We thank the reviewers for their feedback. The reviewers R1 and R3 suggested additional experiments. We are pleased to report that we have completed those experiments. We report those results and address other concerns below. Due to limited space, we couldn't answer all of the reviewers' clarification queries but promise to include in the final version.

[R1] "why the 6 link baselines failed... comparable to humanoids where these agents still work": In Humanoid environments (e.g. in OpenAI Gym), the range of each joint angle is carefully set with hand-designed limits using domain knowledge (e.g., abdomen can't bend as much as knee) which makes it possible to train up to 17 DOF. In contrast, our agents have no joint angle, or torque, limits which makes the task much harder. Hence, as shown in Supplementary Figure 1, the monolithic baseline works until 4 limbs (i.e., 12 DOF), but fails to scale beyond that.

[R1] "...child node connected to the same parent experiences the same torque? What happens when a node does not have any children? coordinate frames are these torques?": The torque on parent/child differs with respect to their location/configuration. The center of rotation of applied torque is the center of mass of the limb, and the axes orientation are aligned with the limb's rotation. Hence, each limb directly only experiences the torque it exerts on itself. However, when it is connected to other limbs, its torque can affect its neighbors physically.
Yes, in absence of children, torque is between the world and the limb itself. We will add it to the environment details.

[R1] "range of magnetic force?": The range of the magnetic force is approximately 1.33 times the length of a limb.

[R1] "...setup is more similar to a RNN than a GNN": We refer to it as graph because the topology is tree structured. Standard RNNs don't have this topology and are usually applied in a linear sequence.

[R1] "generalizations fall to >90%... benefit might be the trainability": This is a good point. We will clarify that one benefit of our method is that it trains better, which partially explains its high performance at test time. However, in the locomotion experiments, we see that the generalization gap (the difference between training and test performance) is substantially lower for our method compared to the baselines.

[R1] plot "max-performance after k steps as a function of number of links": We show this plot for standing task in Figure 1 for our DGN (w/ msgs) model.

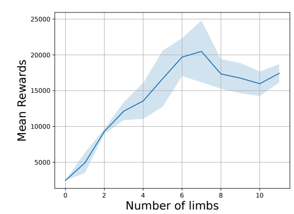

Figure 1: DGN in standing task: performance wrt # of limbs.

[R1, R2] show "standard deviation of runs at test time": We trained two models per experiment with 50 episodic evaluations for each model at every checkpoint time-step. The numbers reported in table are mean performance scores. We couldn't report std-errors in Tables due to space congestion, and will include them in the final version (1 extra page is allowed in camera ready).

[R2] "surprised not to see qualitative evaluation of learned morphologies, policies": Qualitative results are provided in the supplementary as a video (`ProjectVideo.mp4`) for both of the training tasks and 8 evaluation scenarios, together with baselines and narrated audio. In case R2 missed the video, we highly encourage them to check it out.

[R2] "architecture, training procedure details.. important to replicate results": Supplementary material contains (a) training details (Sec A.1), (b) pseudo-code algorithm box (Sec A.4), (c) full source code for reproducibility.

[R2] "environments that are not standard... why not environments built around Mujoco": We did try Mujoco briefly but found it too slow to simulate lots of individual *controllable* limbs in parallel. Hence, we switched to the standard Unity ML-agents framework [Juliani et.al. 2018], which is a dominant platforms for designing realistic games and is efficient. Details like contact forces, control frequency etc. are kept as similar to Mujoco as possible.

[R2] "nature of these messages is not clear... how much the controller on each link knows about the overall morphology": We describe how messages are passed in Lines 166-182. Briefly, they are 32-dim *learned* vectors passed from one limb to its connected neighbors. Each limb only receives its own sensory data (e.g. its touch, depth sensor), and can only get to know about far-away limbs states by *learning to pass* meaningful messages. An algorithm box with pseudo-code, as well as the actual source code, are included in the supplementary.

[R2] "msgs vs. no msgs": In no-msgs case, a limb can't get to know about other limbs it is not directly connected to, while in msg passing, a limb can get to know so by learning to pass meaningful messages.

[R2] "designs hand-crafted by the authors rather than design experts": We made our best effort to design a working monolithic architecture. We tried linear morphology for standing and star-shaped for locomotion. A random morphology doesn't work well for standing, but does okay for locomotion.

[R2] "which axis is the surface height grid projected": It is the y-coordinate height of that block of the floor.

[R3] train modular graph policy on fixed morphology: Upon R3's great suggestion, we trained a modular DGN policy for static morphology. The performance is significantly better than 'monolithic policy, static graph' but worse than our final self-assembling DGN. This suggests that modularity is the key component for generalization. We do not have the space to include plot here, but will include in the paper. Nevertheless, regardless of generalization properties, one of the main contribution of our work is showing how could dynamic agents be trained to self-assemble.

[R3] "From the video it seems that the robot sometimes breaks apart": Those are all due to the individual agent producing the unlink action. The agent cannot break in other conditions in the current set up.