I used a smaller network for the self-supervised learning to make the training faster. It was a block network with 2 blocks of 3 layers each.Two cnn layers with stride 1 and same padding while the third layer acted as a downsampler using a convolutional layer with stride 2.

This network was used for the encoding. 2 additional linear layers were added after which contrastive loss was run. The classifier was a single linear layer with a softmax. As a baseline the classifier was run alone and got an accuracy of 16%. The contrastive loss was not calculating the gradients properly which I suspect was because of adding the outputs to the list. Therefore, I shifted to the Triplet loss margin for testing purposes but gave no results. It gave an accuracy of around 10% which would point to the embeddings not able to capture any significant mapping. Reading a few resources I used the model used for the semi supervised task. The last 2 linear layers were used as the buffer for the embeddings. The following system gave an accuracy of 26 percent after training the embeddings for 2 epochs and classifier for 40. The input were 2 images with their respective transformations.I believe the bottleneck is the classifier because it is just one layer deep. It still shows a significant improvement over training on just the labelled dataset.