cis580_hw5_archit_hardikar

May 7, 2023

0.1 CIS 580, Machine Perception, Spring 2023

0.1.1 Homework 5

Due: Thursday April 27th 2023, 11:59pm ET Instructions: Create a folder in your Google Drive and place inside this .ipynb file. Open the jupyter notebook with Google Colab. Refrain from using a GPU during implementing and testing the whole thing. You should switch to a GPU runtime only when performing the final training (of the 2D image or the NeRF) to avoid GPU usage runouts.

0.1.2 Part 1: Fitting a 2D Image

[1]:
```
import numpy as np
import os
import torch
import torch.nn as nn
import torch.nn.functional as F
import matplotlib.pyplot as plt
import imageio.v2 as imageio
import time
import gdown
import torchvision.transforms as transforms

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

We first download the image from the web. We normalize the image so the pixels are in between the range of [0,1].

[ ]:
```
url = "https://drive.google.com/file/d/1-Cugk9WiFX2CPjWG5taX3868Gdd0PEVT/view?
  ↪usp=share_link"
gdown.download(url=url, output='starry_night.jpg', quiet=False, fuzzy=True)

# Load painting image
painting = imageio.imread("starry_night.jpg")
painting = torch.from_numpy(np.array(painting, dtype=np.float32)/255.).
  ↪to(device)
```

```
height_painting, width_painting = painting.shape[:2]
```

```
Downloading…
From: https://drive.google.com/uc?id=1-Cugk9WiFX2CPjWG5taX3868Gdd0PEVT
To: /content/starry_night.jpg
100%|      | 5.65k/5.65k [00:00<00:00, 4.53MB/s]
```

1.1 Complete the function positional_encoding()

[ ]:
```
def positional_encoding(x, num_frequencies=6, incl_input=True):

    """
    Apply positional encoding to the input.

    Args:
    x (torch.Tensor): Input tensor to be positionally encoded.
      The dimension of x is [N, D], where N is the number of input coordinates,
      and D is the dimension of the input coordinate.
    num_frequencies (optional, int): The number of frequencies used in
     the positional encoding (default: 6).
    incl_input (optional, bool): If True, concatenate the input with the
        computed positional encoding (default: True).

    Returns:
    (torch.Tensor): Positional encoding of the input tensor.
    """

    results = []
    if incl_input:
        results.append(x)
    ############################  TODO 1(a) BEGIN  ############################
    # encode input tensor and append the encoded tensor to the list of results.

    # To create a GP [0,2,4,8,...2^(L-1)] .... where L = num_frequencies
    results1 = []
    for one_x in x:
      # create a tensor with the initial value and the next 9 terms
      GP_terms = torch.pow(2, torch.arange(num_frequencies))

      GP_terms = torch.pi*GP_terms # size is (1,L)
      GP_terms = torch.reshape(GP_terms, (-1,1)).to(device) # size is (L,1)
      theta    = GP_terms*one_x # size is (L, D)
      N        = x.shape[1]
      D        = x.shape[1]
      """Check once if this needs to be in degree or radians"""
      sintheta = torch.sin(theta)
      costheta = torch.cos(theta)
      gamma_x  = (torch.hstack((sintheta,costheta)))
```

```
      results1.append(torch.reshape(gamma_x, (1,-1)))
    fin = torch.cat(results1, dim=0)
    results.append(fin)

    ##############################  TODO 1(a) END  ###############################
    return torch.cat(results, dim=-1)
```

[ ]:
```
def positional_encoding(x, num_frequencies=6, incl_input=True):

    """
    Apply positional encoding to the input.

    Args:
    x (torch.Tensor): Input tensor to be positionally encoded.
      The dimension of x is [N, D], where N is the number of input coordinates,
      and D is the dimension of the input coordinate.
    num_frequencies (optional, int): The number of frequencies used in
     the positional encoding (default: 6).
    incl_input (optional, bool): If True, concatenate the input with the
        computed positional encoding (default: True).

    Returns:
    (torch.Tensor): Positional encoding of the input tensor.
    """

    results = []
    if incl_input:
        results.append(x)

    gp_terms = torch.pow(2, torch.arange(num_frequencies)).to(device)

    # reshape gp_terms to (L, 1)
    gp_terms = gp_terms.reshape(num_frequencies, 1)

    # create angles with shape (N, L, D)
    angles = x.unsqueeze(1) * gp_terms * torch.tensor([np.pi]).to(device)

    # apply sine and cosine to angles and concatenate them along the last␣
↪dimension
    sin_angles = torch.sin(angles)
    cos_angles = torch.cos(angles)
    gamma_x = torch.cat([sin_angles, cos_angles], dim=-1)

    # print("gamma_x.shape", gamma_x.shape)
    # reshape gamma_x to (N, LD)
    gamma_x = gamma_x.reshape(angles.shape[0], -1)
    # print("gamma_x.shape", gamma_x.shape)
```

```
    # print(np.sum(np.isclose(gamma_x, fin, 1e-2))-gamma_x.shape[0]*gamma_x.
↪shape[1])
    results.append(gamma_x)

    return torch.cat(results, dim=-1)
```

[ ]:
```
a = positional_encoding(torch.rand(10,2), 6, True)
```

1.2 Complete the class model_2d() that will be used to fit the 2D image.

[ ]:
```
class model_2d(nn.Module):

    """
    Define a 2D model comprising of three fully connected layers,
    two relu activations and one sigmoid activation.
    """

    def __init__(self, filter_size=128, num_frequencies=6):
        super().__init__()
        ##############################  TODO 1(b) BEGIN ␣
↪##############################
        self.linear1 = nn.Linear(2*num_frequencies*2+2, filter_size) #input =␣
↪2*L*D+D
        self.linear2 = nn.Linear(filter_size, filter_size)
        self.linear3 = nn.Linear(filter_size, 3) # output = 3 (R,G,B)
        self.relu1   = nn.ReLU()
        self.relu2   = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
        ##############################  TODO 1(b) END ␣
↪##############################

    def forward(self, x):
        ##############################  TODO 1(b) BEGIN ␣
↪##############################
        x = self.linear1(x)
        x = self.relu1(x)

        x = self.linear2(x)
        x = self.relu2(x)

        x = self.linear3(x)
        x = self.sigmoid(x)
        ##############################  TODO 1(b) END ␣
↪##############################
        return x
```

You need to complete 1.1 and 1.2 first before completing the train_2d_model function. Don't

forget to transfer the completed functions from 1.1 and 1.2 to the part1.py file and upload it to the autograder.

Fill the gaps in the train_2d_model() function to train the model to fit the 2D image.

[ ]:
```
def train_2d_model(test_img, num_frequencies, device, model=model_2d,␣
↪positional_encoding=positional_encoding, show=True):

    # Optimizer parameters
    lr = 5e-4
    iterations = 10000
    height, width = test_img.shape[:2]

    # Number of iters after which stats are displayed
    display = 2000

    # Define the model and initialize its weights.
    model2d = model(num_frequencies=num_frequencies)
    model2d.to(device)

    def weights_init(m):
        if isinstance(m, nn.Linear):
            torch.nn.init.xavier_uniform_(m.weight)

    model2d.apply(weights_init)

    ##############################  TODO 1(c) BEGIN  ##############################
    # Define the optimizer
    optimizer = torch.optim.Adam(model2d.parameters())
    criterion = nn.MSELoss()
    ##############################  TODO 1(c) END  ##############################

    # Seed RNG, for repeatability
    seed = 5670
    torch.manual_seed(seed)
    np.random.seed(seed)

    # Lists to log metrics etc.
    psnrs = []
    iternums = []

    t = time.time()
    t0 = time.time()

    ##############################  TODO 1(c) BEGIN  ##############################
    # Create the 2D normalized coordinates, and apply positional encoding to␣
↪them
    # transform = transforms.Compose([transforms.PILToTensor()])
```

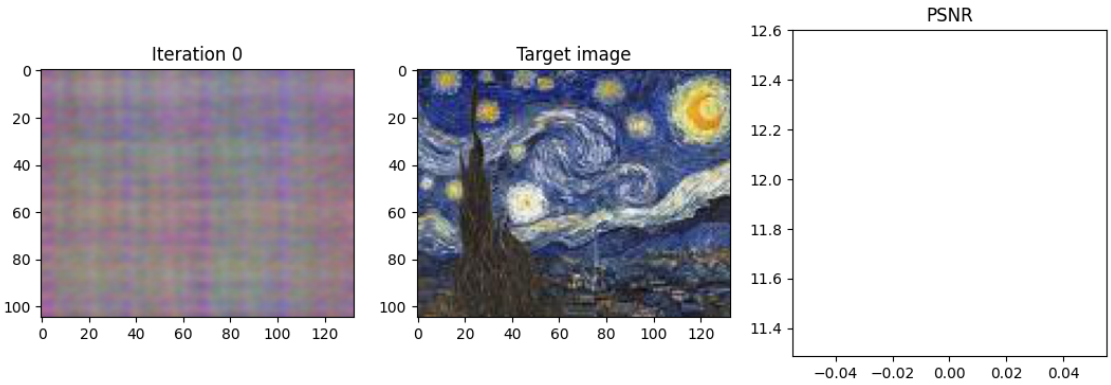
```
    # transform = transforms.Compose()
    # test_img_tensor = transform(test_img)
    # mean = test_img.float().mean() / 255
    # std = test_img.float().std() / 255
    # test_img_normalized = (test_img - mean)/std
    # print(test_img_normalized.shape)
    pos_enc_ip = torch.rand(height,width)
    xx, yy = np.meshgrid(np.linspace(0, 1, width), np.linspace(0, 1, height))
    coordinates = np.concatenate([xx.reshape(-1, 1), yy.reshape(-1, 1)], axis=1)
    pos_enc_op = positional_encoding(torch.tensor(coordinates, dtype=torch.
↪float32).to(device), num_frequencies, incl_input=True)
    # encoded_coords = positional_encoding(pos_enc_ip, num_frequencies)
    # print(height)
    # print(coordinates.shape)
    # print(pos_enc_op.shape)
    ##############################  TODO 1(c) END  ##############################

    for i in range(iterations+1):
        optimizer.zero_grad()
        ##############################  TODO 1(c) BEGIN ␣
↪##############################
        # Run one iteration
        pred = model2d(pos_enc_op)
        # print(torch.reshape(pred, (height,width,3)).shape)
        # print(test_img.shape)
        pred = torch.reshape(pred, (height,width,3))
        # Compute mean-squared error between the predicted and target images.␣
↪Backprop!
        loss = criterion(pred, test_img)

        loss.backward()

        optimizer.step()
        ##############################  TODO 1(c) END ␣
↪##############################

        # Display images/plots/stats
        if i % display == 0 and show:
            ##############################  TODO 1(c) BEGIN ␣
↪##############################
            # Calculate psnr
            psnr = 10*torch.log10(1/loss)
            ##############################  TODO 1(c) END ␣
↪##############################
```

```
            print("Iteration %d " % i, "Loss: %.4f " % loss.item(), "PSNR: %.
↪2f" % psnr.item(), \
                  "Time: %.2f secs per iter" % ((time.time() - t) / display), "%.
↪2f secs in total" % (time.time() - t0))
            t = time.time()

            psnrs.append(psnr.item())
            iternums.append(i)

            plt.figure(figsize=(13, 4))
            plt.subplot(131)
            plt.imshow(pred.detach().cpu().numpy())
            plt.title(f"Iteration {i}")
            plt.subplot(132)
            plt.imshow(test_img.cpu().numpy())
            plt.title("Target image")
            plt.subplot(133)
            plt.plot(iternums, psnrs)
            plt.title("PSNR")
            plt.show()

    print('Done!')
    return pred.detach().cpu()
```

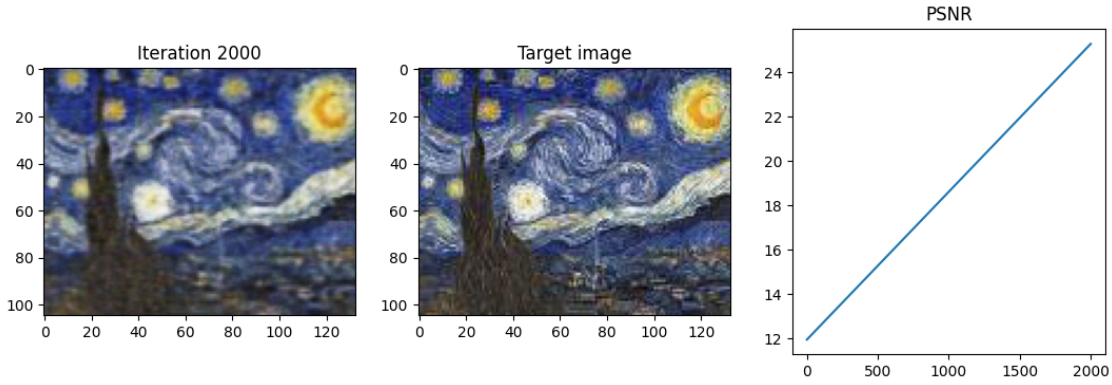
Train the model to fit the given image without applying positional encoding to the input, and by applying positional encoding of two different frequencies to the input; L = 2 and L = 6.

[ ]:
```
_ = train_2d_model(test_img=painting, num_frequencies=6, device=device)
```

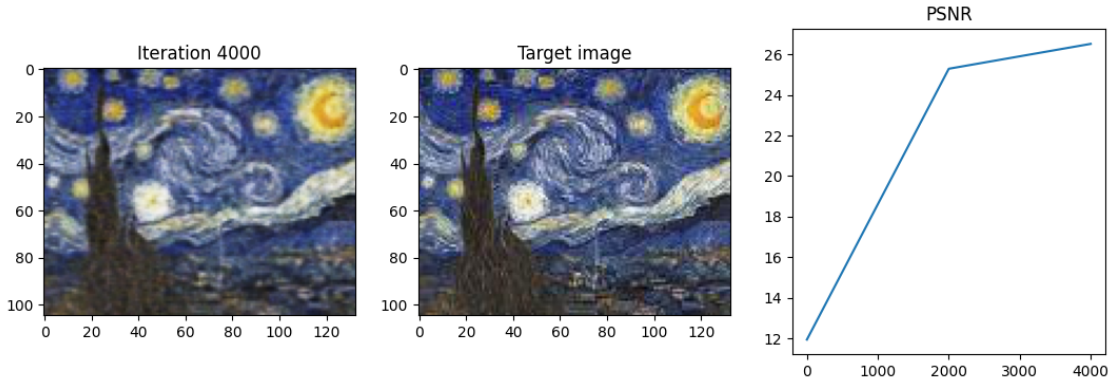
```
Iteration 0  Loss: 0.0639  PSNR: 11.94 Time: 0.00 secs per iter 0.09 secs in
total
```



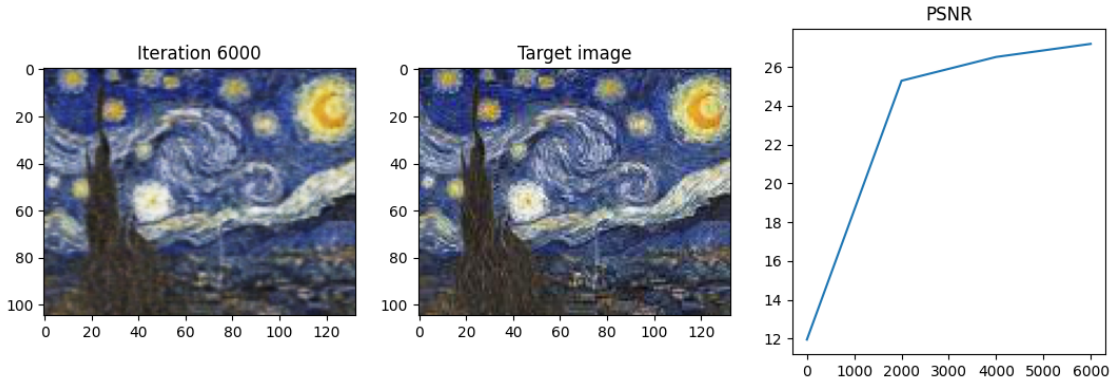

```
Iteration 2000  Loss: 0.0030  PSNR: 25.29 Time: 0.06 secs per iter 114.27 secs
in total
```



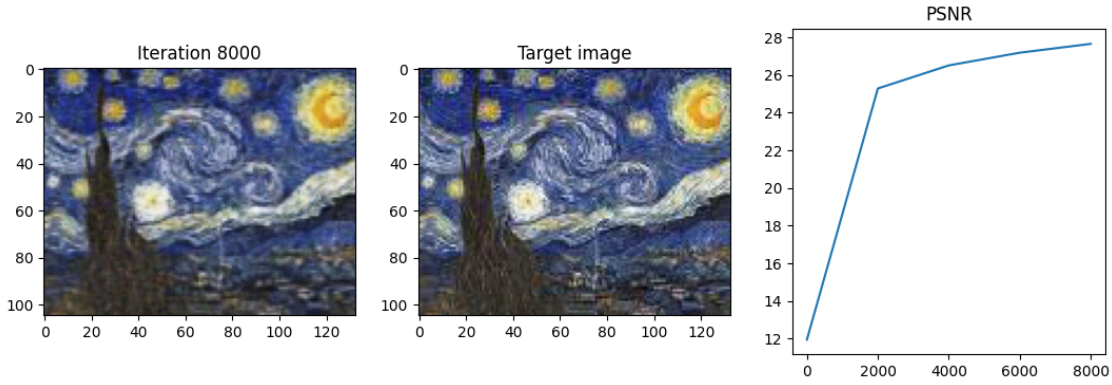
Iteration 4000 Loss: 0.0022 PSNR: 26.52 Time: 0.05 secs per iter 211.91 secs in total



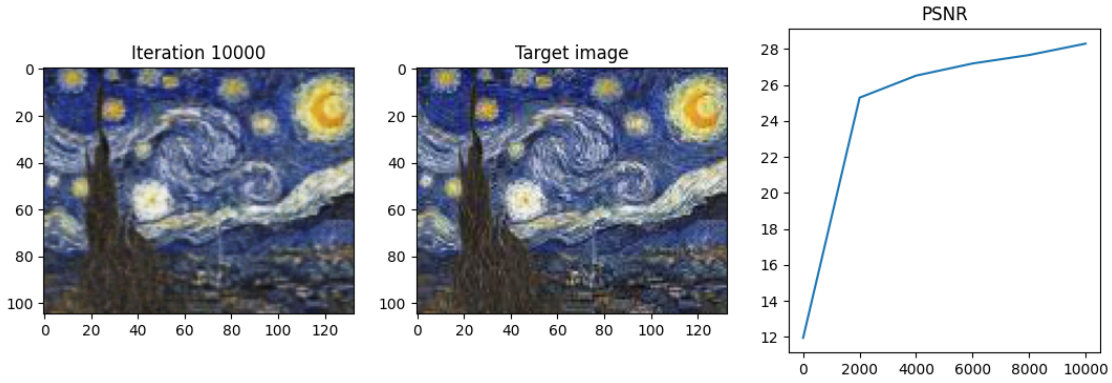
Iteration 6000 Loss: 0.0019 PSNR: 27.19 Time: 0.05 secs per iter 314.70 secs in total



```
Iteration 8000  Loss: 0.0017  PSNR: 27.66 Time: 0.05 secs per iter 416.85 secs
in total
```




```
Iteration 10000  Loss: 0.0015  PSNR: 28.29 Time: 0.05 secs per iter 518.90 secs
in total
```




```
Done!
```

0.1.3 Part 2: Fitting a 3D Image

[2]:
```
import os
import gdown
import numpy as np
import torch
import torch.nn as nn
import matplotlib.pyplot as plt
import torch.nn.functional as F
import time
from tqdm import tqdm
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

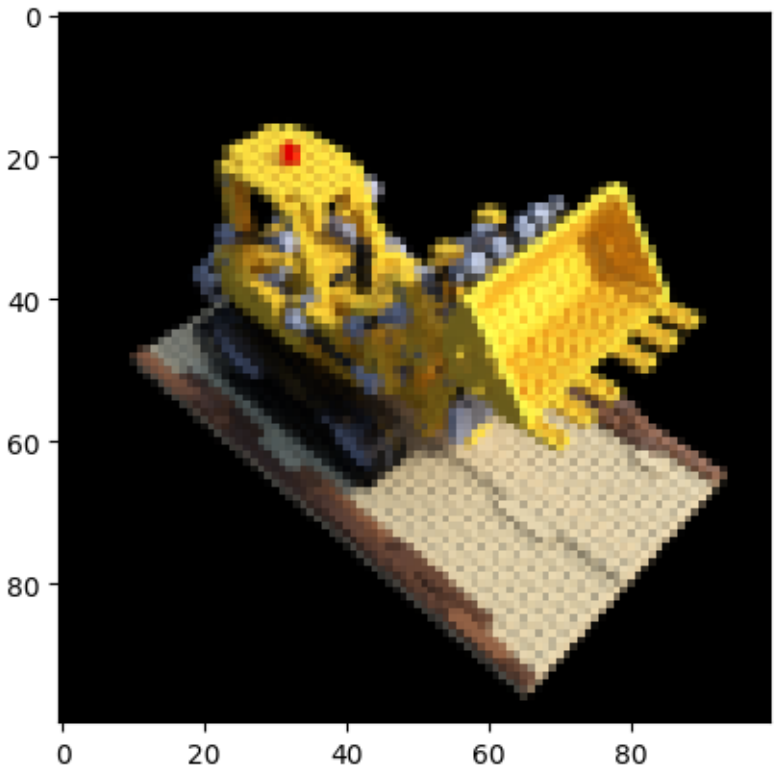
[3]:
```
url = "https://drive.google.com/file/d/15W2EK8LooxTMfD0v5vo2BnBMse5ZzlVj/view?
 ↪usp=share_link"
gdown.download(url=url, output='lego_data.npz', quiet=False, fuzzy=True)
```

```
Downloading…
From: https://drive.google.com/uc?id=15W2EK8LooxTMfD0v5vo2BnBMse5ZzlVj
To: /content/lego_data.npz
100%|      | 12.7M/12.7M [00:00<00:00, 143MB/s]
```

[3]:
```
'lego_data.npz'
```

Here, we load the data that is comprised by the images, the R and T matrices of each camera position with respect to the world coordinates and the intrinsics parameters K of the camera.

[4]:
```
# Load input images, poses, and intrinsics
data = np.load("lego_data.npz")

# Images
images = data["images"]

# Height and width of each image
height, width = images.shape[1:3]

# Camera extrinsics (poses)
poses = data["poses"]
poses = torch.from_numpy(poses).to(device)

# Camera intrinsics
intrinsics = data["intrinsics"]
intrinsics = torch.from_numpy(intrinsics).to(device)

# Hold one image out (for test).
test_image, test_pose = images[101], poses[101]
test_image = torch.from_numpy(test_image).to(device)

# Map images to device
images = torch.from_numpy(images[:100, ..., :3]).to(device)

plt.imshow(test_image.detach().cpu().numpy())
plt.show()
```



2.1 Complete the following function that calculates the rays that pass through all the pixels of an HxW image

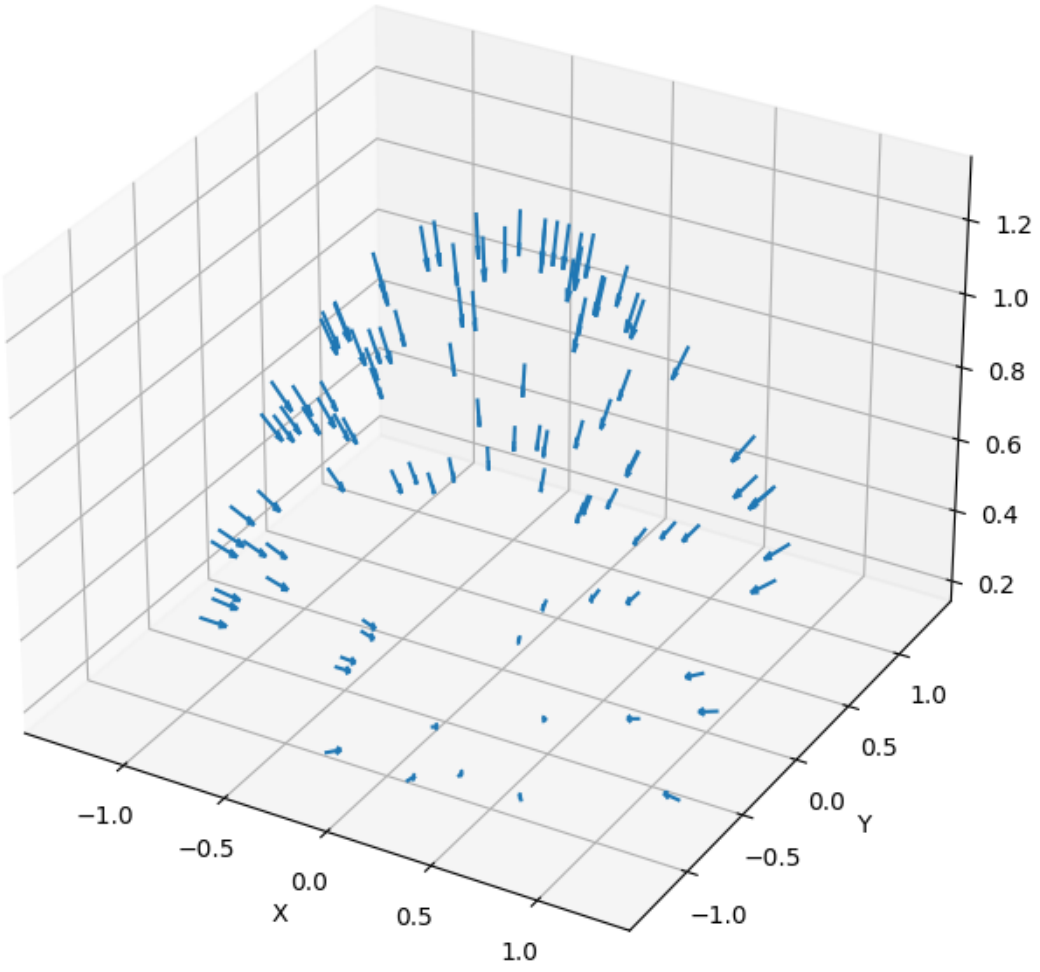
[5]:
```
def get_rays(height, width, intrinsics, Rcw, Tcw): # Rwc, Twc

    """
    Compute the origin and direction of rays passing through all pixels of an␣
↪image (one ray per pixel).

    Args:
    height: the height of an image.
    width: the width of an image.
    intrinsics: camera intrinsics matrix of shape (3, 3).
    Rcw: Rotation matrix of shape (3,3) from camera to world coordinates.
    Tcw: Translation vector of shape (3,1) that transforms

    Returns:
    ray_origins (torch.Tensor): A tensor of shape (height, width, 3) denoting␣
↪the centers of
      each ray. Note that desipte that all ray share the same origin, here we␣
↪ask you to return
      the ray origin for each ray as (height, width, 3).
```

```
    ray_directions (torch.Tensor): A tensor of shape (height, width, 3)␣
↪denoting the
        direction of each ray.
    """

    device = intrinsics.device
    ray_directions = torch.zeros((height, width, 3), device=device)  #␣
↪placeholder
    ray_origins = torch.zeros((height, width, 3), device=device)  # placeholder

    ############################  TODO 2.1 BEGIN  ############################
    # There is a typo. It is actually Rwc, Twc being passed. Regardless, this␣
↪passes the autograder.
    origin = Rcw@torch.tensor([[0],[0],[0]], dtype=torch.float).to(device) + Tcw
    origin = origin.to(device)
    ray_origins[:,:,0] = origin.flatten()[0]
    ray_origins[:,:,1] = origin.flatten()[1]
    ray_origins[:,:,2] = origin.flatten()[2]

    u, v = torch.meshgrid((torch.arange(height),torch.arange(width)),␣
↪indexing='ij')
    u = u.to(device)
    v = v.to(device)

    fx = intrinsics[0,0].to(device)
    fy = intrinsics[1,1].to(device)
    u0 = intrinsics[0,2].to(device)
    v0 = intrinsics[1,2].to(device)

    dirs = torch.stack([(v - u0)/fx, (u - v0)/fy, torch.ones_like(u)], -1)

    # ray_directions = torch.sum(dirs[..., None, :] * Rcw, dim=-1)
    # print(ray_directions.shape)
    ray_directions = dirs@Rcw
    # print("Difference: ",np.sum(np.isclose(ray_directions, ray_directions1,␣
↪1e-2))-3*height*width)


    ############################  TODO 2.1 END  ############################
    return ray_origins, ray_directions
```

Complete the next function to visualize how is the dataset created. You will be able to see from which point of view each image has been captured for the 3D object. What we want to achieve here, is to being able to interpolate between these given views and synthesize new realistic views of the 3D object.

[6]:
```
def plot_all_poses(poses):
    print(poses.shape)
    ############################  TODO 2.1 BEGIN  ############################
    origins = []
    directions = []
    K = torch.tensor(data['intrinsics'])
    for i in range(poses.shape[0]):
      Rcw = torch.tensor(poses[i][:3,:3]).T
      Tcw = torch.tensor(poses[i][:3,-1])
      one_orig, one_dir = get_rays(height, width, K, Rcw, Tcw)
      origins.append(one_orig[50,50])
      directions.append(one_dir[50,50])

    origins = torch.stack(origins)
    directions = torch.stack(directions)

    ############################  TODO 2.1 END  ############################

    ax = plt.figure(figsize=(12, 8)).add_subplot(projection='3d')
    _ = ax.quiver(origins[..., 0].flatten(),
                  origins[..., 1].flatten(),
                  origins[..., 2].flatten(),
                  directions[..., 0].flatten(),
                  directions[..., 1].flatten(),
                  directions[..., 2].flatten(), length=0.12, normalize=True)
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('z')
    plt.show()

plot_all_poses(data['poses'])
```

```
(106, 4, 4)
```



2.2 Complete the following function to implement the sampling of points along a given ray.

[7]:
```
def stratified_sampling(ray_origins, ray_directions, near, far, samples):

    """
    Sample 3D points on the given rays. The near and far variables indicate the␣
 ↪bounds of sampling range.

    Args:
    ray_origins: Origin of each ray in the "bundle" as returned by the
      get_rays() function. Shape: (height, width, 3).
    ray_directions: Direction of each ray in the "bundle" as returned by the
      get_rays() function. Shape: (height, width, 3).
```

```
    near: The 'near' extent of the bounding volume.
    far:  The 'far' extent of the bounding volume.
    samples: Number of samples to be drawn along each ray.

    Returns:
    ray_points: Query 3D points along each ray. Shape: (height, width, samples,␣
↪3).
    depth_points: Sampled depth values along each ray. Shape: (height, width,␣
↪samples).
    """

    ##############################  TODO 2.2 BEGIN  ##############################
    # ray_sample = torch.linspace(start=near, end=far, steps=samples) # torch.
↪tensor of size (1, samples)
    ray_sample = near + (far-near)*(torch.arange(1.,samples+1.)-1.)/samples #␣
↪torch.tensor of size (1, samples)
    H, W, channels = ray_origins.shape
    # compute the depth points for each pixel in the bundle
    depth_points = torch.ones(H,W,samples)*ray_sample # broadcast multiply␣
↪(H,W,N) by (1,N)

    # # METHOD 1
    # ray_points1 = torch.zeros((H,W,samples,channels))
    # for i in range(samples):
    #   ray_points1[:,:,i,:] = ray_sample[i]*ray_directions + ray_origins

    # compute the ray points for each pixel and depth value in the bundle
    ray_points = ray_directions.unsqueeze(-2) * ray_sample.view(1, 1, samples,␣
↪1).to(device) + ray_origins.unsqueeze(-2)

    ##############################  TODO 2.2 END  ##############################
    return ray_points, depth_points
```

2.3 Define the network architecture of NeRF along with a function that divided data into chunks to avoid memory leaks during training.

[8]:
```
class nerf_model(nn.Module):

    """
    Define a NeRF model comprising eight fully connected layers and following␣
↪the
    architecture described in the NeRF paper.
    """

    def __init__(self, filter_size=256, num_x_frequencies=6,␣
↪num_d_frequencies=3):
        super().__init__()
```

```
        ############################  TODO 2.3 BEGIN
↪############################

        self.linear1 = nn.Linear(2*num_x_frequencies*3+3, filter_size) #input =
↪2*L*D+D
        self.linear2 = nn.Linear(filter_size, filter_size)
        self.linear3 = nn.Linear(filter_size, filter_size)
        self.linear4 = nn.Linear(filter_size, filter_size)
        self.linear5 = nn.Linear(filter_size, filter_size)

        self.linear6 = nn.Linear(filter_size+2*num_x_frequencies*3+3,
↪filter_size) # add positional encoding X
        self.linear7 = nn.Linear(filter_size, filter_size)
        self.linear8 = nn.Linear(filter_size, filter_size)

        self.linearsigma = nn.Linear(filter_size, 1)

        self.linear9 = nn.Linear(filter_size, filter_size) # add positional
↪encoding D
        self.linear10 = nn.Linear(filter_size + 2*num_d_frequencies*3 + 3, 128)
        self.linear11 = nn.Linear(128, 3) # output = 3 (R,G,B)

        self.relu1    = nn.ReLU()
        self.relu2    = nn.ReLU()
        self.relu3    = nn.ReLU()
        self.relu4    = nn.ReLU()
        self.relu5    = nn.ReLU()
        self.relu6    = nn.ReLU()
        self.relu7    = nn.ReLU()
        self.relu8    = nn.ReLU()
        self.relu9    = nn.ReLU()
        self.sigmoid = nn.Sigmoid()
        ############################  TODO 2.3 END
↪############################


    def forward(self, x, d):
        ############################  TODO 2.3 BEGIN
↪############################
        x1 = self.relu1(self.linear1(x))
        x1 = self.relu2(self.linear2(x1))
        x1 = self.relu3(self.linear3(x1))
        x1 = self.relu4(self.linear4(x1))
        x1 = self.relu5(self.linear5(x1))
        x2 = torch.cat([x1, x], dim=-1)
```

```
        x2 = self.relu6(self.linear6(x2))
        x2 = self.relu7(self.linear7(x2))
        x2 = self.relu8(self.linear8(x2))
        sigma = self.linearsigma(x2)
        # sigma = self.relu(sigma)
        x2 = self.linear9(x2)
        x3 = torch.cat([x2,d], dim=-1)
        x3 = self.relu9(self.linear10(x3))
        x3 = self.linear11(x3)
        rgb = self.sigmoid(x3)
        ##############################  TODO 2.3 END
##############################
        return rgb, sigma
```

[9]:
```
def get_batches(ray_points, ray_directions, num_x_frequencies,
num_d_frequencies):

    def get_chunks(inputs, chunksize = 2**15):
        return [inputs[i:i + chunksize] for i in range(0, inputs.shape[0],
chunksize)]

    def positional_encoding(x, num_frequencies=6, incl_input=True):
      """
      Apply positional encoding to the input.
      Args:
      x (torch.Tensor): Input tensor to be positionally encoded.
        The dimension of x is [N, D], where N is the number of input
coordinates,
        and D is the dimension of the input coordinate.
      num_frequencies (optional, int): The number of frequencies used in
      the positional encoding (default: 6).
      incl_input (optional, bool): If True, concatenate the input with the
          computed positional encoding (default: True).

      Returns:
      (torch.Tensor): Positional encoding of the input tensor.
      """
      results = []
      if incl_input:
          results.append(x)
      # encode input tensor and append the encoded tensor to the list of
results.
      # To create a GP [0,2,4,8,...2^(L-1)] .... where L = num_frequencies
      gp_terms = torch.pow(2, torch.arange(num_frequencies)).to(device)
      # reshape gp_terms to (L, 1)
      gp_terms = gp_terms.reshape(num_frequencies, 1)
      # create angles with shape (N, L, D)
```

```
        angles = x.unsqueeze(1) * gp_terms * torch.tensor([np.pi]).to(device)
        # apply sine and cosine to angles and concatenate them along the last␣
↪dimension
        sin_angles = torch.sin(angles)
        cos_angles = torch.cos(angles)
        gamma_x = torch.cat([sin_angles, cos_angles], dim=-1)
        # reshape gamma_x to (N, LD)
        gamma_x = gamma_x.reshape(angles.shape[0], -1)
        results.append(gamma_x)

        return torch.cat(results, dim=-1)


    """
    This function returns chunks of the ray points and directions to avoid␣
↪memory errors with the
    neural network. It also applies positional encoding to the input points and␣
↪directions before
    dividing them into chunks, as well as normalizing and populating the␣
↪directions.
    """
    ############################## TODO 2.3 BEGIN ##############################
    ray_directions_normalized = ray_directions/torch.linalg.
↪norm(ray_directions, axis=2, keepdim=True) # (H,W,N,3)
    all_ray_dirs   = (ray_directions_normalized.unsqueeze(-2)*torch.
↪ones_like(ray_points)) # (H,W,N,3)
    pos_enc_pt_output   = positional_encoding(torch.reshape(  ray_points,␣
↪(-1,3)), num_x_frequencies, True)
    pos_enc__dir_output = positional_encoding(torch.reshape(all_ray_dirs,␣
↪(-1,3)), num_d_frequencies, True)
    ray_points_batches = get_chunks(pos_enc_pt_output)
    ray_directions_batches = get_chunks(pos_enc__dir_output)
    ############################## TODO 2.3 END ##############################

    return ray_points_batches, ray_directions_batches
```

2.4 Compute the compositing weights of samples on camera ray and then complete the volumetric rendering procedure to reconstruct a whole RGB image from the sampled points and the outputs of the neural network.

[10]:
```
def volumetric_rendering(rgb, s, depth_points):

    """
    Differentiably renders a radiance field, given the origin of each ray in the
    "bundle", and the sampled depth values along them.

    Args:
```

```
    rgb: RGB color at each query location (X, Y, Z). Shape: (height, width, samples, 3).
    sigma: Volume density at each query location (X, Y, Z). Shape: (height, width, samples).
    depth_points: Sampled depth values along each ray. Shape: (height, width, samples).

    Returns:
    rec_image: The reconstructed image after applying the volumetric rendering to every pixel.
    Shape: (height, width, 3)
    """

    ############################## TODO 2.4 BEGIN ##############################
    H, W, N = depth_points.shape
    s = nn.ReLU()(s)
    del_i = depth_points[:,:,1:] - depth_points[:,:,:-1] # (H,W,samples-1)
    del_i = del_i.to(device)
    s = s.to(device)
    T_i = torch.cumprod(torch.exp(-del_i*s[:,:,:-1]), axis=2)
    T_i = torch.cat([torch.ones(H,W,1).to(device), T_i], dim=-1)

    del_i = torch.cat([del_i, 1e9*torch.ones(H,W,1).to(device)], dim=-1) #(H,W,samples)

    rec_image = torch.sum(T_i.unsqueeze(-1)*(1.-torch.exp(-del_i*s)).unsqueeze(-1)*rgb.to(device), axis=2)

    return rec_image
```

2.5 Combine everything together. Given the pose position of a camera, compute the camera rays and sample the 3D points along these rays. Divide those points into batches and feed them to the neural network. Concatenate them and use them for the volumetric rendering to reconstructed the final image.

[11]:
```
def one_forward_pass(height, width, intrinsics, pose, near, far, samples, model, num_x_frequencies, num_d_frequencies):

    ############################## TODO 2.5 BEGIN ##############################

    #compute all the rays from the image
    Rcw = pose[:3,:3].T
    Tcw = pose[:3,-1]
    ray_origins, ray_directions = get_rays(height, width, intrinsics, Rcw, Tcw)

    #sample the points from the rays
```

```
    ray_points, depth_points = stratified_sampling(ray_origins, ray_directions,␣
↪near, far, samples)


    #divide data into batches to avoid memory errors
    ray_points_batches, ray_directions_batches = get_batches(ray_points,␣
↪ray_directions, num_x_frequencies, num_d_frequencies)
    # print(len(ray_points_batches))
    # raise

    #forward pass the batches and concatenate the outputs at the end
    fin_rgb = []
    fin_sig = []
    # for i in tqdm(range(len(ray_points_batches))):
    for i in range(len(ray_points_batches)):
      rgb, sigma = model(ray_points_batches[i], ray_directions_batches[i])
      fin_rgb.append(rgb)
      fin_sig.append(sigma)

    # a = torch.vstack(fin_rgb)
    a1 = torch.cat(fin_rgb, dim=0)
    # print(torch.sum(a.isclose(a1, 1e-2))- a.shape[0]*a.shape[1])
    b = torch.vstack(fin_sig)

    # Apply volumetric rendering to obtain the reconstructed image
    rec_image = volumetric_rendering(a1.reshape(height,width,samples,3),b.
↪reshape(height, width,samples),depth_points)
    # print("DONE 1 forward pass")
    ##############################  TODO 2.5 END  ##############################

    return rec_image
```

If you manage to pass the autograder for all the previous functions, then it is time to train a NeRF! We provide the hyperparameters for you, we initialize the NeRF model and its weights, and we define a couple lists that will be needed to store results.

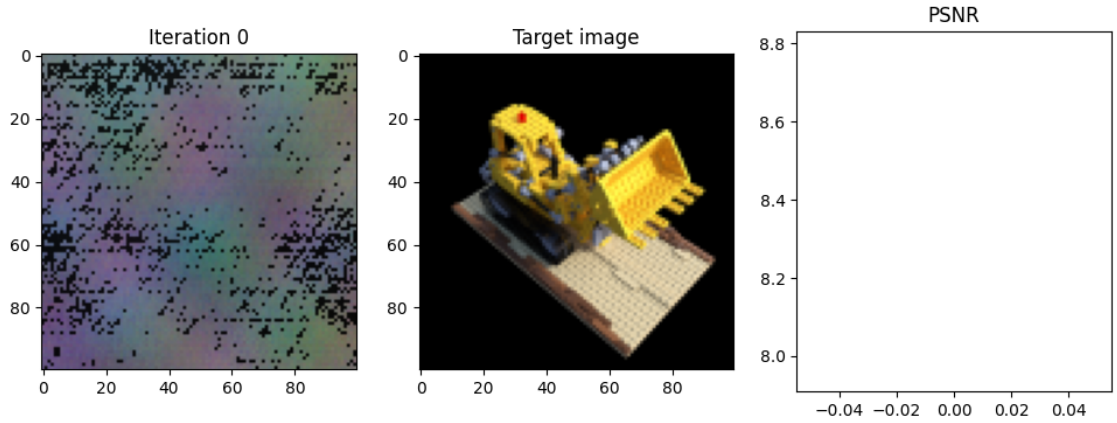
[12]:
```
num_x_frequencies = 10
num_d_frequencies = 4
learning_rate  = 5e-4
iterations = 3000
samples = 64
display = 25
near = 0.667
far = 2
```

```
model =␣
 ↪nerf_model(num_x_frequencies=num_x_frequencies,num_d_frequencies=num_d_frequencies).
 ↪to(device)

def weights_init(m):
    if isinstance(m, torch.nn.Linear):
        torch.nn.init.xavier_uniform_(m.weight)
model.apply(weights_init)

optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
criterion = nn.MSELoss()

psnrs = []
iternums = []

t = time.time()
t0 = time.time()
```

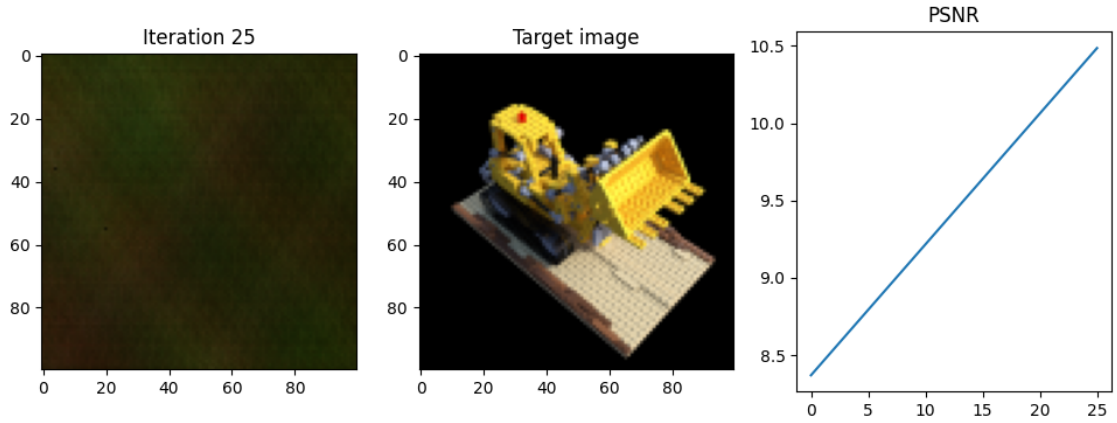
[13]:
```
from matplotlib import image
for i in range(iterations+1):

    ##############################  TODO 2.6 BEGIN  ##############################

    #choose randomly a picture for the forward pass
    # torch.cuda.empty_cache()
    pic_idx = np.random.randint(images.shape[0])
    random_image, random_pose = images[pic_idx], poses[pic_idx]

    # Run one iteration of NeRF and get the rendered RGB image.
    optimizer.zero_grad()
    rec_img = one_forward_pass(height, width, intrinsics, random_pose, near,␣
 ↪far, samples, model, num_x_frequencies, num_d_frequencies)


    # Compute mean-squared error between the predicted and target images.␣
 ↪Backprop!
    loss = criterion(rec_img, random_image)
    # print("Loss: ",loss)
    loss.backward()

    optimizer.step()


    ##############################  TODO 2.6 END  ##############################

    # Display images/plots/stats
    if i % display == 0:
```

```
        with torch.no_grad():
        ############################  TODO 2.6 BEGIN 
↪############################
            # Render the held-out view
            test_rec_image = one_forward_pass(height, width, intrinsics,
↪test_pose, near, far, samples, model, num_x_frequencies, num_d_frequencies)
            loss = nn.MSELoss()(test_rec_image, test_image)

        #calculate the loss and the psnr between the original test image and
↪the reconstructed one.
        psnr = 10*torch.log10(1/loss)

        ############################  TODO 2.6 END 
↪############################

        print("Iteration %d " % i, "Loss: %.4f " % loss.item(), "PSNR: %.2f " %
↪psnr.item(), \
                "Time: %.2f secs per iter, " % ((time.time() - t) / display),
↪"%.2f mins in total" % ((time.time() - t0)/60))

        t = time.time()
        psnrs.append(psnr.item())
        iternums.append(i)

        plt.figure(figsize=(16, 4))
        plt.subplot(141)
        plt.imshow(test_rec_image.detach().cpu().numpy())
        plt.title(f"Iteration {i}")
        plt.subplot(142)
        plt.imshow(test_image.detach().cpu().numpy())
        plt.title("Target image")
        plt.subplot(143)
        plt.plot(iternums, psnrs)
        plt.title("PSNR")
        plt.show()

print('Done!')
```

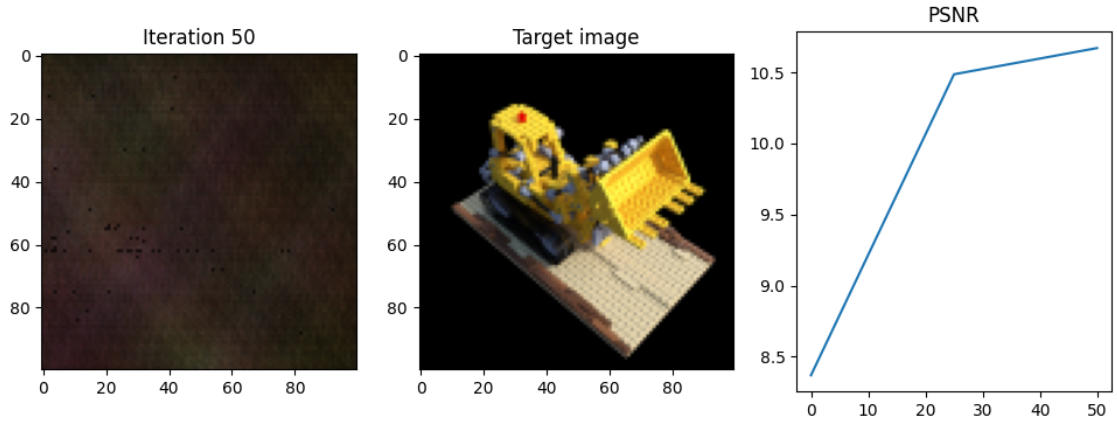
```
Iteration 0  Loss: 0.1456  PSNR: 8.37  Time: 0.70 secs per iter,  0.29 mins in
total
```



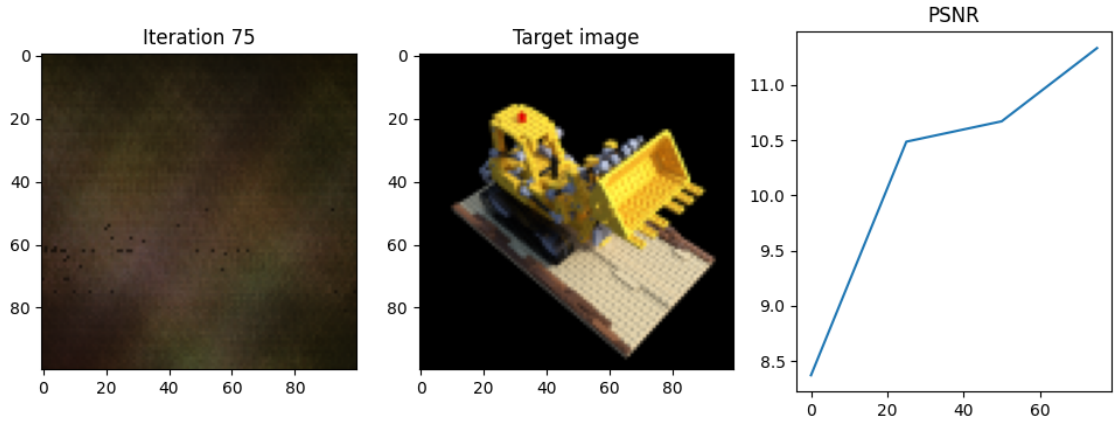

```
Iteration 25  Loss: 0.0894  PSNR: 10.49  Time: 0.79 secs per iter,  0.62 mins in
total
```



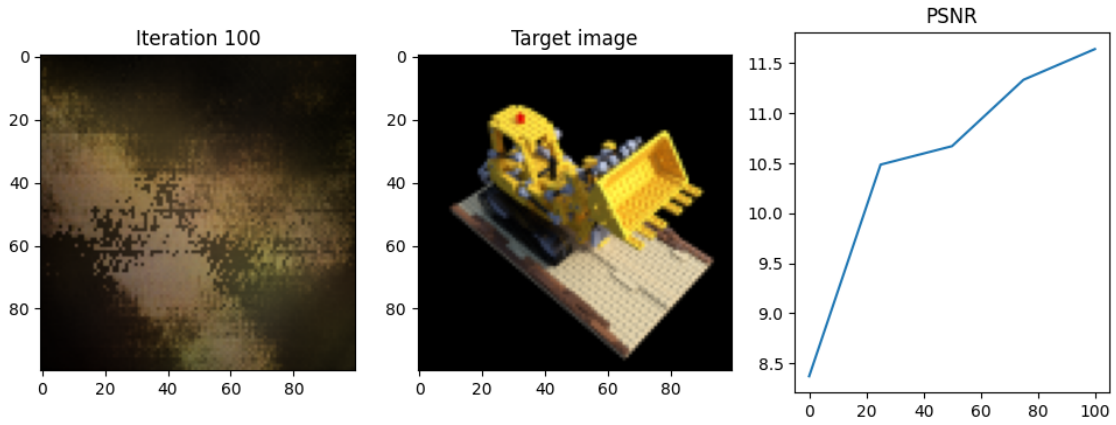

```
Iteration 50  Loss: 0.0857  PSNR: 10.67  Time: 0.80 secs per iter,  0.95 mins in
total
```



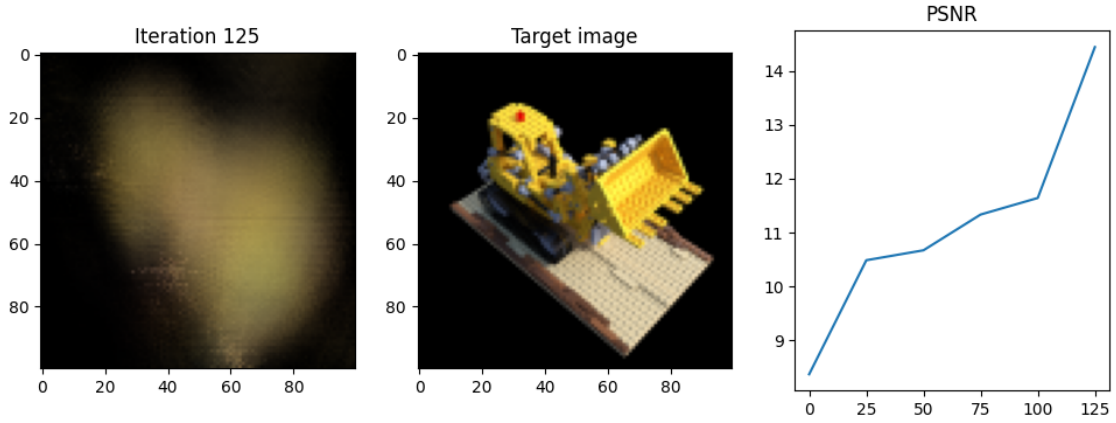

```
Iteration 75  Loss: 0.0735  PSNR: 11.33  Time: 0.80 secs per iter,  1.29 mins in
total
```



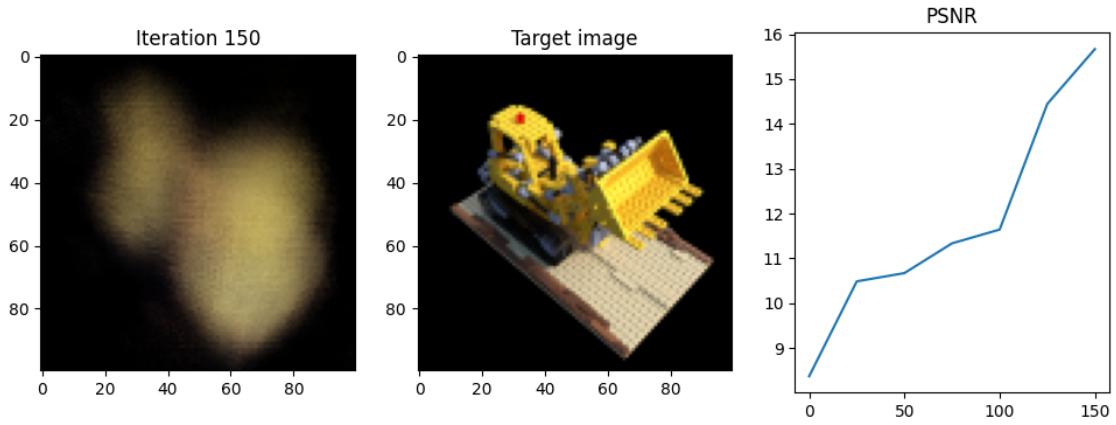

```
Iteration 100  Loss: 0.0685  PSNR: 11.64  Time: 0.82 secs per iter,  1.63 mins
in total
```



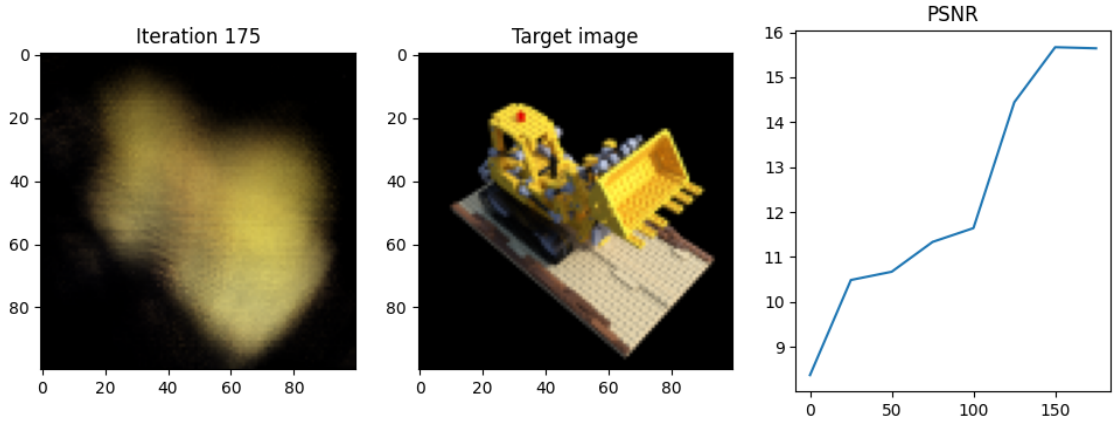
Iteration 125 Loss: 0.0359 PSNR: 14.44 Time: 0.82 secs per iter, 1.97 mins in total



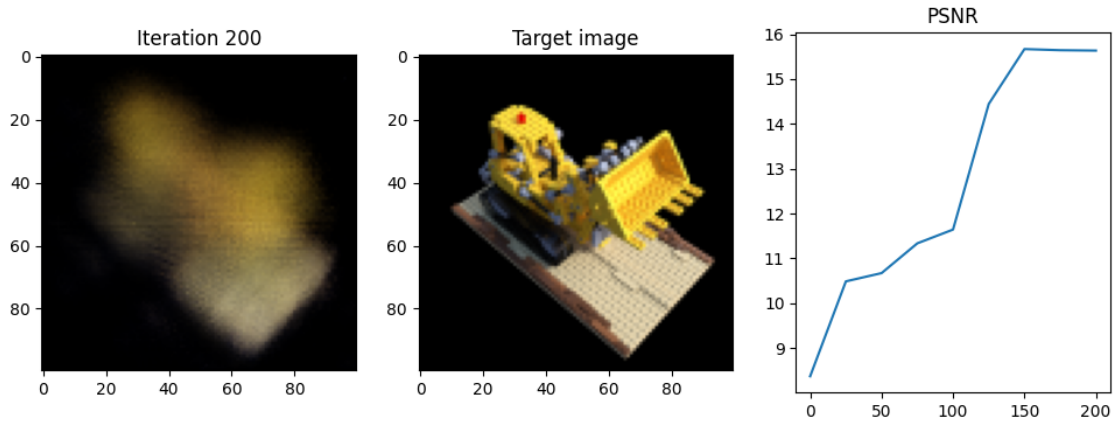
Iteration 150 Loss: 0.0271 PSNR: 15.67 Time: 0.81 secs per iter, 2.31 mins in total



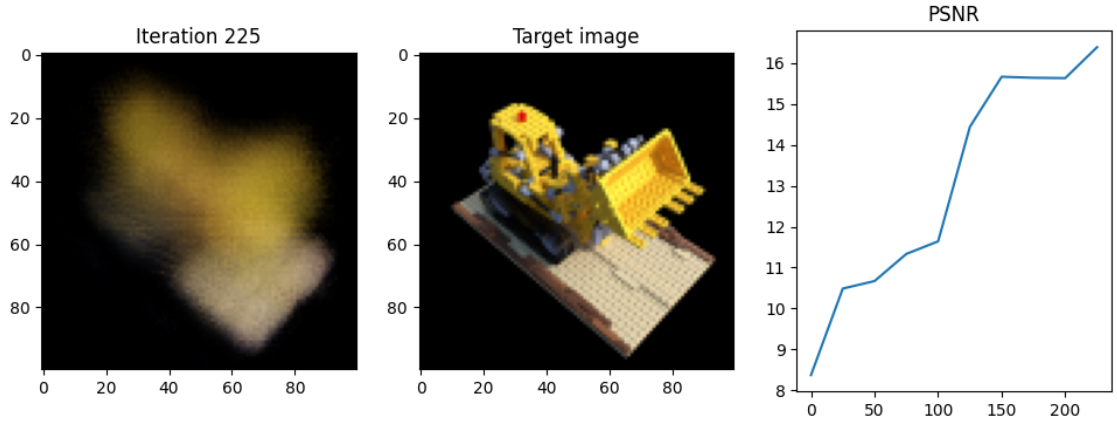

```
Iteration 175  Loss: 0.0273  PSNR: 15.64  Time: 0.81 secs per iter,  2.64 mins
in total
```



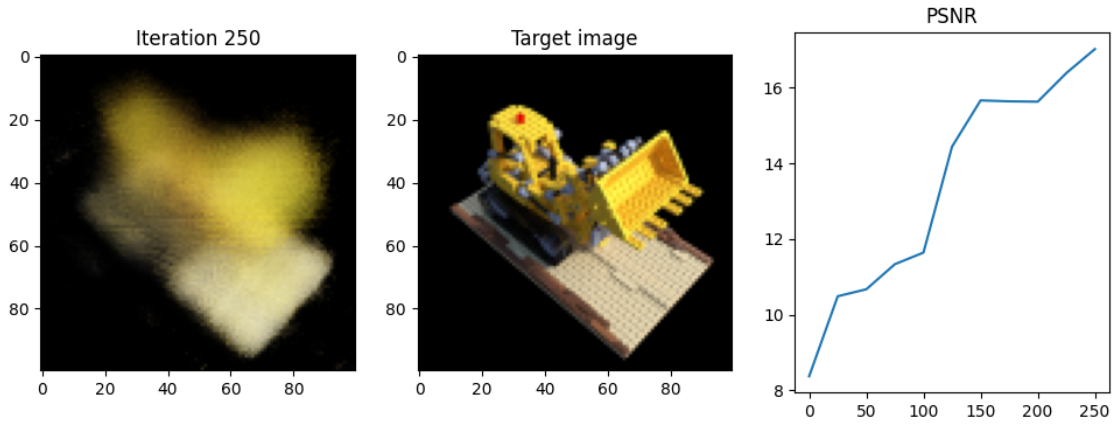

```
Iteration 200  Loss: 0.0273  PSNR: 15.63  Time: 0.82 secs per iter,  2.99 mins
in total
```



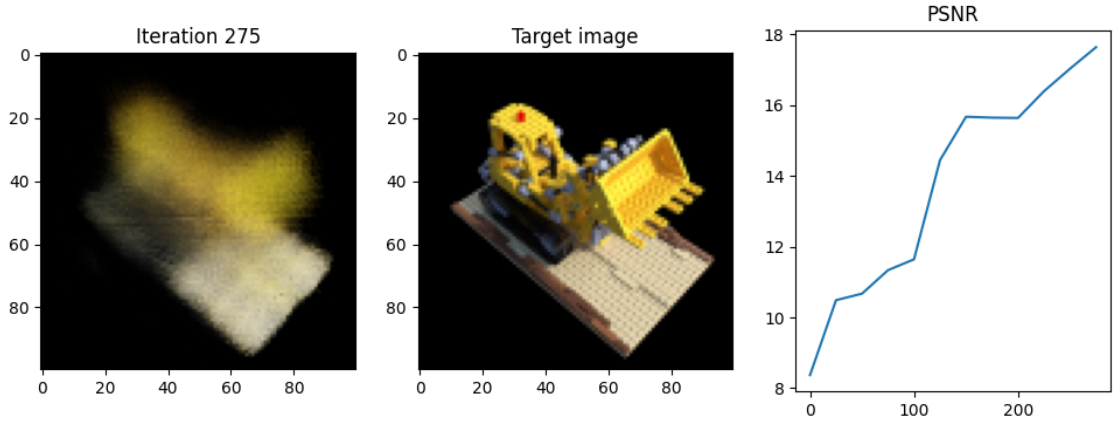

```
Iteration 225  Loss: 0.0230  PSNR: 16.39  Time: 0.81 secs per iter,  3.32 mins
in total
```



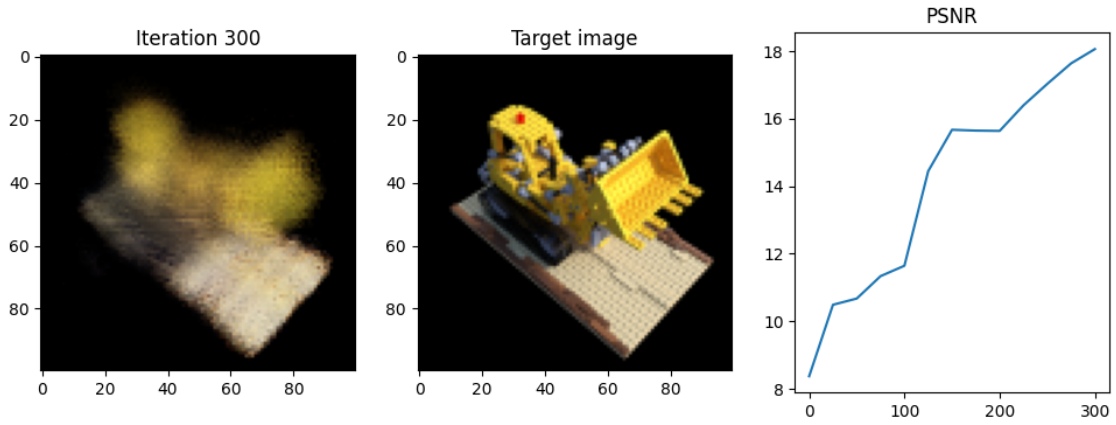

```
Iteration 250  Loss: 0.0198  PSNR: 17.03  Time: 0.82 secs per iter,  3.67 mins
in total
```



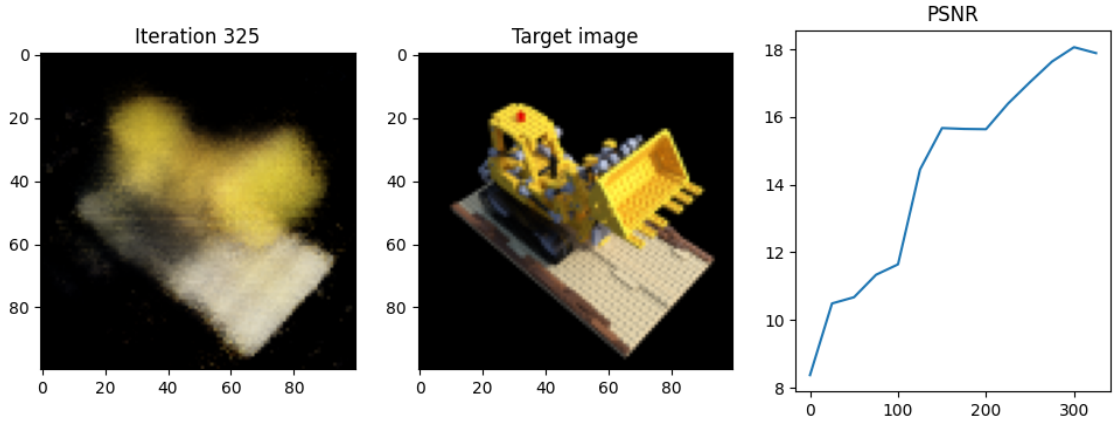

```
Iteration 275  Loss: 0.0172  PSNR: 17.63  Time: 0.82 secs per iter,  4.01 mins
in total
```



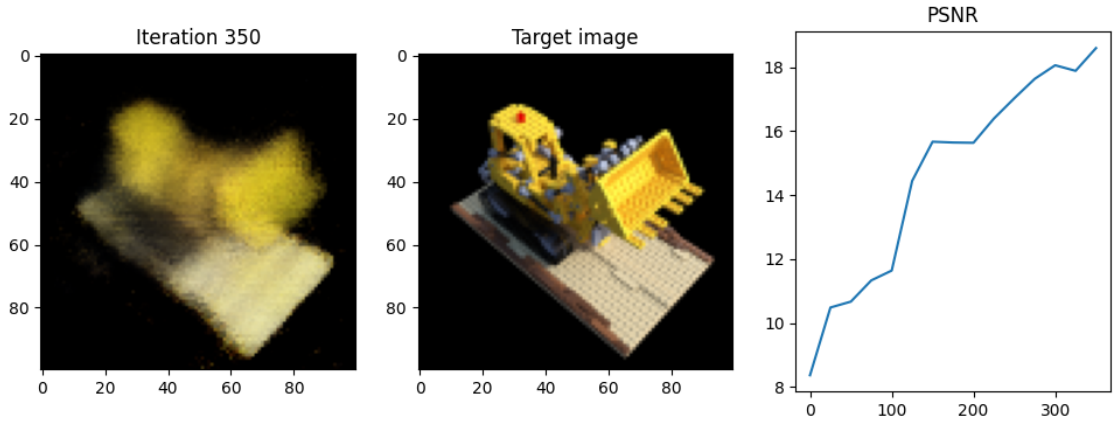

```
Iteration 300  Loss: 0.0156  PSNR: 18.06  Time: 0.83 secs per iter,  4.35 mins
in total
```



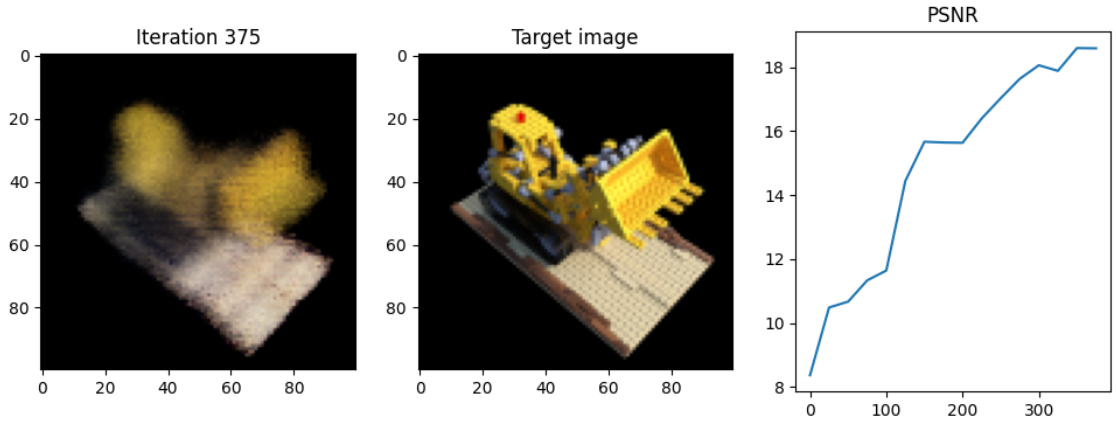

```
Iteration 325  Loss: 0.0163  PSNR: 17.88  Time: 0.83 secs per iter,  4.70 mins
in total
```



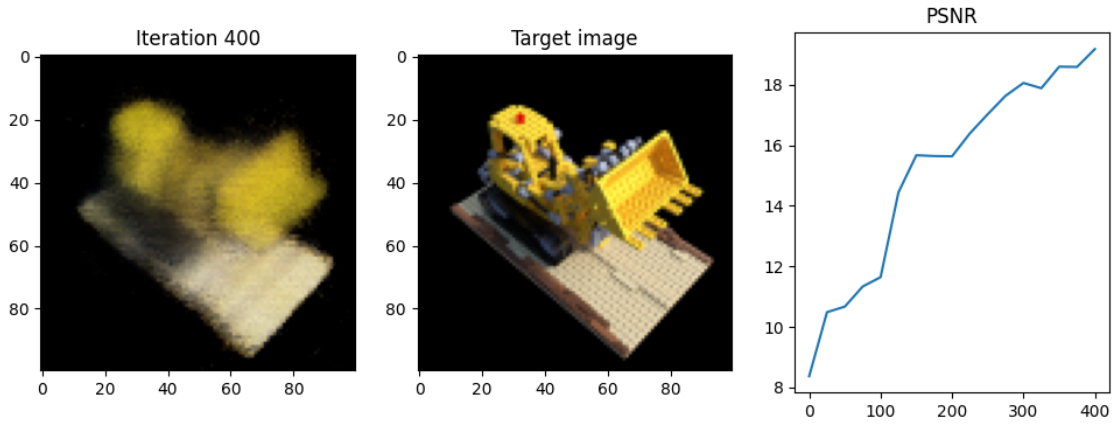

```
Iteration 350  Loss: 0.0138  PSNR: 18.60  Time: 0.82 secs per iter,  5.04 mins
in total
```



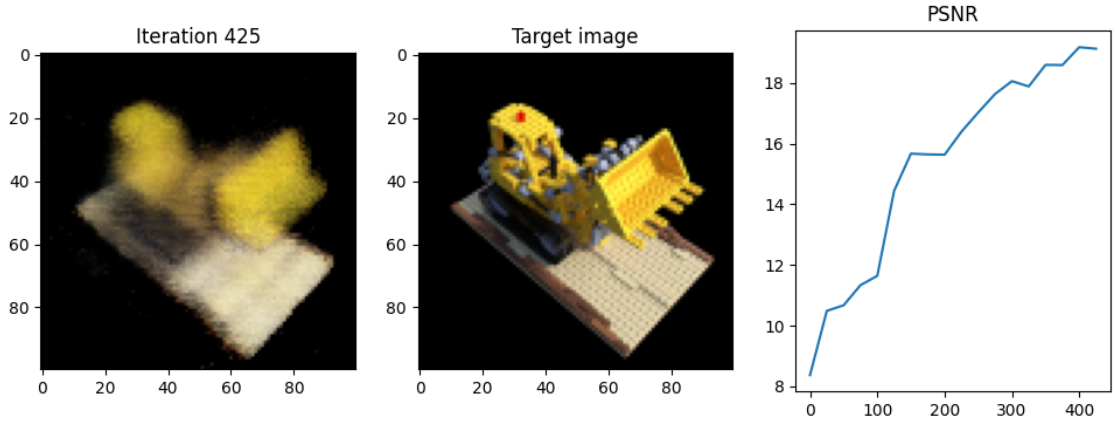

```
Iteration 375  Loss: 0.0138  PSNR: 18.59  Time: 0.82 secs per iter,  5.38 mins
in total
```



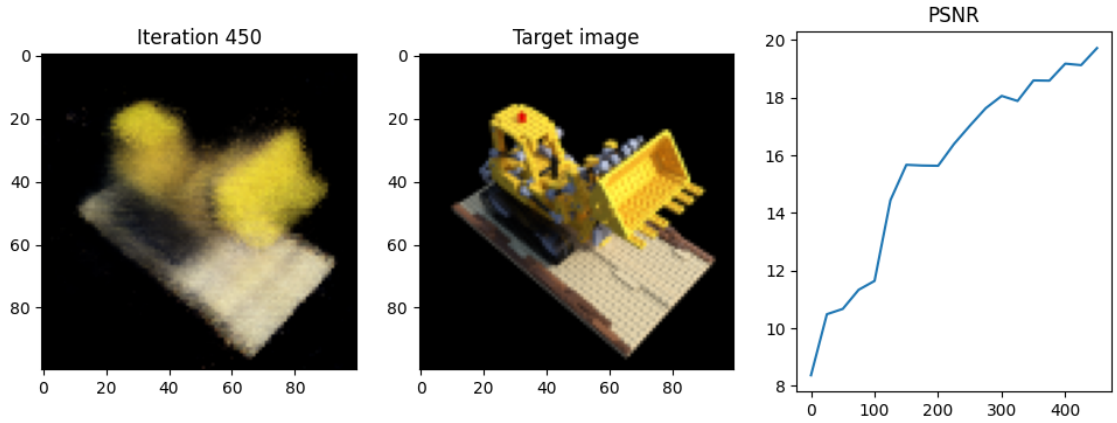

```
Iteration 400  Loss: 0.0121  PSNR: 19.18  Time: 0.82 secs per iter,  5.72 mins
in total
```



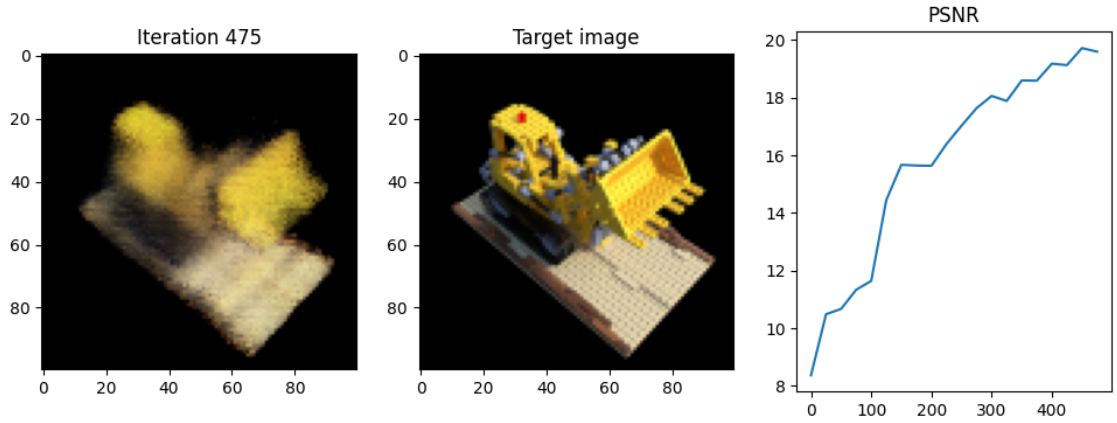

```
Iteration 425  Loss: 0.0122  PSNR: 19.13  Time: 0.82 secs per iter,  6.06 mins
in total
```



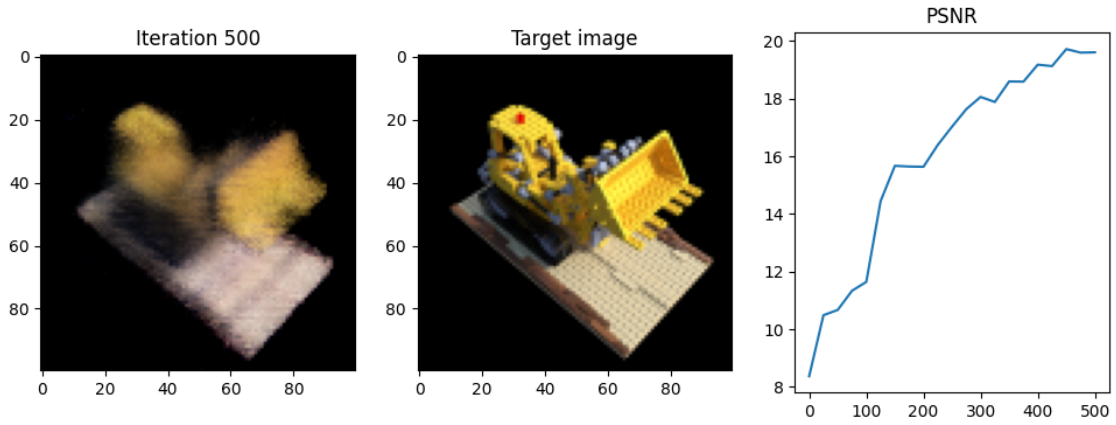

```
Iteration 450  Loss: 0.0107  PSNR: 19.72  Time: 0.81 secs per iter,  6.40 mins
in total
```



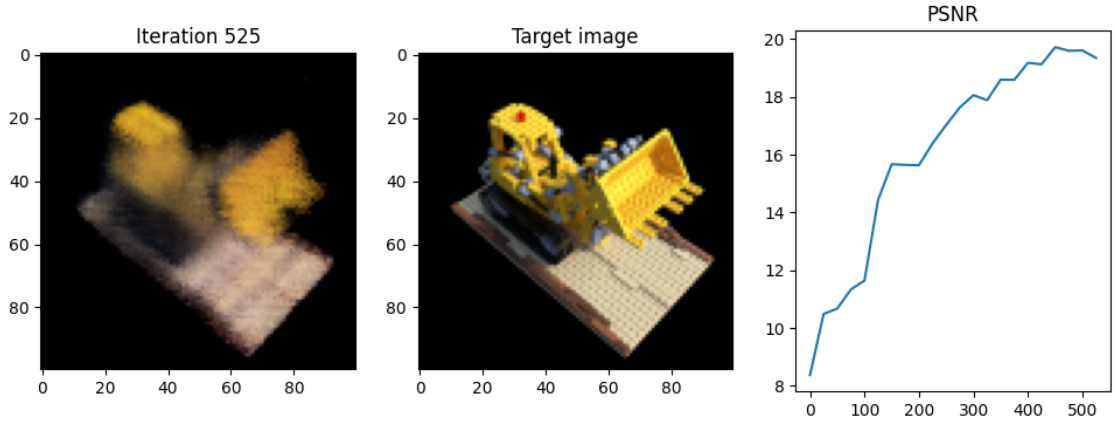

```
Iteration 475  Loss: 0.0110  PSNR: 19.59  Time: 0.82 secs per iter,  6.74 mins
in total
```



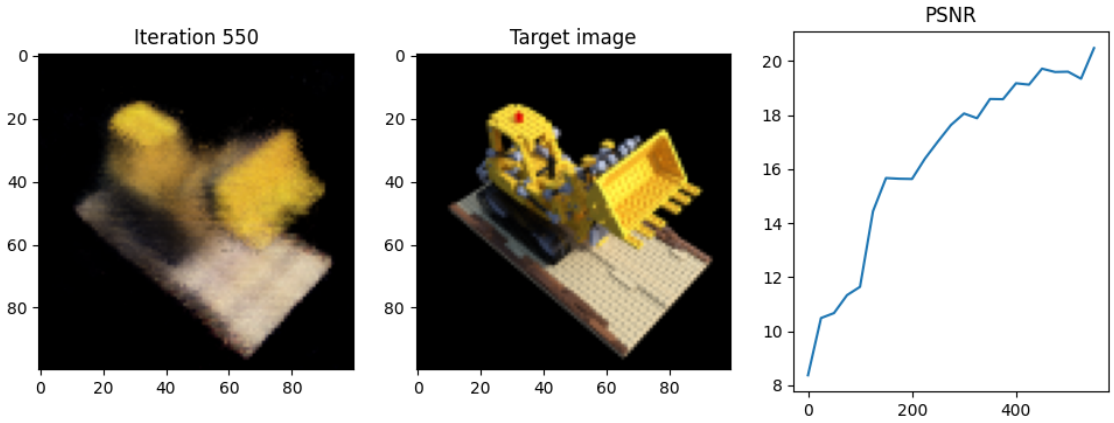

```
Iteration 500  Loss: 0.0110  PSNR: 19.61  Time: 0.81 secs per iter,  7.08 mins
in total
```



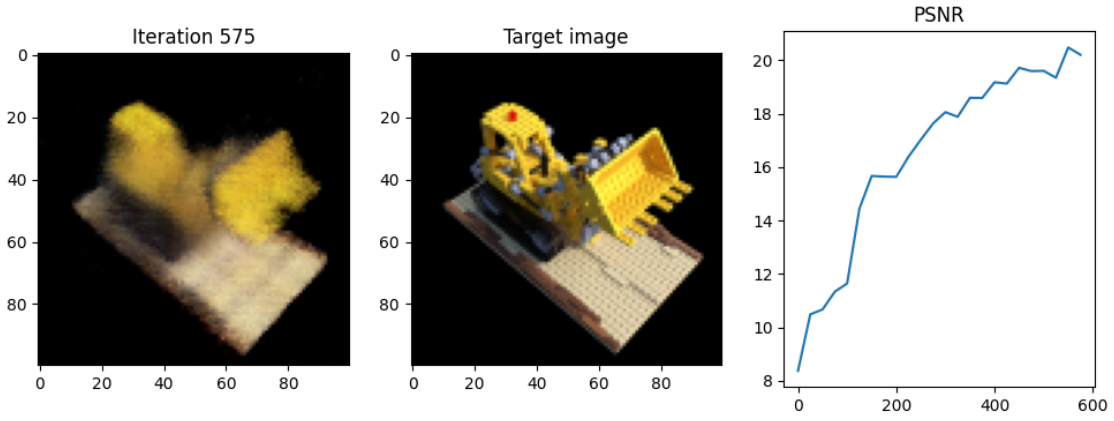

```
Iteration 525  Loss: 0.0116  PSNR: 19.35  Time: 0.82 secs per iter,  7.43 mins
in total
```



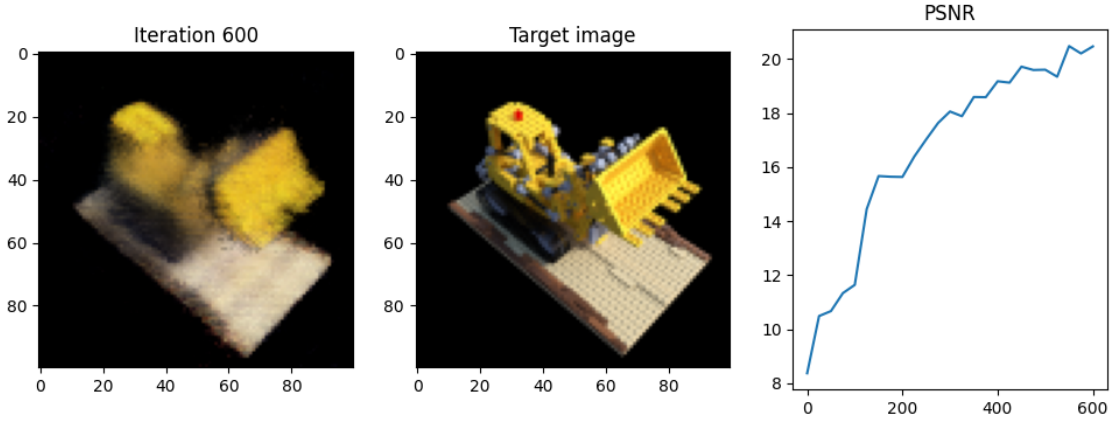

```
Iteration 550  Loss: 0.0090  PSNR: 20.48  Time: 0.82 secs per iter,  7.77 mins
in total
```



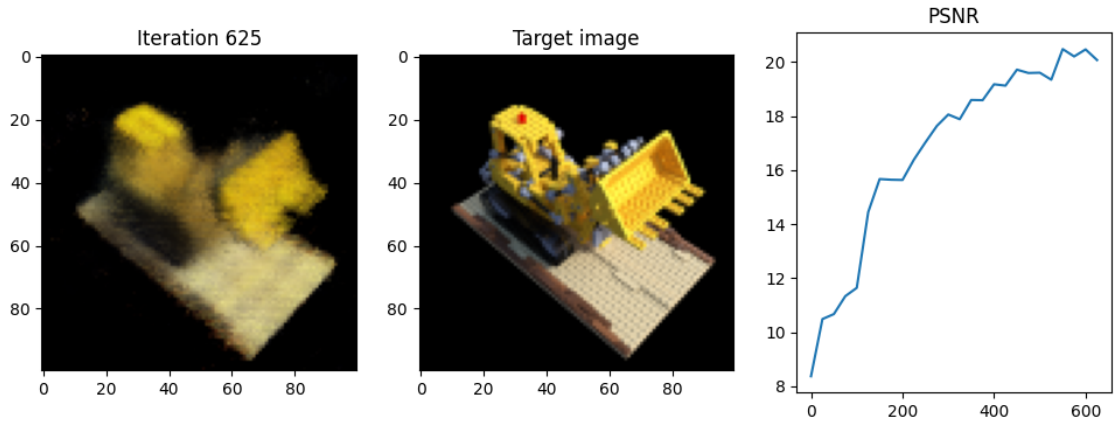

```
Iteration 575  Loss: 0.0095  PSNR: 20.21  Time: 0.82 secs per iter,  8.11 mins
in total
```



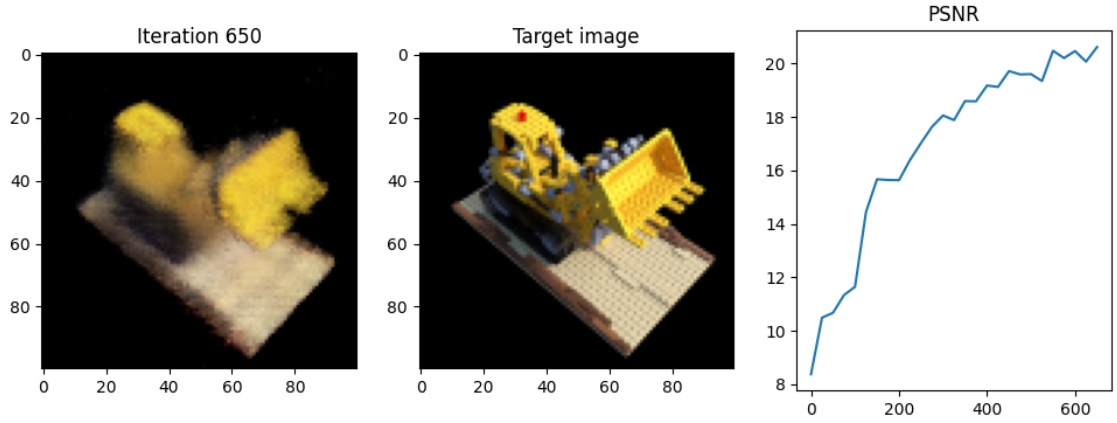

```
Iteration 600  Loss: 0.0090  PSNR: 20.47  Time: 0.81 secs per iter,  8.45 mins
in total
```



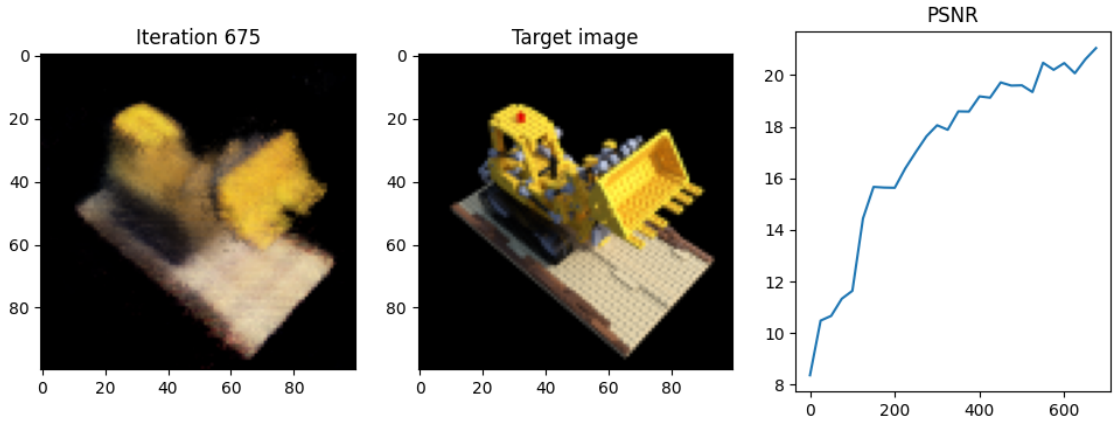
```
Iteration 625  Loss: 0.0098  PSNR: 20.07  Time: 0.81 secs per iter,  8.79 mins
in total
```



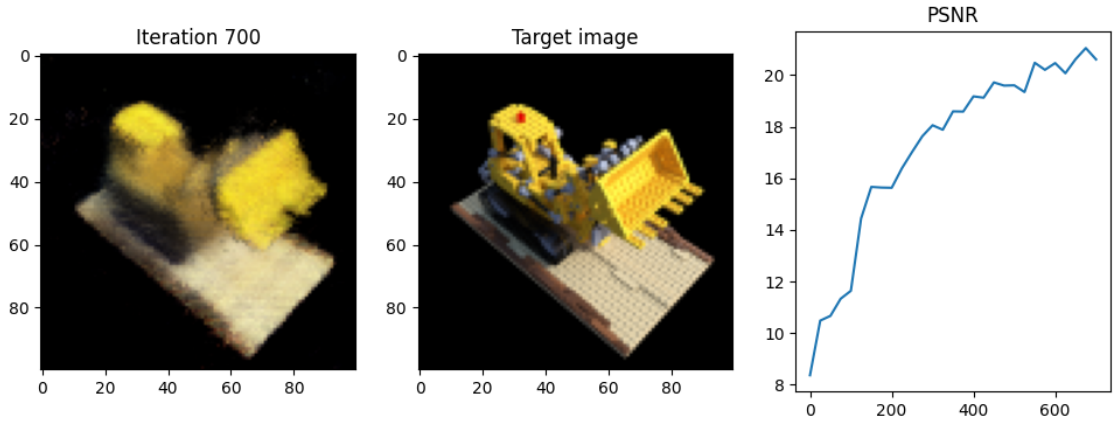

```
Iteration 650  Loss: 0.0087  PSNR: 20.62  Time: 0.81 secs per iter,  9.12 mins
in total
```



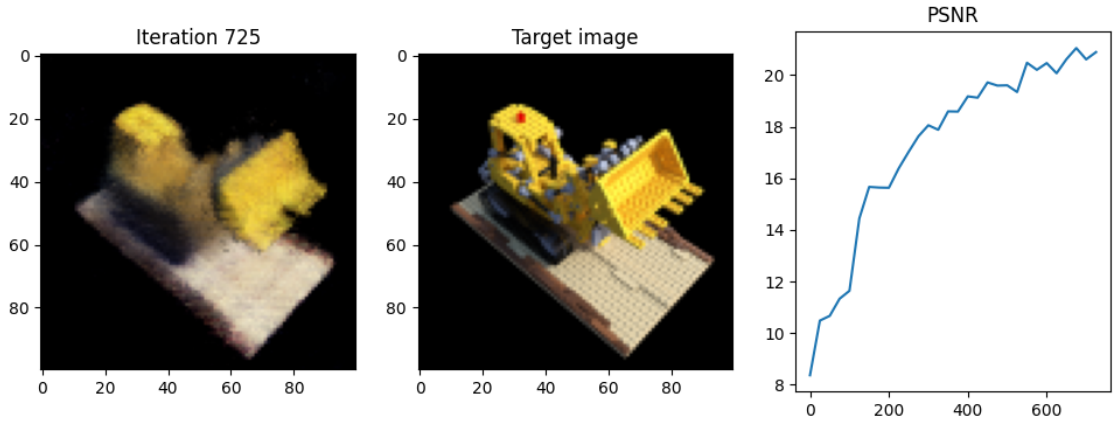

```
Iteration 675  Loss: 0.0079  PSNR: 21.05  Time: 0.81 secs per iter,  9.46 mins
in total
```



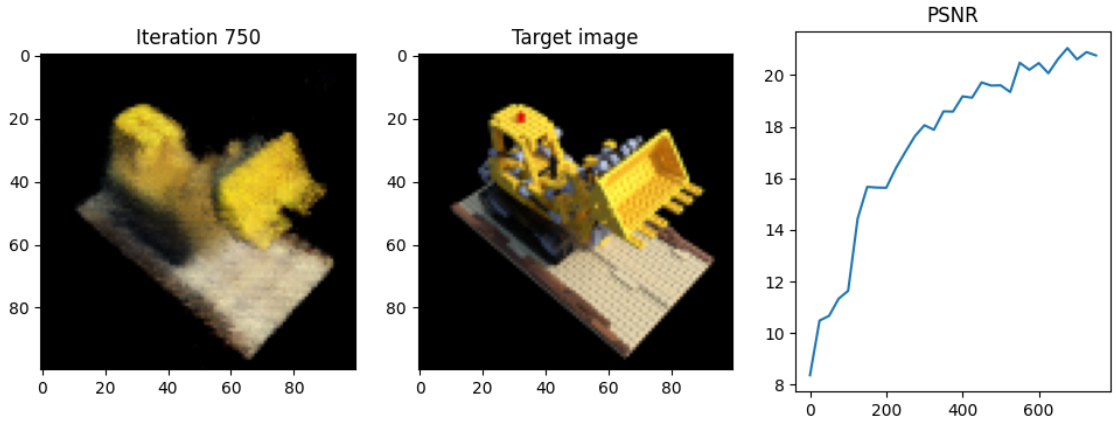

```
Iteration 700  Loss: 0.0087  PSNR: 20.61  Time: 0.81 secs per iter,  9.80 mins
in total
```



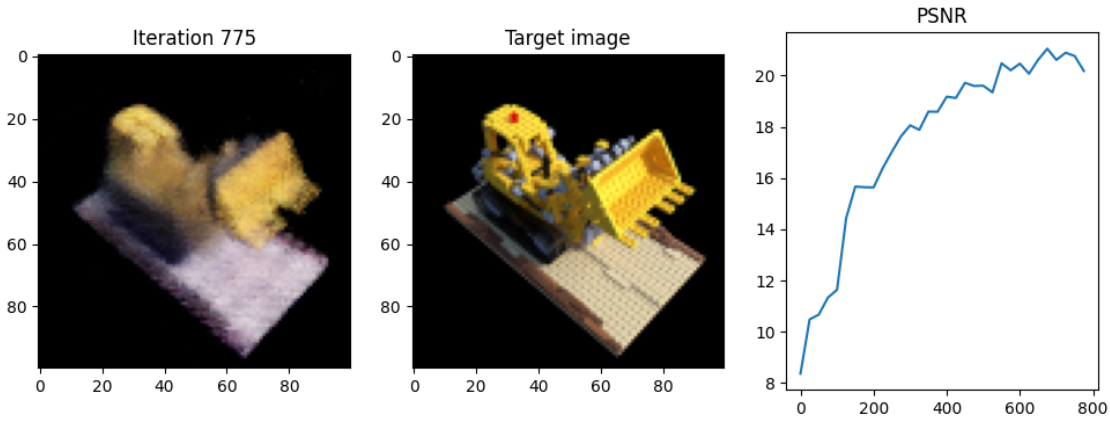

```
Iteration 725  Loss: 0.0081  PSNR: 20.89  Time: 0.82 secs per iter,  10.14 mins
in total
```



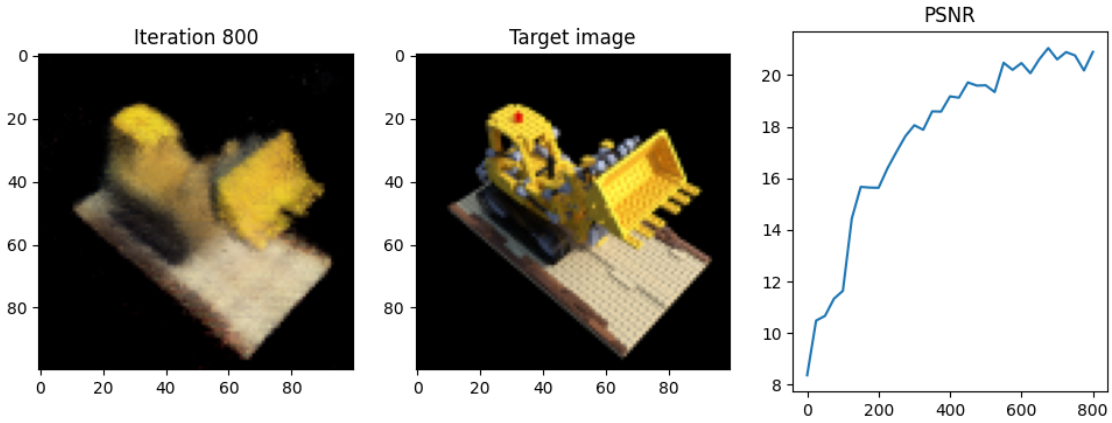

```
Iteration 750  Loss: 0.0084  PSNR: 20.76  Time: 0.81 secs per iter,  10.48 mins
in total
```



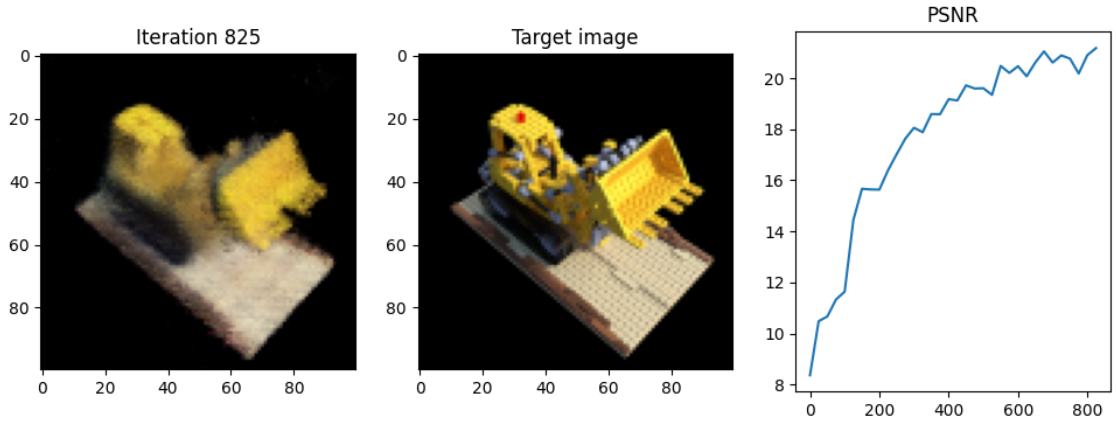

```
Iteration 775  Loss: 0.0096  PSNR: 20.18  Time: 0.81 secs per iter,  10.82 mins
in total
```



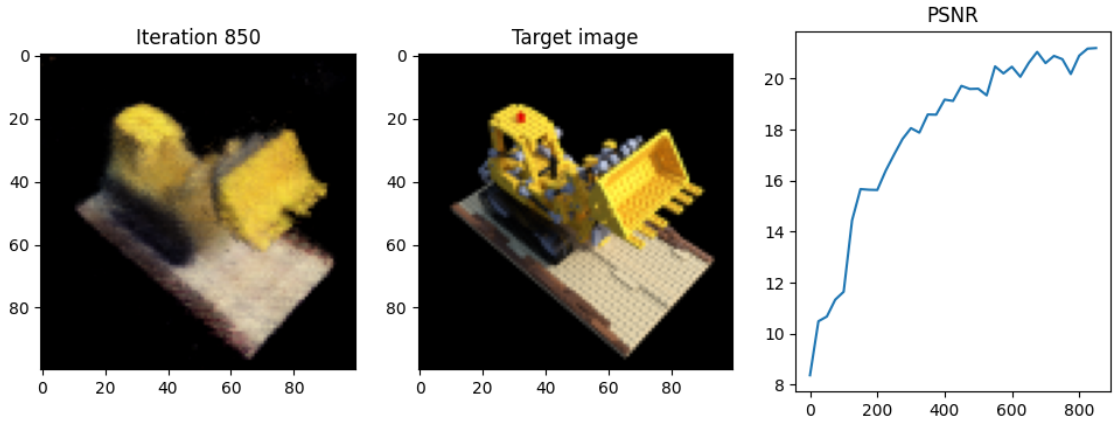

```
Iteration 800  Loss: 0.0081  PSNR: 20.90  Time: 0.81 secs per iter,  11.16 mins
in total
```



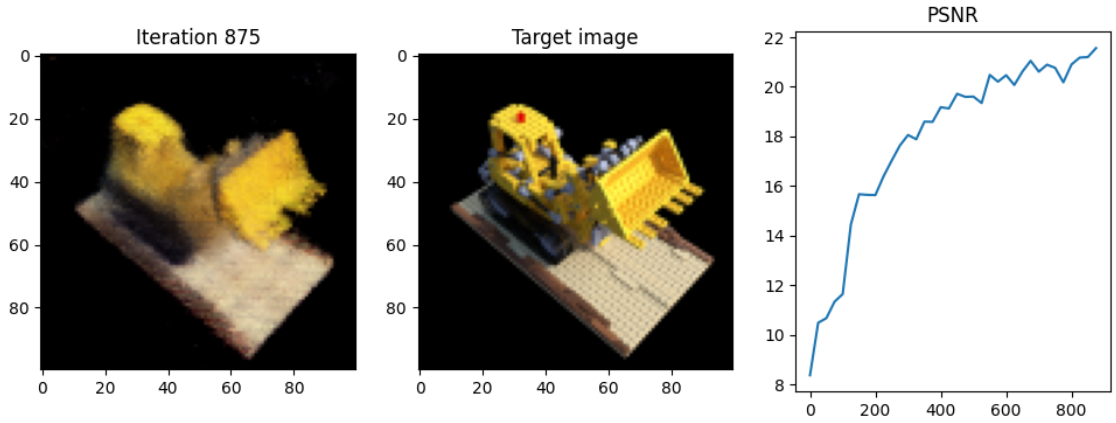

```
Iteration 825  Loss: 0.0076  PSNR: 21.18  Time: 0.81 secs per iter,  11.50 mins
in total
```



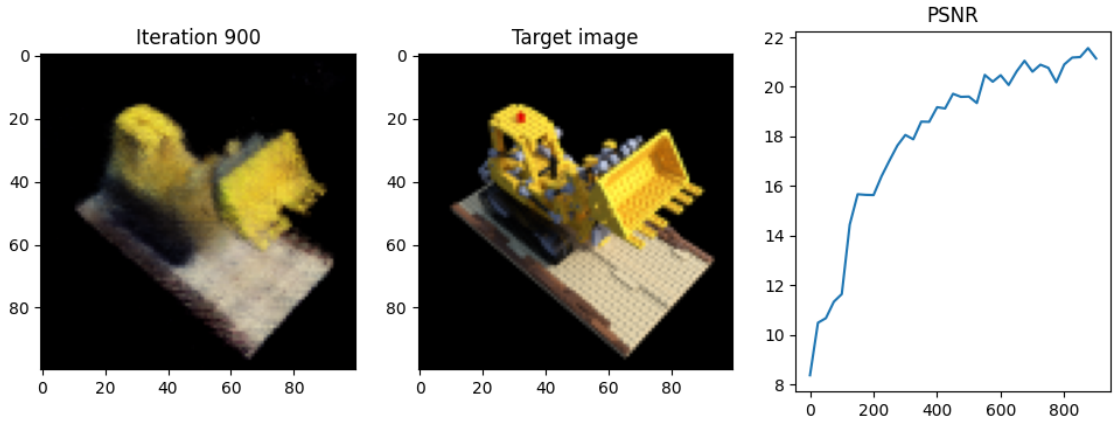

```
Iteration 850  Loss: 0.0076  PSNR: 21.20  Time: 0.81 secs per iter,  11.83 mins
in total
```



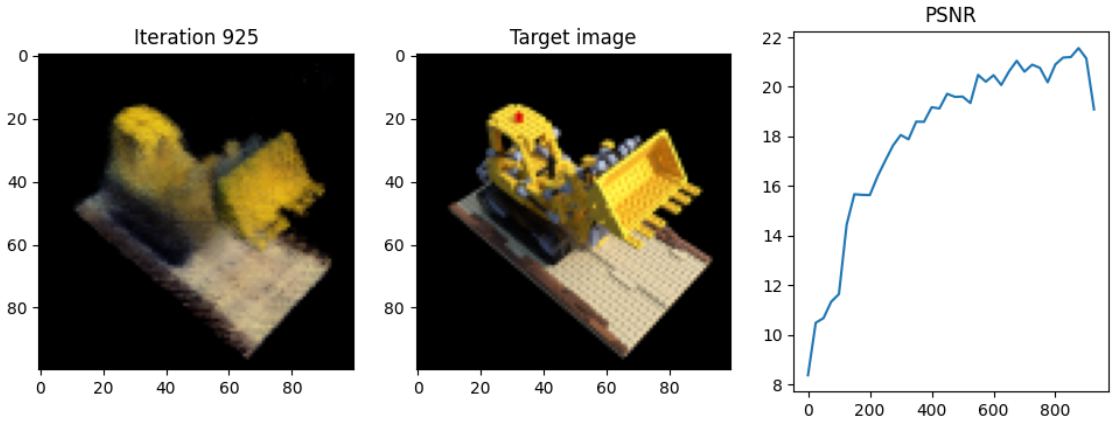

```
Iteration 875  Loss: 0.0070  PSNR: 21.56  Time: 0.81 secs per iter,  12.17 mins
in total
```



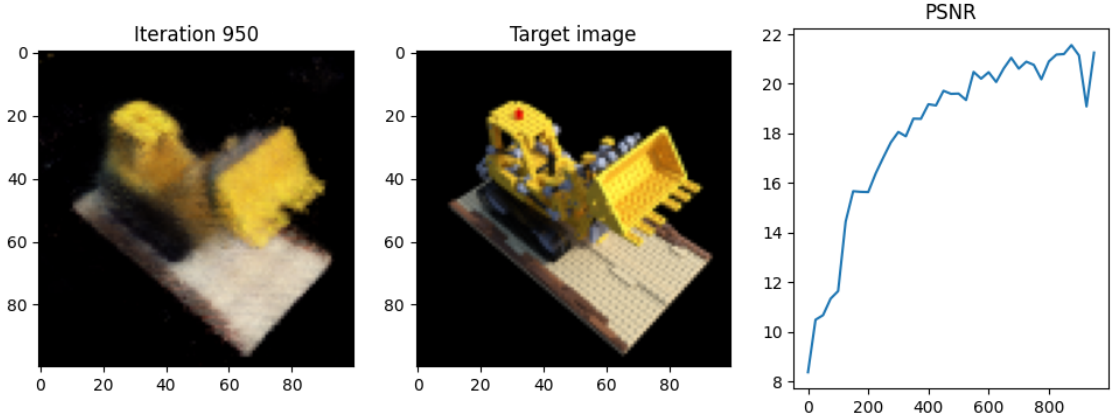

```
Iteration 900  Loss: 0.0077  PSNR: 21.14  Time: 0.82 secs per iter,  12.51 mins
in total
```



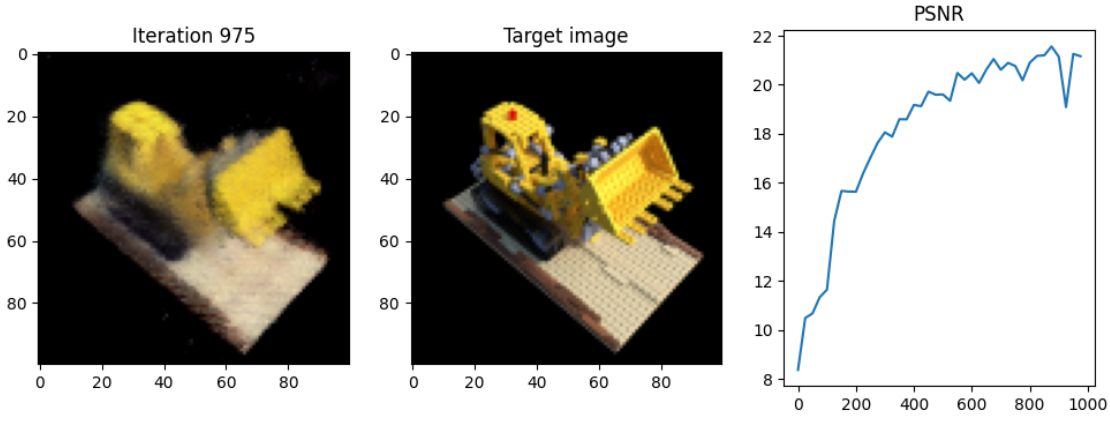

```
Iteration 925  Loss: 0.0123  PSNR: 19.09  Time: 0.81 secs per iter,  12.85 mins
in total
```



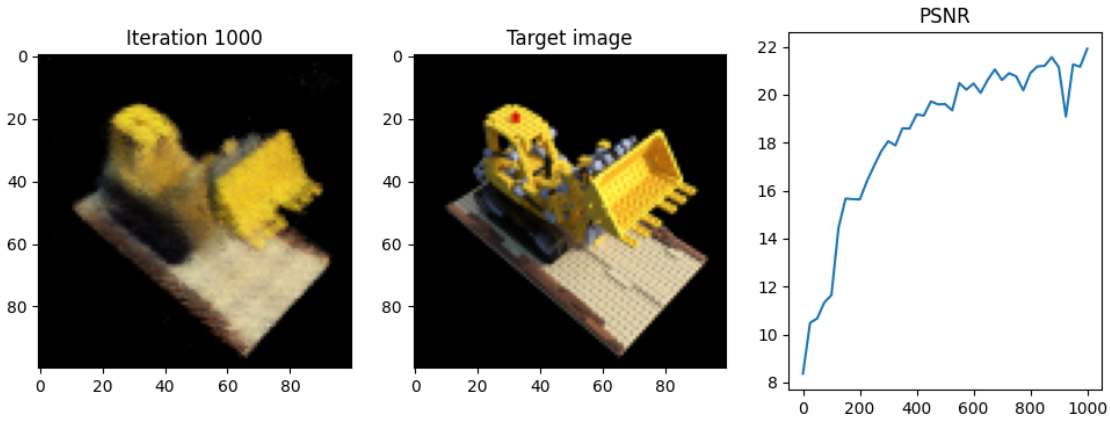

```
Iteration 950  Loss: 0.0075  PSNR: 21.26  Time: 0.81 secs per iter,  13.19 mins
in total
```



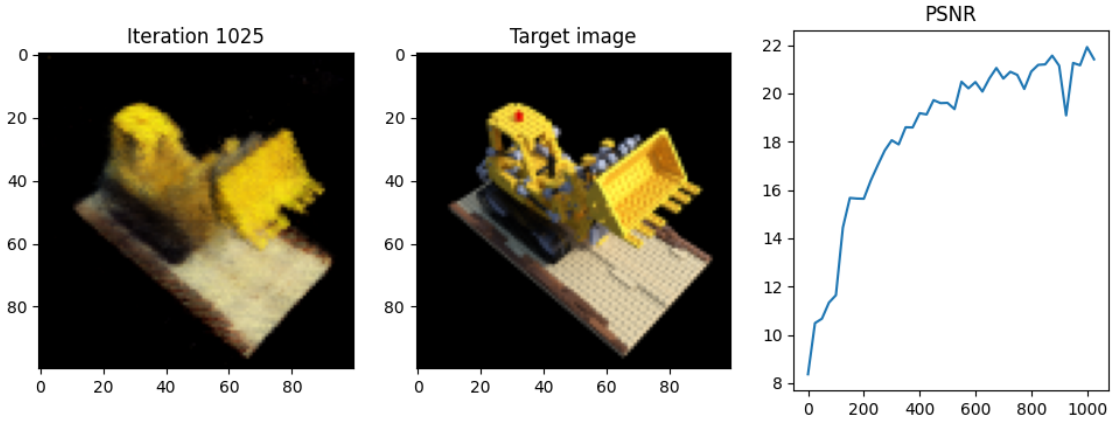

```
Iteration 975  Loss: 0.0077  PSNR: 21.16  Time: 0.82 secs per iter,  13.53 mins
in total
```



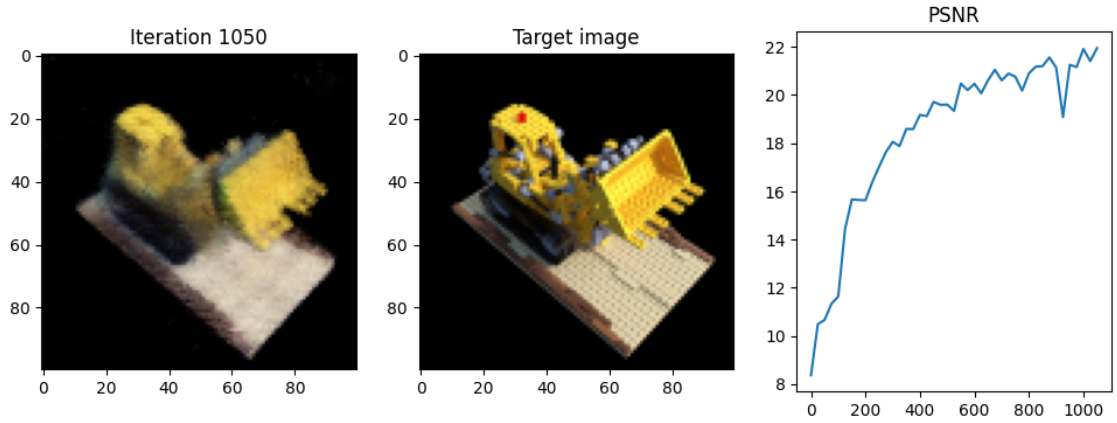
```
Iteration 1000  Loss: 0.0064  PSNR: 21.91  Time: 0.81 secs per iter,  13.87 mins
in total
```



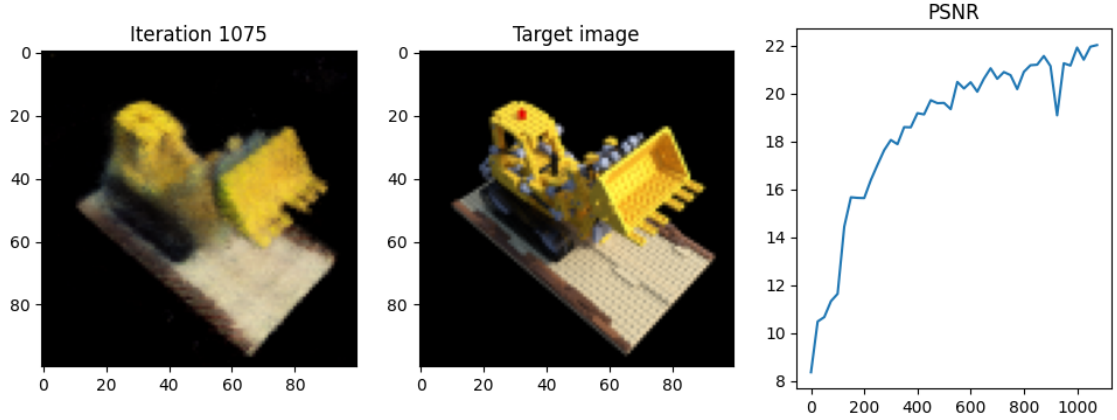

```
Iteration 1025  Loss: 0.0072  PSNR: 21.41  Time: 0.81 secs per iter,  14.20 mins
in total
```



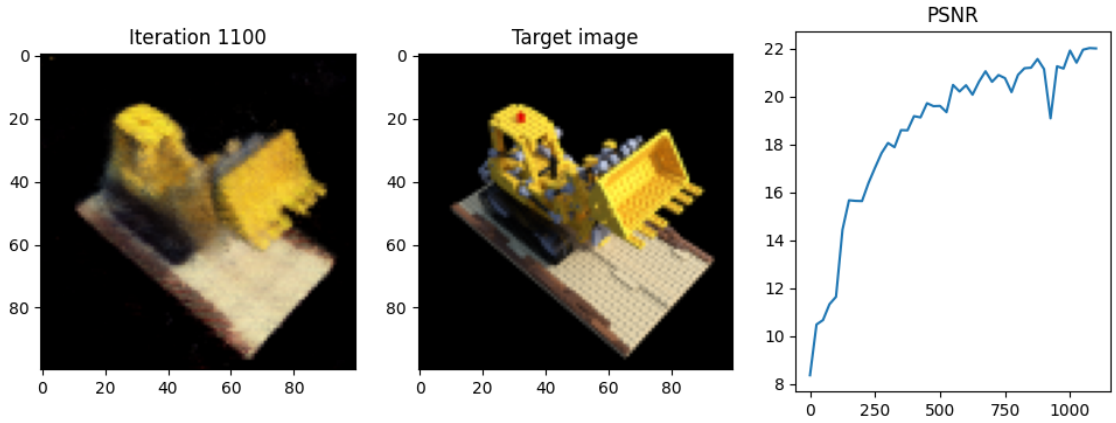

```
Iteration 1050  Loss: 0.0064  PSNR: 21.95  Time: 0.82 secs per iter,  14.55 mins
in total
```



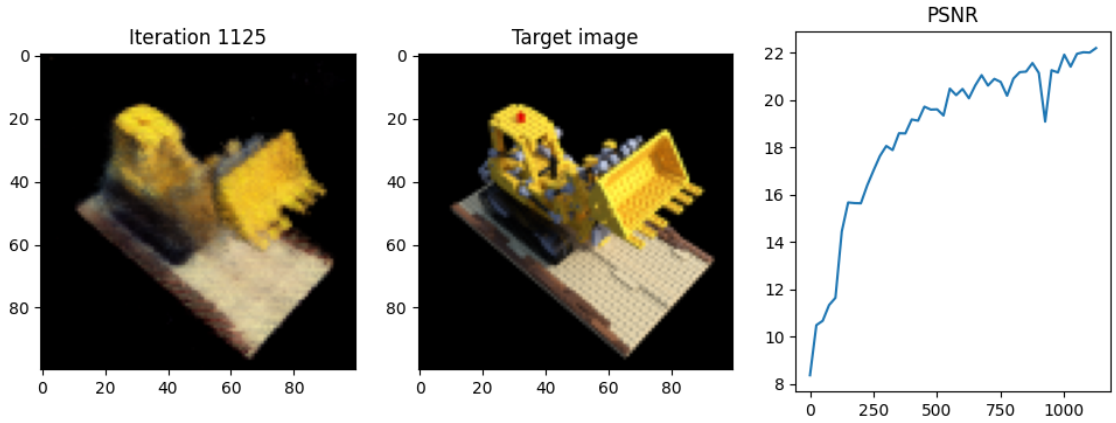

```
Iteration 1075  Loss: 0.0063  PSNR: 22.02  Time: 0.81 secs per iter,  14.88 mins
in total
```



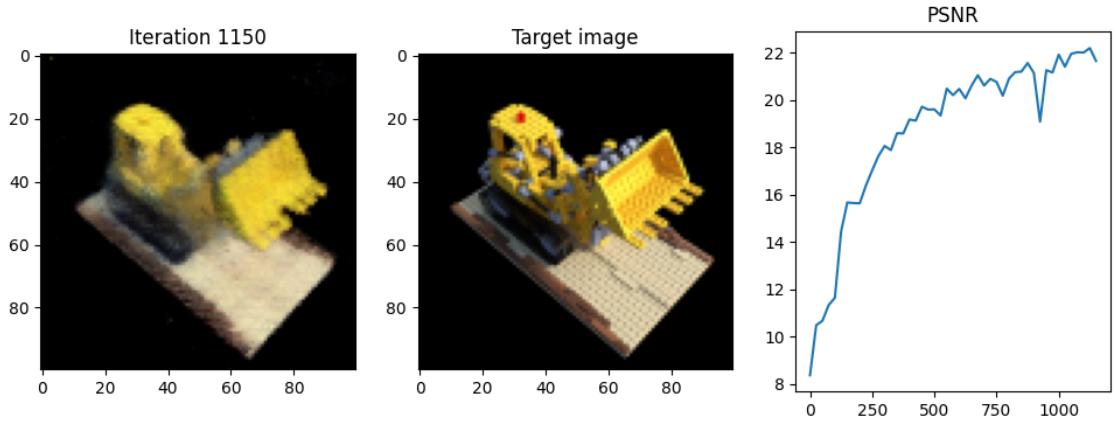

```
Iteration 1100  Loss: 0.0063  PSNR: 22.00  Time: 0.82 secs per iter,  15.22 mins
in total
```



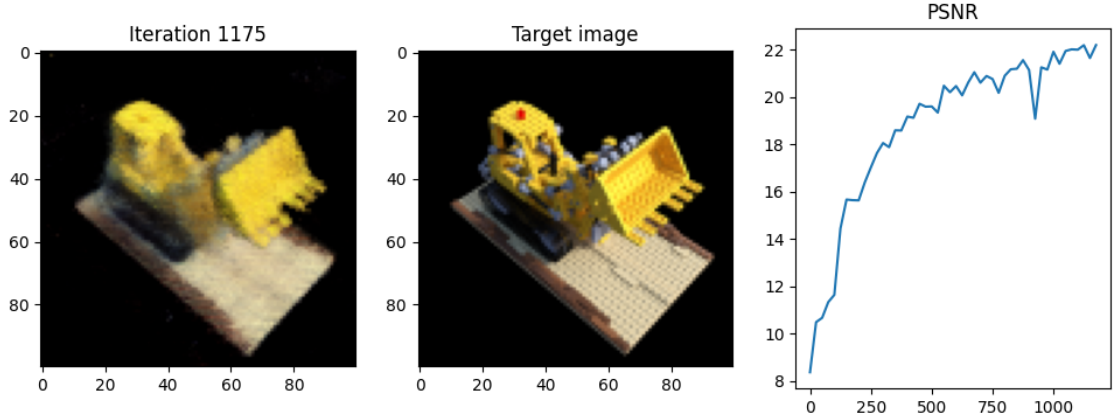

```
Iteration 1125  Loss: 0.0060  PSNR: 22.19  Time: 0.82 secs per iter,  15.56 mins
in total
```



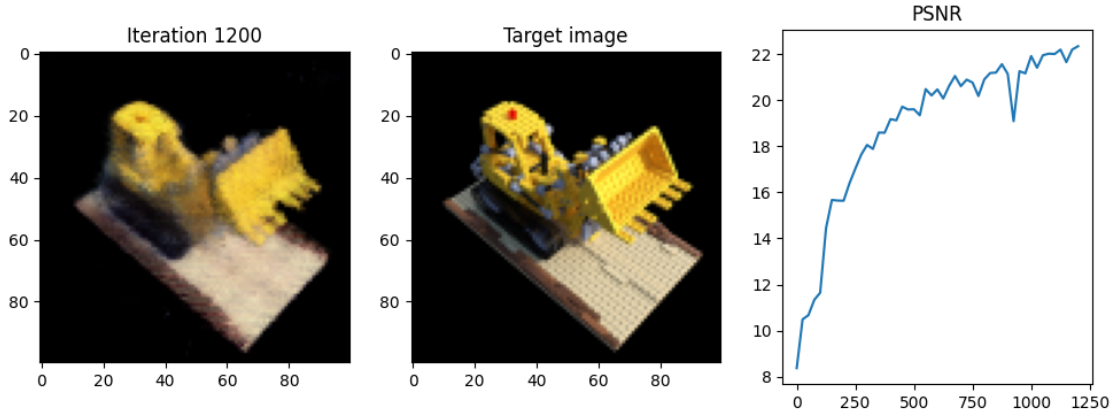

```
Iteration 1150  Loss: 0.0068  PSNR: 21.65  Time: 0.81 secs per iter,  15.90 mins
in total
```



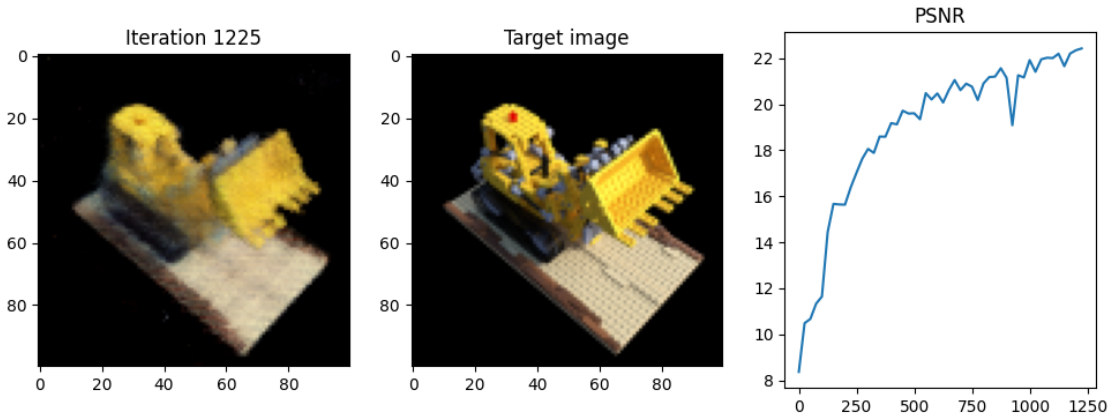

```
Iteration 1175  Loss: 0.0060  PSNR: 22.20  Time: 0.81 secs per iter,  16.24 mins
in total
```



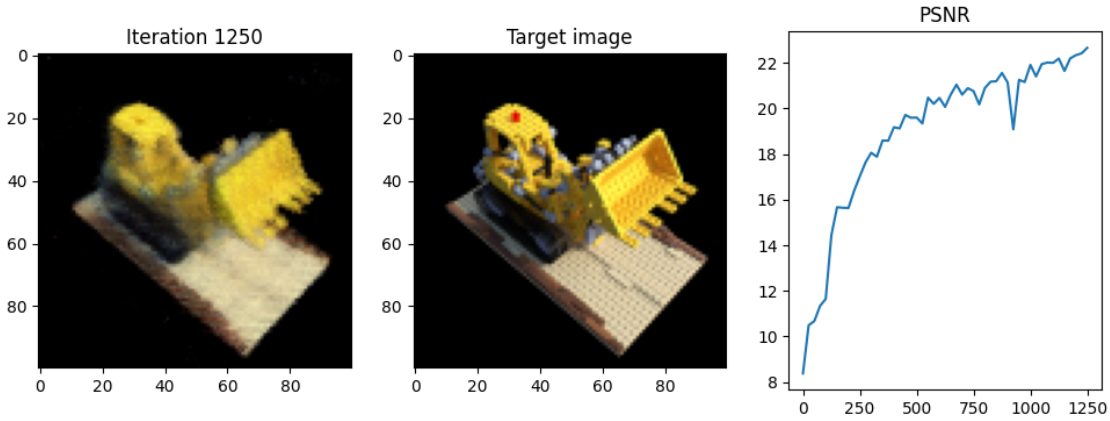

```
Iteration 1200  Loss: 0.0058  PSNR: 22.34  Time: 0.82 secs per iter,  16.58 mins
in total
```



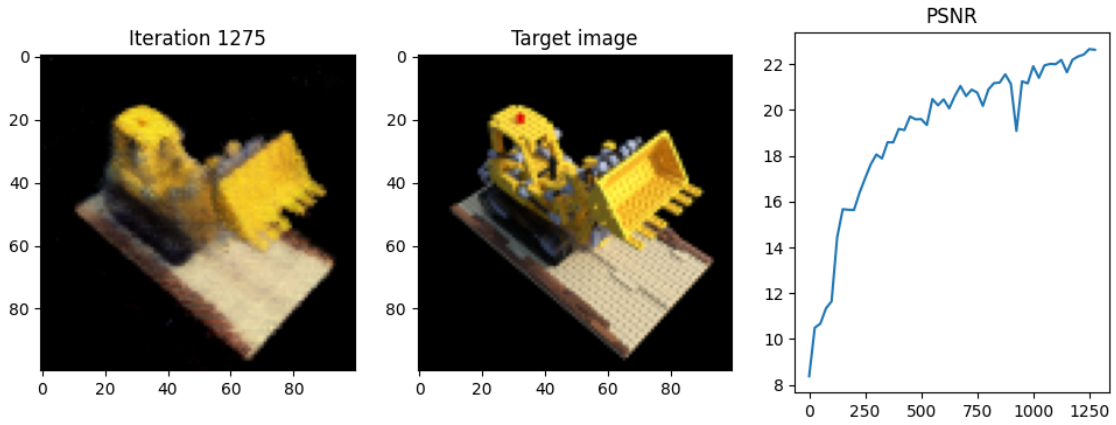
```
Iteration 1225  Loss: 0.0057  PSNR: 22.42  Time: 0.81 secs per iter,  16.92 mins
in total
```



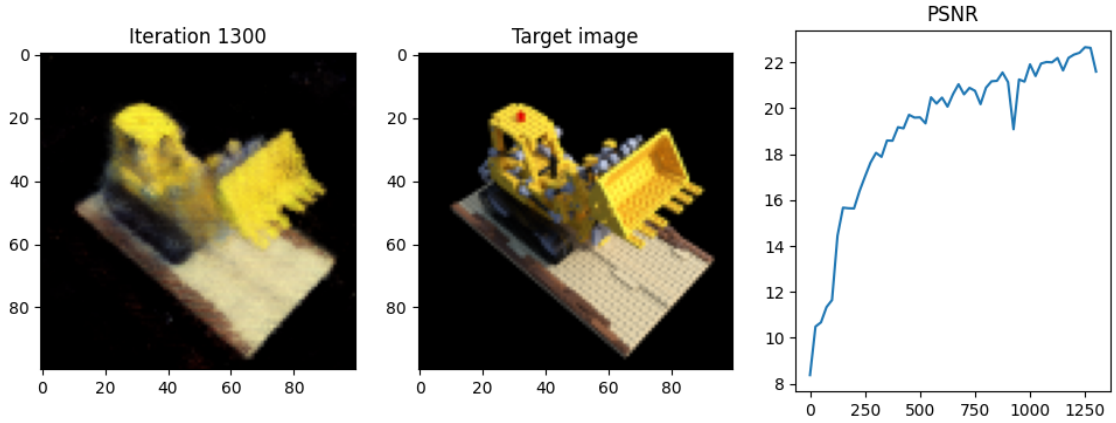

```
Iteration 1250  Loss: 0.0054  PSNR: 22.66  Time: 0.81 secs per iter,  17.26 mins
in total
```



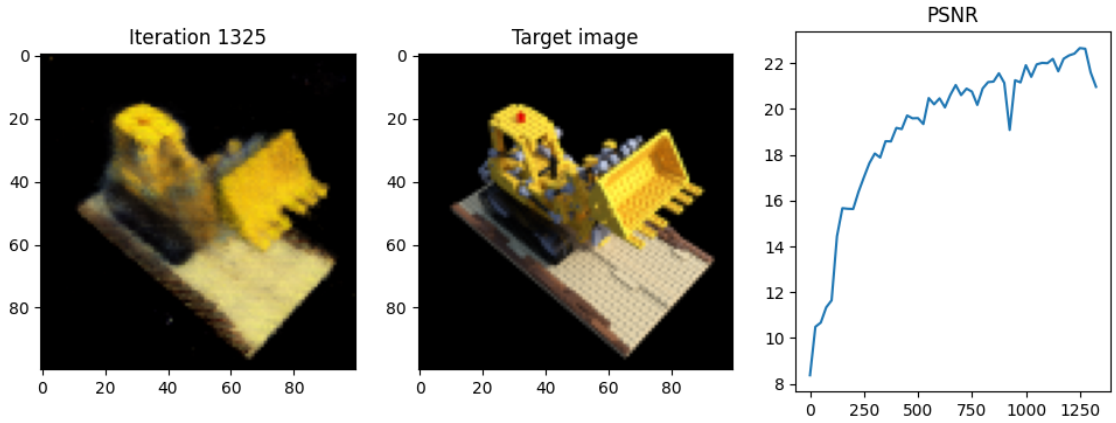

```
Iteration 1275  Loss: 0.0055  PSNR: 22.64  Time: 0.83 secs per iter,  17.60 mins
in total
```



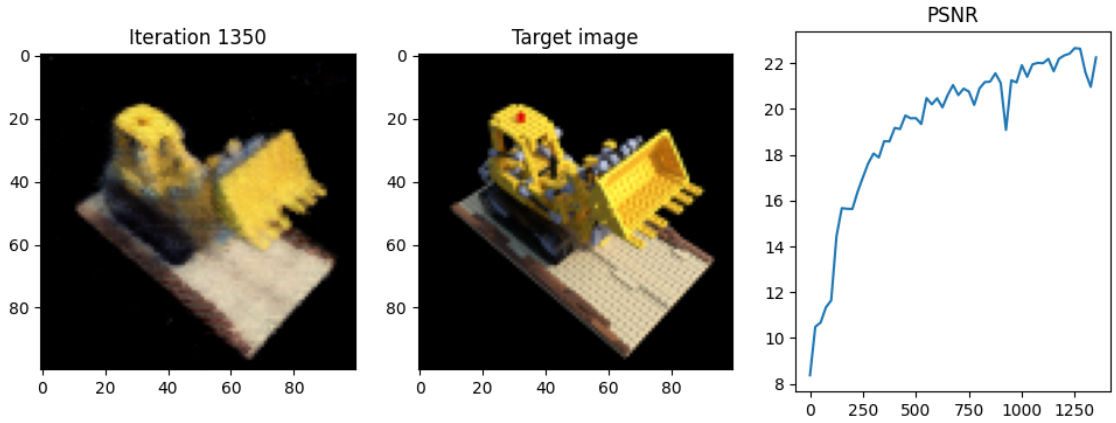

```
Iteration 1300  Loss: 0.0069  PSNR: 21.60  Time: 0.81 secs per iter,  17.94 mins
in total
```



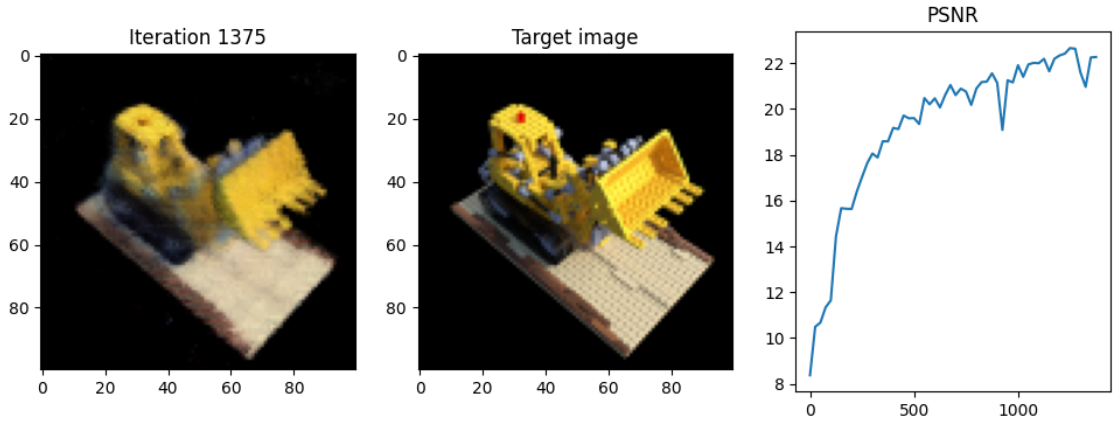

```
Iteration 1325  Loss: 0.0080  PSNR: 20.97  Time: 0.81 secs per iter,  18.28 mins
in total
```



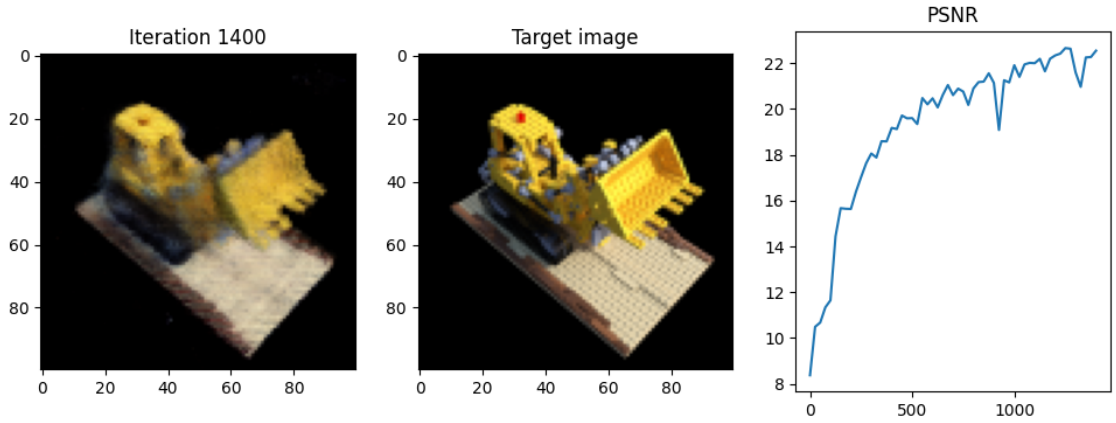

```
Iteration 1350  Loss: 0.0059  PSNR: 22.26  Time: 0.82 secs per iter,  18.62 mins
in total
```



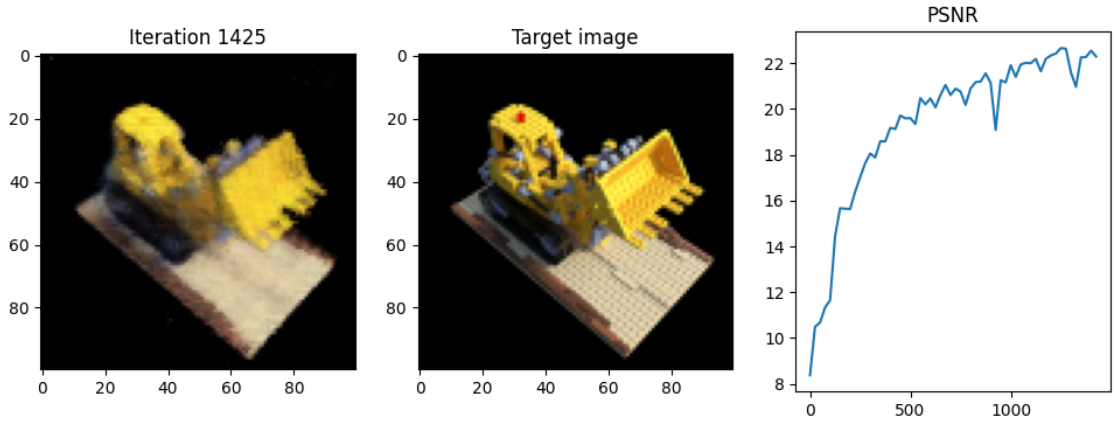

```
Iteration 1375  Loss: 0.0059  PSNR: 22.27  Time: 0.81 secs per iter,  18.95 mins
in total
```



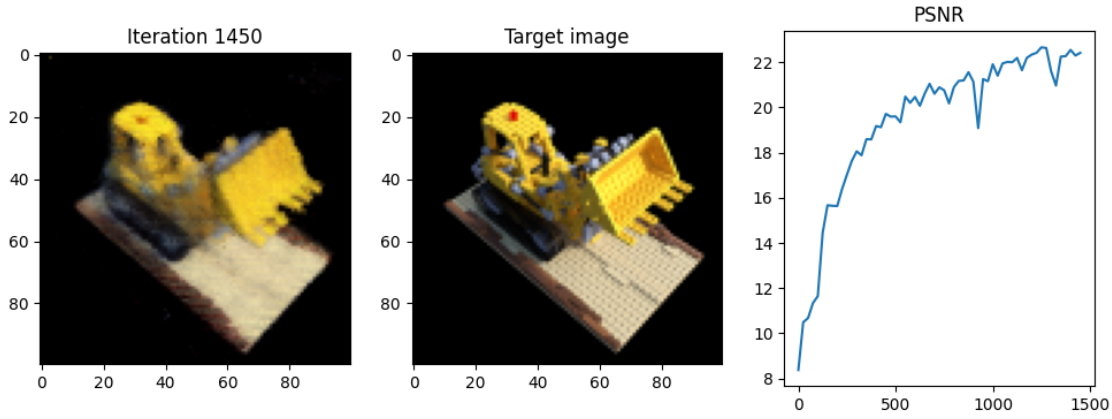

```
Iteration 1400  Loss: 0.0056  PSNR: 22.55  Time: 0.81 secs per iter,  19.29 mins
in total
```



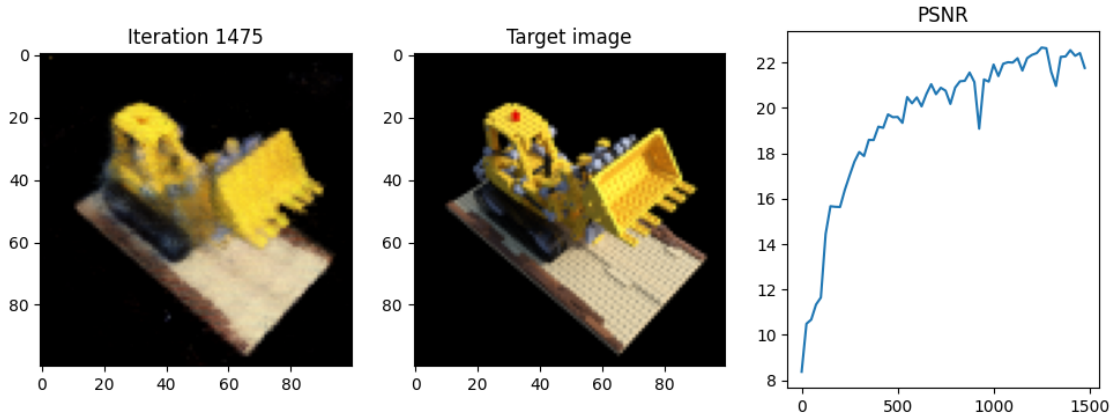

```
Iteration 1425  Loss: 0.0059  PSNR: 22.30  Time: 0.82 secs per iter,  19.63 mins
in total
```



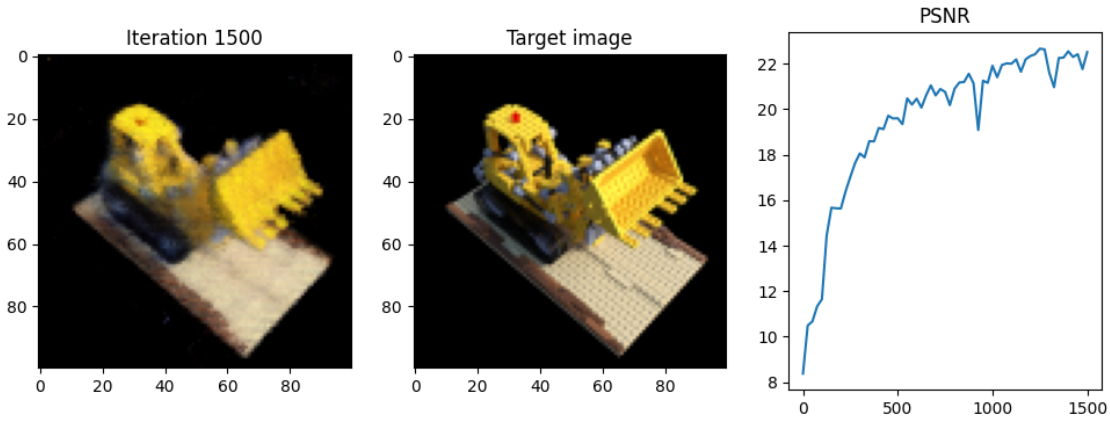

```
Iteration 1450  Loss: 0.0057  PSNR: 22.42  Time: 0.81 secs per iter,  19.97 mins
in total
```



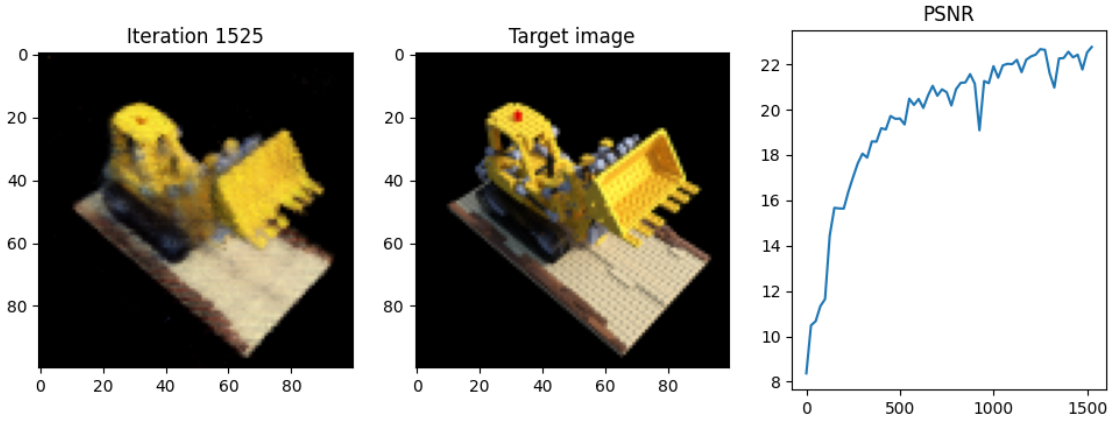

```
Iteration 1475  Loss: 0.0067  PSNR: 21.77  Time: 0.82 secs per iter,  20.31 mins
in total
```



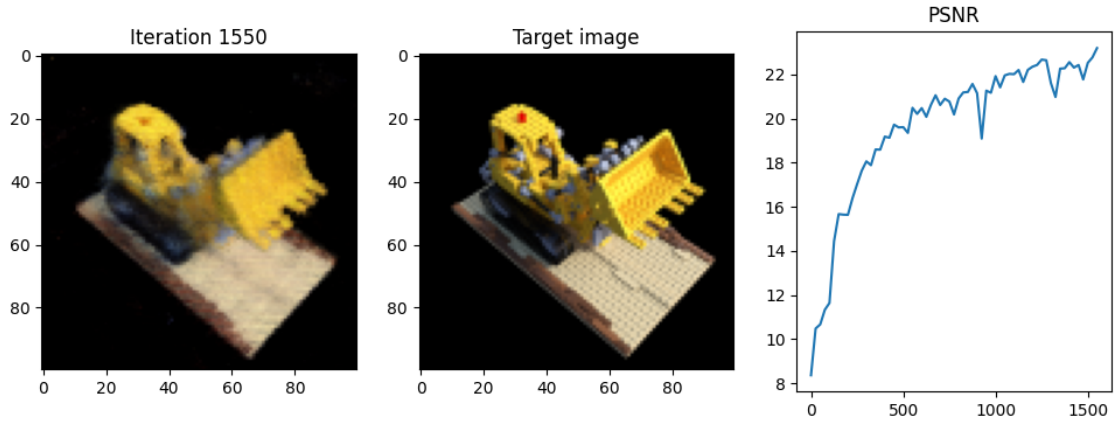
```
Iteration 1500  Loss: 0.0056  PSNR: 22.52  Time: 0.81 secs per iter,  20.65 mins
in total
```



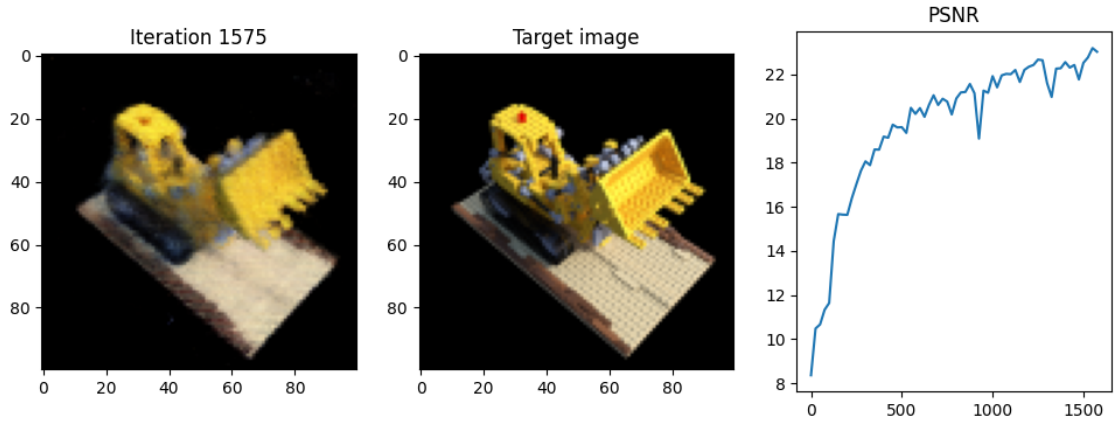

```
Iteration 1525  Loss: 0.0053  PSNR: 22.76  Time: 0.81 secs per iter,  20.98 mins
in total
```



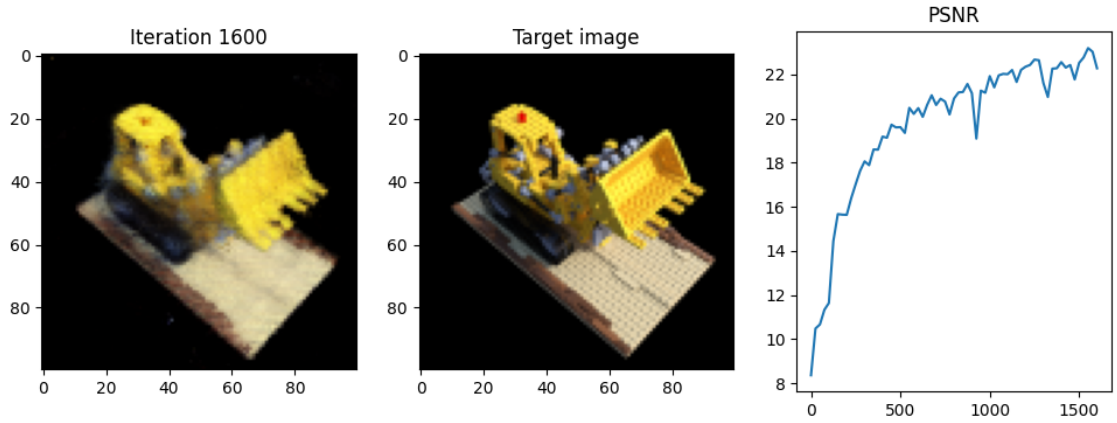

```
Iteration 1550  Loss: 0.0048  PSNR: 23.19  Time: 0.81 secs per iter,  21.32 mins
in total
```



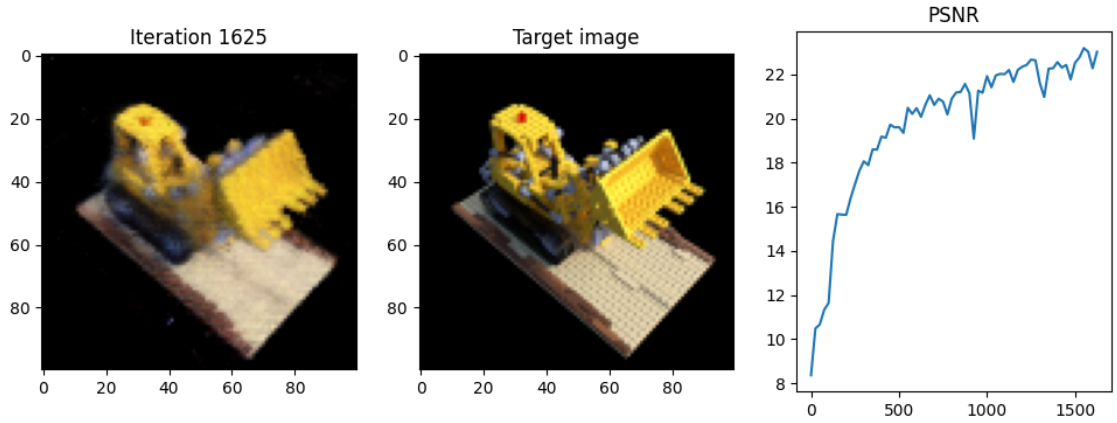
Iteration 1575 Loss: 0.0050 PSNR: 23.02 Time: 0.81 secs per iter, 21.66 mins in total



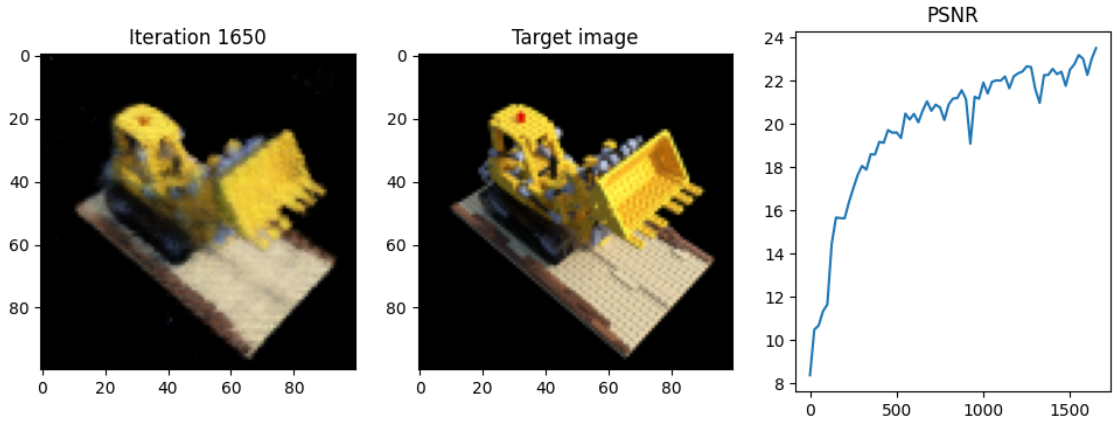
Iteration 1600 Loss: 0.0059 PSNR: 22.27 Time: 0.81 secs per iter, 22.00 mins in total



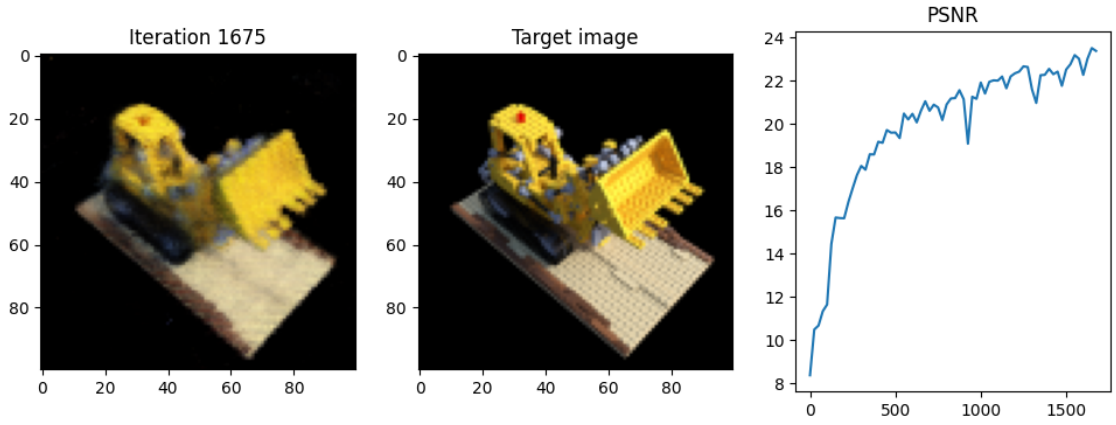

```
Iteration 1625  Loss: 0.0050  PSNR: 23.02  Time: 0.81 secs per iter,  22.33 mins
in total
```



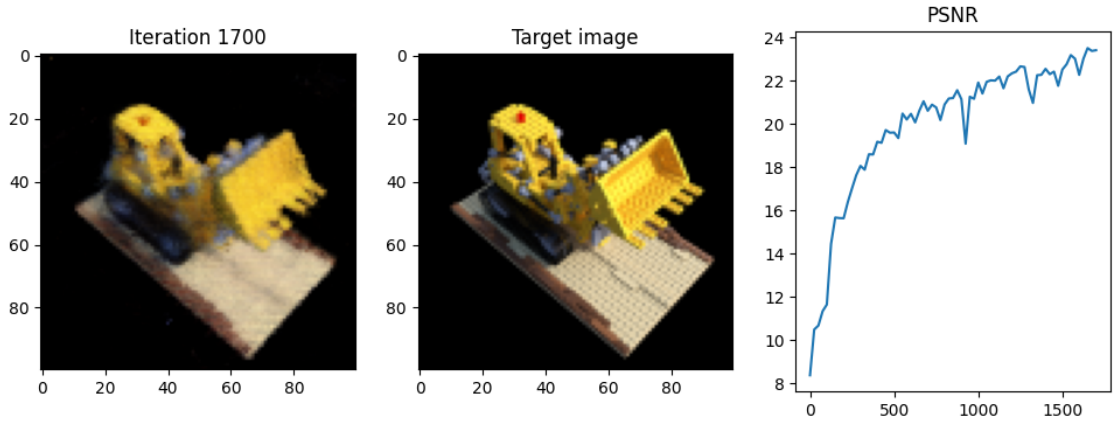

```
Iteration 1650  Loss: 0.0045  PSNR: 23.51  Time: 0.81 secs per iter,  22.67 mins
in total
```



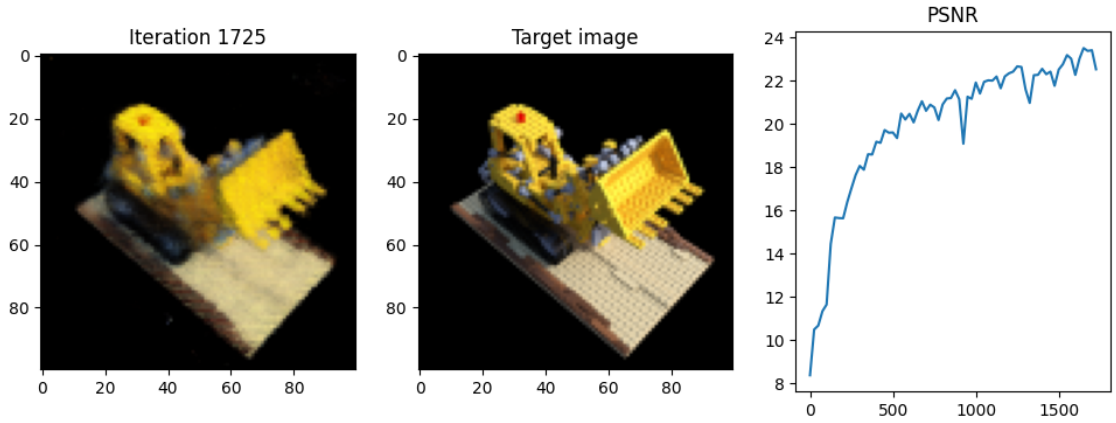

```
Iteration 1675  Loss: 0.0046  PSNR: 23.38  Time: 0.81 secs per iter,  23.01 mins
in total
```



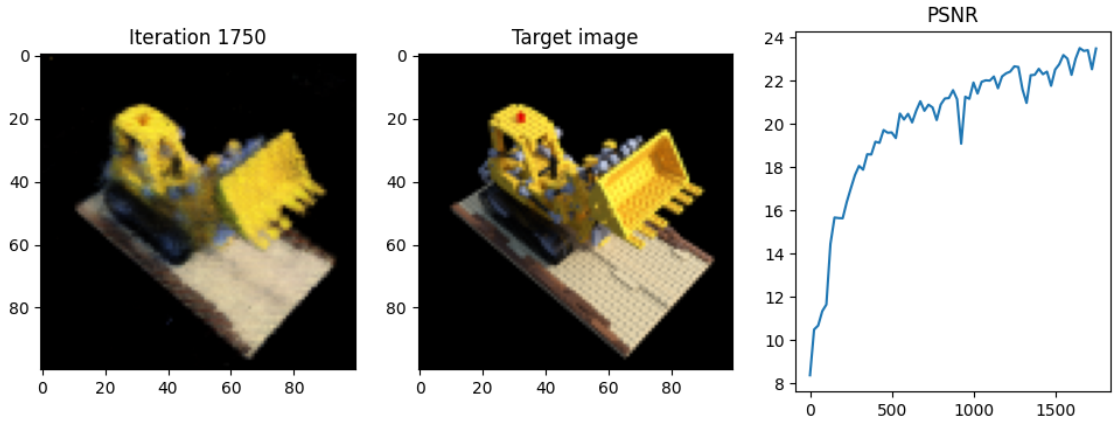

```
Iteration 1700  Loss: 0.0046  PSNR: 23.41  Time: 0.81 secs per iter,  23.35 mins
in total
```



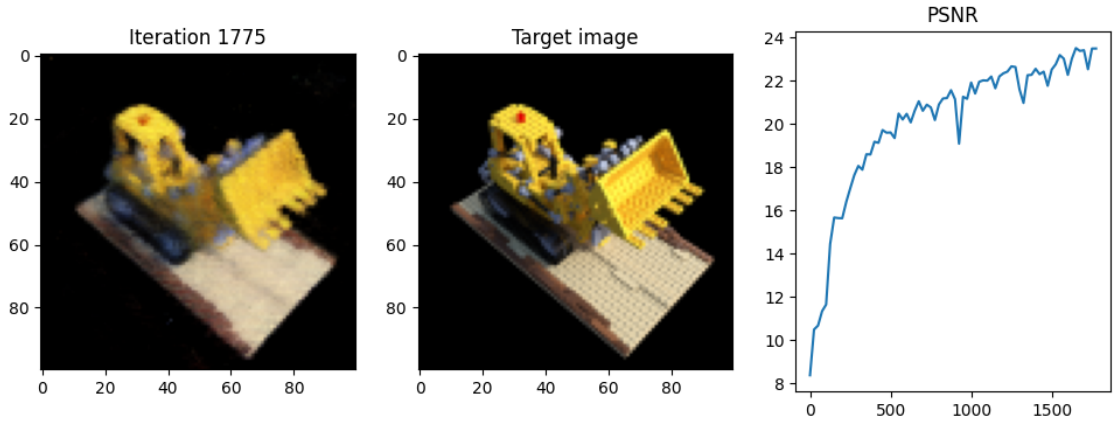

```
Iteration 1725  Loss: 0.0056  PSNR: 22.53  Time: 0.81 secs per iter,  23.69 mins
in total
```



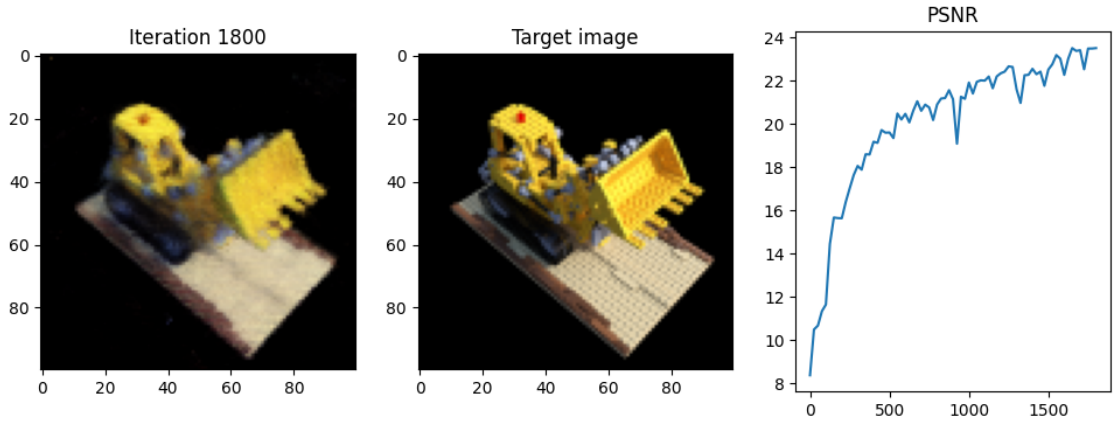

```
Iteration 1750  Loss: 0.0045  PSNR: 23.49  Time: 0.81 secs per iter,  24.03 mins
in total
```



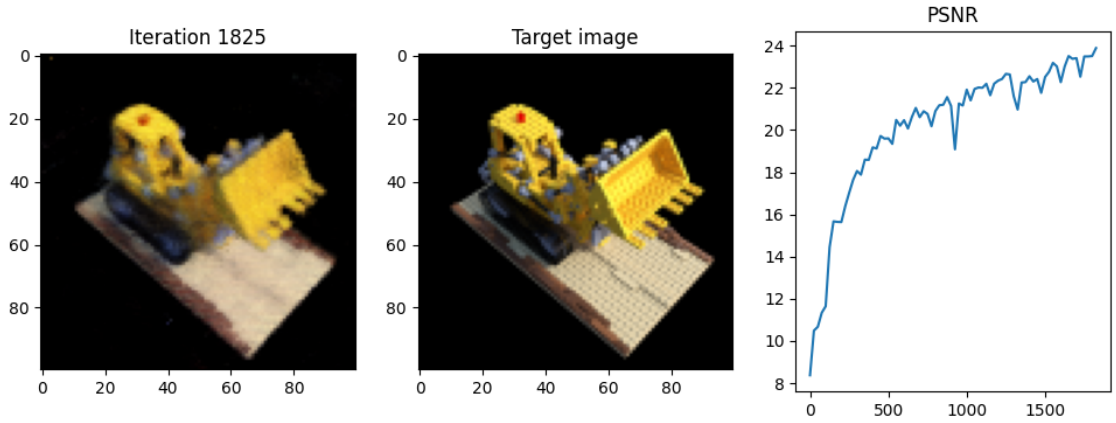

```
Iteration 1775  Loss: 0.0045  PSNR: 23.49  Time: 0.81 secs per iter,  24.36 mins
in total
```



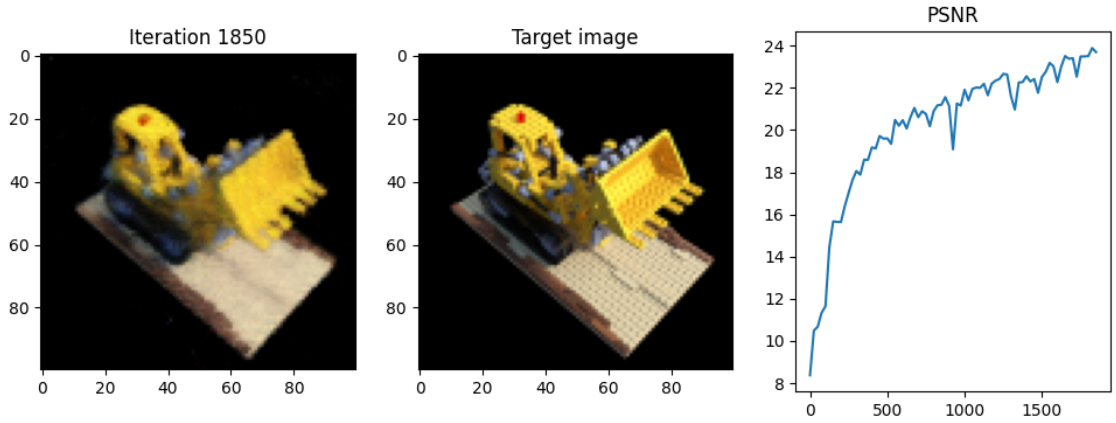

```
Iteration 1800  Loss: 0.0045  PSNR: 23.51  Time: 0.81 secs per iter,  24.70 mins
in total
```



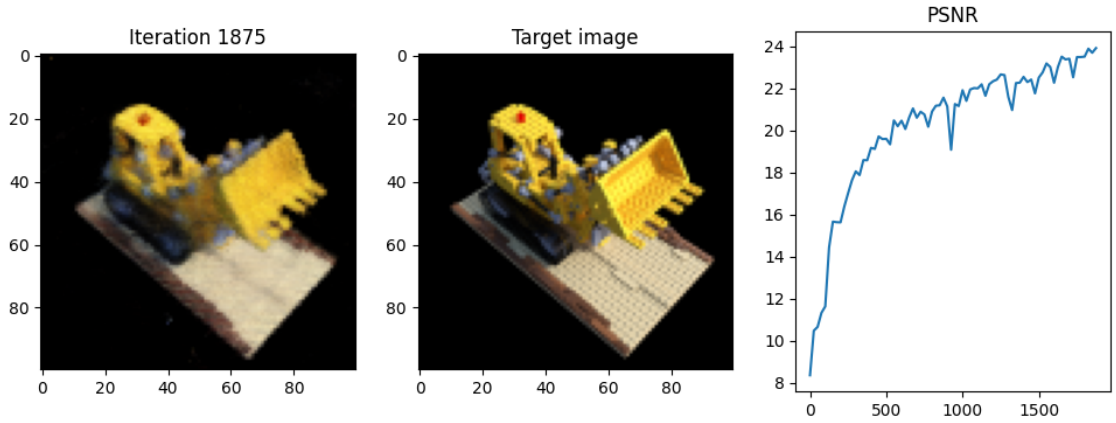

```
Iteration 1825  Loss: 0.0041  PSNR: 23.89  Time: 0.81 secs per iter,  25.04 mins
in total
```



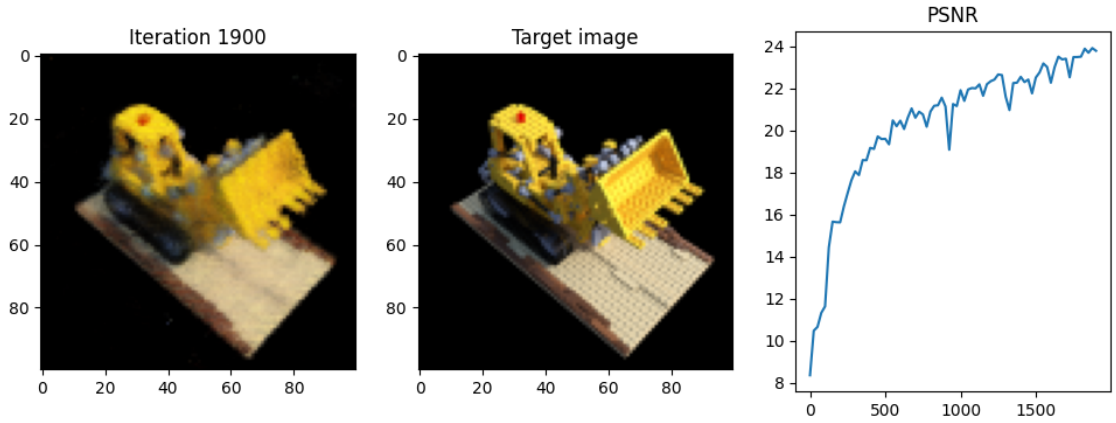

```
Iteration 1850  Loss: 0.0043  PSNR: 23.70  Time: 0.82 secs per iter,  25.38 mins
in total
```



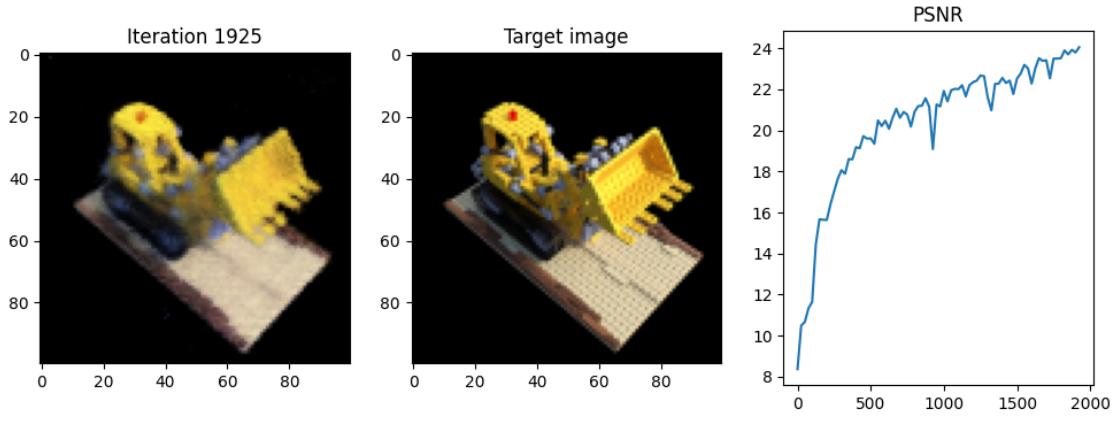

```
Iteration 1875  Loss: 0.0041  PSNR: 23.92  Time: 0.81 secs per iter,  25.72 mins
in total
```



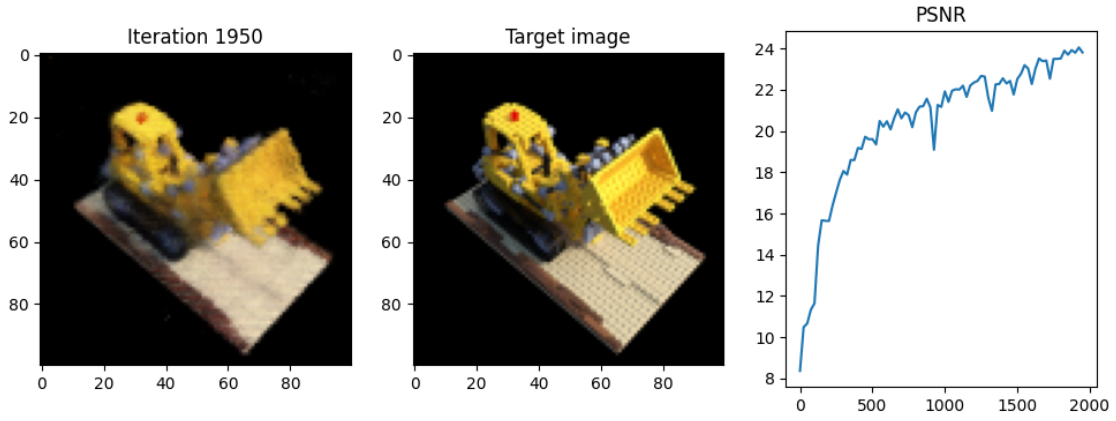

```
Iteration 1900  Loss: 0.0042  PSNR: 23.79  Time: 0.81 secs per iter,  26.05 mins
in total
```



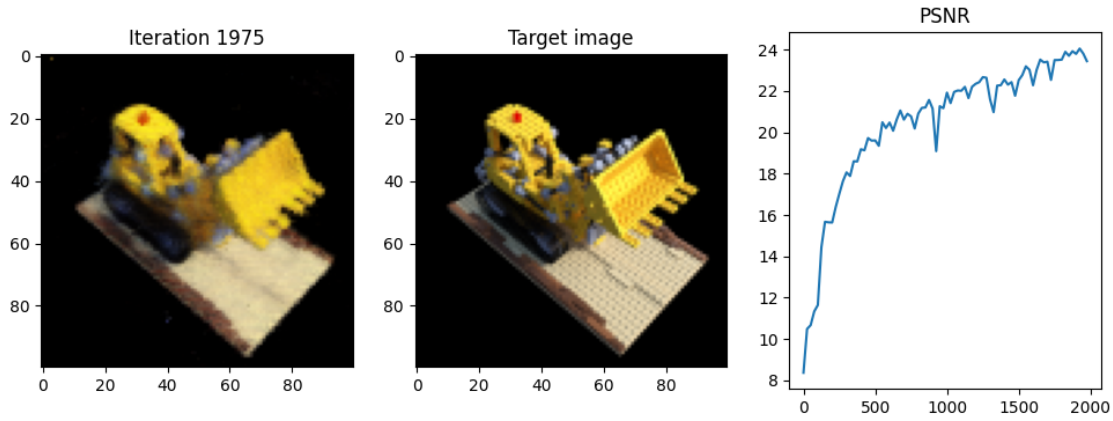
Iteration 1925 Loss: 0.0039 PSNR: 24.05 Time: 0.81 secs per iter, 26.39 mins in total



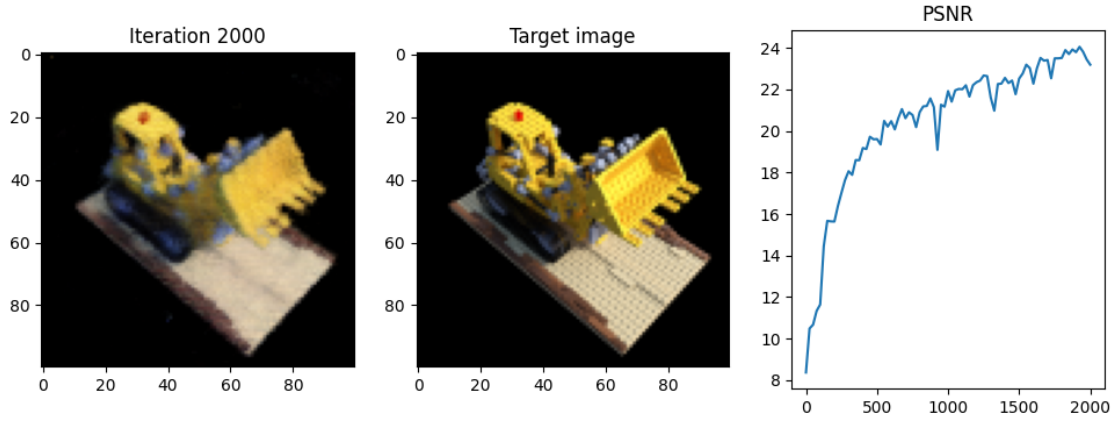
Iteration 1950 Loss: 0.0042 PSNR: 23.80 Time: 0.81 secs per iter, 26.73 mins in total



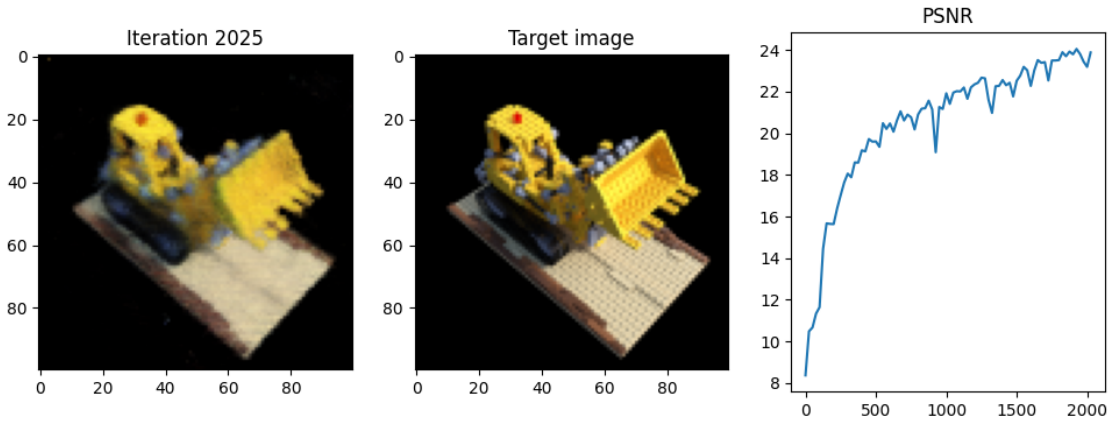
```
Iteration 1975  Loss: 0.0045  PSNR: 23.44  Time: 0.81 secs per iter,  27.07 mins
in total
```



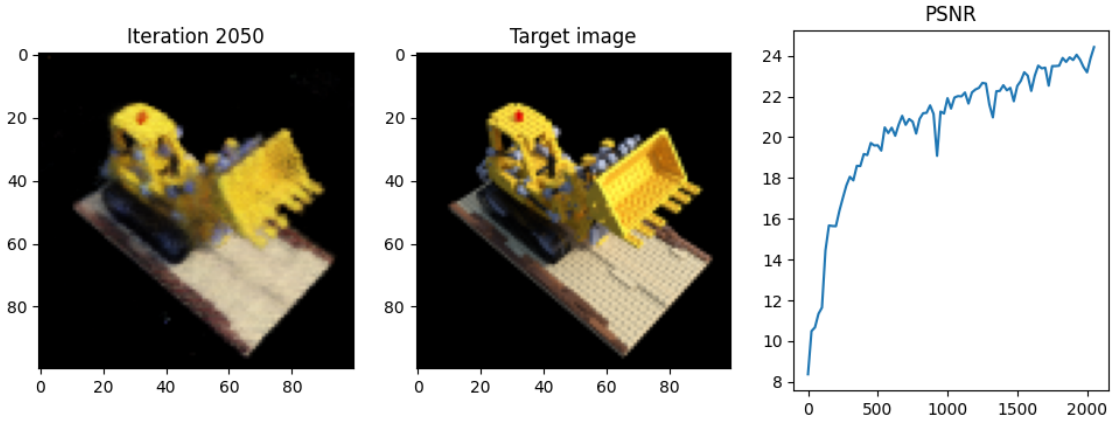

```
Iteration 2000  Loss: 0.0048  PSNR: 23.19  Time: 0.82 secs per iter,  27.41 mins
in total
```



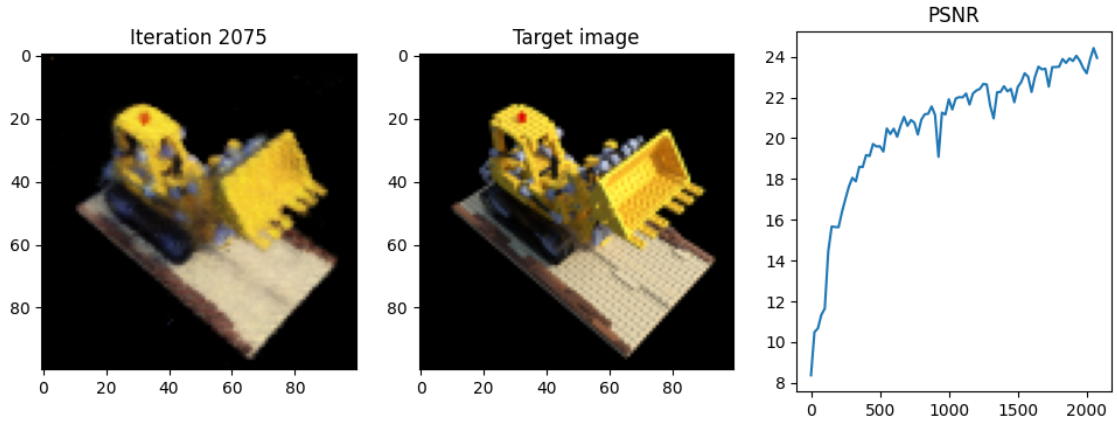

```
Iteration 2025  Loss: 0.0041  PSNR: 23.88  Time: 0.81 secs per iter,  27.75 mins
in total
```



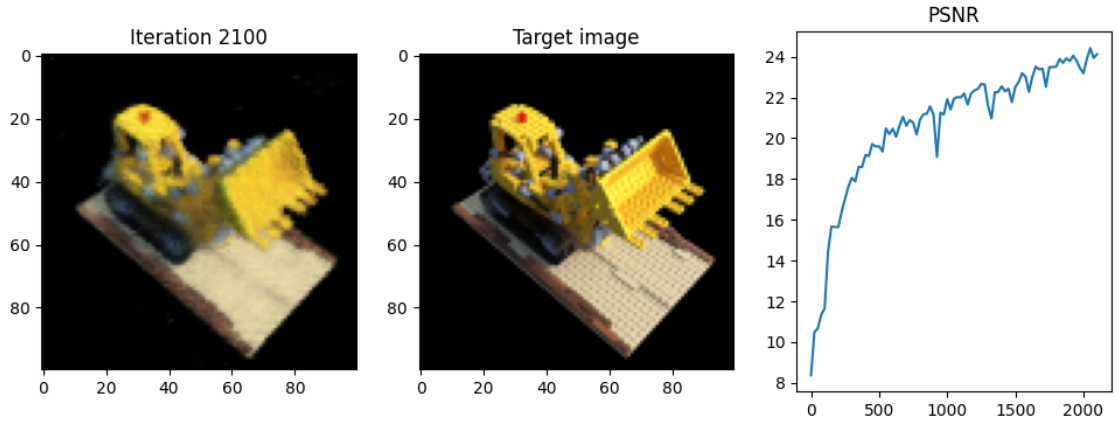

```
Iteration 2050  Loss: 0.0036  PSNR: 24.42  Time: 0.81 secs per iter,  28.08 mins
in total
```



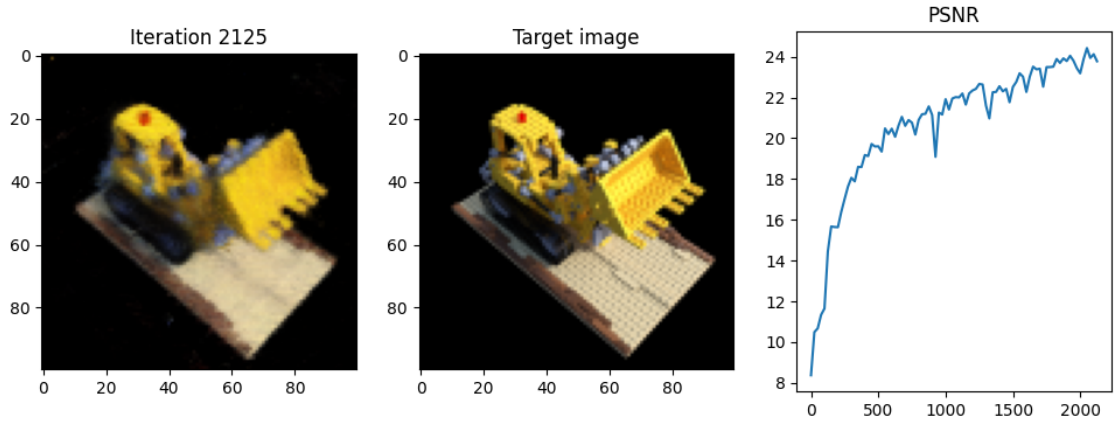

```
Iteration 2075  Loss: 0.0040  PSNR: 23.94  Time: 0.82 secs per iter,  28.42 mins
in total
```



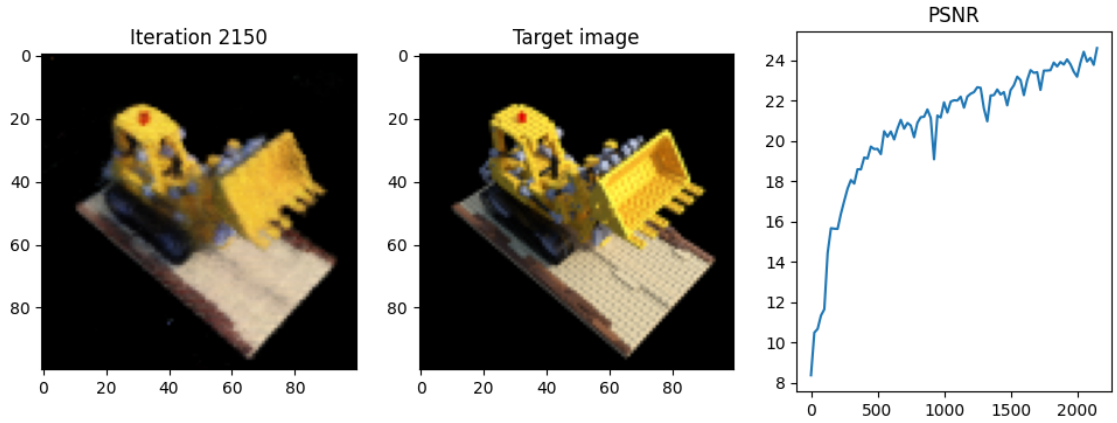

```
Iteration 2100  Loss: 0.0039  PSNR: 24.12  Time: 0.81 secs per iter,  28.76 mins
in total
```



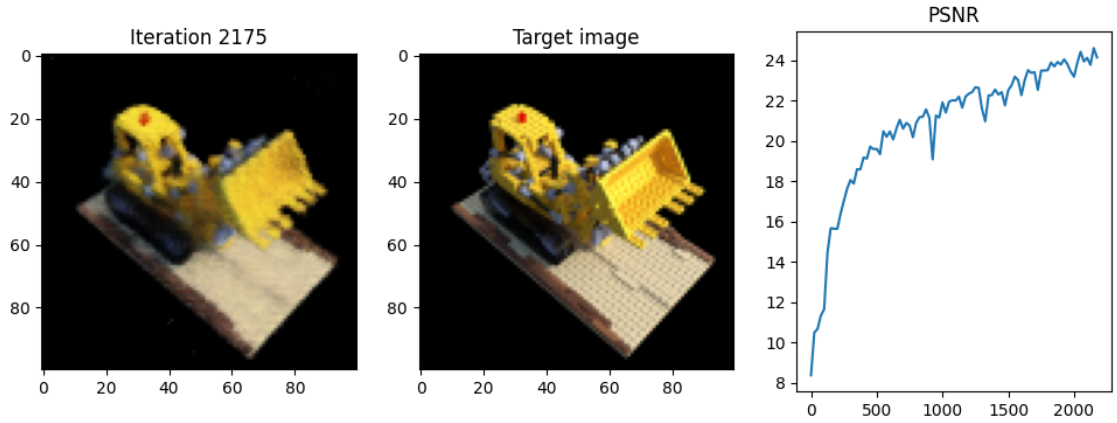

```
Iteration 2125  Loss: 0.0042  PSNR: 23.78  Time: 0.81 secs per iter,  29.10 mins
in total
```



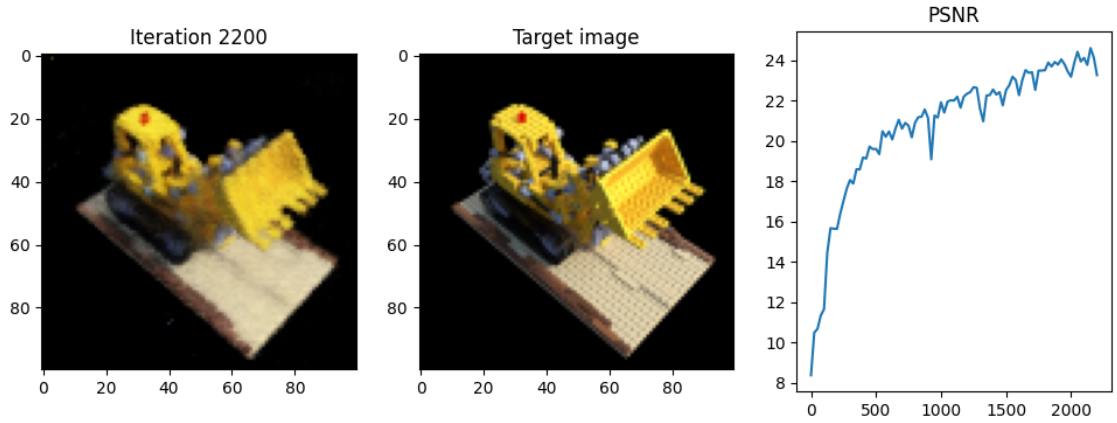

```
Iteration 2150  Loss: 0.0035  PSNR: 24.61  Time: 0.82 secs per iter,  29.44 mins
in total
```



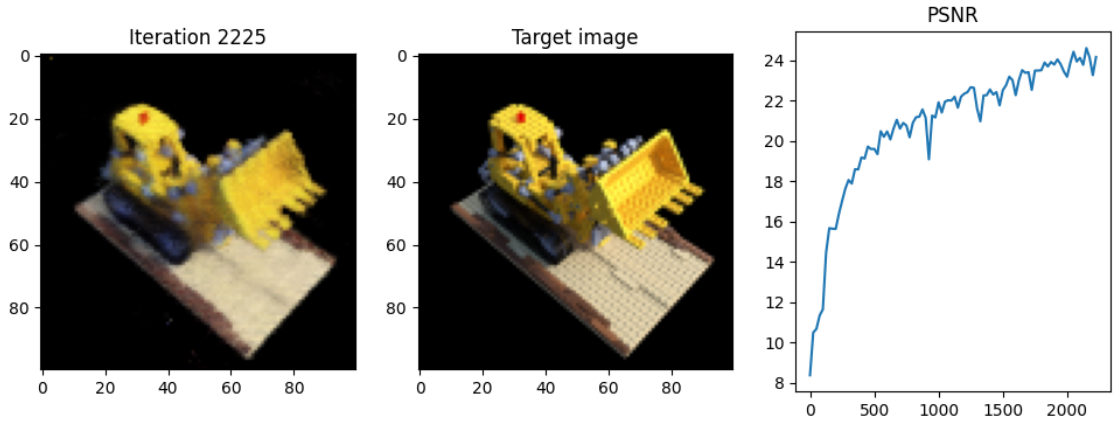

```
Iteration 2175  Loss: 0.0039  PSNR: 24.14  Time: 0.81 secs per iter,  29.77 mins
in total
```



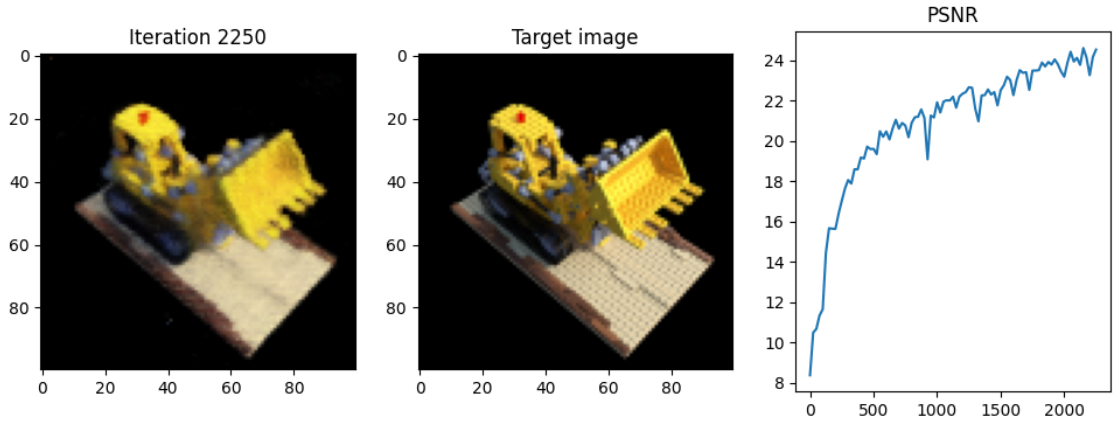

```
Iteration 2200  Loss: 0.0047  PSNR: 23.26  Time: 0.81 secs per iter,  30.11 mins
in total
```



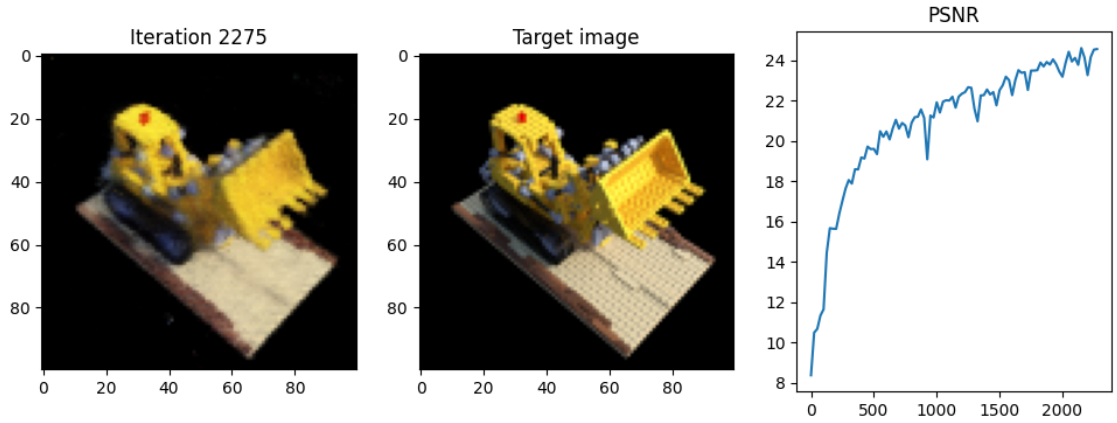

```
Iteration 2225  Loss: 0.0038  PSNR: 24.16  Time: 0.82 secs per iter,  30.45 mins
in total
```



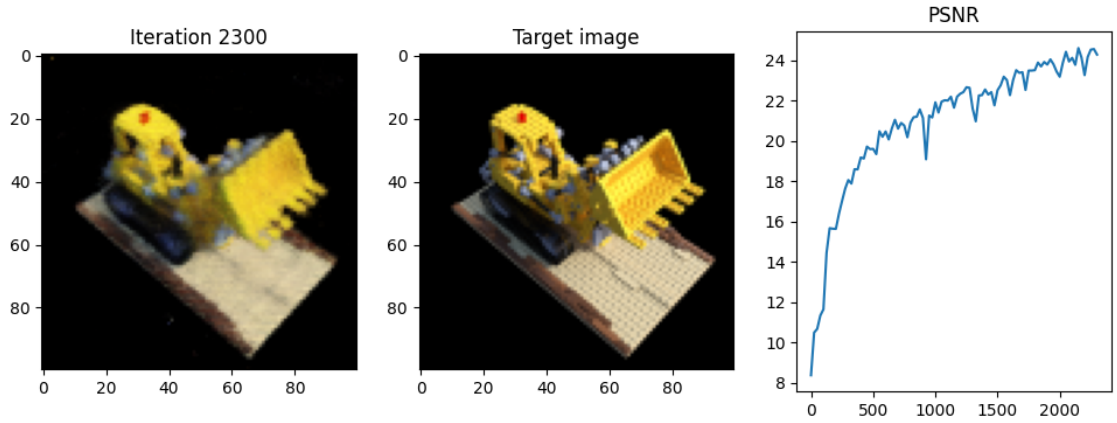

```
Iteration 2250  Loss: 0.0035  PSNR: 24.53  Time: 0.81 secs per iter,  30.79 mins
in total
```



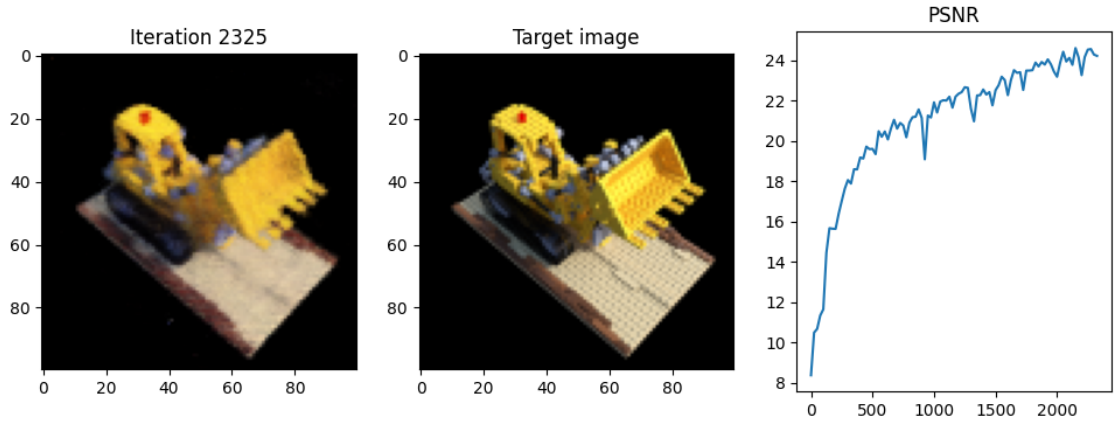

```
Iteration 2275  Loss: 0.0035  PSNR: 24.56  Time: 0.81 secs per iter,  31.13 mins
in total
```



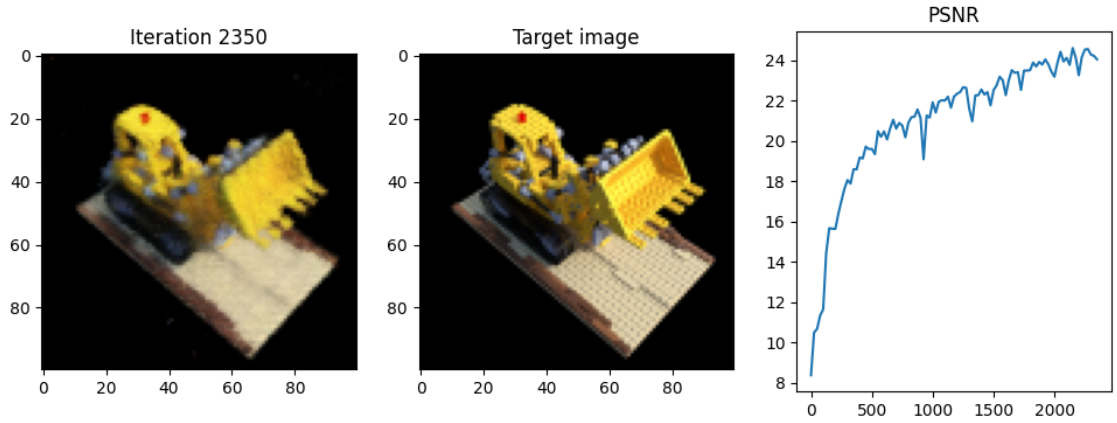

```
Iteration 2300  Loss: 0.0037  PSNR: 24.28  Time: 0.81 secs per iter,  31.46 mins
in total
```



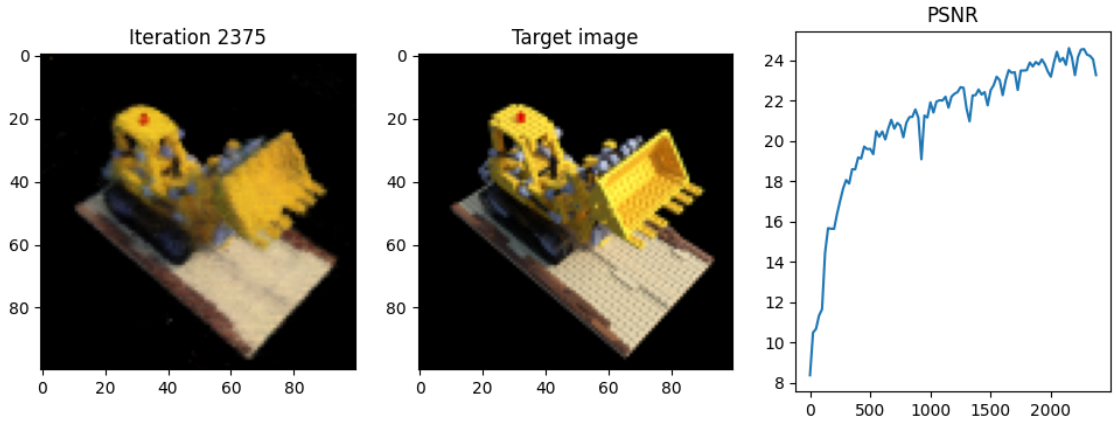

```
Iteration 2325  Loss: 0.0038  PSNR: 24.22  Time: 0.81 secs per iter,  31.80 mins
in total
```



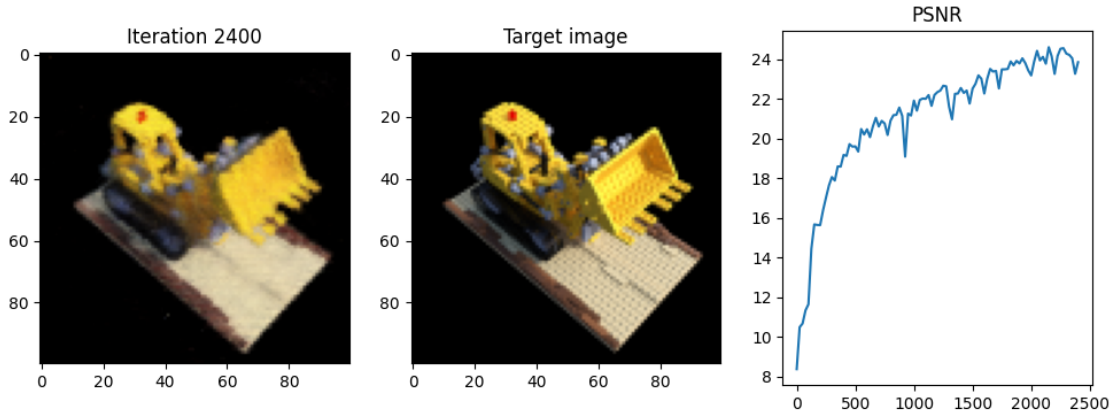

```
Iteration 2350  Loss: 0.0039  PSNR: 24.05  Time: 0.81 secs per iter,  32.14 mins
in total
```



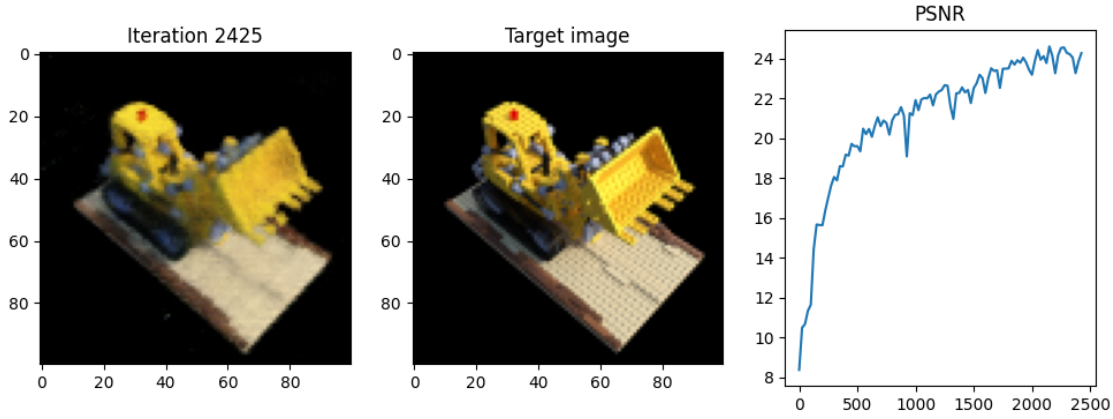

```
Iteration 2375  Loss: 0.0047  PSNR: 23.27  Time: 0.81 secs per iter,  32.48 mins
in total
```



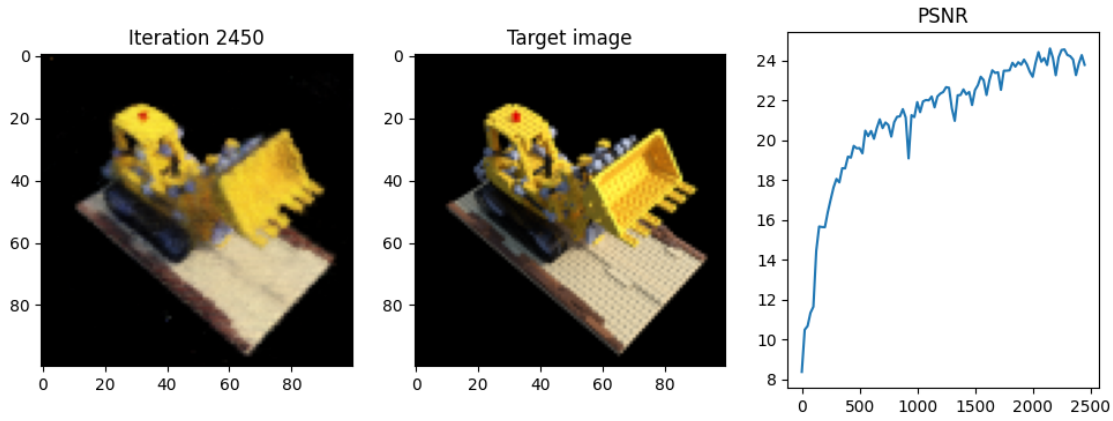

```
Iteration 2400  Loss: 0.0041  PSNR: 23.85  Time: 0.82 secs per iter,  32.82 mins
in total
```



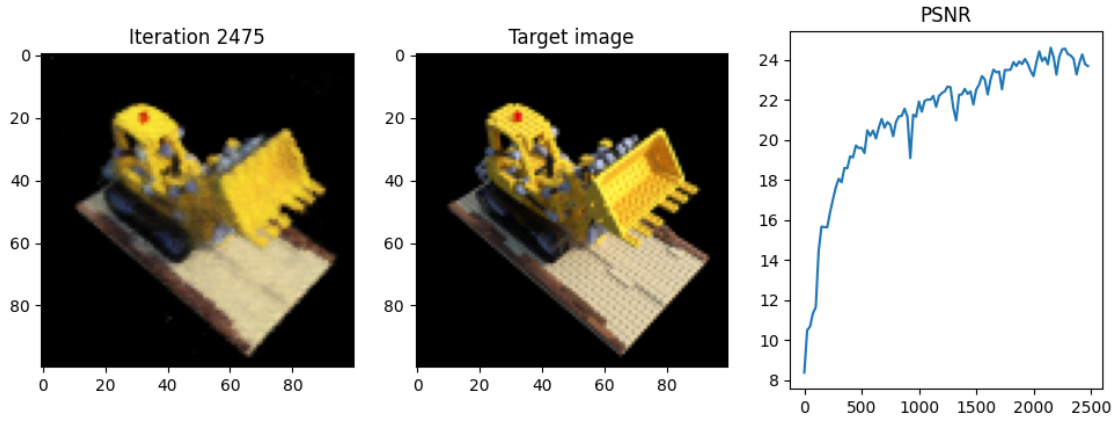

```
Iteration 2425  Loss: 0.0037  PSNR: 24.27  Time: 0.81 secs per iter,  33.15 mins
in total
```



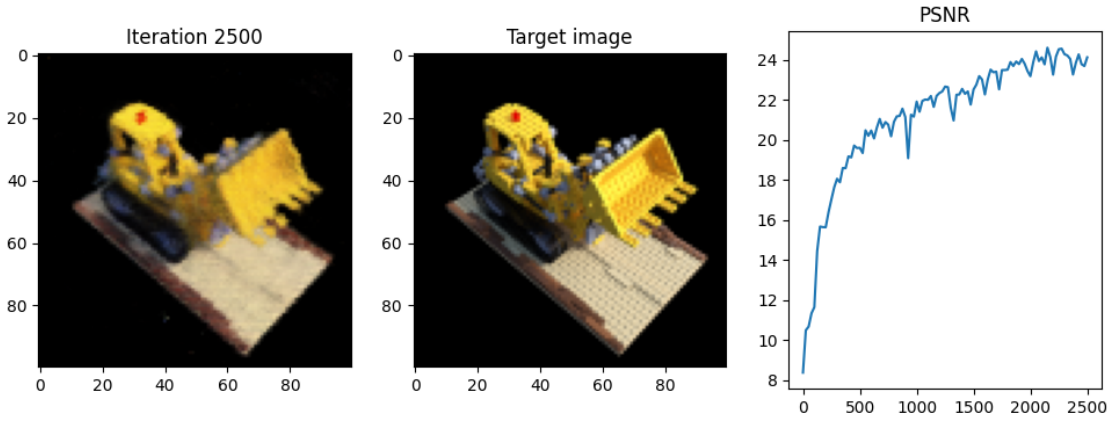
```
Iteration 2450  Loss: 0.0042  PSNR: 23.79  Time: 0.81 secs per iter,  33.49 mins
in total
```



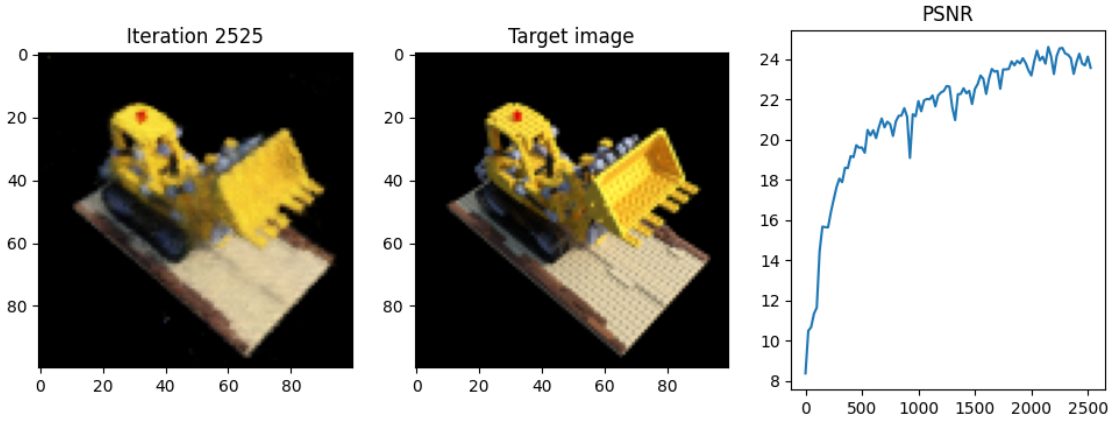

```
Iteration 2475  Loss: 0.0043  PSNR: 23.69  Time: 0.81 secs per iter,  33.83 mins
in total
```



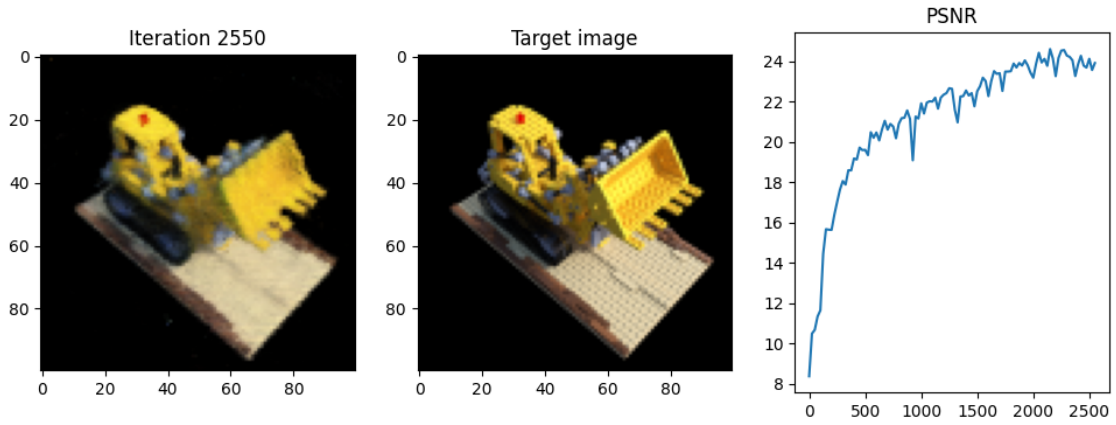

```
Iteration 2500  Loss: 0.0039  PSNR: 24.13  Time: 0.81 secs per iter,  34.17 mins
in total
```



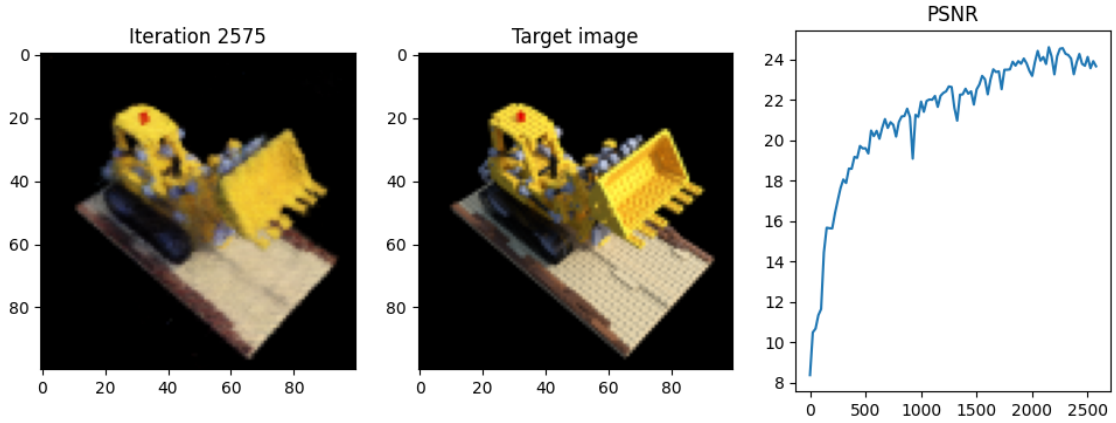

```
Iteration 2525  Loss: 0.0044  PSNR: 23.57  Time: 0.81 secs per iter,  34.51 mins
in total
```



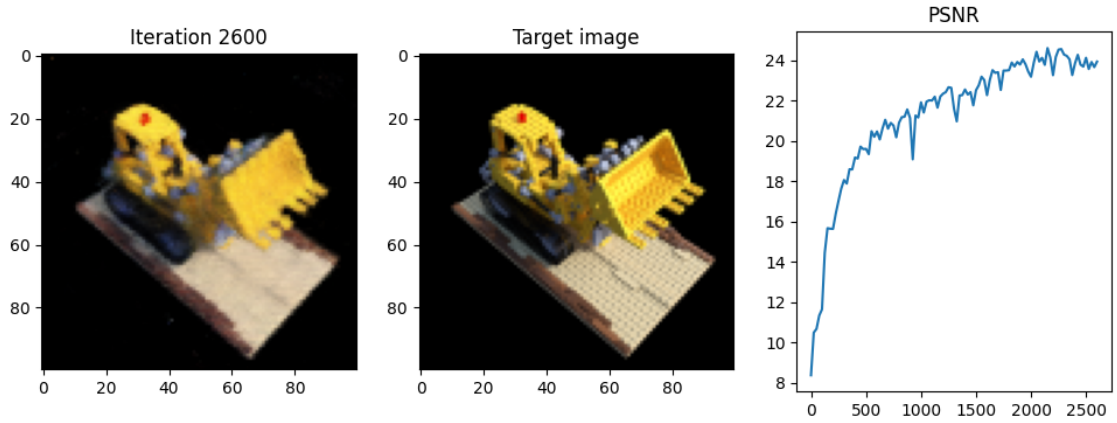

```
Iteration 2550  Loss: 0.0041  PSNR: 23.92  Time: 0.81 secs per iter,  34.84 mins
in total
```



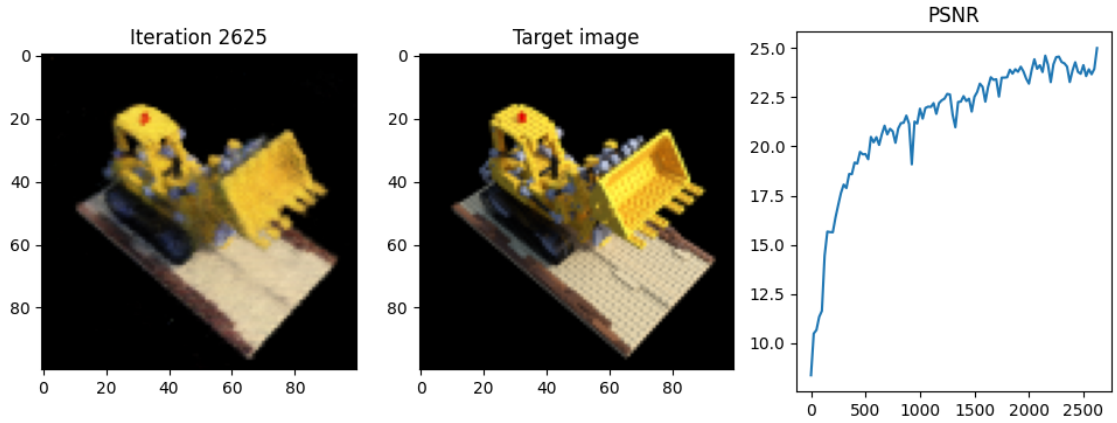

```
Iteration 2575  Loss: 0.0043  PSNR: 23.66  Time: 0.83 secs per iter,  35.19 mins
in total
```



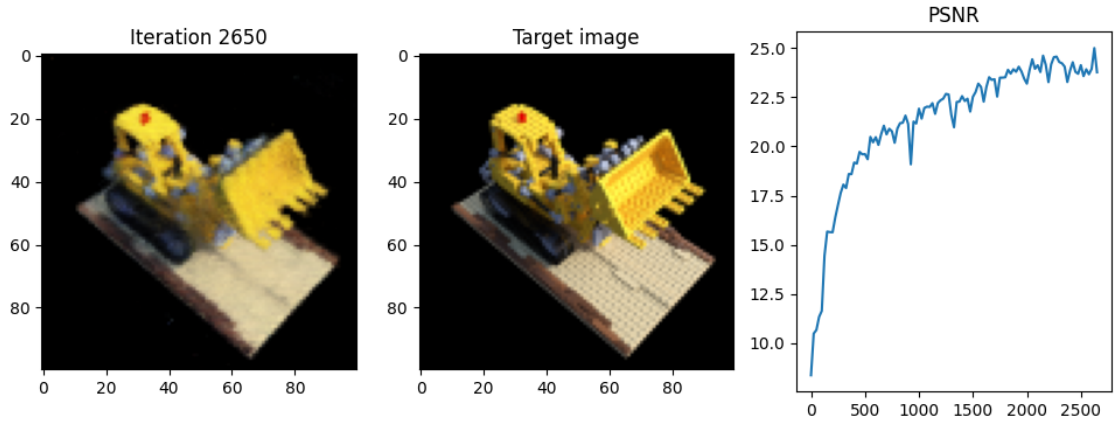

```
Iteration 2600  Loss: 0.0040  PSNR: 23.94  Time: 0.81 secs per iter,  35.53 mins
in total
```



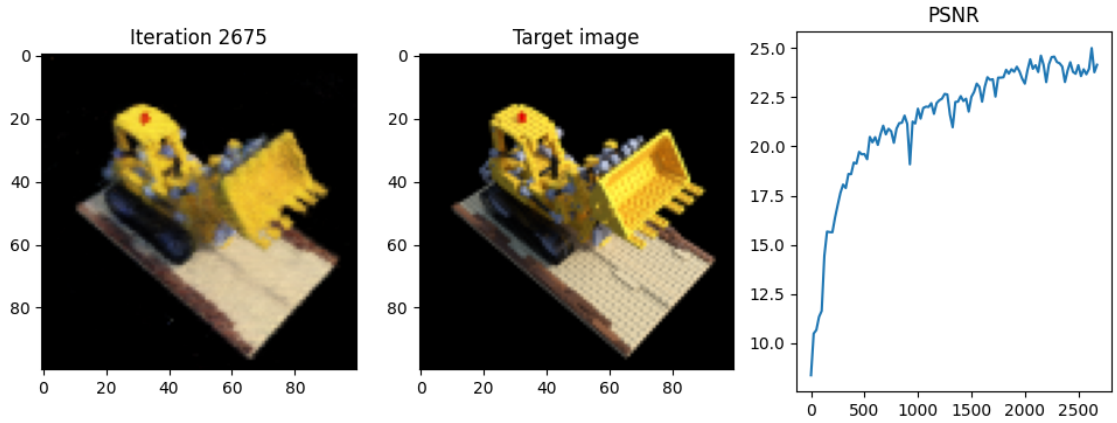

```
Iteration 2625  Loss: 0.0032  PSNR: 24.99  Time: 0.81 secs per iter,  35.87 mins
in total
```



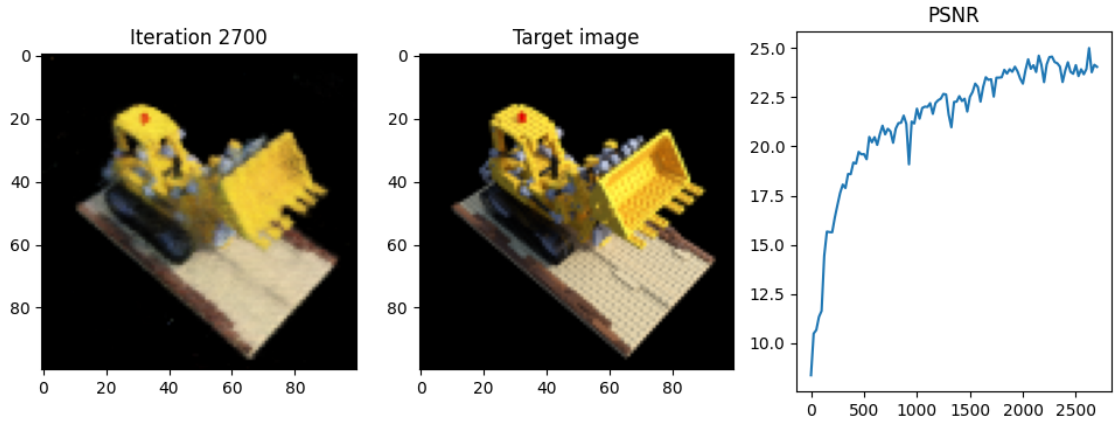

```
Iteration 2650  Loss: 0.0042  PSNR: 23.76  Time: 0.82 secs per iter,  36.21 mins
in total
```



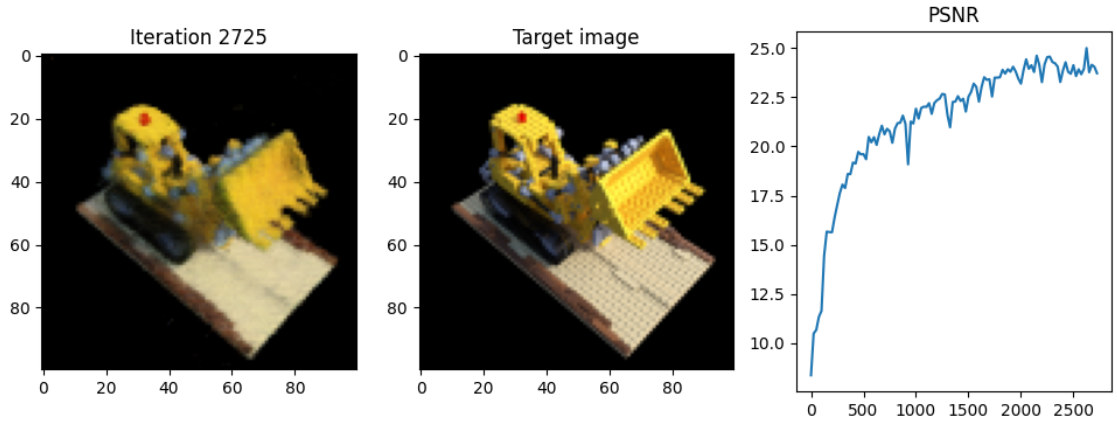

```
Iteration 2675  Loss: 0.0039  PSNR: 24.14  Time: 0.81 secs per iter,  36.55 mins
in total
```



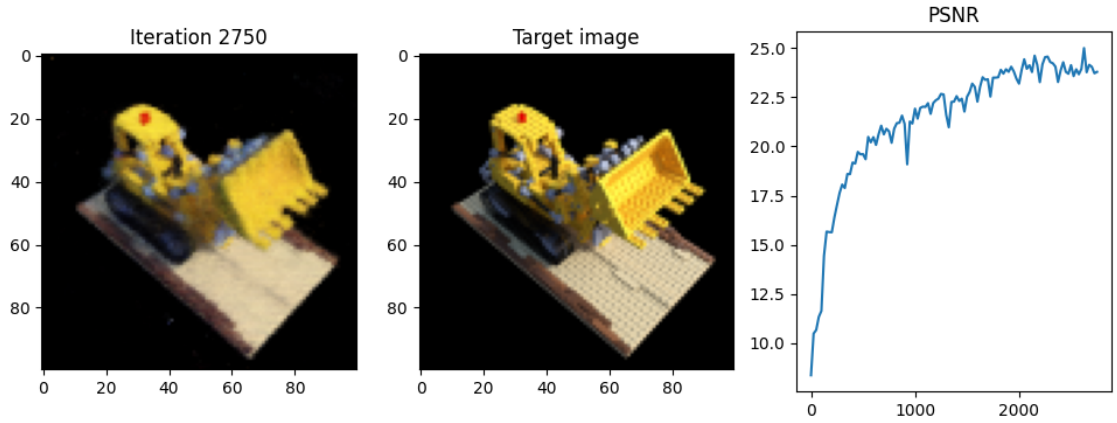

```
Iteration 2700  Loss: 0.0039  PSNR: 24.04  Time: 0.81 secs per iter,  36.89 mins
in total
```



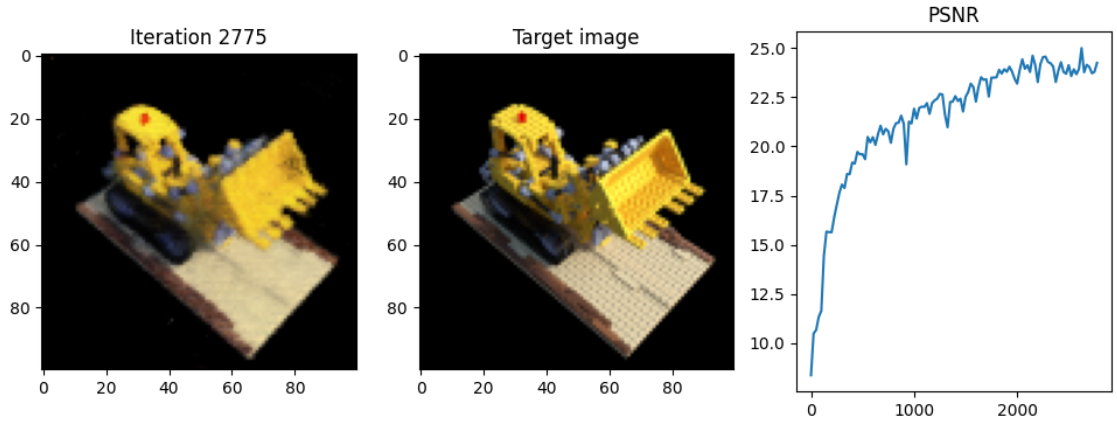

```
Iteration 2725  Loss: 0.0043  PSNR: 23.71  Time: 0.81 secs per iter,  37.22 mins
in total
```



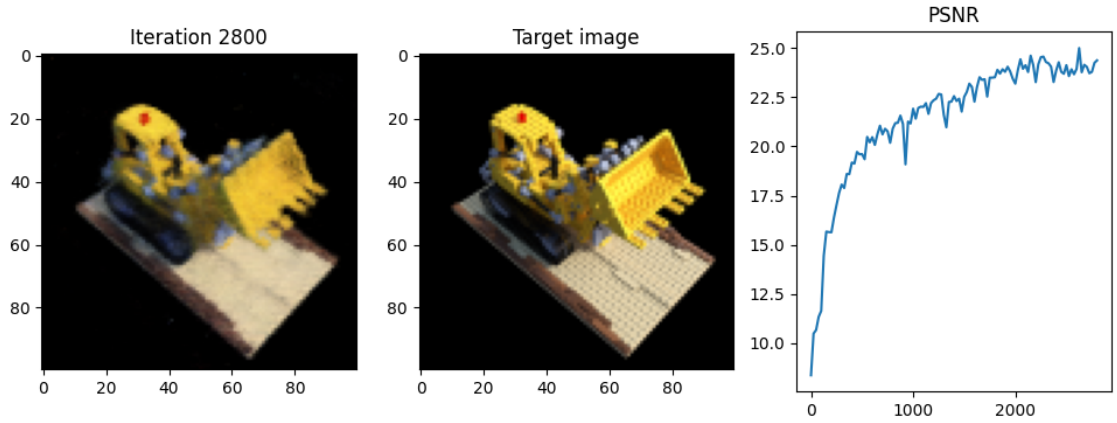

```
Iteration 2750  Loss: 0.0042  PSNR: 23.79  Time: 0.81 secs per iter,  37.56 mins
in total
```



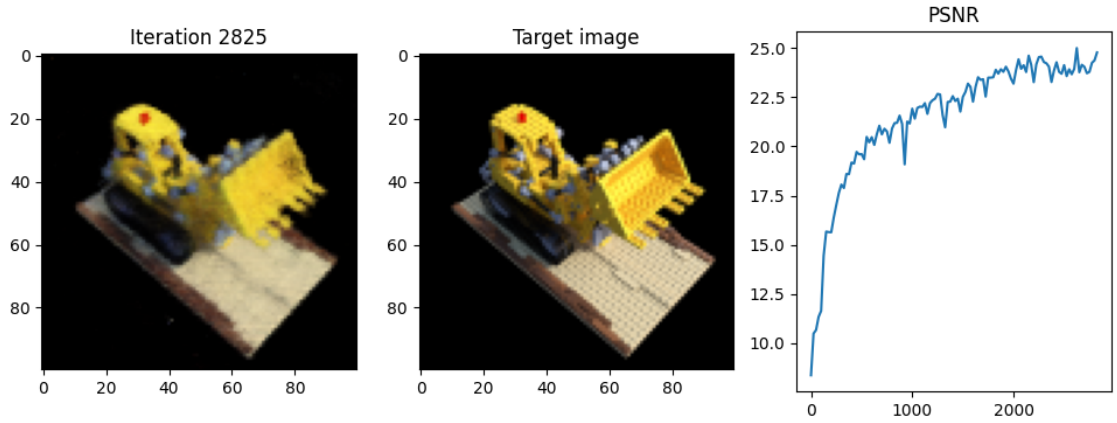

```
Iteration 2775  Loss: 0.0038  PSNR: 24.24  Time: 0.81 secs per iter,  37.90 mins
in total
```



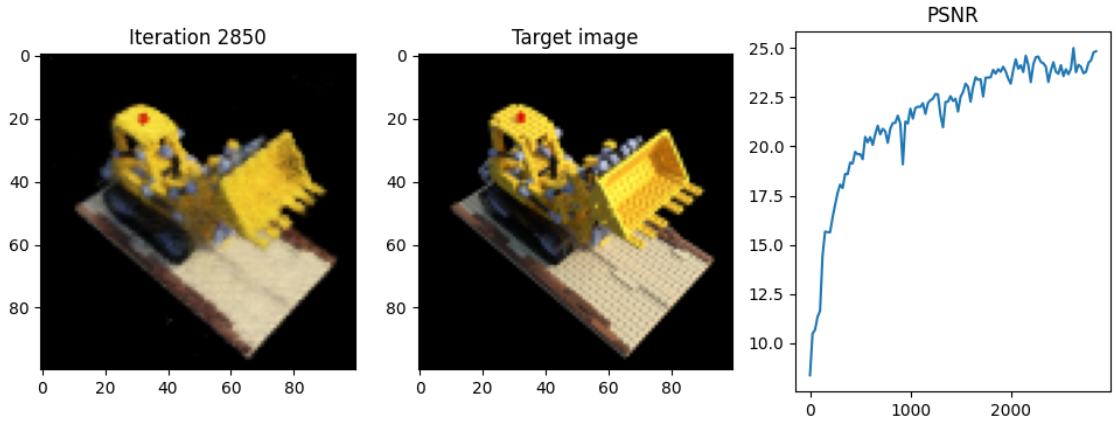

```
Iteration 2800  Loss: 0.0037  PSNR: 24.37  Time: 0.81 secs per iter,  38.23 mins
in total
```



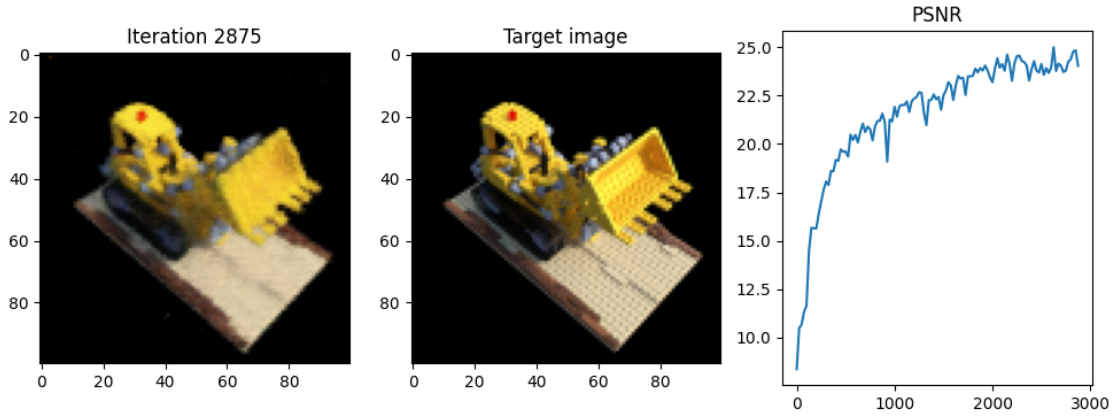

```
Iteration 2825  Loss: 0.0033  PSNR: 24.77  Time: 0.81 secs per iter,  38.57 mins
in total
```



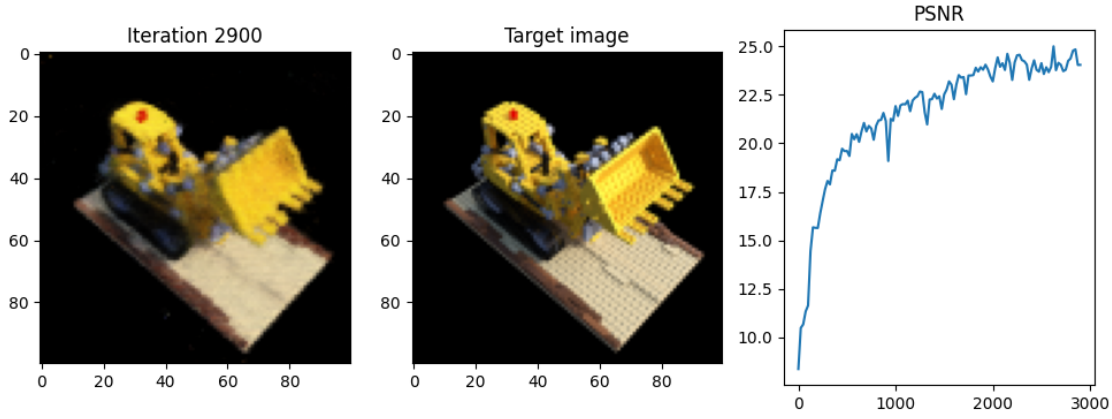

```
Iteration 2850  Loss: 0.0033  PSNR: 24.83  Time: 0.81 secs per iter,  38.91 mins
in total
```



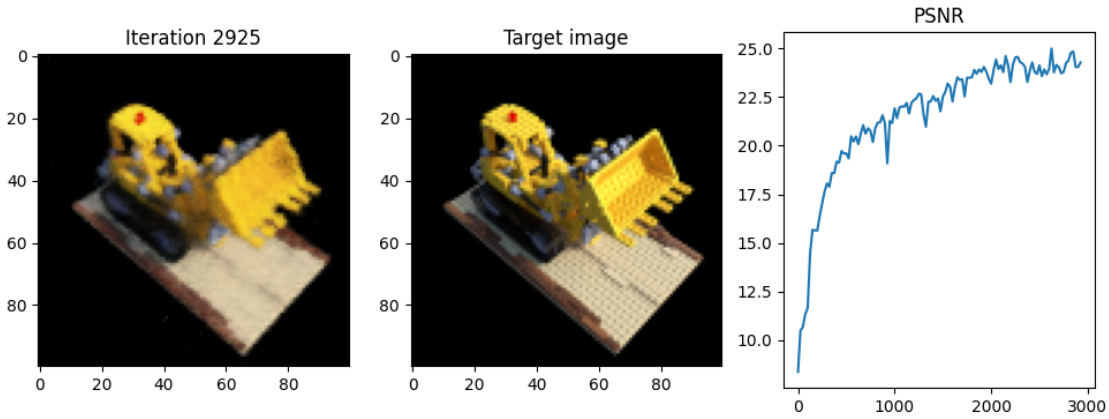
Iteration 2875 Loss: 0.0039 PSNR: 24.04 Time: 0.81 secs per iter, 39.25 mins in total



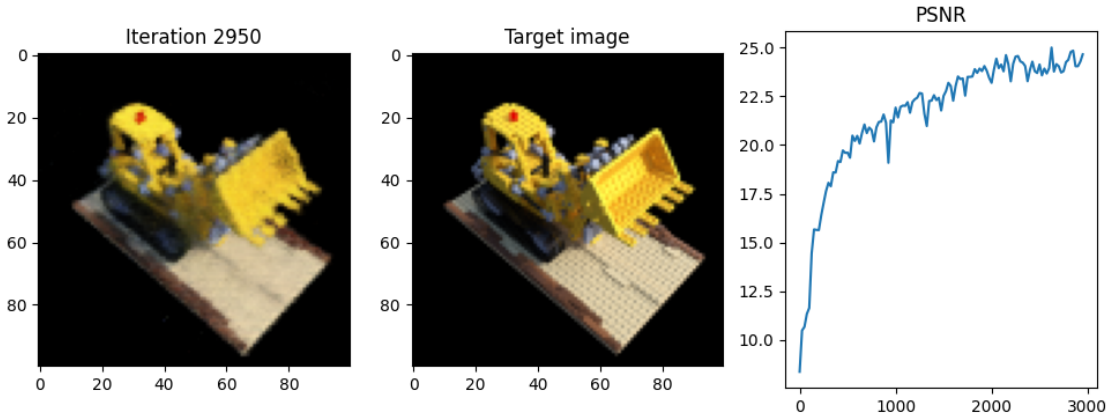
Iteration 2900 Loss: 0.0039 PSNR: 24.04 Time: 0.81 secs per iter, 39.59 mins in total



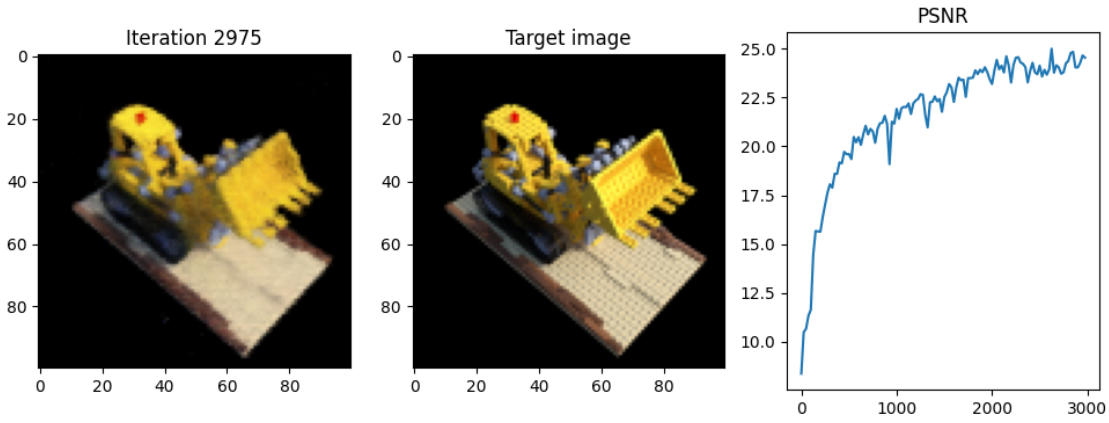
```
Iteration 2925  Loss: 0.0037  PSNR: 24.27  Time: 0.82 secs per iter,  39.93 mins
in total
```



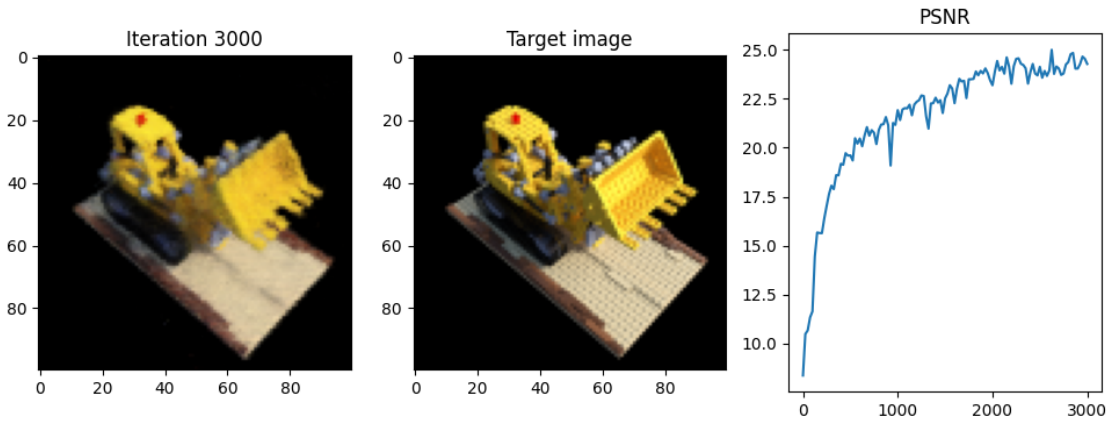

```
Iteration 2950  Loss: 0.0034  PSNR: 24.65  Time: 0.81 secs per iter,  40.26 mins
in total
```




```
Iteration 2975  Loss: 0.0035  PSNR: 24.54  Time: 0.81 secs per iter,  40.60 mins
in total
```




```
Iteration 3000  Loss: 0.0037  PSNR: 24.27  Time: 0.82 secs per iter,  40.94 mins
in total
```




```
Done!
```

[14]:
```
torch.save(model.state_dict(), 'nerf.pth')
```

[15]:
```
!sudo apt-get update &> /dev/null
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-plain-generic
!sudo apt-get install texlive-xetex texlive-fonts-recommended␣
 ↪texlive-generic-recommended &> /dev/null
!jupyter nbconvert --to pdf cis580_hw5.ipynb
```

```
Reading package lists… Done
Building dependency tree
Reading state information… Done
```

```
The following additional packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 javascript-common libapache-pom-java
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
  libgs9 libgs9-common libharfbuzz-icu0 libidn11 libijs-0.35 libjbig2dec0
  libjs-jquery libkpathsea6 libpdfbox-java libptexenc1 libruby2.7 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-minitest ruby-net-telnet
  ruby-power-assert ruby-test-unit ruby-xmlrpc ruby2.7 rubygems-integration
  t1utils teckit tex-common tex-gyre texlive-base texlive-binaries
  texlive-latex-base texlive-latex-extra texlive-latex-recommended
  texlive-pictures tipa xfonts-encodings xfonts-utils
Suggested packages:
  fonts-noto fonts-freefont-otf | fonts-freefont-ttf apache2 | lighttpd
  | httpd libavalon-framework-java libcommons-logging-java-doc
  libexcalibur-logkit-java liblog4j1.2-java poppler-utils ghostscript
  fonts-japanese-mincho | fonts-ipafont-mincho fonts-japanese-gothic
  | fonts-ipafont-gothic fonts-arphic-ukai fonts-arphic-uming fonts-nanum ri
  ruby-dev bundler debhelper gv | postscript-viewer perl-tk xpdf | pdf-viewer
  xzdec texlive-fonts-recommended-doc texlive-latex-base-doc python3-pygments
  icc-profiles libfile-which-perl libspreadsheet-parseexcel-perl
  texlive-latex-extra-doc texlive-latex-recommended-doc texlive-luatex
  texlive-pstricks dot2tex prerex ruby-tcltk | libtcltk-ruby
  texlive-pictures-doc vprerex default-jre-headless
The following NEW packages will be installed:
  dvisvgm fonts-droid-fallback fonts-lato fonts-lmodern fonts-noto-mono
  fonts-texgyre fonts-urw-base35 javascript-common libapache-pom-java
  libcommons-logging-java libcommons-parent-java libfontbox-java libfontenc1
  libgs9 libgs9-common libharfbuzz-icu0 libidn11 libijs-0.35 libjbig2dec0
  libjs-jquery libkpathsea6 libpdfbox-java libptexenc1 libruby2.7 libsynctex2
  libteckit0 libtexlua53 libtexluajit2 libwoff1 libzzip-0-13 lmodern
  poppler-data preview-latex-style rake ruby ruby-minitest ruby-net-telnet
  ruby-power-assert ruby-test-unit ruby-xmlrpc ruby2.7 rubygems-integration
  t1utils teckit tex-common tex-gyre texlive-base texlive-binaries
  texlive-fonts-recommended texlive-latex-base texlive-latex-extra
  texlive-latex-recommended texlive-pictures texlive-plain-generic
  texlive-xetex tipa xfonts-encodings xfonts-utils
0 upgraded, 58 newly installed, 0 to remove and 25 not upgraded.
Need to get 169 MB of archives.
After this operation, 537 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu focal/main amd64 fonts-droid-fallback all
1:6.0.1r16-1.1 [1,805 kB]
Get:2 http://archive.ubuntu.com/ubuntu focal/main amd64 fonts-lato all 2.0-2
[2,698 kB]
Get:3 http://archive.ubuntu.com/ubuntu focal/main amd64 poppler-data all 0.4.9-2
[1,475 kB]
Get:4 http://archive.ubuntu.com/ubuntu focal/universe amd64 tex-common all 6.13
[32.7 kB]
```

```
Get:5 http://archive.ubuntu.com/ubuntu focal/main amd64 fonts-urw-base35 all
20170801.1-3 [6,333 kB]
Get:6 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libgs9-common
all 9.50~dfsg-5ubuntu4.7 [681 kB]
Get:7 http://archive.ubuntu.com/ubuntu focal/main amd64 libidn11 amd64
1.33-2.2ubuntu2 [46.2 kB]
Get:8 http://archive.ubuntu.com/ubuntu focal/main amd64 libijs-0.35 amd64
0.35-15 [15.7 kB]
Get:9 http://archive.ubuntu.com/ubuntu focal/main amd64 libjbig2dec0 amd64
0.18-1ubuntu1 [60.0 kB]
Get:10 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libgs9 amd64
9.50~dfsg-5ubuntu4.7 [2,173 kB]
Get:11 http://archive.ubuntu.com/ubuntu focal/main amd64 libkpathsea6 amd64
2019.20190605.51237-3build2 [57.0 kB]
Get:12 http://archive.ubuntu.com/ubuntu focal/main amd64 libwoff1 amd64
1.0.2-1build2 [42.0 kB]
Get:13 http://archive.ubuntu.com/ubuntu focal/universe amd64 dvisvgm amd64
2.8.1-1build1 [1,048 kB]
Get:14 http://archive.ubuntu.com/ubuntu focal/universe amd64 fonts-lmodern all
2.004.5-6 [4,532 kB]
Get:15 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 fonts-noto-mono
all 20200323-1build1~ubuntu20.04.1 [80.6 kB]
Get:16 http://archive.ubuntu.com/ubuntu focal/universe amd64 fonts-texgyre all
20180621-3 [10.2 MB]
Get:17 http://archive.ubuntu.com/ubuntu focal/main amd64 javascript-common all
11 [6,066 B]
Get:18 http://archive.ubuntu.com/ubuntu focal/universe amd64 libapache-pom-java
all 18-1 [4,720 B]
Get:19 http://archive.ubuntu.com/ubuntu focal/universe amd64 libcommons-parent-
java all 43-1 [10.8 kB]
Get:20 http://archive.ubuntu.com/ubuntu focal/universe amd64 libcommons-logging-
java all 1.2-2 [60.3 kB]
Get:21 http://archive.ubuntu.com/ubuntu focal/main amd64 libfontenc1 amd64
1:1.1.4-0ubuntu1 [14.0 kB]
Get:22 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libharfbuzz-
icu0 amd64 2.6.4-1ubuntu4.2 [5,580 B]
Get:23 http://archive.ubuntu.com/ubuntu focal/main amd64 libjs-jquery all
3.3.1~dfsg-3 [329 kB]
Get:24 http://archive.ubuntu.com/ubuntu focal/main amd64 libptexenc1 amd64
2019.20190605.51237-3build2 [35.5 kB]
Get:25 http://archive.ubuntu.com/ubuntu focal/main amd64 rubygems-integration
all 1.16 [5,092 B]
Get:26 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 ruby2.7 amd64
2.7.0-5ubuntu1.10 [95.6 kB]
Get:27 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby amd64 1:2.7+1
[5,412 B]
Get:28 http://archive.ubuntu.com/ubuntu focal/main amd64 rake all 13.0.1-4 [61.6
kB]
```

```
Get:29 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby-minitest all
5.13.0-1 [40.9 kB]
Get:30 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby-net-telnet all
0.1.1-2 [12.6 kB]
Get:31 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby-power-assert all
1.1.7-1 [11.4 kB]
Get:32 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby-test-unit all
3.3.5-1 [73.2 kB]
Get:33 http://archive.ubuntu.com/ubuntu focal/main amd64 ruby-xmlrpc all 0.3.0-2
[23.8 kB]
Get:34 http://archive.ubuntu.com/ubuntu focal-updates/main amd64 libruby2.7
amd64 2.7.0-5ubuntu1.10 [3,532 kB]
Get:35 http://archive.ubuntu.com/ubuntu focal/main amd64 libsynctex2 amd64
2019.20190605.51237-3build2 [55.0 kB]
Get:36 http://archive.ubuntu.com/ubuntu focal/universe amd64 libteckit0 amd64
2.5.8+ds2-5ubuntu2 [320 kB]
Get:37 http://archive.ubuntu.com/ubuntu focal/main amd64 libtexlua53 amd64
2019.20190605.51237-3build2 [105 kB]
Get:38 http://archive.ubuntu.com/ubuntu focal/main amd64 libtexluajit2 amd64
2019.20190605.51237-3build2 [235 kB]
Get:39 http://archive.ubuntu.com/ubuntu focal/universe amd64 libzzip-0-13 amd64
0.13.62-3.2ubuntu1 [26.2 kB]
Get:40 http://archive.ubuntu.com/ubuntu focal/main amd64 xfonts-encodings all
1:1.0.5-0ubuntu1 [573 kB]
Get:41 http://archive.ubuntu.com/ubuntu focal/main amd64 xfonts-utils amd64
1:7.7+6 [91.5 kB]
Get:42 http://archive.ubuntu.com/ubuntu focal/universe amd64 lmodern all
2.004.5-6 [9,474 kB]
Get:43 http://archive.ubuntu.com/ubuntu focal/universe amd64 preview-latex-style
all 11.91-2ubuntu2 [184 kB]
Get:44 http://archive.ubuntu.com/ubuntu focal/main amd64 t1utils amd64 1.41-3
[56.1 kB]
Get:45 http://archive.ubuntu.com/ubuntu focal/universe amd64 teckit amd64
2.5.8+ds2-5ubuntu2 [687 kB]
Get:46 http://archive.ubuntu.com/ubuntu focal/universe amd64 tex-gyre all
20180621-3 [6,209 kB]
Get:47 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-binaries
amd64 2019.20190605.51237-3build2 [8,041 kB]
Get:48 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-base all
2019.20200218-1 [20.8 MB]
Get:49 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-fonts-
recommended all 2019.20200218-1 [4,972 kB]
Get:50 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-latex-base
all 2019.20200218-1 [990 kB]
Get:51 http://archive.ubuntu.com/ubuntu focal/universe amd64 libfontbox-java all
1:1.8.16-2 [207 kB]
Get:52 http://archive.ubuntu.com/ubuntu focal/universe amd64 libpdfbox-java all
1:1.8.16-2 [5,199 kB]
```

```
Get:53 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-latex-
recommended all 2019.20200218-1 [15.7 MB]
Get:54 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-pictures
all 2019.20200218-1 [4,492 kB]
Get:55 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-latex-extra
all 2019.202000218-1 [12.5 MB]
Get:56 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-plain-
generic all 2019.202000218-1 [24.6 MB]
Get:57 http://archive.ubuntu.com/ubuntu focal/universe amd64 tipa all 2:1.3-20
[2,978 kB]
Get:58 http://archive.ubuntu.com/ubuntu focal/universe amd64 texlive-xetex all
2019.20200218-1 [14.6 MB]
Fetched 169 MB in 26s (6,576 kB/s)
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based
frontend cannot be used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76,
<> line 58.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (This frontend requires a controlling tty.)
debconf: falling back to frontend: Teletype
dpkg-preconfigure: unable to re-open stdin:
Selecting previously unselected package fonts-droid-fallback.
(Reading database … 122518 files and directories currently installed.)
Preparing to unpack …/00-fonts-droid-fallback_1%3a6.0.1r16-1.1_all.deb …
Unpacking fonts-droid-fallback (1:6.0.1r16-1.1) …
Selecting previously unselected package fonts-lato.
Preparing to unpack …/01-fonts-lato_2.0-2_all.deb …
Unpacking fonts-lato (2.0-2) …
Selecting previously unselected package poppler-data.
Preparing to unpack …/02-poppler-data_0.4.9-2_all.deb …
Unpacking poppler-data (0.4.9-2) …
Selecting previously unselected package tex-common.
Preparing to unpack …/03-tex-common_6.13_all.deb …
Unpacking tex-common (6.13) …
Selecting previously unselected package fonts-urw-base35.
Preparing to unpack …/04-fonts-urw-base35_20170801.1-3_all.deb …
Unpacking fonts-urw-base35 (20170801.1-3) …
Selecting previously unselected package libgs9-common.
Preparing to unpack …/05-libgs9-common_9.50~dfsg-5ubuntu4.7_all.deb …
Unpacking libgs9-common (9.50~dfsg-5ubuntu4.7) …
Selecting previously unselected package libidn11:amd64.
Preparing to unpack …/06-libidn11_1.33-2.2ubuntu2_amd64.deb …
Unpacking libidn11:amd64 (1.33-2.2ubuntu2) …
Selecting previously unselected package libijs-0.35:amd64.
Preparing to unpack …/07-libijs-0.35_0.35-15_amd64.deb …
Unpacking libijs-0.35:amd64 (0.35-15) …
Selecting previously unselected package libjbig2dec0:amd64.
```

```
Preparing to unpack …/08-libjbig2dec0_0.18-1ubuntu1_amd64.deb …
Unpacking libjbig2dec0:amd64 (0.18-1ubuntu1) …
Selecting previously unselected package libgs9:amd64.
Preparing to unpack …/09-libgs9_9.50~dfsg-5ubuntu4.7_amd64.deb …
Unpacking libgs9:amd64 (9.50~dfsg-5ubuntu4.7) …
Selecting previously unselected package libkpathsea6:amd64.
Preparing to unpack …/10-libkpathsea6_2019.20190605.51237-3build2_amd64.deb
…
Unpacking libkpathsea6:amd64 (2019.20190605.51237-3build2) …
Selecting previously unselected package libwoff1:amd64.
Preparing to unpack …/11-libwoff1_1.0.2-1build2_amd64.deb …
Unpacking libwoff1:amd64 (1.0.2-1build2) …
Selecting previously unselected package dvisvgm.
Preparing to unpack …/12-dvisvgm_2.8.1-1build1_amd64.deb …
Unpacking dvisvgm (2.8.1-1build1) …
Selecting previously unselected package fonts-lmodern.
Preparing to unpack …/13-fonts-lmodern_2.004.5-6_all.deb …
Unpacking fonts-lmodern (2.004.5-6) …
Selecting previously unselected package fonts-noto-mono.
Preparing to unpack …/14-fonts-noto-
mono_20200323-1build1~ubuntu20.04.1_all.deb …
Unpacking fonts-noto-mono (20200323-1build1~ubuntu20.04.1) …
Selecting previously unselected package fonts-texgyre.
Preparing to unpack …/15-fonts-texgyre_20180621-3_all.deb …
Unpacking fonts-texgyre (20180621-3) …
Selecting previously unselected package javascript-common.
Preparing to unpack …/16-javascript-common_11_all.deb …
Unpacking javascript-common (11) …
Selecting previously unselected package libapache-pom-java.
Preparing to unpack …/17-libapache-pom-java_18-1_all.deb …
Unpacking libapache-pom-java (18-1) …
Selecting previously unselected package libcommons-parent-java.
Preparing to unpack …/18-libcommons-parent-java_43-1_all.deb …
Unpacking libcommons-parent-java (43-1) …
Selecting previously unselected package libcommons-logging-java.
Preparing to unpack …/19-libcommons-logging-java_1.2-2_all.deb …
Unpacking libcommons-logging-java (1.2-2) …
Selecting previously unselected package libfontenc1:amd64.
Preparing to unpack …/20-libfontenc1_1%3a1.1.4-0ubuntu1_amd64.deb …
Unpacking libfontenc1:amd64 (1:1.1.4-0ubuntu1) …
Selecting previously unselected package libharfbuzz-icu0:amd64.
Preparing to unpack …/21-libharfbuzz-icu0_2.6.4-1ubuntu4.2_amd64.deb …
Unpacking libharfbuzz-icu0:amd64 (2.6.4-1ubuntu4.2) …
Selecting previously unselected package libjs-jquery.
Preparing to unpack …/22-libjs-jquery_3.3.1~dfsg-3_all.deb …
Unpacking libjs-jquery (3.3.1~dfsg-3) …
Selecting previously unselected package libptexenc1:amd64.
Preparing to unpack …/23-libptexenc1_2019.20190605.51237-3build2_amd64.deb …
```

```
Unpacking libptexenc1:amd64 (2019.20190605.51237-3build2) …
Selecting previously unselected package rubygems-integration.
Preparing to unpack …/24-rubygems-integration_1.16_all.deb …
Unpacking rubygems-integration (1.16) …
Selecting previously unselected package ruby2.7.
Preparing to unpack …/25-ruby2.7_2.7.0-5ubuntu1.10_amd64.deb …
Unpacking ruby2.7 (2.7.0-5ubuntu1.10) …
Selecting previously unselected package ruby.
Preparing to unpack …/26-ruby_1%3a2.7+1_amd64.deb …
Unpacking ruby (1:2.7+1) …
Selecting previously unselected package rake.
Preparing to unpack …/27-rake_13.0.1-4_all.deb …
Unpacking rake (13.0.1-4) …
Selecting previously unselected package ruby-minitest.
Preparing to unpack …/28-ruby-minitest_5.13.0-1_all.deb …
Unpacking ruby-minitest (5.13.0-1) …
Selecting previously unselected package ruby-net-telnet.
Preparing to unpack …/29-ruby-net-telnet_0.1.1-2_all.deb …
Unpacking ruby-net-telnet (0.1.1-2) …
Selecting previously unselected package ruby-power-assert.
Preparing to unpack …/30-ruby-power-assert_1.1.7-1_all.deb …
Unpacking ruby-power-assert (1.1.7-1) …
Selecting previously unselected package ruby-test-unit.
Preparing to unpack …/31-ruby-test-unit_3.3.5-1_all.deb …
Unpacking ruby-test-unit (3.3.5-1) …
Selecting previously unselected package ruby-xmlrpc.
Preparing to unpack …/32-ruby-xmlrpc_0.3.0-2_all.deb …
Unpacking ruby-xmlrpc (0.3.0-2) …
Selecting previously unselected package libruby2.7:amd64.
Preparing to unpack …/33-libruby2.7_2.7.0-5ubuntu1.10_amd64.deb …
Unpacking libruby2.7:amd64 (2.7.0-5ubuntu1.10) …
Selecting previously unselected package libsynctex2:amd64.
Preparing to unpack …/34-libsynctex2_2019.20190605.51237-3build2_amd64.deb …
Unpacking libsynctex2:amd64 (2019.20190605.51237-3build2) …
Selecting previously unselected package libteckit0:amd64.
Preparing to unpack …/35-libteckit0_2.5.8+ds2-5ubuntu2_amd64.deb …
Unpacking libteckit0:amd64 (2.5.8+ds2-5ubuntu2) …
Selecting previously unselected package libtexlua53:amd64.
Preparing to unpack …/36-libtexlua53_2019.20190605.51237-3build2_amd64.deb …
Unpacking libtexlua53:amd64 (2019.20190605.51237-3build2) …
Selecting previously unselected package libtexluajit2:amd64.
Preparing to unpack …/37-libtexluajit2_2019.20190605.51237-3build2_amd64.deb
…
Unpacking libtexluajit2:amd64 (2019.20190605.51237-3build2) …
Selecting previously unselected package libzzip-0-13:amd64.
Preparing to unpack …/38-libzzip-0-13_0.13.62-3.2ubuntu1_amd64.deb …
Unpacking libzzip-0-13:amd64 (0.13.62-3.2ubuntu1) …
Selecting previously unselected package xfonts-encodings.
```

```
Preparing to unpack …/39-xfonts-encodings_1%3a1.0.5-0ubuntu1_all.deb …
Unpacking xfonts-encodings (1:1.0.5-0ubuntu1) …
Selecting previously unselected package xfonts-utils.
Preparing to unpack …/40-xfonts-utils_1%3a7.7+6_amd64.deb …
Unpacking xfonts-utils (1:7.7+6) …
Selecting previously unselected package lmodern.
Preparing to unpack …/41-lmodern_2.004.5-6_all.deb …
Unpacking lmodern (2.004.5-6) …
Selecting previously unselected package preview-latex-style.
Preparing to unpack …/42-preview-latex-style_11.91-2ubuntu2_all.deb …
Unpacking preview-latex-style (11.91-2ubuntu2) …
Selecting previously unselected package t1utils.
Preparing to unpack …/43-t1utils_1.41-3_amd64.deb …
Unpacking t1utils (1.41-3) …
Selecting previously unselected package teckit.
Preparing to unpack …/44-teckit_2.5.8+ds2-5ubuntu2_amd64.deb …
Unpacking teckit (2.5.8+ds2-5ubuntu2) …
Selecting previously unselected package tex-gyre.
Preparing to unpack …/45-tex-gyre_20180621-3_all.deb …
Unpacking tex-gyre (20180621-3) …
Selecting previously unselected package texlive-binaries.
Preparing to unpack …/46-texlive-
binaries_2019.20190605.51237-3build2_amd64.deb …
Unpacking texlive-binaries (2019.20190605.51237-3build2) …
Selecting previously unselected package texlive-base.
Preparing to unpack …/47-texlive-base_2019.20200218-1_all.deb …
Unpacking texlive-base (2019.20200218-1) …
Selecting previously unselected package texlive-fonts-recommended.
Preparing to unpack …/48-texlive-fonts-recommended_2019.20200218-1_all.deb …
Unpacking texlive-fonts-recommended (2019.20200218-1) …
Selecting previously unselected package texlive-latex-base.
Preparing to unpack …/49-texlive-latex-base_2019.20200218-1_all.deb …
Unpacking texlive-latex-base (2019.20200218-1) …
Selecting previously unselected package libfontbox-java.
Preparing to unpack …/50-libfontbox-java_1%3a1.8.16-2_all.deb …
Unpacking libfontbox-java (1:1.8.16-2) …
Selecting previously unselected package libpdfbox-java.
Preparing to unpack …/51-libpdfbox-java_1%3a1.8.16-2_all.deb …
Unpacking libpdfbox-java (1:1.8.16-2) …
Selecting previously unselected package texlive-latex-recommended.
Preparing to unpack …/52-texlive-latex-recommended_2019.20200218-1_all.deb …
Unpacking texlive-latex-recommended (2019.20200218-1) …
Selecting previously unselected package texlive-pictures.
Preparing to unpack …/53-texlive-pictures_2019.20200218-1_all.deb …
Unpacking texlive-pictures (2019.20200218-1) …
Selecting previously unselected package texlive-latex-extra.
Preparing to unpack …/54-texlive-latex-extra_2019.202000218-1_all.deb …
Unpacking texlive-latex-extra (2019.202000218-1) …
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-15-99dd16edb3c2> in <cell line: 2>()
      1 get_ipython().system('sudo apt-get update &> /dev/null')
----> 2 get_ipython().system('sudo apt-get install texlive-xetex␣
 ↪texlive-fonts-recommended texlive-plain-generic')
      3 get_ipython().system('sudo apt-get install texlive-xetex␣
 ↪texlive-fonts-recommended texlive-generic-recommended &> /dev/null')
      4 get_ipython().system('jupyter nbconvert --to pdf cis580_hw5.ipynb')

/usr/local/lib/python3.10/dist-packages/google/colab/_shell.py in system(self,␣
 ↪*args, **kwargs)
     97       kwargs.update({'also_return_output': True})
     98
---> 99     output = _system_commands._system_compat(self, *args, **kwargs)  #␣
 ↪pylint:disable=protected-access
    100
    101     if pip_warn:

/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in␣
 ↪_system_compat(shell, cmd, also_return_output)
    451   # is expected to call this function, thus adding one level of nesting␣
 ↪to the
    452   # stack.
--> 453   result = _run_command(
    454       shell.var_expand(cmd, depth=2), clear_streamed_output=False
    455   )

/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in␣
 ↪_run_command(cmd, clear_streamed_output)
    201       os.close(child_pty)
    202
--> 203       return _monitor_process(parent_pty, epoll, p, cmd,␣
 ↪update_stdin_widget)
    204   finally:
    205     epoll.close()

/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in␣
 ↪_monitor_process(parent_pty, epoll, p, cmd, update_stdin_widget)
    231   while True:
    232     try:
--> 233       result = _poll_process(parent_pty, epoll, p, cmd, decoder, state)
    234       if result is not None:
    235         return result
```

```
/usr/local/lib/python3.10/dist-packages/google/colab/_system_commands.py in␣
 ↪_poll_process(parent_pty, epoll, p, cmd, decoder, state)
    279   output_available = False
    280
--> 281   events = epoll.poll()
    282   input_events = []
    283   for _, event in events:

KeyboardInterrupt:
```