Neural Topological SLAM for Visual Navigation: Supplementary Material

Devendra Singh Chaplot[1†], Ruslan Salakhutdinov[1*], Abhinav Gupta[1,2*], Saurabh Gupta[3*]
[1] Carnegie Mellon University, [2] Facebook AI Research, [3] UIUC

Project webpage: https://devendrachaplot.github.io/projects/Neural-Topological-SLAM

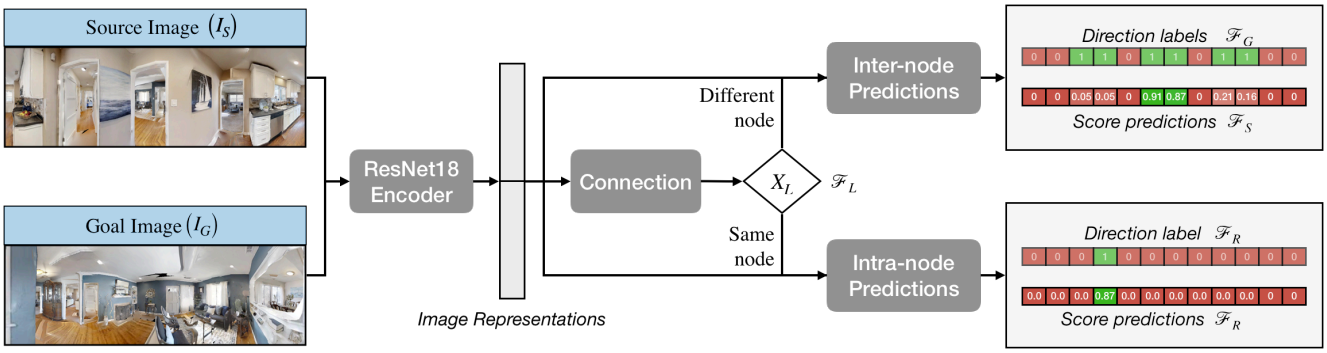

Figure 1: NTS Multi-task Learning Model. Figure showing an overview of the NTS Multi-task Learning Model. It takes a Source Image (I_S) and a Goal Image (I_G) as input and encodes them using a shared ResNet18 encoder. It first predicts whether the two images belong to the same node or not using the Connection model. If they belong to the same node, it makes Intra-Node predictions which include the direction and score (or equivalently) distance of the Goal Image relative to the Source Image. If they belong to different nodes, it makes Inter-Node Predictions which include directions of explorable areas and a semantic score corresponding to each explorable area denoting its proximity to the Goal Image.

A. NTS Multi-task Learning Model Architecture and Training Details

The NTS Multi-task Learning model consists of 4 modules, a ResNet18 encoder [1], a Connection model, an Inter-node Predictions model, and an Intra-node Predictions model as shown in Figure 1. Given panoramic source (I_S) and goal (I_G) images, each of size 128×512, we first construct $n_\theta = 12$ square patches from each panoramic image. The i^{th} patch consists of 128×128 image centered at $i/n_\theta * 2\pi$ radians. These resulting patches are such that each patch overlaps with 1/3 portion of the adjacent patch on both sides (patches are wrapped around at the edge of the panorama). Each of the n_θ patches of both source and target images is passed through the ResNet18 encoder to get a patch representation of size 128. The architecture of the ResNet18 encoder is shown in Figure 2 for RGB setting. For RGBD, we pass the depth channel through separate non-pretrained ResNet18 Conv layers, followed by 1x1 convolution to get a representation of size 512. This representation is concatenated with the 512 size RGB representation and passed through FC1 and FC2 to finally get the same 128 size patch representation. We use a dropout of 0.5 in FC1 and FC2 and ReLU non-linearity between all layers.

Given 12 patch representations for source and goal images each, the connection model predicts whether the two images belong to the same node or not. The architecture of the Connection Model is shown in Figure 3. If the two images belong to the same node, the Intra-Node Predictions model predicts the direction and score (or equivalently distance) of the goal image relative to the source image. The architecture of the Intra-Node Predictions Model is shown in Figure 4. If the two images belong to different nodes, the Inter-Node Predictions model predicts the explorable area directions and the score of each direction. The architecture of the Inter-Node Predictions Model is shown in Figure 5. As shown in the figure, the explorable area directions are specific to the source image patches and independent of the goal image. We use ReLU non-linearity between all layers in all the modules.

The whole model is trained using supervised learning with 5 different losses. We use Cross-Entropy Loss for the Connection and direction labels and MSE Loss for score labels. We use a loss coefficient of 10 for the score labels and a loss coefficient of 1 for the connection and direction labels. We train the model jointly with all the 5 losses using the Adam optimizer [2] with a learning rate of 5e-4. We use a batch size of 64.

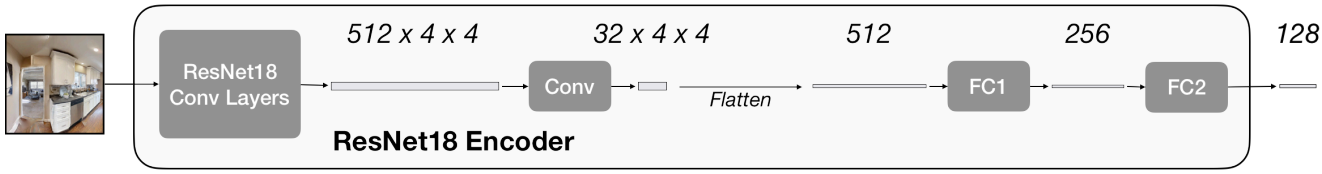

Figure 2: ResNet18 Encoder. Figure showing the architecture of the ResNet18 Encoder.

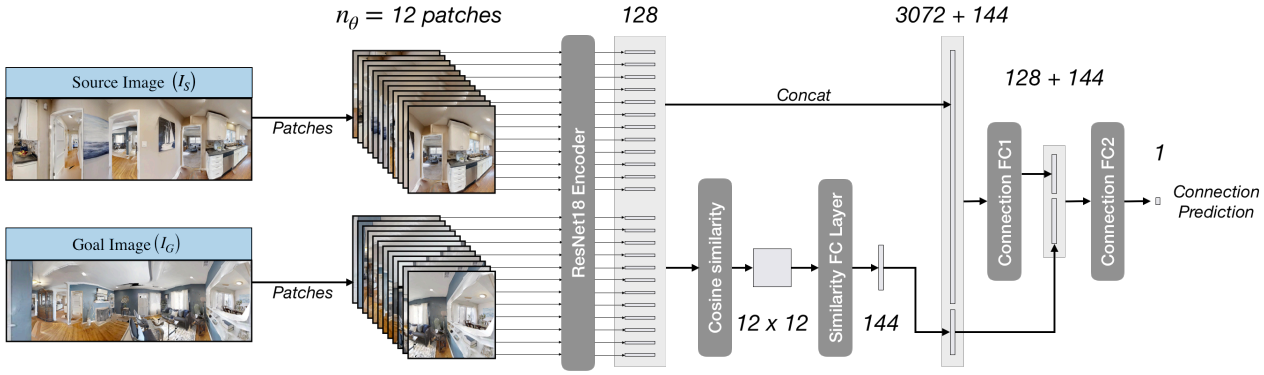

Figure 3: Connection Model. Figure showing the architecture of the Connection Model.

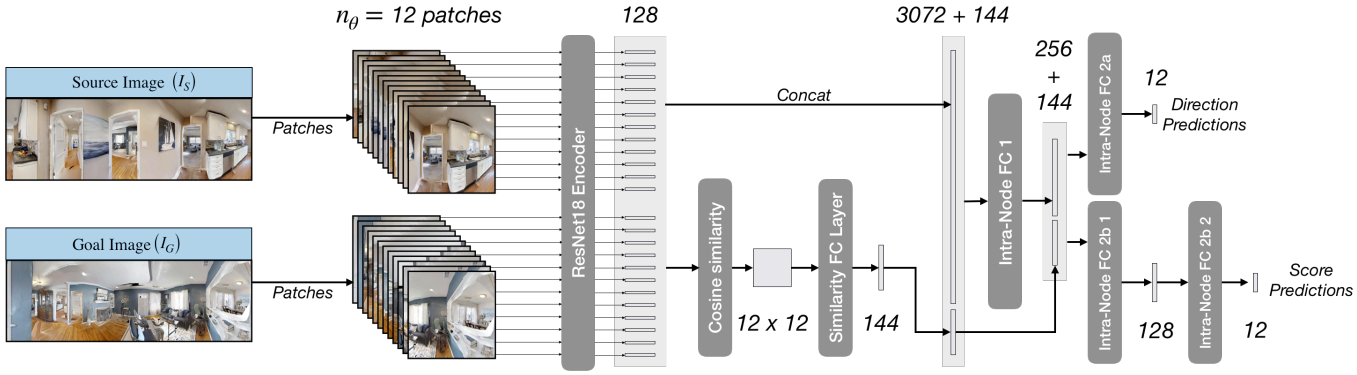

Figure 4: Intra-Node Predictions Model. Figure showing the architecture of the Intra-Node Predictions Model.

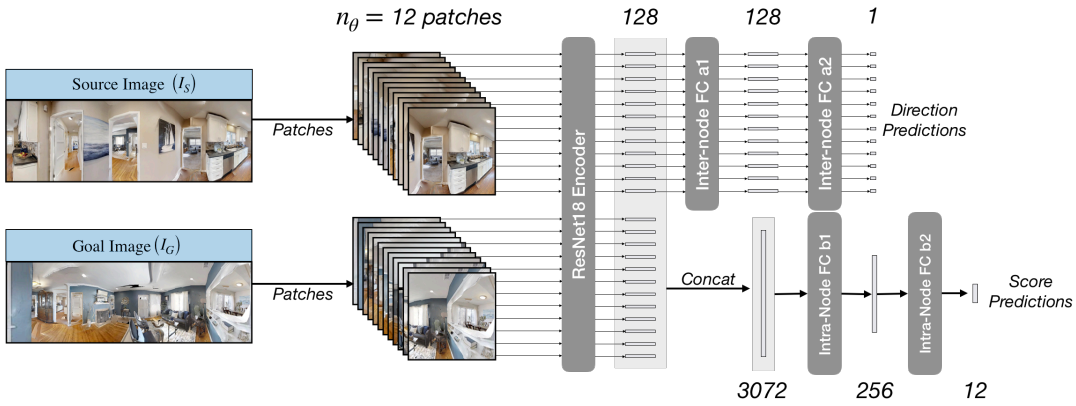

Figure 5: Inter-Node Predictions Model. Figure showing the architecture of the Inter-Node Predictions Model.

B. Dataset collection and automated labeling

For training the NTS Multi-task Learning model, we need to collect the data and different types of labels. We randomly sample 300 points in each training scene. For each ordered pair of images, source image (I_S) and goal image (I_G), we gather the different labels as follow:

Connection label: This label denotes whether the source and goal image belong to the same node or not. We get this label by detecting whether the goal image location is visible from the source image. To compute this, we take a 5-degree patch of depth image centered at the relative direction of the goal image from the source image. If the maximum depth in this patch is greater than the distance to the goal image, then the goal image belongs to the same node as the source image and the label is 1. Intuitively, the above checks whether the maximum depth value in the direction of the goal image is greater than the distance to the goal. If the maximum depth in the direction of goal image is greater than the distance to the goal image or if the distance to the goal image is greater than $r = 3m$, then the goal image belongs to a different node and the label is 0.

Intra-node labels: If the source and goal image belong to the same node, i.e. the above label is 1, then the intra-node direction and score labels are directly obtained from relative position of the goal image from the source image. Let the relative position of the goal image be $\Delta p = (d, \theta)$, where d is the distance and θ is the angle to the goal image from the source image. Intra-node direction label is just the $360/n_\theta = 30$ degree bin in which θ falls, i.e. $nint(\theta/2\pi \times n_\theta)$ where $nint(\cdot)$ is the nearest integer function. For the intra-node score label (s), we just convert the distance d to a score between 0 and 1 using the following function:

$$s = max((1 - d/r), 0)$$

where s denotes the score, $r = 3m$ denotes the radius of the node, d is the distance. Note that the direction label is discrete and the score label is continuous.

Inter-node labels: If the source and goal image do not belong to the same node, i.e. the connection label is 0, we compute the inter-node labels. For computing the inter-node direction labels, we first project the depth channel in the panoramic source image to compute an obstacle map. We ignore obstacles beyond $r = 3m$. In this obstacle map, we take $n_\theta = 12$ points at angles $i/n_\theta \times 2\pi, \forall i \in [1, 2, \ldots, 12]$ and distance $r = 3m$ away. For every, $i \in [1, 2, \ldots, 12]$, if the shortest path distance to the corresponding is less than $1.05 \times r$, then the corresponding inter-node direction label is 1 and otherwise 0.

Let the inter-score labels be denoted by $s_i, \forall i \in [1, 2, \ldots, 12]$. For the inter-node score label s_i, we find the farthest explored and traversable point in direction $i \times (2\pi/n_\theta)$ in the above obstacle map. Let the shortest path distance (geodesic distance) of the goal image from this point be d_i. Then the score label is given by:

$$s_i = max((1 - d_i/d_{max}), 0)$$

where $d_{max} = 20m$ is the maximum distance above which the score is 0.

C. Visualizations

In Figure 6, we show visualizations of an example trajectory using the NTS model. The figure shows the agent observations at different time steps, the goal image, the node locations in the trajectory and the ghost nodes.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1


(a)
Goal Image
Start Location
Goal Location
Current Location
t = 0
Current Observation
(b)
Goal Image
Start Location
Goal Location
Current Location
Regular Nodes
Ghost Nodes
Selected Ghost Node
t = 1
Current Observation
(c)
Goal Image
Start Location
Goal Location
Current Location
Regular Nodes
Ghost Nodes
Selected Ghost Node
t = 20
Current Observation

Figure 6: Example Trajectory Visualization. Figure showing a visualization of an example trajectory using the NTS model. **(a)** Starting and goal locations at the beginning of an episode, $t = 0$ shown on the ground-truth map. Note that the map is not available to the NTS model. **(b)** NTS model creates the first node at $t = 1$. Figure showing the ghost nodes and the selected ghost node. Note that the ghost node locations are for visualization only, they are not visible or predicted by the NTS model. The model predicts only the direction of ghost nodes. **(c)** The agent observation at $t = 20$ and trajectory till $t = 20$ shown on the map. **(d)** The NTS model creates a new node at $t = 27$. **(e)** The ghost nodes and selected ghost nodes after creating the second node. The model correctly selects the correct ghost node closest to the goal. Note that two ghost nodes in the direction of the second node from the first node are removed. **(f)** The agent reaches the goal at $t = 61$ after creating 4 nodes as shown in the trajectory on the map.