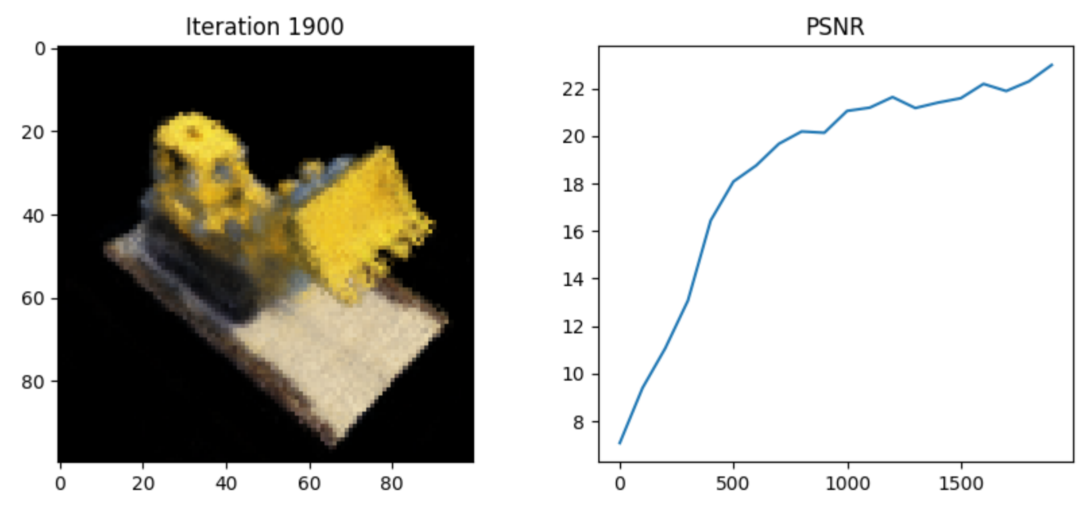

PSNR

4. **Change Skip connection parameter and remove skip connection (Assignment 4)**

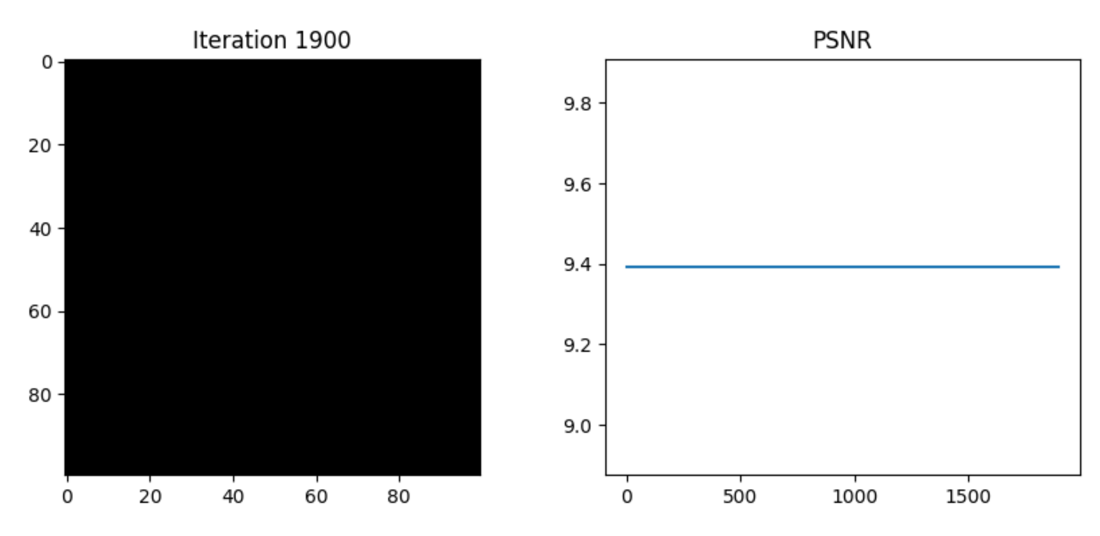
- **Take a shot at last iteration image**
- **(Total 3 images – apply skip connection at third layer, apply skip connection at fifth layer, remove all skip connections)**



skip_connection = 5



skip_connection = 3



Iteration 1900

PSNR

skip_connection = 0