Homework 2

Methods Defined

- **compute_camera_matrix** : generating intrinsic matrix, which tells us the specification of our camera, as well as projection matrix, which is used to create the depth, mask, and color image for our dataset
- **RGBDataset** : used to specify the transformation performed on our image dataset as well as specifications on how to retrieve the images
- **MiniUNet** : Model specification, consists of the layers that we use to perform a forward pass as well as the structure of the Neural Network
- **train** : Our training function, consists of backward propagation to update the weights as well as metrics computation
- **val**: Our validation function, consists of metrics computation of our validation dataset to make sure that the mask prediction that we have is accurate
- **main**: Composing our function to perform the training, validation, as well as saving our prediction and model checkpoints
- **gen_obj_depth**: Filtering out the depth image to only consist the object of interest, using ground truth image mask / predicted image mask
- **obj_depth2pts**: Converting the depth image that we have into 3-dimensional points in world coordinate for a single object, using ground truth image mask / predicted image mask
- **align_pts**: Perform ICP algorithm, using Procrustes analysis as the initial transformation matrix
- **estimate_pose**: Specifying the transformation required to transform the mesh points that we generate from our model object to match the depth points that we generate through our camera, in order to improve our 3D model prediction. This is done for every object that we have
- **main**: Combining all of the functions, generating pose for our model based on the ground truth mask and predicted mask, as well as the ply points that combine transformed mesh point cloud and our depth points.

2.1

The input of the task is the rgb image, and the output is the mask image, where each colour indicates the presence of that object in that pixel (including background). The supervision that we need is the ground truth on this image segmentation, and that is why we generate the dataset by dropping different objects on our simulation and simulate picture taking, knowing the ground truth of the image segmentation.

2.4.1

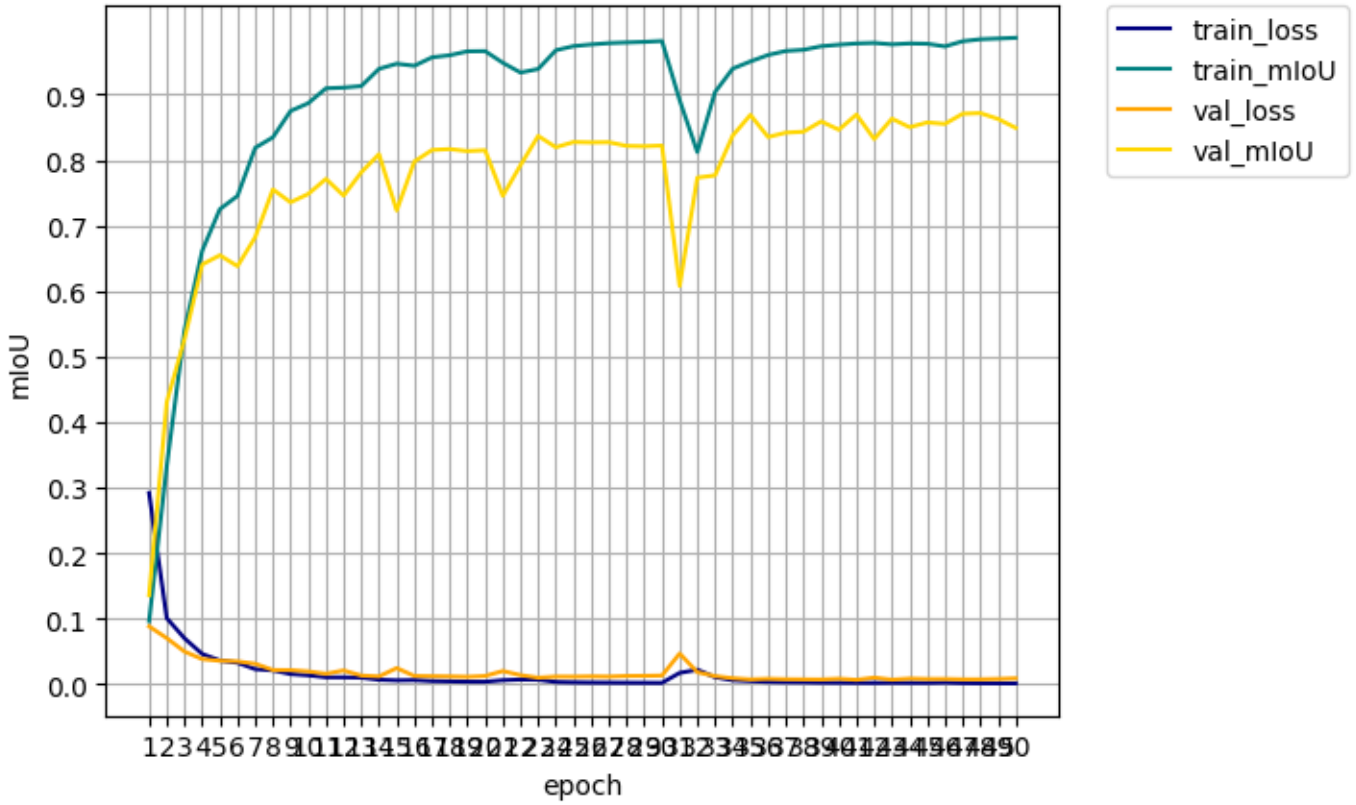
The 3x3 kernels is mainly used to capture the information about specific area of the image, where we want to identify features present in each field of the pictures. 1x1 kernels is used to 1.) combine information from the same pixel from different channels into one and 2.) change the number of channel to 6, the required output

2.4.2

The output require 6 channel, because each of the channel represent the "likelihood" (if its normalized through softmax, then it might represent probability) our our prediction on that pixel belonging to that object class. The ground truth only have one channel because 1.) that is the format needed by PyTorch CrossEntropyLoss on the target variable and 2.) Because its a definite value, not probability, we can represent each pixel with a single number, the class it belongs to.

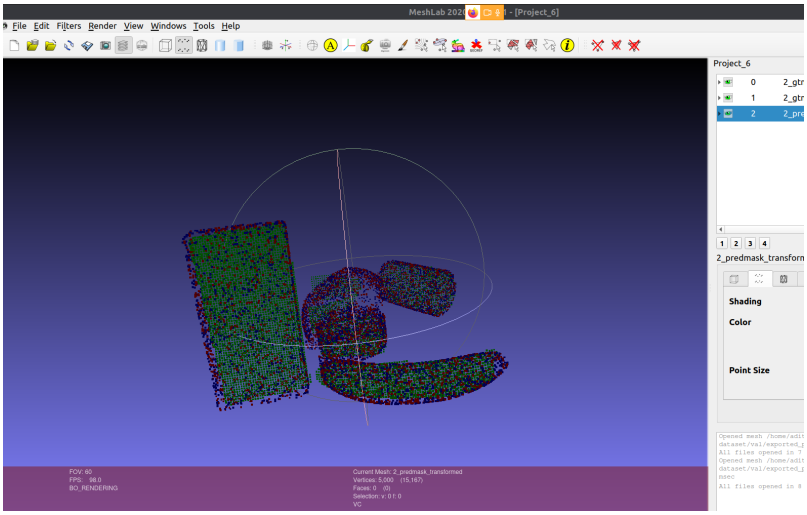
2.5

I decided to increase the number of epoch, as the learning curve has yet to converge. I used the learning rate specified in the Ed Discussion (lr=$1e^{-3}$).



3.2

Upon multiple experiments, I found out that increasing the number of iteration and decreasing the threshold (the minimum cost to be reached so that the algorithm stops) causes an increase in the performance.



pybullet build time: Mar 1 2022 23:24:40
Evaluating predicted pose using ground truth mask
Average closest point distance of sugar_box
average: 0.0001521307563875675
min: 0.0001195201068567397 max: 0.00018868298692726977
Average closest point distance of tomato_soup_can
average: 0.00014247274234925557
min: 0.00010712678143929556 max: 0.0002495566293216567
Average closest point distance of tuna_fish_can
average: 8.947437402711251e-05
min: 5.573124511679464e-05 max: 0.0001271486458973312
Average closest point distance of banana
average: 0.0001606813130795856
min: 5.906361152646962e-05 max: 0.00039194352985052474
Average closest point distance of bowl
average: 0.0003052187938015659
min: 0.00010443860590918366 max: 0.0005339901217370343
Evaluating predicted pose using predicted mask
Average closest point distance of sugar_box
average: 0.00017813881060496162
min: 0.00013005241983562047 max: 0.0002515289283932871
Average closest point distance of tomato_soup_can
average: 0.000253880287563217
min: 9.001365295765625e-05 max: 0.0008670089534424794
Average closest point distance of tuna_fish_can
average: 0.0001243468512612829
min: 7.610991876967224e-05 max: 0.0002725496893955053
Average closest point distance of banana
average: nan
min: 6.002099077728995e-05 max: 0.00238360511101044 5
Average closest point distance of bowl
average: 0.0004694681478624713
min: 0.0001112274456995409 max: 0.000592657746612707