# Supplementary Materials for Neural Task Graphs: Generalizing to Unseen Tasks from a Single Video Demonstration

## 1. Implementation Details

Image inputs are first encoded ($Enc(\cdot)$ in the paper) by the SqueezeNet to a 128D embedding. Demo Interpreter is a 2-layer GRU seq2seq model with attention and 128 hidden units using the OpenNMT implementation [1]. Both the Node Localizer and the Edge Classifier are 4-layer MLPs. GCN uses 128D node embedding and 2-layer MLPs for $f_{\texttt{set}}$ and $f_{\texttt{reset}}$ in Equation (1) in the paper. Each component is trained with a separate loss function, but the full model is end-to-end trainable to the input images. An overview of our model architecture is shown in Figure 1.

## 2. Generalizing to Unseen States and Execution Orders

The main advantage of using the Conjugate Task Graph is that we do not have to enumerate the unseen states. The states are implicitly related to the actions by the Edge Classifier. As for generalizing the execution order, we train our GCN to learn to generate the edges (action transitions) that are not observed in a single demonstration (supervision comes from multiple demonstrations per task). This allows the execution engine to handle actions whose order can be permuted at test time.

## 3. Approach Clarification

We do not use object detector in this work because the whole image is encoded by the CNN. In addition, we do not explicitly estimate the state in this work because state estimation is only required when there is a "fork" in the conjugate task graph (Fig. 5 in the paper), where actions are nodes and transition state are edges. In this case, we use the Edge Classifier to determine the next node (action), which implicitly understands the current state. Finally, We do not perform temporal action detection/localization in this paper. As discussed in the paper, we do not need to know the the exact temporal extent of each action in the demonstration to construct the task graph. Thus, we only use the seq2seq model to capture the order of actions in the demonstration.

## References

[1] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*, 2017.

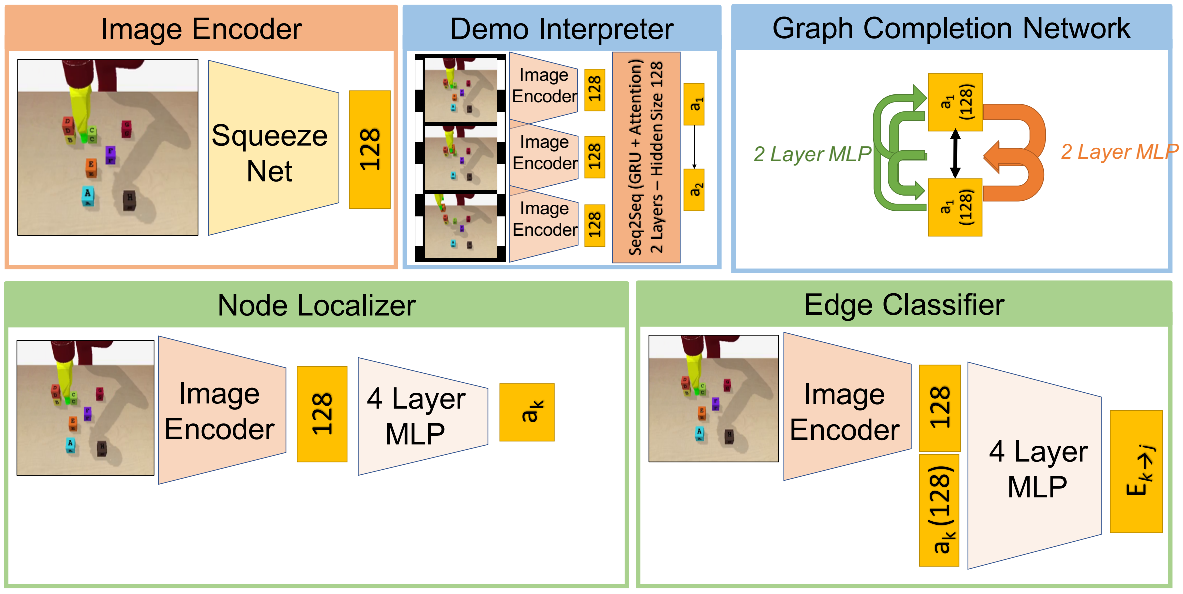

Figure 1. Architecture of each module of our Neural Task Graph (NTG) Networks. The Image Encoder uses SqueezeNet to compress the 64x64x3 image to a 128 dim vector. The Demo Interpreter passes each frame of the demo video through the image interpreter, then passes the resulting sequence through a 2 layer Seq2Seq model with GRU and Attention and a 128 dimension hidden state. The Graph completion network uses 2 layer MLP networks to propagate the node embedding to their neighbors, and a 2 Layer MLP to update the edges based on the nodes. The Node Localizer uses the Image Encoder to encode the image, then a 4 Layer MLP to predict the current node in the graph. The Edge Classifier then takes the 128 dim node embedding from the current node, as well as the encoded image state, concatenates them, then passes it through a 4 layer MLP to predict the next node.