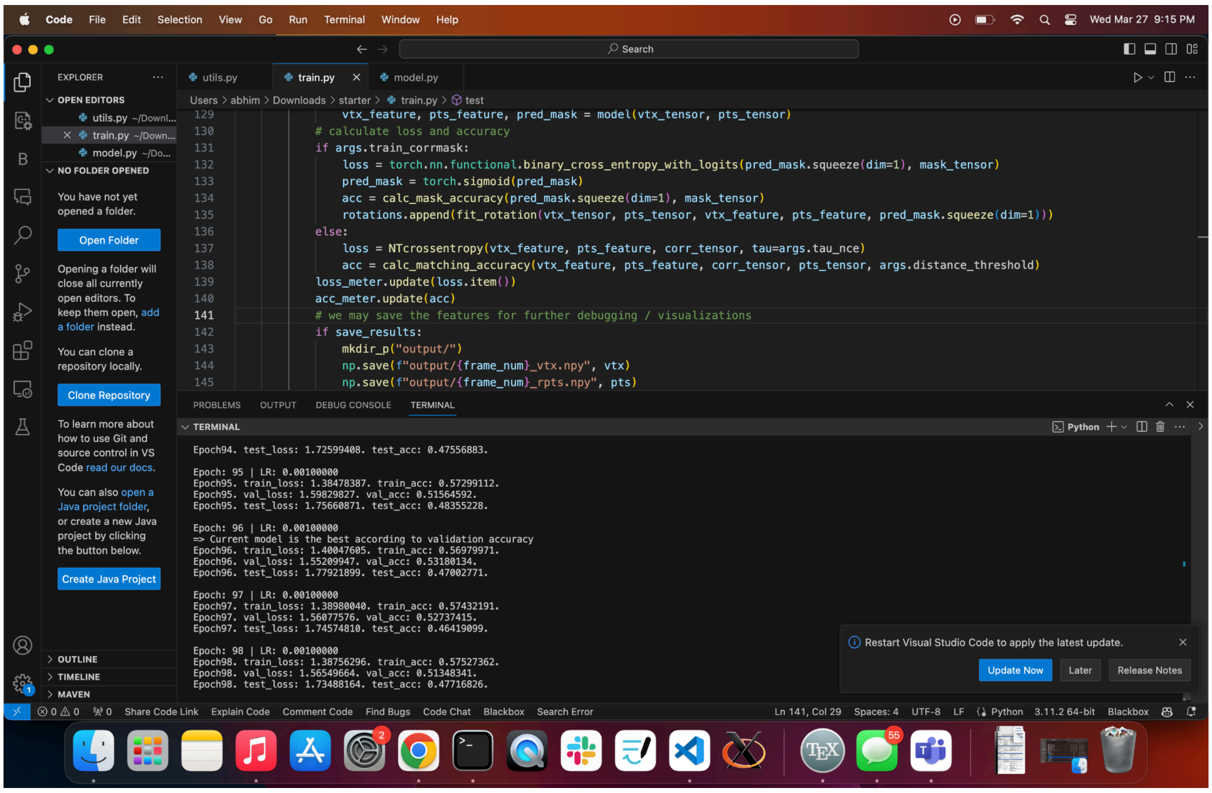
Case 1: train_cormmask=0

distance_threshold=0.01
val_acc: 0.53180134 and corresponding match accuracy: 0.47002771.

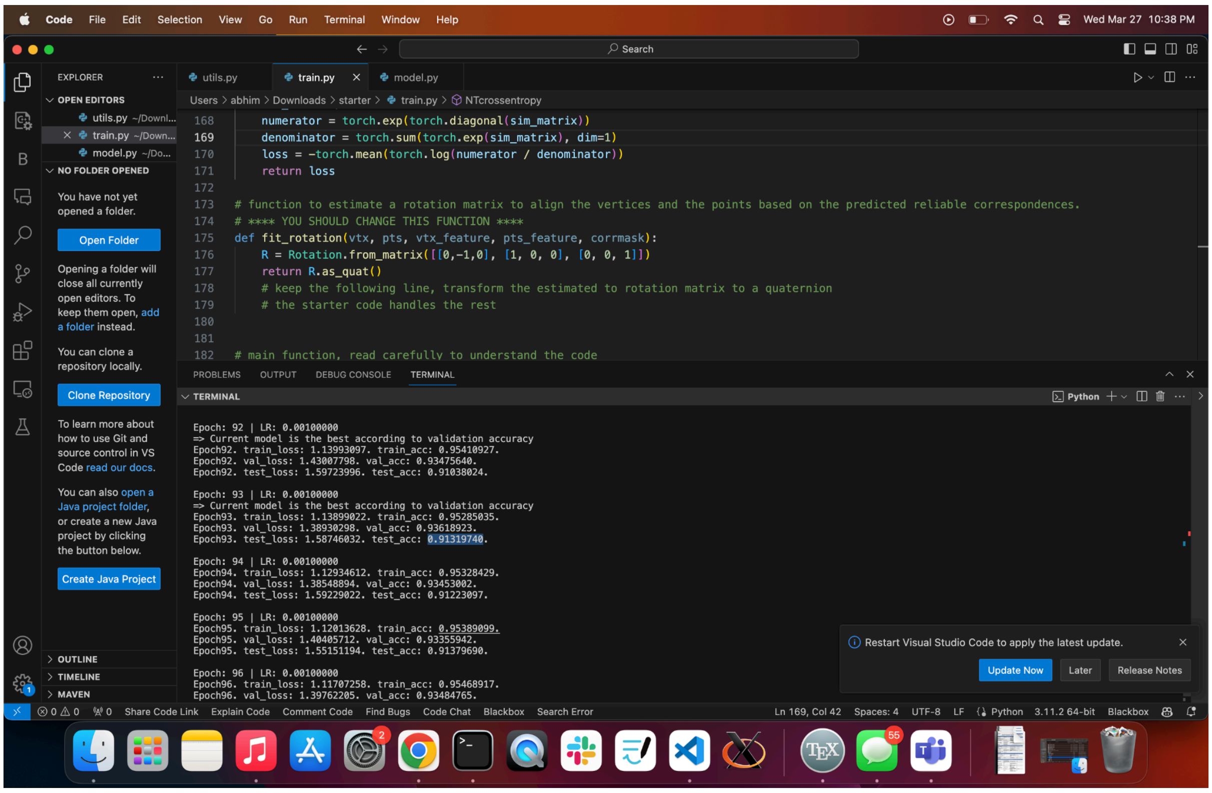


```python
            vtx_feature, pts_feature, pred_mask = model(vtx_tensor, pts_tensor)
        # calculate loss and accuracy
        if args.train_corrmask:
            loss = torch.nn.functional.binary_cross_entropy_with_logits(pred_mask.squeeze(dim=1), mask_tensor)
            pred_mask = torch.sigmoid(pred_mask)
            acc = calc_mask_accuracy(pred_mask.squeeze(dim=1), mask_tensor)
            rotations.append(fit_rotation(vtx_tensor, pts_tensor, vtx_feature, pts_feature, pred_mask.squeeze(dim=1)))
        else:
            loss = NTcrossentropy(vtx_feature, pts_feature, corr_tensor, tau=args.tau_nce)
            acc = calc_matching_accuracy(vtx_feature, pts_feature, corr_tensor, pts_tensor, args.distance_threshold)
        loss_meter.update(loss.item())
        acc_meter.update(acc)
        # we may save the features for further debugging / visualizations
        if save_results:
            mkdir_p("output/")
            np.save(f"output/{frame_num}_vtx.npy", vtx)
            np.save(f"output/{frame_num}_rpts.npy", pts)
```

```
Epoch94. test_loss: 1.72599408. test_acc: 0.47556883.

Epoch: 95 | LR: 0.00100000
Epoch95. train_loss: 1.38478387. train_acc: 0.57299112.
Epoch95. val_loss: 1.59829827. val_acc: 0.51564592.
Epoch95. test_loss: 1.75660871. test_acc: 0.48355228.

Epoch: 96 | LR: 0.00100000
=> Current model is the best according to validation accuracy
Epoch96. train_loss: 1.40047605. train_acc: 0.56979971.
Epoch96. val_loss: 1.55209947. val_acc: 0.53180134.
Epoch96. test_loss: 1.77921899. test_acc: 0.47002771.

Epoch: 97 | LR: 0.00100000
Epoch97. train_loss: 1.38980040. train_acc: 0.57432191.
Epoch97. val_loss: 1.56077576. val_acc: 0.52737415.
Epoch97. test_loss: 1.74574810. test_acc: 0.46419099.

Epoch: 98 | LR: 0.00100000
Epoch98. train_loss: 1.38756296. train_acc: 0.57527362.
Epoch98. val_loss: 1.56549664. val_acc: 0.51348341.
Epoch98. test_loss: 1.73488164. test_acc: 0.47716826.
```

Restart Visual Studio Code to apply the latest update.

distance_threshold=0.02
Best epoch 98 with val_acc: 0.93618923 and corresponding match accuracy: 0.91319740

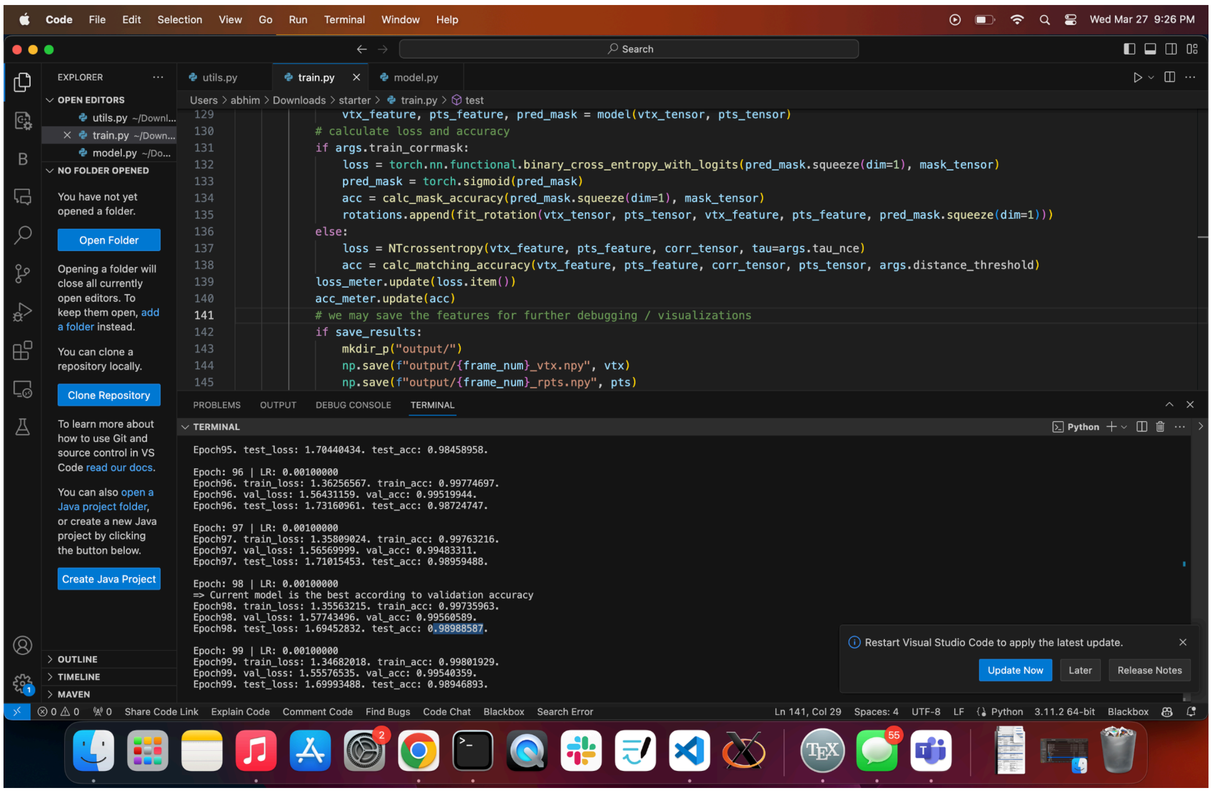


```python
    numerator = torch.exp(torch.diagonal(sim_matrix))
    denominator = torch.sum(torch.exp(sim_matrix), dim=1)
    loss = -torch.mean(torch.log(numerator / denominator))
    return loss

# function to estimate a rotation matrix to align the vertices and the points based on the predicted reliable correspondences.
# **** YOU SHOULD CHANGE THIS FUNCTION ****
def fit_rotation(vtx, pts, vtx_feature, pts_feature, cormmask):
    R = Rotation.from_matrix([[0,-1,0], [1, 0, 0], [0, 0, 1]])
    return R.as_quat()
    # keep the following line, transform the estimated to rotation matrix to a quaternion
    # the starter code handles the rest


# main function, read carefully to understand the code
```

```
Epoch: 92 | LR: 0.00100000
=> Current model is the best according to validation accuracy
Epoch92. train_loss: 1.13993097. train_acc: 0.95410927.
Epoch92. val_loss: 1.43007798. val_acc: 0.93475640.
Epoch92. test_loss: 1.59723996. test_acc: 0.91038024.

Epoch: 93 | LR: 0.00100000
=> Current model is the best according to validation accuracy
Epoch93. train_loss: 1.13899022. train_acc: 0.95285035.
Epoch93. val_loss: 1.38930298. val_acc: 0.93618923.
Epoch93. test_loss: 1.58746032. test_acc: 0.91319740.

Epoch: 94 | LR: 0.00100000
Epoch94. train_loss: 1.12934612. train_acc: 0.95328429.
Epoch94. val_loss: 1.38548894. val_acc: 0.93453002.
Epoch94. test_loss: 1.59229022. test_acc: 0.91223097.

Epoch: 95 | LR: 0.00100000
Epoch95. train_loss: 1.12013628. train_acc: 0.95389099.
Epoch95. val_loss: 1.40405712. val_acc: 0.93355942.
Epoch95. test_loss: 1.55151194. test_acc: 0.91379690.

Epoch: 96 | LR: 0.00100000
Epoch96. train_loss: 1.11707258. train_acc: 0.95468917.
Epoch96. val_loss: 1.39762205. val_acc: 0.93484765.
```

Restart Visual Studio Code to apply the latest update.

distance_threshold=0.04
Best epoch 98 with val_acc: 0.99560589. and corresponding match accuracy: 0.98988587

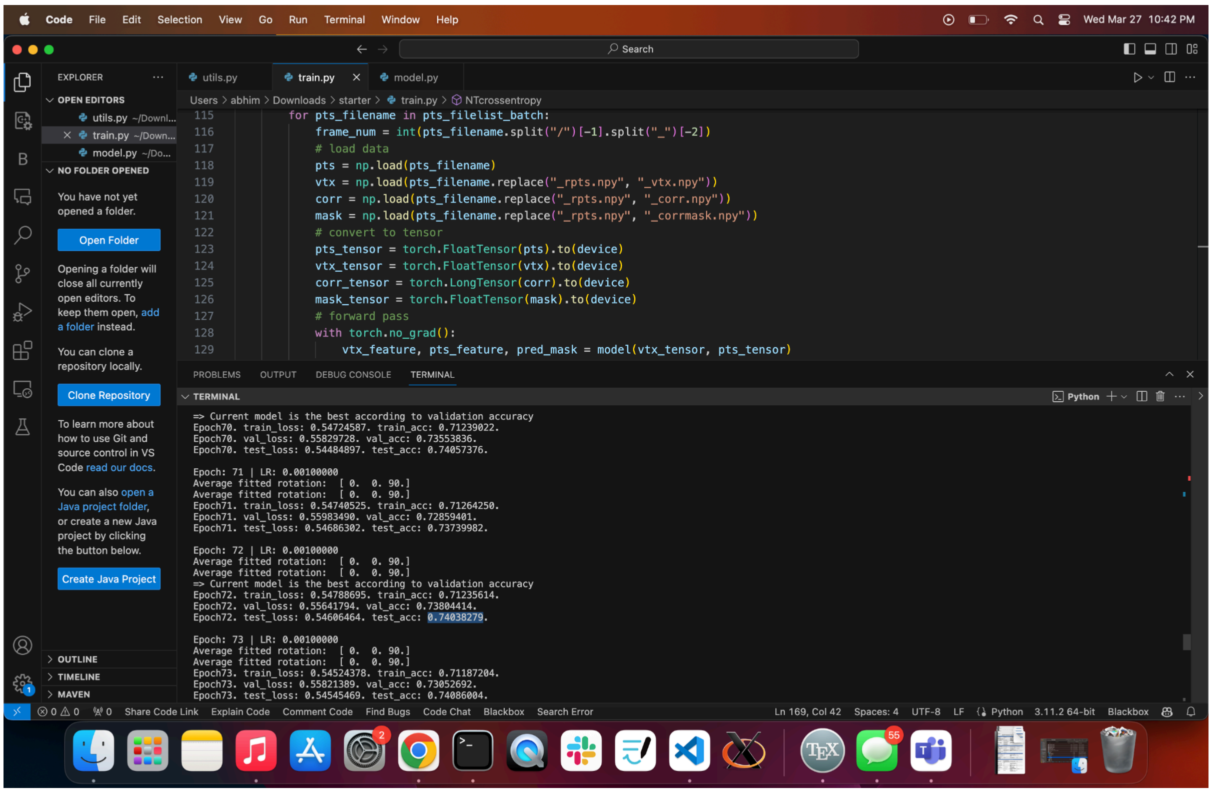


```python
            vtx_feature, pts_feature, pred_mask = model(vtx_tensor, pts_tensor)
        # calculate loss and accuracy
        if args.train_corrmask:
            loss = torch.nn.functional.binary_cross_entropy_with_logits(pred_mask.squeeze(dim=1), mask_tensor)
            pred_mask = torch.sigmoid(pred_mask)
            acc = calc_mask_accuracy(pred_mask.squeeze(dim=1), mask_tensor)
            rotations.append(fit_rotation(vtx_tensor, pts_tensor, vtx_feature, pts_feature, pred_mask.squeeze(dim=1)))
        else:
            loss = NTcrossentropy(vtx_feature, pts_feature, corr_tensor, tau=args.tau_nce)
            acc = calc_matching_accuracy(vtx_feature, pts_feature, corr_tensor, pts_tensor, args.distance_threshold)
        loss_meter.update(loss.item())
        acc_meter.update(acc)
        # we may save the features for further debugging / visualizations
        if save_results:
            mkdir_p("output/")
            np.save(f"output/{frame_num}_vtx.npy", vtx)
            np.save(f"output/{frame_num}_rpts.npy", pts)
```

```
Epoch95. test_loss: 1.70440434. test_acc: 0.98458958.

Epoch: 96 | LR: 0.00100000
Epoch96. train_loss: 1.36256567. train_acc: 0.99774697.
Epoch96. val_loss: 1.56431159. val_acc: 0.99519944.
Epoch96. test_loss: 1.73160961. test_acc: 0.98724747.

Epoch: 97 | LR: 0.00100000
Epoch97. train_loss: 1.35809024. train_acc: 0.99763216.
Epoch97. val_loss: 1.56569999. val_acc: 0.99483311.
Epoch97. test_loss: 1.71015453. test_acc: 0.98959488.

Epoch: 98 | LR: 0.00100000
=> Current model is the best according to validation accuracy
Epoch98. train_loss: 1.35563215. train_acc: 0.99735963.
Epoch98. val_loss: 1.57743496. val_acc: 0.99560589.
Epoch98. test_loss: 1.69452832. test_acc: 0.98988587.

Epoch: 99 | LR: 0.00100000
Epoch99. train_loss: 1.34682018. train_acc: 0.99801929.
Epoch99. val_loss: 1.55576535. val_acc: 0.99540359.
Epoch99. test_loss: 1.69993488. test_acc: 0.98946893.
```

Restart Visual Studio Code to apply the latest update.

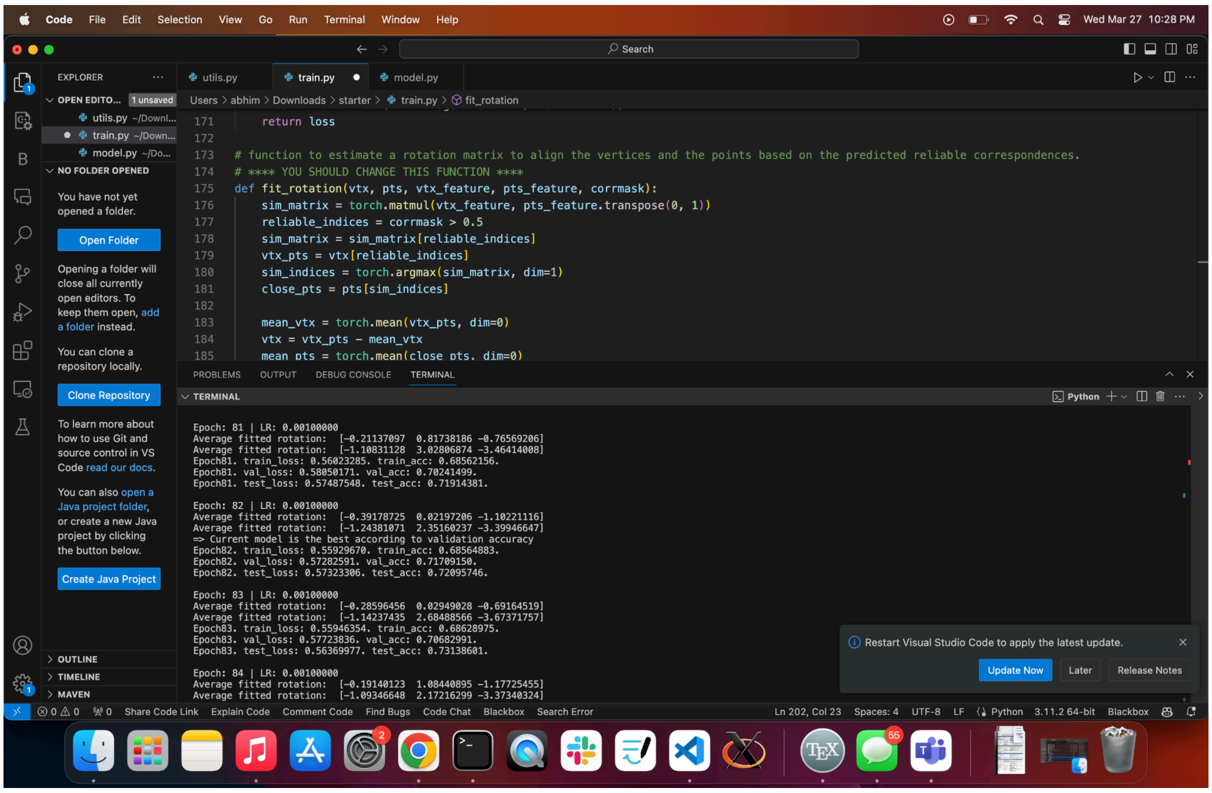
Case 2: train_corrmask=1 (distance_threshold 0.02)
Best epoch has val_accuracy: 0.73804414 with match accuracy: 0.74038279



```python
        for pts_filename in pts_filelist_batch:
            frame_num = int(pts_filename.split("/")[-1].split("_")[-2])
            # load data
            pts = np.load(pts_filename)
            vtx = np.load(pts_filename.replace("_rpts.npy", "_vtx.npy"))
            corr = np.load(pts_filename.replace("_rpts.npy", "_corr.npy"))
            mask = np.load(pts_filename.replace("_rpts.npy", "_corrmask.npy"))
            # convert to tensor
            pts_tensor = torch.FloatTensor(pts).to(device)
            vtx_tensor = torch.FloatTensor(vtx).to(device)
            corr_tensor = torch.LongTensor(corr).to(device)
            mask_tensor = torch.FloatTensor(mask).to(device)
            # forward pass
            with torch.no_grad():
                vtx_feature, pts_feature, pred_mask = model(vtx_tensor, pts_tensor)
```

```
=> Current model is the best according to validation accuracy
Epoch70. train_loss: 0.54724587. train_acc: 0.71239022.
Epoch70. val_loss: 0.55829728. val_acc: 0.73553836.
Epoch70. test_loss: 0.54484897. test_acc: 0.74057376.

Epoch: 71 | LR: 0.00100000
Average fitted rotation:  [ 0.  0. 90.]
Average fitted rotation:  [ 0.  0. 90.]
Epoch71. train_loss: 0.54740525. train_acc: 0.71264250.
Epoch71. val_loss: 0.55983490. val_acc: 0.72859401.
Epoch71. test_loss: 0.54686302. test_acc: 0.73739982.

Epoch: 72 | LR: 0.00100000
Average fitted rotation:  [ 0.  0. 90.]
Average fitted rotation:  [ 0.  0. 90.]
=> Current model is the best according to validation accuracy
Epoch72. train_loss: 0.54788695. train_acc: 0.71235614.
Epoch72. val_loss: 0.55641794. val_acc: 0.73804414.
Epoch72. test_loss: 0.54606464. test_acc: 0.74038279.

Epoch: 73 | LR: 0.00100000
Average fitted rotation:  [ 0.  0. 90.]
Average fitted rotation:  [ 0.  0. 90.]
Epoch73. train_loss: 0.54524378. train_acc: 0.71187204.
Epoch73. val_loss: 0.55821389. val_acc: 0.73052692.
Epoch73. test_loss: 0.54545469. test_acc: 0.74086004.
```

Case 3: Task E
train_corrmask=1
distance_threshold = 0.02
val acc 0.71709150
match acc 0.72095746.
Rotation matrix:
Average fitted rotation: [-0.39178725 0.02197206 -1.10221116]
Average fitted rotation: [-1.24381071 2.35160237 -3.39946647]



```python
    return loss

# function to estimate a rotation matrix to align the vertices and the points based on the predicted reliable correspondences.
# **** YOU SHOULD CHANGE THIS FUNCTION ****
def fit_rotation(vtx, pts, vtx_feature, pts_feature, cormask):
    sim_matrix = torch.matmul(vtx_feature, pts_feature.transpose(0, 1))
    reliable_indices = corrmask > 0.5
    sim_matrix = sim_matrix[reliable_indices]
    vtx_pts = vtx[reliable_indices]
    sim_indices = torch.argmax(sim_matrix, dim=1)
    close_pts = pts[sim_indices]

    mean_vtx = torch.mean(vtx_pts, dim=0)
    vtx = vtx_pts - mean_vtx
    mean_pts = torch.mean(close_pts, dim=0)
```

```
Epoch: 81 | LR: 0.00100000
Average fitted rotation:  [-0.21137097  0.81738186 -0.76569206]
Average fitted rotation:  [-1.10831128  3.02806874 -3.46414008]
Epoch81. train_loss: 0.56023285. train_acc: 0.68562156.
Epoch81. val_loss: 0.58050171. val_acc: 0.70241499.
Epoch81. test_loss: 0.57487548. test_acc: 0.71914381.

Epoch: 82 | LR: 0.00100000
Average fitted rotation:  [-0.39178725  0.02197206 -1.10221116]
Average fitted rotation:  [-1.24381071  2.35160237 -3.39946647]
=> Current model is the best according to validation accuracy
Epoch82. train_loss: 0.55929670. train_acc: 0.68564883.
Epoch82. val_loss: 0.57282591. val_acc: 0.71709150.
Epoch82. test_loss: 0.57323306. test_acc: 0.72095746.

Epoch: 83 | LR: 0.00100000
Average fitted rotation:  [-0.28596456  0.02949028 -0.69164519]
Average fitted rotation:  [-1.14237435  2.68488566 -3.67371757]
Epoch83. train_loss: 0.55946354. train_acc: 0.68628975.
Epoch83. val_loss: 0.57723836. val_acc: 0.70682991.
Epoch83. test_loss: 0.56369977. test_acc: 0.73138601.

Epoch: 84 | LR: 0.00100000
Average fitted rotation:  [-0.19140123  1.08440895 -1.17725455]
Average fitted rotation:  [-1.09346648  2.17216299 -3.37340324]
```

Restart Visual Studio Code to apply the latest update.