Chenghao Wang
CS7180 Geometric Deep Learning (Spring 2023)

HW1

Part 1: Training a convolutional network on MNIST, measuring translation equivariance.

Question 1. Implement a convolutional neural network and reports its learning curve (loss vs training steps) and validation accuracy compared to the MLP baseline we implemented.

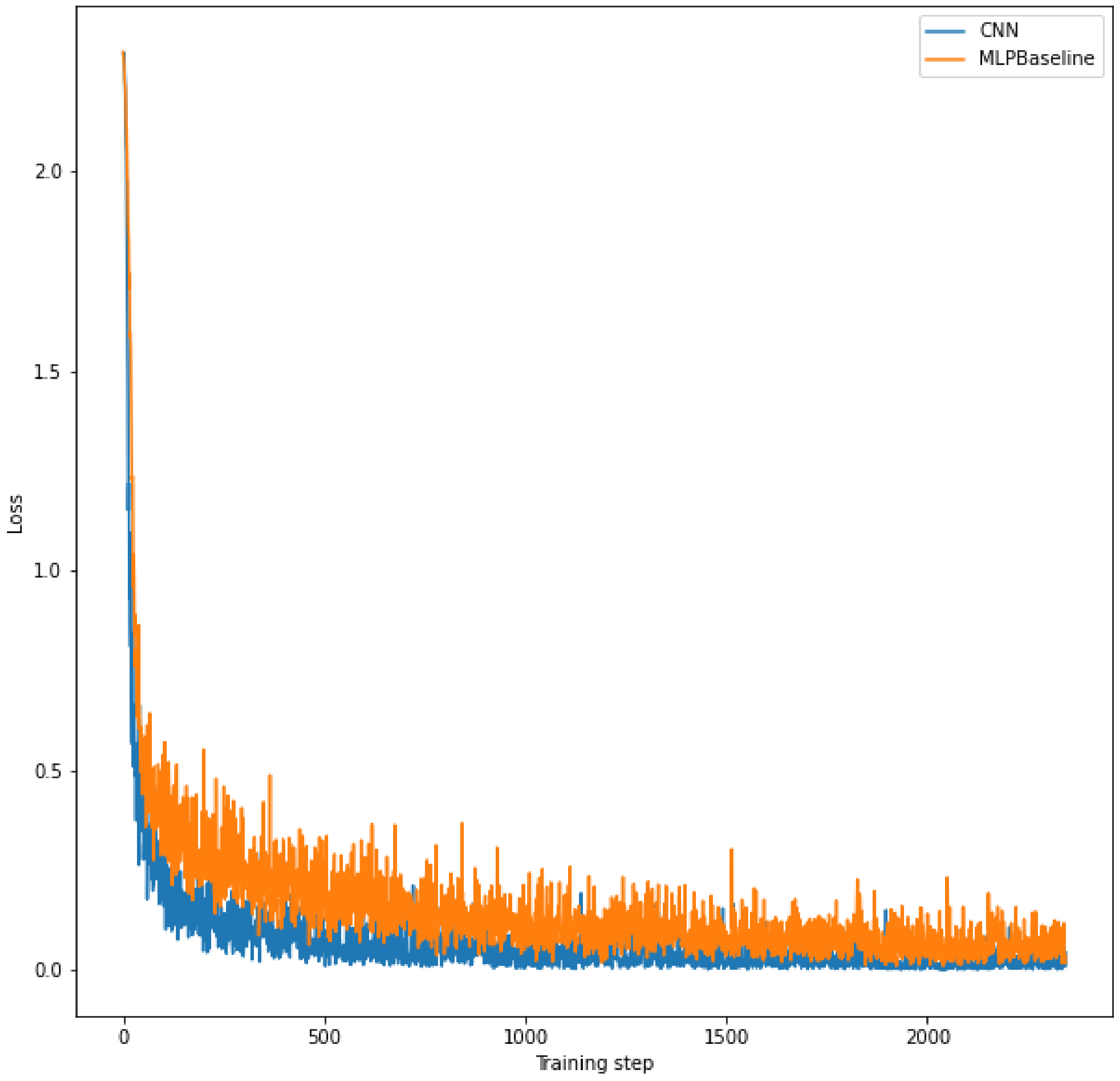

Figure 1. Comparison between MLP and CNN in full size MNIST.

Validation Accuracy of MLP baseline is 97.2%
Validation Accuracy of CNN is 97.2%

Question 2. Plot validation accuracy vs training dataset size for the CNN and the MLP.

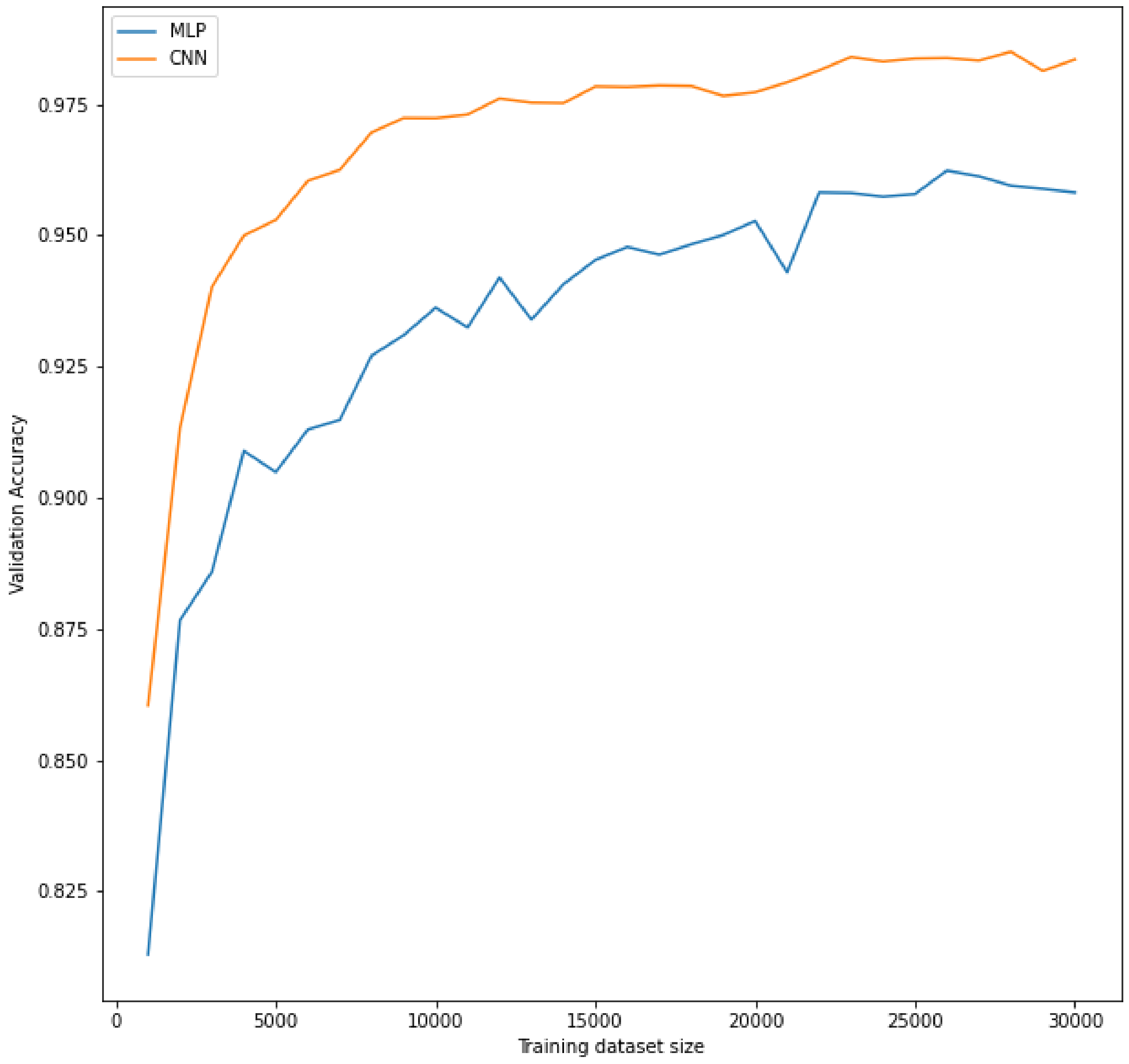

Figure 2. Comparison of accuracy between MLP and CNN in MNIST in different size.

In this problem, I tested data from size 500 to 30,000 with a sampling interval of 1000. Compared to MLP, CNN's Validation Accuracy rises more significantly with Training Dataset Size growth.

Question 3. Measure the invariance of the CNN and MLP predictions to translations of the input image. You can do so by randomly translating the input image (with the limit that the digit is still somewhat within the image) and measuring the change of the predicted probabilities compared to the image of a centered digit. Repeat the same measurement for untrained (randomly initialized) networks. It is ok to perform this and all following experiments with only a few validation images. Discuss the results.

In this problem, I created multiple datasets, each containing 9 different translations of the same number. The translation offsets of both X and Y are [-3, 2, 0, 2, 3]. The following table shows the result.

Label	Accuracy_{CNN}	Accuracy_{MLP}	$\text{Accuracy}_{randCNN}$	$\text{Accuracy}_{randMLP}$
1	73.1%	23.1%	0.0%	0.0%
2	92.3%	84.6%	0.0%	0.0%
3	69.2%	61.5%	0.0%	0.0%
4	96.2%	38.5%	0.0%	0.0 %
5	88.5%	69.2%	0.0%	0.0%
6	80.8%	80.8%	0.0%	0.0%
7	92.3%	69.2%	0.0%	0.0%
8	69.2%	42.3%	0.0%	3.8%
9	69.2%	46.2%	0.0%	0.0%
0	80.8%	46.2%	0.0%	0.0%

The validation accuracy of the trained models of the CNN and the MLP is presented in the first two columns of the table. A comparison of these values clearly shows that the CNN model has a significantly higher accuracy than the MLP model, which implies that the CNN model has a better ability to handle transformations in the input data. This superior invariance property of the CNN model is an important aspect to consider when choosing the appropriate model for a particular problem.

The last two columns of the table represent the validation accuracy of untrained models, meaning models with random initializations. As expected, the majority of these untrained models have a 0.0% accuracy, as they have not yet been optimized through training to make accurate predictions.

Question 4. Add several max pooling or average pooling layers to your convolutional network (you can set convolutional strides to 1 to avoid downsampling too quickly). Does pooling make the network more invariant?

Label	Accuracy_{CNN}	$\text{Accuracy}_{CNNwithPooling}$
1	73.1%	100.0%
2	92.3%	100.0%
3	69.2%	92.3%
4	96.2%	100.0%
5	88.5%	84.6%
6	80.8%	80.8%
7	92.3%	100.0%
8	69.2%	84.6%
9	69.2%	80.8%
0	80.8%	88.5%

The results presented above clearly show that the addition of a MaxPooling layer in the CNN has had a significant impact on the network's prediction accuracy for images subjected to random panning. This improvement in accuracy is an indication of the enhanced invariance property of the CNN network with the MaxPooling layer.

Question 5. Remove any stride and max pooling from your network, so that the resolution of your feature map is always 28 * 28 (the original image size). The final prediction layer can first flatten the feature map, or maybe perform average pooling, followed by some fully-connected layers. Measure the translation equivariance of any two intermediate feature maps in your convolutional network. You want to translate the input image and see if the feature map (of the same size) translates accordingly. Repeat the same measurement for an untrained (randomly initialized) network.

First of all, I implemented some changes to the structure of the CNN to include function that visualizes the feature maps.

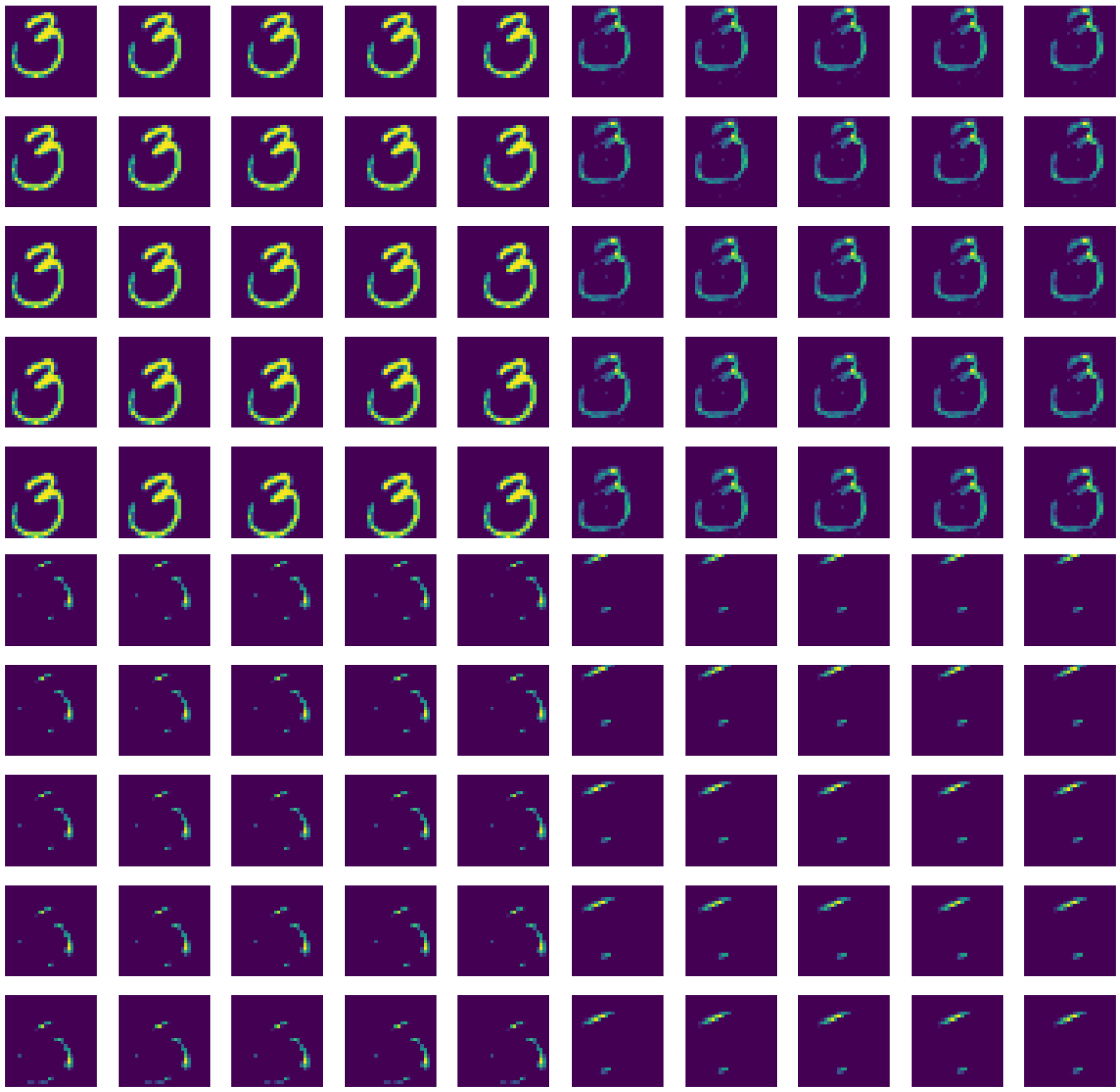

FIGURE 3. Up-Left: Raw Image; Up-Right: Conv1; Bottom-Left: Conv2; Bottom-Right: Conv3

After visualized all feature maps, I measured the translation equivariance of any two intermediate feature maps in CNN by using following equation:

$$ManhattanDistance\left(\text{conv}(Img), T^{-1}\left[\text{conv}\left(T * Img\right)\right]\right)$$

Label	$Distance(img, conv1)$	$Distance(conv1, conv2)$	$Distance(conv2, conv3)$
1	29.987438	82.45204	473.76416
2	1043.4941	1208.9978	416.14935
3	13.048479	324.64767	149.62613

Apply same experiment to an untrained CNN network:

Label	$Distance(img, conv1)$	$Distance(conv1, conv2)$	$Distance(conv2, conv3)$
1	3296.0454	292.7112	20668.889
2	3857.684	836.3973	15651.227
3	2740.1072	1378.1427	12096.751

The results demonstrate that the trained CNN network exhibits improved equivariance compared to the untrained CNN network for most of the data.

Part 2: Training a Graph Convolutional Network on a citation graph dataset.

Question 6. Implement a Graph Convolutional Network and train it on the Cora dataset. The network should make a prediction for each of the 2708 nodes. Plot its learning curve and report its validation accuracy. Note that you can run the entire graph through the network in a single training step (unless you run out of memory).

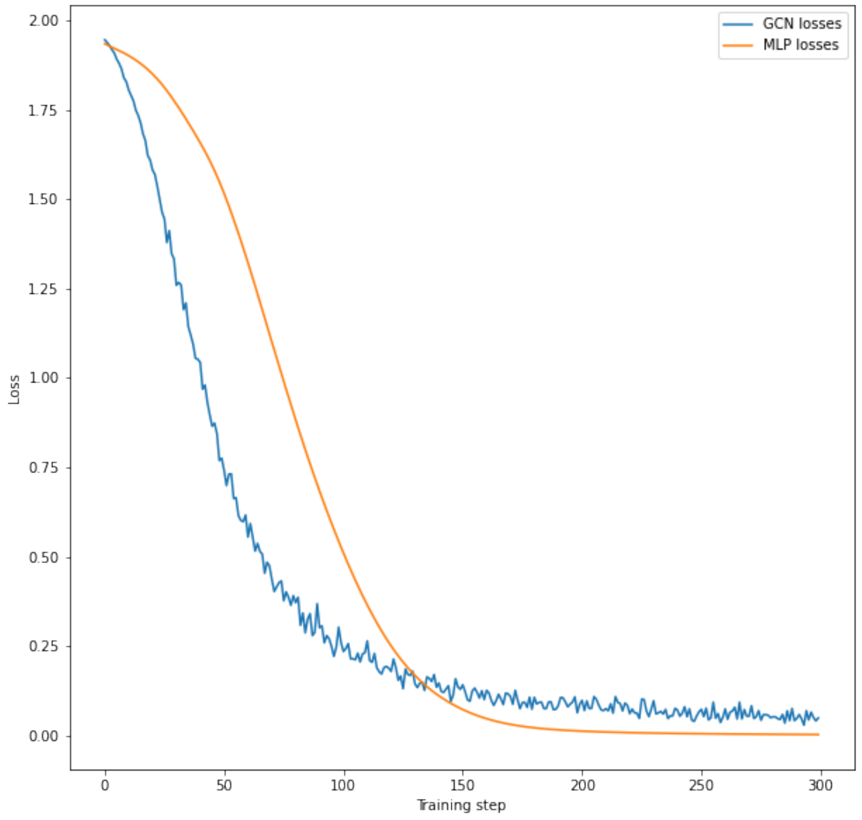

Figure 4. Comparison between MLP and GCN in Cora dataset.

Validation Accuracy of MLP baseline is 51.7%
Validation Accuracy of GCN is 74.3%

Question 7. Is the MLP baseline equivariant to the permutation of nodes (and the corresponding permutation of the adjacency matrix)? What about the Graph Convolutional Network?

The baseline MLP model typically treats graph data as a vector of node features and does not take into account the graph structure. As a result, it is not equivariant to node permutations, meaning that changing the order of the nodes in the graph would change the output of the model.

On the other hand, GCN models are specifically designed to preserve graph structure by incorporating graph information in their computations. The GCN model aggregates information from the node's neighbors by computing dot products between node representations and the graph's adjacency matrix. This process is equivariant to node permutations, meaning that the output of the model remains the same even if the node order is changed.

Question 8. Plot the number of training steps (you can choose a few discrete values) vs validation accuracy. Is the MLP baseline or the Graph Convolutional Network more prone to overfitting? Why?

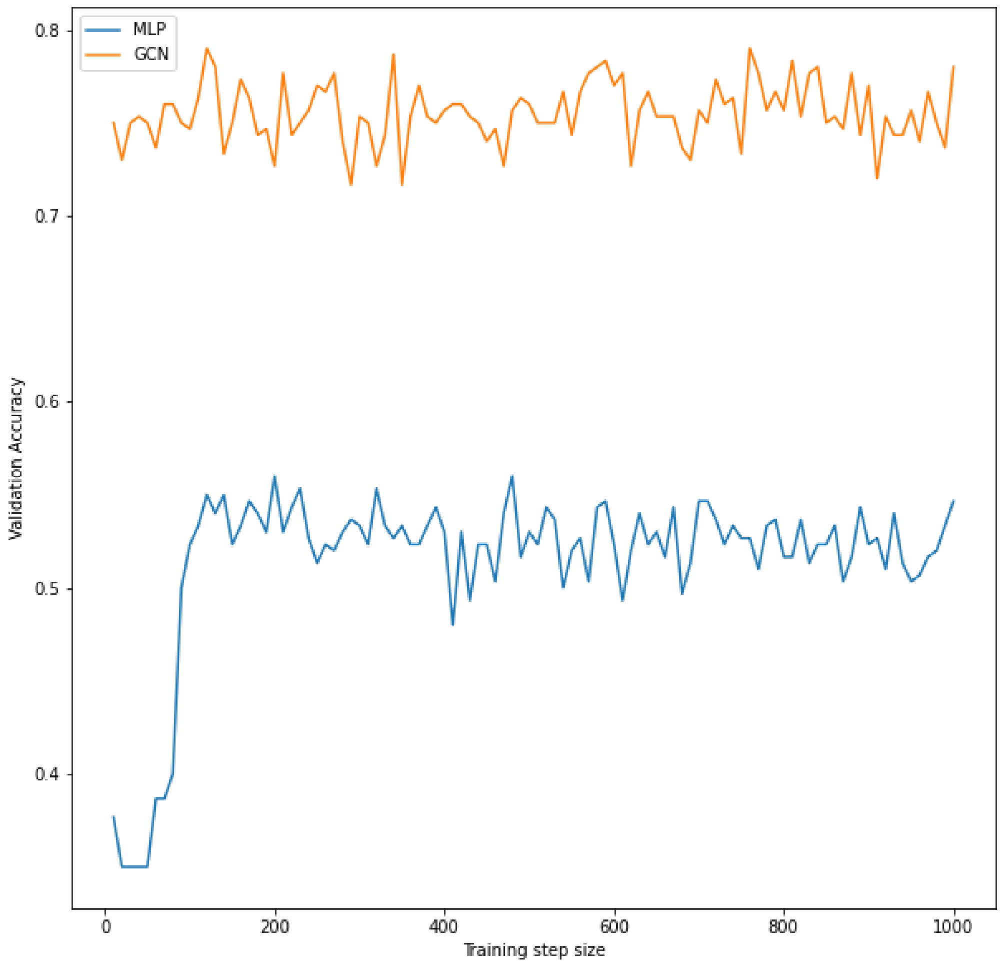

FIGURE 5. Comparison between MLP and GCN in Cora dataset in different train steps.

The MLP model does not experience overfitting until 200 training steps have been completed. After this point, both the MLP and GCN models are overfitted, as evidenced by the fact that the validation accuracy for both models does not continue to improve beyond 200 steps.