CS 763: LAB7b

Group 10
Piyush Onkar(160050012) Apoorva Agarwal(203050018)

Entire data without splitting is used for training and testing the model considering that it is mentioned in the Task document that the test file will be used by TAs to test our model.
To use the pre-trained model for testing change the file name in line7 in eval.py with test_data = <file_path>.

Information available in dataset :
View 1 extrinsic parameter i.e. rotation and translation vectors
View 2 intrinsic parameter i.e. focal length
As the information given in the dataset is not complete for any of the views we have tried 2 different approaches.

Approach1
In this approach the model used is identical to the one used in 7a task.
With the assumption that both cameras have the same intrinsic and extrinsic parameters a camera matrix is computed using computeCameraMatrix() function in the code.
After we get predicted 3d points as output from the last layer of the model, the camera matrix is used to get reprojected 2d points.
Then mean per joint position loss between the ground truth 2d points and reprojected 2d points is used to train the model.
2D Reprojection Loss (cal_mpjpe loss) = 5.14
3D Loss (Test_loss) = 1.46

Loss Computation
There are 2 loss functions

1. cal_mpjpe(pose_1, mtx, pred_3d, avg=True)
    - This loss is used to train the model.
    - It takes ground truth 2d points, camera matrix and predicted 3d points as input.
    - It calls another function(reconstruct(x_3d,mtx)) to reproject 2d points from predicted 3d points. Mean per joint l2 norm as loss, is computed between the ground truth 2d points and reprojected 2d points.
2. test_loss(pose_1, pose_2, avg=True)
    - This loss is used just to test the model
    - It computes mean per joint position loss between ground truth 3d points and predicted 3d points.

Model1

```
LiftModel1(
  (inputLayer): Linear(in_features=30, out_features=1024, bias=True)
  (residualLayer): ModuleList(
    (0): ResiBlock(
      (res_units): ModuleList(
        (0): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.1, inplace=False)
        )
        (1): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.1, inplace=False)
        )
      )
    )
    (1): ResiBlock(
      (res_units): ModuleList(
        (0): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.1, inplace=False)
        )
        (1): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.1, inplace=False)
        )
      )
    )
  )
  (outputLayer): Linear(in_features=1024, out_features=45, bias=True)
)
```

Approach2

In this approach the model is modified.
Concatenation of joint_2d_1, joint_2d_2, rot, transl, focal_length is given as input feature vector to model.
Extra fully connected layer is added that transforms the 3d points to 2d points dimension.
The model is trained on mean per joint position loss between the ground truth 2d points and predicted 2d points.
The predicted_3d points are taken from the output of the intermediate layer(interLayer in model).
2D Reprojection Loss (cal_mpjpe loss) = 0.66
3D Loss (Test_loss) = 1.67

Loss Computation

There are 2 loss functions

1. cal_mpjpe(pose_1, pose_2, avg=True)
    - This loss is used to train the model.
    - It takes ground truth 2d points and output of the last layer of model as predicted 2d points as input.
    - It computes Mean per joint l2 norm as loss between the ground truth 2d points and predicted 2d points.

2. test_loss(pose_1, pose_2, avg=True)
    - This loss is used just to test the model
    - It computes mean per joint position loss between ground truth 3d points and predicted 3d points. Predicted 3d points are taken from the output of the intermediate layer(interLayer in model).

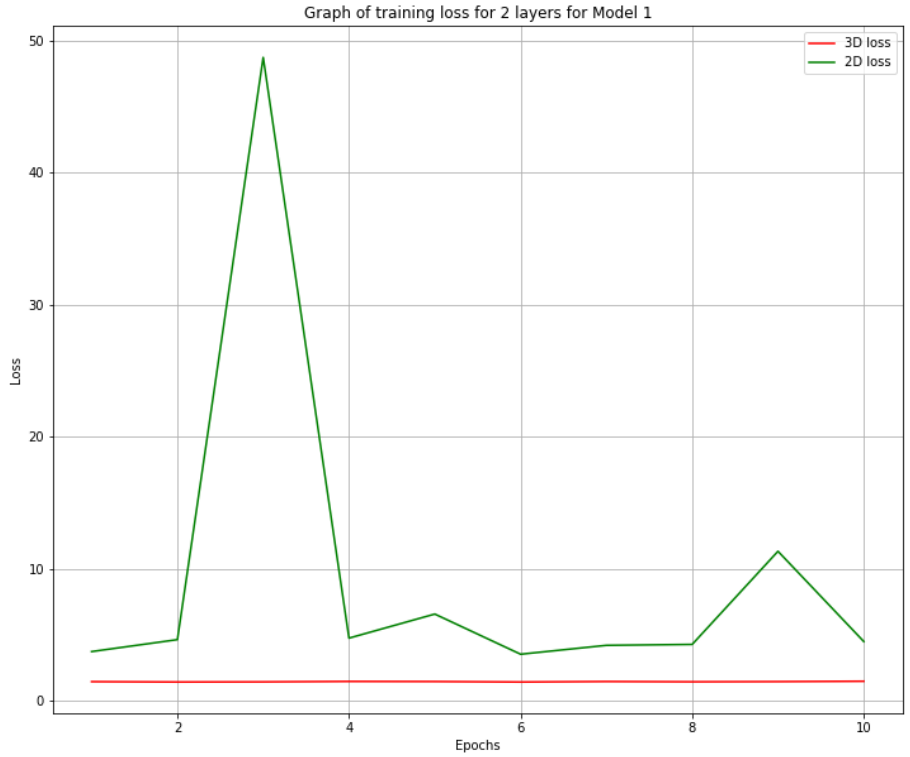
Model2

```
LiftModel2(
  (inputLayer): Linear(in_features=74, out_features=1024, bias=True)
  (residualLayer): ModuleList(
    (0): ResiBlock(
      (res_units): ModuleList(
        (0): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.05, inplace=False)
        )
        (1): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.05, inplace=False)
        )
      )
    )
    (1): ResiBlock(
      (res_units): ModuleList(
        (0): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.05, inplace=False)
        )
        (1): Sequential(
          (0): Linear(in_features=1024, out_features=1024, bias=True)
          (1): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (2): ReLU()
          (3): Dropout(p=0.05, inplace=False)
        )
      )
    )
  )
  (interLayer): Linear(in_features=1024, out_features=45, bias=True)
  (outputLayer): Linear(in_features=45, out_features=30, bias=True)
)
```

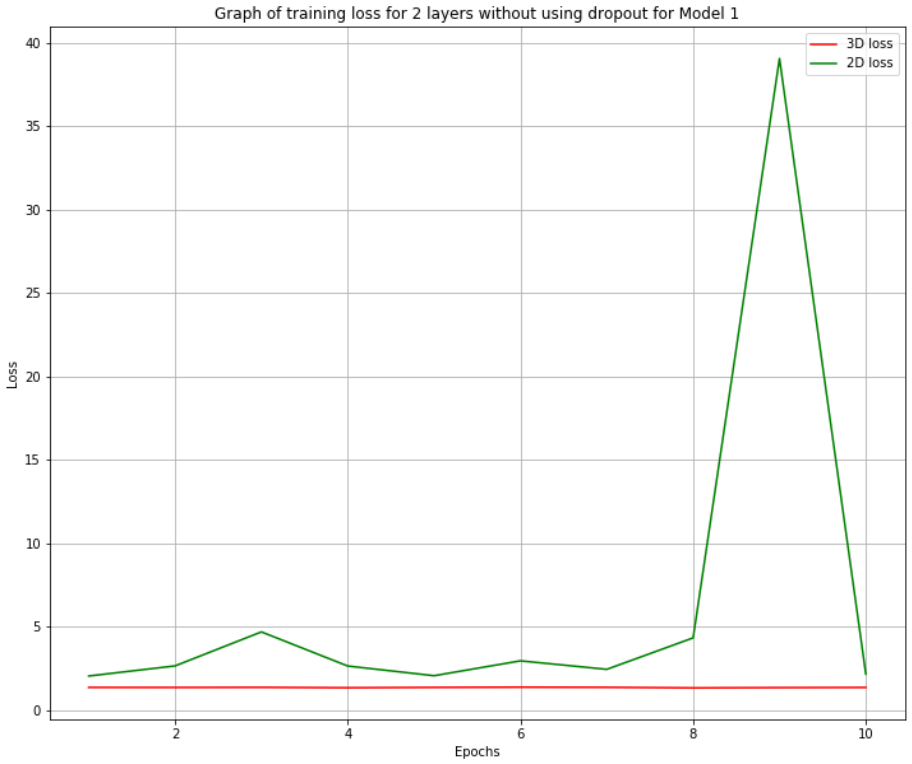
Experiments Performed :

For Model1 :

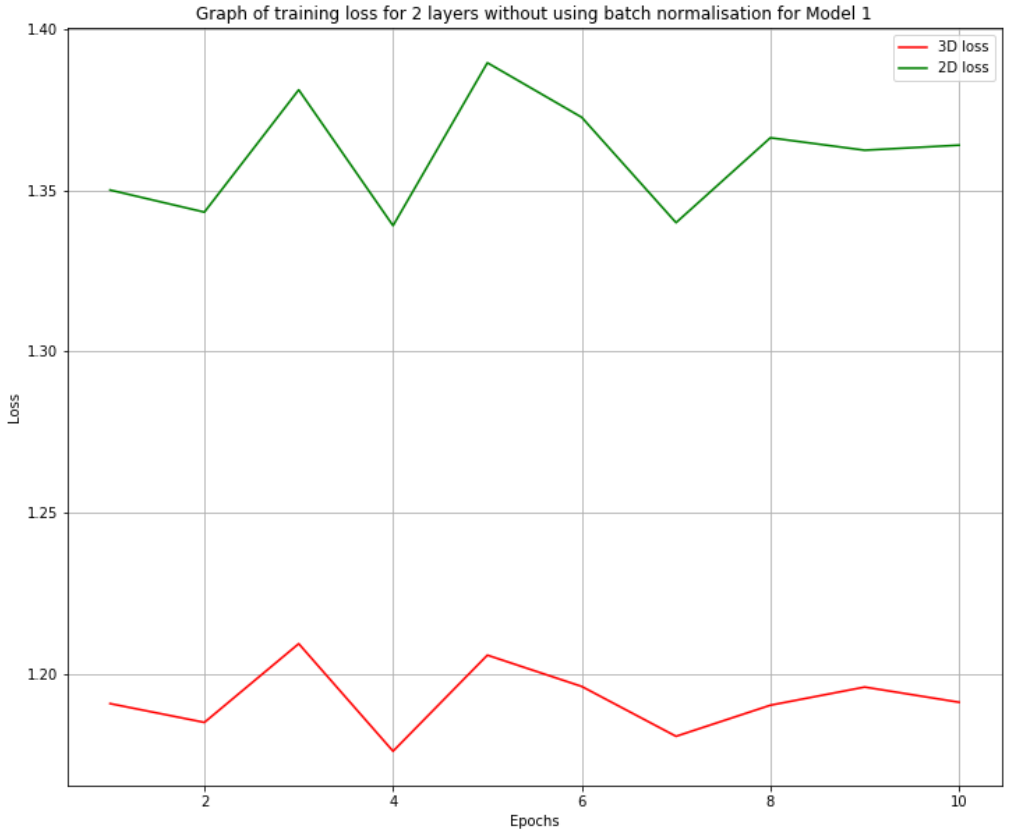
Plot of Loss vs Number of epochs for 2 Layers :



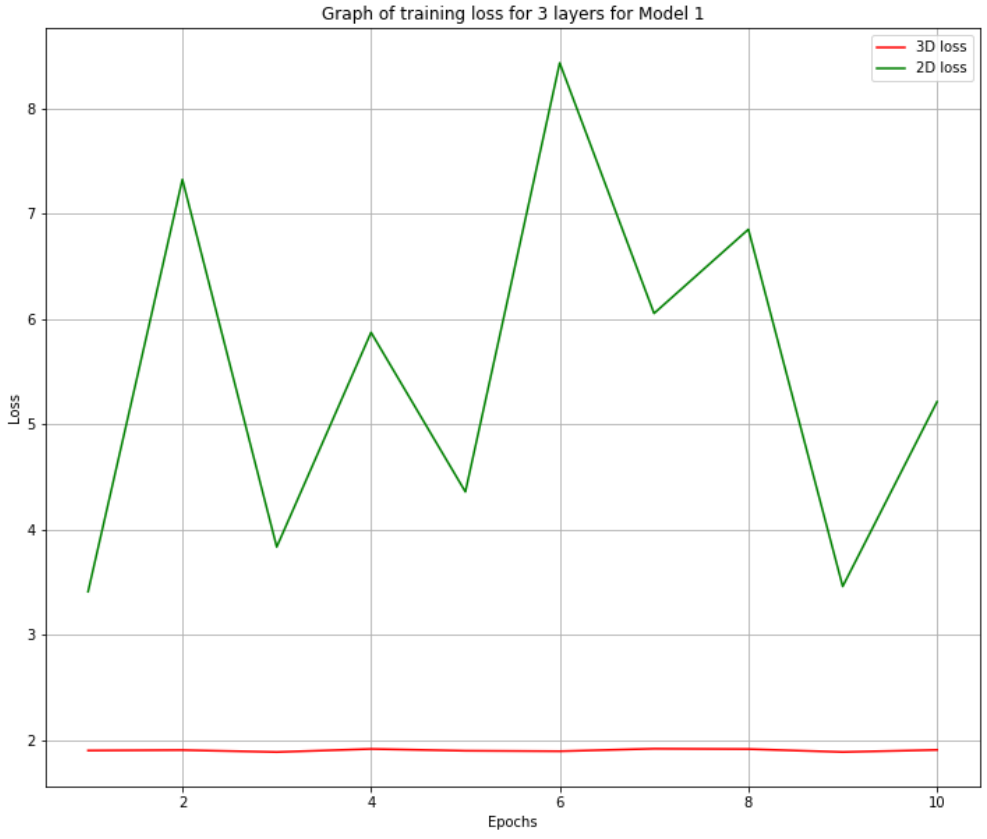
Plot of Loss vs Number of epochs for 2 Layers without using Dropout Layer in Residual Block :



Plot of Loss vs Number of epochs for 2 Layers without using Batch Normalisation in Residual Block :

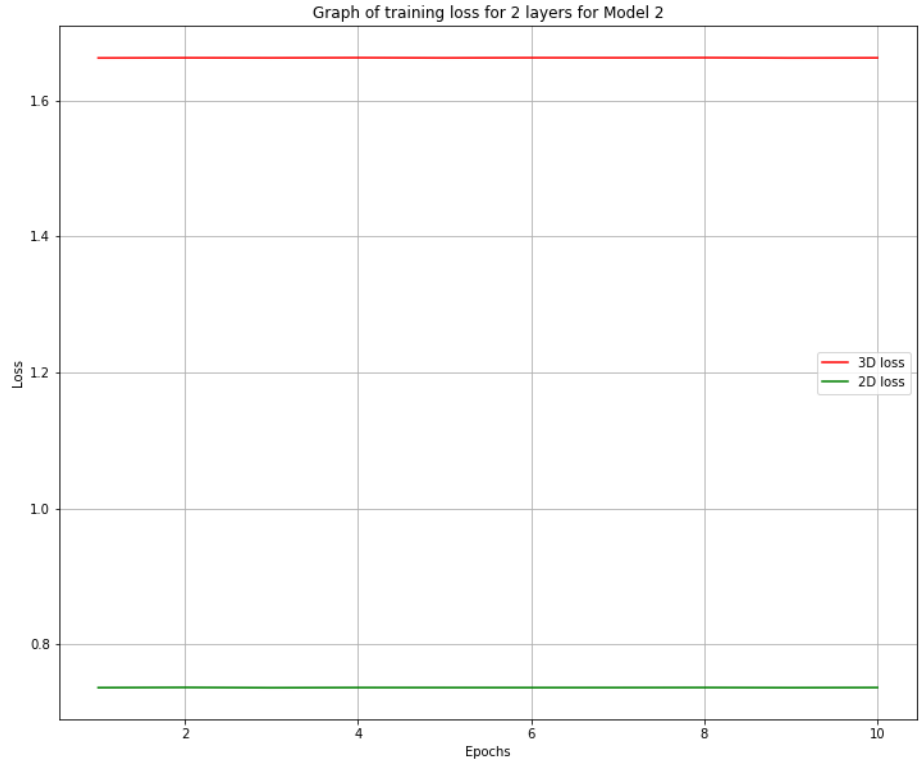


Plot of Loss vs Number of epochs for 3 Layers :

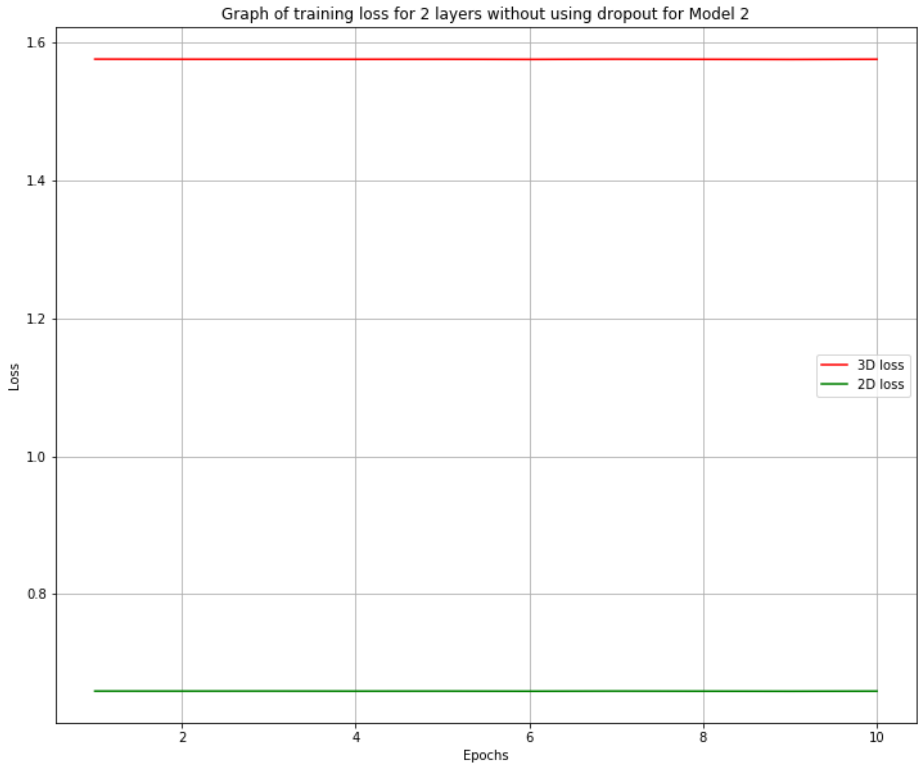


For Model 2:

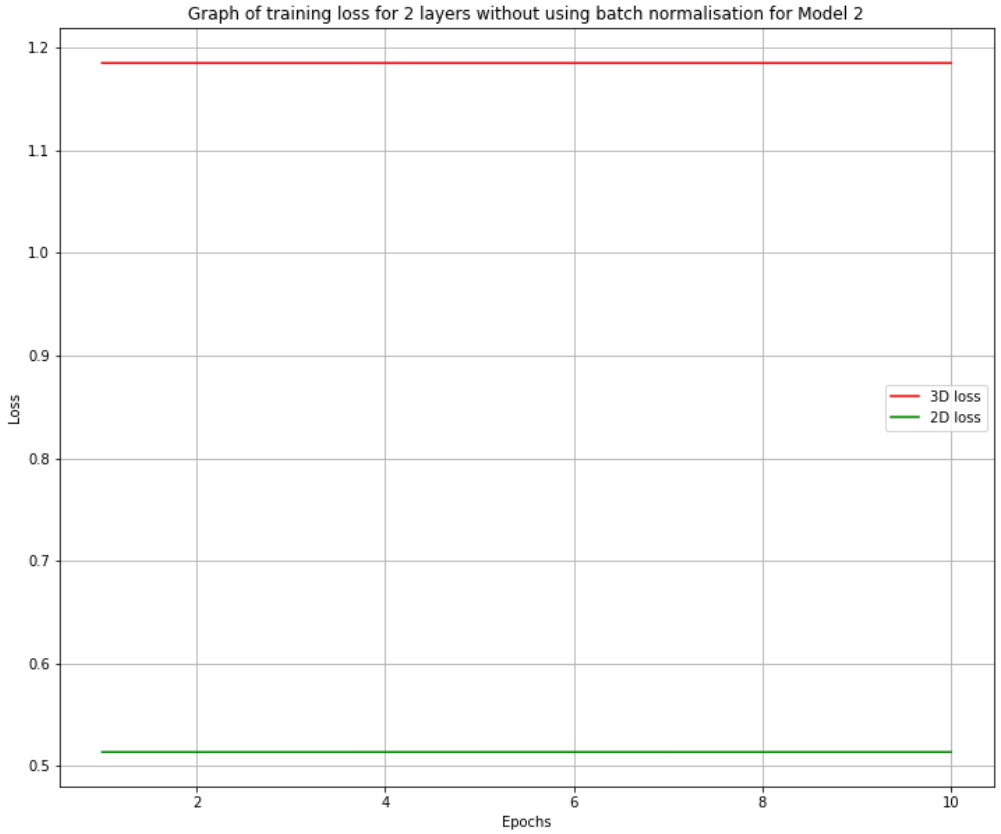
Plot of Loss vs Number of epochs for 2 Layers :



Plot of Loss vs Number of epochs for 2 Layers without using Dropout Layer in Residual Block :



Plot of Loss vs Number of epochs for 2 Layers without using Batch Normalisation in Residual Block :



Plot of Loss vs Number of epochs for 3 Layers :