SymNet 3.0: Exploiting Long-Range Influences in Learning Generalized Neural Policies for Relational MDPs (Supplementary Material)

Vishal Sharma[*1] **Daman Arora**[*1] **Mausam**[1] **Parag Singla**[1]

[1]Indian Institute of Technology Delhi {vishal.sharma, cs5180404, mausam, parags}@cse.iitd.ac.in

1 PROOFS

Theorem 4. *For a node n in the influence graph, let $L(n, k)$ denote the multi-set of node features of nodes that are exactly k hops away from node n in the influence graph. In reference to theorem 1, given the features of nodes n_1 and n_2, if there exists a $k > 0$ such that $L(n_1, k) \neq L(n_2, k)$, then, given a sufficiently powerful attention function* SYMNET3.0 *has the power to learn the parameters that break the symmetry induced between s_1 and s_2 which have the features of nodes n_1 and n_2 swapped.*

Proof (Sketch). The high level intuition of the proof is that their will exist certain "key-nodes" in the graph which have unique features. For example, in the Navigation domain, it will be the Goal location. For the Pizza domain, it would be the location of the Pizza and the Customer. If a node is using distance from these key-nodes, then it is possible to break symmetries induced by a fixed-depth GAT. The more the number of key-nodes, the easier it is to break the symmetry. The formal proof is as follows:

Let $C_{f,n,d}$ denote the number of times f occurs in $L(n, d)$. As $L(n_1, k) \neq L(n_2, k)$, $\exists f'$, s.t. $C_{f',n_1,k} \neq C_{f',n_2,k}$. With a sufficiently powerful attention function, in the state s_1, node n_1 can focus attention on f' to learn a node embedding different from that of n_2.

We construct one such set of parameters for SYMNET3.0 that will break the symmetry among s_1 and s_2 with respect to nodes n_1 and n_2.

1. GAT_{pre} can learn an identity mapping for each node by focusing all attention on itself and those nodes in its neighbourhood that have exactly the same features as itself while ignoring all other neighbours.
2. Next, consider the following (un-normalized) attention function in the influence layer.

$$e(f_i, f_j, d_{ij}) = \begin{cases} 0 & d = 0 \\ 0 & f_j = f', d_{ij} = k \\ -INF & \text{otherwise} \end{cases}$$

   where INF is a very large positive number.
3. Next, GAT_{post} can also learn an identity mapping (similar to GAT_{pre}).

The above parameters ensure that, in the influence layer, any given node gives a non-zero attention weight (after normalization) to itself and to any other node at a distance k having features f'. In state s_1, n_1's attention is spread over n_1 and those nodes at a distance k that have f' as their features. Therefore the influence embedding for n_1 in s_1 will be $\frac{C_{f',n_1,k}}{C_{f',n_1,k}+1} * k$.

[*]Equal Contribution

Similarly for n_2 in s_2, it will be $\frac{C_{f',n_2,k}}{C_{f',n_2,k}+1} * k$. Since $C_{f',n_1,k} \neq C_{f',n_2,k}$, the embeddings will be different when the features are swapped, thus breaking the symmetry among n_1 in s_1 and n_2 in s_2. $\square$

An example of attention function in the influence layer that can break symmetry: Additionally, we also provide an explicit construction of attention weights of the influence layer, that is independent of f'. Assume that the features of nodes come from a finite-ordered set F and there exists a function $idx_F : F \rightarrow \mathbb{N}$ that returns the index of a feature in the ordered-set F. Consider the un-normalized attention function for $m, n \geq 1$,

$$a_{m,n}(f_i, f_j, d_{ij}) = \begin{cases} 0 & d_{ij} = 0 \\ 0 & idx(f_j) = m \text{ and } d_{ij} = n \\ -INF & \text{otherwise} \end{cases} \tag{1}$$

where $-INF$ is a very large negative number. Since the influence layer has multi-head attention, we can assign each head with a different attention function. Specifically, we assign $a_{m,n}$ to the $(n|F| + m)^{th}$ attention head. Note that if a graph has $|G|$ nodes, this ensures there are atmost $|G||F|$ attention heads.

Note that given these attention heads, it is possible to encode the multi-set of neighbours at a distance k in the influence embedding! Let $C_{f,n,d}$ denote the number of times feature f occurs in the d-hop neighbour of node n. If we're given that $L(n_1, k) \neq L(n_2, k)$, we can say that $\exists f' \in F$ such that $C_{f,n_1,k} \neq C_{f,n_2,k}$.

Consider the embedding of node n_1 in state s_1, specifically the $a^{th}_{idx(f'),k}$ attention head which would correspond to the $(k|F| + idx(f'))^{th}$ element of the influence embedding. Equal attention would be spread over n_1 and $C_{f',n_1,k}$ nodes. Therefore the aggregated distance would be $\frac{C_{f',n_1,k}}{1+C_{f',n_1,k}} * k$. Correspondingly for n_2 in s_2, this element would be $\frac{C_{f',n_2,k}}{1+C_{f,n_2,k}} * k$. Since $C_{f',n_1,k} \neq C_{f',n_2,k}$ the embedding for n_1 in s_1 would not equal the embedding for n_2 in s_2. A similar argument can be made for n_2 in s_1 and n_1 in s_2. In practice this kind of a construction would require the dimension of the heads to scale with the size of the graph, however this is an exaggeration in the practical setting. In practical domains, there are only a fixed-small number of key features, and just considering the distance from them is sufficient for computing the policy.

2 RDDL EXAMPLE

The IPPC domain of Navigation is a 2D grid world where a robot has to reach a goal cell. Each cell in the grid has a death probability with which the robot can die. The agent receives a $+1$ reward for reaching the goal and 0 otherwise.

Object Types: `x, y`

Non-Fluents: `north(y, y), south(y, y), east(x, x), west(x, x), min_x(x), max_x(x), prob(x, y), goal(x, y)`

State-Fluents: `robot_at(x, y)`

Actions: `move_north, move_south, move_east, move_west`

3 EXPERIMENTAL DETAILS

- **Data generation:** For each domain, we generate 1000 training, 100 validation, and 200 test instances with size increasing from train to val to test instances. Similar to SymNet2.0, we use state-of-the-art online planner PROST and generate 30 trajectories of each training instance using the default settings.
- **Architectural Details**: For our experiments with SymNet2.0, we use a GAT with depth 4, having shared weights across layers, each layer having 10 attention heads. For SymNet3.0, both the pre-processing and post-processing GATs are of depth 2 and have 10 attention heads, with shared weights. For SymNet3.0, the influence layer uses 10 attention heads. The final node embedding dimension for both models is 20, and action decoders used are MLPs with 1 hidden layer of dimension 20.
- **Training details**: We train all models for 48 hours on a $K40$ GPU using imitation learning for LR domains and for 24 hours for IPPC domains. Each checkpoint is evaluated on validation instances and we pick the one with best average for testing.

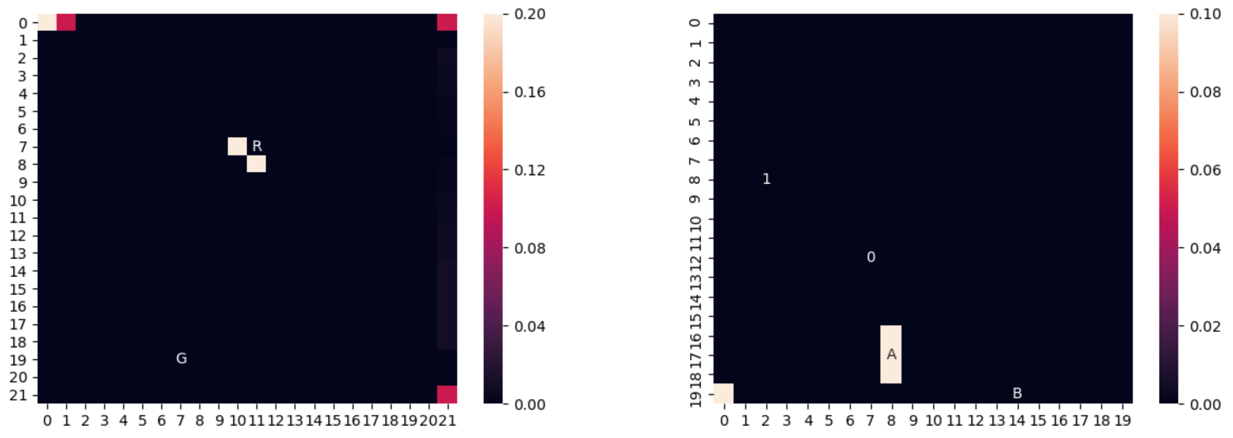

Figure 1: (left) Figure shows the attention map averaged across all heads for the robot's location computed by SYMNET3.0+KL for the DNav domain. We note that the attention heads focus on the corners of the grid helping in the localization of all nodes. (right) Figure shows the attention map averaged across all heads for the robot's location computed by SYMNET3.0+KL for the SRecon domain. Here, 0 and 1 denote the object 0 and object 1. We note that the attention heads focus on one of the corners of the grid.

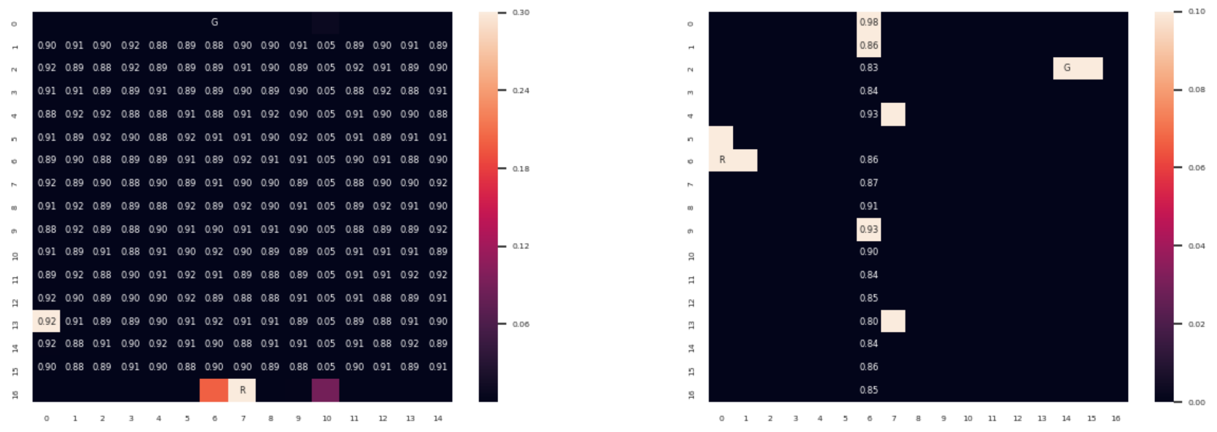

Figure 2: (left) Figure shows the attention map averaged across all heads for the robot's location computed by SYMNET3.0+KL for the StNav domain. Here, the probability of death of each cell is written on the cell. We note that the attention is focused on the entrance of the column which is safest. (right) Figure shows the attention map averaged across all heads for the robot's location computed by SYMNET3.0+KL for the StWall domain. Here, the attention is focuses on the goal and the cells near the safe passage cell.

4 DETAILED RESULTS AND ATTENTION MAPS

The detailed results of experiments for each run of various models for LR domains is shown in Table 1 and for IPPC domains is shown in Table 2

Tables 3 and 4 show the results when the best model among SYMNET3.0-KL, SYMNET3.0+KL$_D$ and SYMNET3.0+KL is chosen based on validation scores..

Model	SRecon	Pizza	DNav	StWall	EAcad	StNav	Mean
PROST	0.34	0.09	0.94	0.69	0.37	0	0.67
SymNet2[1]	0.49	0.22	0.74	0.26	0.89	0	0.63
SymNet2[2]	0.47	0.35	0.57	0.23	0.89	0.01	0.56
SymNet2[3]	0.49	0.27	0.46	0.27	0.9	0	0.54
SymNet2[4]	0.43	0.11	0.43	0.31	0.9	0.13	0.55
SymNet2[5]	0.47	0.33	0.57	0.27	0.9	0.03	0.58
SymNet3-KL[1]	0.63	0.65	0.86	0.47	0.95	0	0.76
SymNet3-KL[2]	0.63	0.43	0.83	0.31	0.72	0.2	0.62
SymNet3-KL[3]	0.73	0.69	0.87	0.42	0.81	0	0.7
SymNet3-KL[4]	0.7	0.64	0.8	0.24	0.9	0.08	0.65
SymNet3-KL[5]	0.72	0.66	0.86	0.24	0.95	0.09	0.68
SymNet3+KL$_D$[1]	0.64	0.42	0.96	0.44	0.96	0.43	0.79
SymNet3+KL$_D$[2]	0.55	0.82	0.91	0.31	0.9	0.01	0.71
SymNet3+KL$_D$[3]	0.59	0.55	0.88	0.4	0.86	0.11	0.71
SymNet3+KL$_D$[4]	0.72	0.44	0.92	0.31	0.94	0.04	0.72
SymNet3+KL$_D$[5]	0.61	0.67	0.88	0.41	0.95	0.17	0.75
SymNet3+KL[1]	0.46	0.09	0.92	0.43	0.92	0.05	0.76
SymNet3+KL[2]	0.65	0.31	0.93	0.33	0.91	0.02	0.72
SymNet3+KL[3]	0.67	0.14	0.97	0.27	0.94	0.03	0.73
SymNet3+KL[4]	0.66	0.18	0.93	0.36	0.95	0.15	0.75
SymNet3+KL[5]	0.61	0.18	0.98	0.36	0.83	0	0.72

Table 1: Performance of all runs of different models on 6 LR domains.

5 SIZES OF INSTANCES

In the spirit of transfer, the sizes of instances increase from training to validation to test instances. A measure of size is the number of state fluents present in the instance. We report the minimum and maximum of train, validation and the test sets for LR domains in table 5 and for IPPC domains in table 6.

6 DOMAINS AND GENERATORS

1. **Deterministic Navigation (DNav)**

   Deterministic Navigation involves a Robot and a Goal cell located in a square grid. For each step that the Robot is not in the Goal cell, it receives a reward of -1. The optimal policy requires the Robot to reach the Goal in the minimum number of timesteps. To generate instances, first the grid size is sampled uniformly from $[D_{min}, D_{max}]$ and then the goal and start cell of the robot is samples uniformly from the grid cells. Parameters for generation:

   (a) D_{min}: Minimum allowed grid dimension

   (b) D_{max}: Maximum allowed grid dimension.

   To generate the train, validation, and test sets, we use the parameters mentioned in Table 7

2. **Extreme Academic Advising (EAcad)** Extreme Academic Advising consists of various courses which are arrange in a Directed Acyclic Graph. Certain courses are program requirements. For each time step, every program requirement that is not completed adds a negative reward to the total reward. In order to get high reward, an agent must complete program requirements in the shortest amount of time possible. If all the pre-requisites of a course have been completed then the probaility of completion of the course when attempted is 0.95. Otherwise, the probability of completion is 0.05. This incentivizes an agent to complete courses in the DAG order specifically leaving out courses which are not ancestors to a requirement course. To generate the courses, we set L which is the number of levels, and C the number

Model	Tam	Traffic	Sys	Skill	Nav	TT	Recon	Elev	Acad	CT	Wild	GoL	Mean
PROST	0.86	0.91	0.76	0.84	0	0.03	0.59	0.91	0.64	0.34	0.32	0.57	0.56
SYMNET2[1]	0.91	0.9	0.76	0.84	0.09	0.76	0.41	0.9	0.88	0.65	0.71	0.82	0.72
SYMNET2[2]	0.92	0.88	0.72	0.78	0.69	0.79	0.3	0.91	0.85	0.91	0.51	0.83	0.76
SYMNET2[3]	0.89	0.87	0.85	0.86	0.59	0.82	0.3	0.92	0.9	0.76	0.47	0.72	0.75
SYMNET2[4]	0.89	0.9	0.81	0.83	0.45	0.76	0.35	0.95	0.64	0.89	0.63	0.8	0.74
SYMNET2[5]	0.89	0.86	0.82	0.79	0.88	0.77	0.37	0.93	0.86	0.82	0.78	0.75	0.79
SYMNET3-KL[1]	0.92	0.85	0.87	0.85	0.78	0.77	0.21	0.92	0.89	0.76	0.88	0.82	0.79
SYMNET3-KL[2]	0.91	0.89	0.82	0.84	0.29	0.59	0.41	0.88	0.68	0.84	0.8	0.75	0.73
SYMNET3-KL[3]	0.93	0.83	0.75	0.84	0.2	0.8	0.61	0.88	0.62	0.81	0.64	0.78	0.72
SYMNET3-KL[4]	0.91	0.84	0.79	0.77	0.88	0.53	0.48	0.89	0.65	0.77	0.74	0.8	0.75
SYMNET3-KL[5]	0.89	0.82	0.83	0.73	0.5	0.79	0.41	0.76	0.79	0.8	0.73	0.82	0.74
SYMNET3+KL$_D$[1]	0.9	0.85	0.81	0.76	0.87	0.77	0.23	0.94	0.92	0.77	0.66	0.79	0.77
SYMNET3+KL$_D$[2]	0.89	0.86	0.83	0.83	0.84	0.82	0.36	0.28	0.55	0.86	0.66	0.83	0.72
SYMNET3+KL$_D$[3]	0.9	0.83	0.85	0.75	0.86	0.76	0.19	0.81	0.78	0.81	0.58	0.69	0.73
SYMNET3+KL$_D$[4]	0.91	0.87	0.84	0.84	0.8	0.8	0.37	0.86	0.8	0.72	0.7	0.76	0.77
SYMNET3+KL$_D$[5]	0.9	0.86	0.8	0.67	0.87	0.22	0.31	0.66	0.86	0.76	0.47	0.77	0.68
SYMNET3+KL[1]	0.91	0.86	0.82	0.63	0.85	0.81	0.18	0.93	0.8	0.76	0.53	0.65	0.73
SYMNET3+KL[2]	0.9	0.86	0.78	0.7	0.85	0.8	0.29	0.9	0.89	0.72	0.56	-0.27	0.67
SYMNET3+KL[3]	0.91	0.86	0.84	0.68	0.87	0.66	0.27	0.9	0.75	0.89	0.21	0.76	0.72
SYMNET3+KL[4]	0.89	0.81	0.83	0.74	0.72	0.74	0.21	0.92	0.88	0.83	0.4	-0.1	0.66
SYMNET3+KL[5]	0.9	0.86	0.83	0.75	0.27	0.7	0.26	0.9	0.67	0.8	0.36	-0.13	0.6

Table 2: Performance of all runs of different models on 12 IPPC domains.

of courses per level. Additionally, each courses has, on average p number of prerequisites from the previous level. The number of course requirements is R, and the are sampled with a probability proportional to the square of their level. This is done so as to choose courses which require a lot of pre-requisites to be completed in order. Parameters of generation are:

(a) L_{min}: Minimum number of levels
(b) L_{max}: Maximum number of levels
(c) C_{min}: Minimum number of courses per level
(d) C_{max}: Maximum number of courses per level
(e) p: Average number of pre-requisites

To generate train, val, and test sets, we use the parameters mentioned in Table 8.

3. **Safe Recon (SRecon)** In Safe Recon, an agent has to traverse on a rectangular grid and take pictures of object where it detects life. Once an agent reaches at an object, it must apply tools("water" and "life"), in the correct order(first water, then life). Tools can fail with some probability. Once life has been detected, the agent can take pictures which gives it positive reward until the end of the episode. If an tool is damaged, it can go back to BASE to repair its tool or use damaged tools. Using damaged tools is risky because the a negative reward is given for each photo clicked which doesn't have life. This domain is identical to the one used for IPPC 2014, with the difference being that we do not use HAZARDS in our version. The parameters for instance generation are:

(a) D_{min} Minimum grid size
(b) D_{max} Maximum grid size
(c) O_{min} Minimum number of objects
(d) O_{max} Maximum number of objects
(e) p_{min} Minimum threshold for tool damage probability
(f) p_{max} Maximum threshold for tool damage probability

Model	SRecon	Pizza	DNav	StWall	EAcad	StNav	Mean
PROST	0.34	0.09	0.94	0.69	0.37	0	0.41
SYMNET2	0.47	0.26	0.55	0.27	0.9	0.03	0.41
SYMNET3-KL	**0.68**	**0.62**	0.84	0.33	0.87	0.08	0.57
SYMNET3+KL$_D$	0.62	0.58	0.91	**0.38**	**0.92**	**0.15**	0.59
SYMNET3+KL	0.61	0.18	**0.95**	0.35	0.91	0.05	0.51
SYMNET3(best val)	0.68	0.58	**0.95**	0.38	0.91	0.15	**0.61**

Table 3: Comparison of SYMNET3.0 variants with the baselines on 6 LR domains. The last row denotes the score of the best among SYMNET3.0+KL, SYMNET3.0-KL and SYMNET3.0-KL_D chosen on the basis of average validation reward.

Model	Tam	Traffic	Sys	Skill	Nav	TT	Recon	Elev	Acad	CT	GoL	Wild	Mean
PROST	0.86	0.91	0.76	0.84	0	0.03	0.59	0.91	0.64	0.34	0.32	0.57	0.56
SYMNET2	0.9	**0.88**	0.79	**0.82**	0.54	**0.78**	0.35	**0.92**	**0.83**	**0.81**	0.62	0.78	**0.75**
SYMNET3-KL	**0.91**	0.85	0.81	0.81	0.53	0.7	**0.42**	0.87	0.73	0.8	**0.76**	**0.79**	0.75
SYMNET3+KL$_D$	0.9	0.85	**0.83**	0.77	**0.85**	0.67	0.29	0.71	0.78	0.78	0.61	0.77	0.73
SYMNET3+KL	0.9	0.85	0.82	0.7	0.71	0.74	0.24	0.91	0.8	0.8	0.41	0.18	0.67
SYMNET3(best val)	0.91	0.85	0.83	0.81	0.85	0.67	0.42	0.91	0.78	0.8	0.76	0.79	**0.78**

Table 4: Comparison of SYMNET3.0 variants with the baselines on 12 IPPC domains. The last row denotes the score of the best among SYMNET3.0+KL, SYMNET3.0-KL and SYMNET3.0-KL_D chosen on the basis of average validation reward.

To generate the train, validation, and test sets we use the parameters mentioned in Table 9.

4. **Pizza Delivery (Pizza)** Pizza Delivery consists of a rectangular grid of width w and height h. In addition, the grid contains d pizza shops which are subgoals. The agent must collect the pizza from any pizza shop and deliver it to the customer in the minimum time possible. Additionally, a wind blows which can randomly push you to any neighbouring cell. To generate domains, a start location s, customer location c and a special pizza shop location p' is sampled uniformly randomly. Next, to sample the remaining $d-1$ goals, we remove all cells p such that

$$dist(s,p) \geq dist(s,p')$$

or

$$dist(s,p) + dist(c,p) \leq dist(s,p') + dist(c,p')$$

This is done so that the planner doesn't go to the nearest pizza shop but learns to minimize the sum of both distances(distance to shop+distance to customer). Out of the candidate cells, we sample with probability proportional to the distance from s. Parameters for generation:

(a) w_{min}: Minimum grid width
(b) w_{max}: Maximum grid width
(c) h_{min}: Minimum grid height
(d) h_{max} Maximum grid height
(e) d_{min}: Minimum number of pizza shops
(f) d_{max} Maximum number of pizza shops

To generate the train, validation, and test sets we use the parameters mentioned in Table 10.

Domain	Train(min)	Train(max)	Val(min)	Val(max)	Test(min)	Test(max)
SRecon	48	193	208	249	373	924
Pizza	34	153	205	265	409	493
EAcad	8	36	42	192	120	320
StWall	25	100	121	225	256	400
DNav	81	196	225	324	400	625

Table 5: Number of state fluents for LR domains for training, validation, and test sets.

Domain	**Train(min)**	**Train(max)**	**Val(min)**	**Val(max)**	**Test(min)**	**Test(max)**
Acad	4	50	60	96	120	240
CT	12	84	112	144	180	312
GoL	4	36	42	64	81	100
Skill	6	42	12	42	30	60
Recon	29	81	68	107	120	257
TT	12	75	108	147	192	300
Wild	10	72	72	128	128	288
Tam	2	48	28	84	48	140
Elev	9	32	18	32	24	60
Traffic	32	80	32	80	56	104
Sys	2	15	2	14	15	25
Nav	9	49	25	90	120	120

Table 6: Number of state fluents for IPPC domains for training, validation and test sets.

	D_{min}	D_{max}	Horizon
Train	9	14	40
Val	15	18	60
Test	20	25	60

Table 7: Table shows the parameters used in to generate instances in the DNav domain.

	L_{min}	L_{max}	C_{min}	C_{max}	p	Horizon
Train	2	8	2	8	1	40
Val	7	12	3	8	1	100
Test	12	20	5	8	1	200

Table 8: Table shows the parameters used in to generate instances in the EAcad domain.

	D_{min}	D_{max}	O_{min}	O_{max}	p_{min}	p_{max}	Horizon
Train	6	13	2	4	0	0.5	40
Val	13	14	2	4	0	0.5	80
Test	18	19	2	4	0	0.5	80

Table 9: Table shows the parameters used in to generate instances in the SRecon domain.

	w_{min}	w_{max}	h_{min}	h_{max}	d_{min}	d_{max}	Horizon
Train	5	12	5	12	2	4	100
Val	14	16	14	16	2	4	150
Test	20	22	20	22	2	4	200

Table 10: Table shows the parameters used in to generate instances in the Pizza domain.

	n_{min}	n_{max}	Horizon
Train	5	10	40
Val	11	15	40
Test	16	20	40

Table 11: Table shows the parameters used in to generate instances in the StWall domain.

5. **Stochastic Wall (StWall)** Stochastic Wall consists of a square grid of dimension n. Uniformly randomly, a row or a column is sampled to form a barrier. The start and goal location are sampled such that they lie on opposite sides of the barrier. In the barrier, a cell is selected to be a safe passageway from one part of the grid to the other. Hitting the barrier can cause death with some probability sampled from $\mathcal{U}(0.8, 1.0)$. The agent must navigate from his initial position to the goal. The generation parameters are:

	w_{min}	w_{max}	h_{min}	h_{max}	Horizon
Train	5	10	5	10	40
Val	11	15	11	15	40
Test	15	20	15	20	40

Table 12: Table shows the parameters used in to generate instances in the StNav domain.

(a) n_{min}: Minimum size of the grid.

(b) n_{max}: Maximum size of the grid.

To refer to the generation parameters for train, val and test splits, refer to table 11.

6. **Stochastic Navigation (StNav)** This domains consists of a grid of width w and height h. The bottom and top rows are safe. The start state is sampled from the bottom row and the goal state is sampled in the top row. In the middle rows, the robot can die with some probability. However, there exists a single column which has very low death probability from $\mathcal{U}(0.045, 0.055)$. For all other column, the death probability in each cell is from $\mathcal{U}(0.88, 0.92)$. This task has long range dependencies because the agent has to decide which column to enter into. The column could be very far and thus SymNet2.0might not be able to decide which direction to take. The generation parameters are:

   (a) w_{min}: Minimum width of the grid.

   (b) w_{max}: Maximum width of the grid.

   (c) h_{min}: Minimum height of the grid.

   (d) h_{max}: Maximum height of the grid.

   To refer to the generation parameters for train, val and test splits, refer to table 12