APPENDIX

A NON-LINEAR REWARD DERIVATION

Here we provide the derivation for the non-linear reward in the successor framework. First, we start by assuming the reward r_t has the following form:

$$r_t = \phi_t^\top \mathbf{o} + \phi_t^\top \mathbf{A} \phi_t \tag{10}$$

where $\{\phi_t, \mathbf{o}\} \in \mathbb{R}^{z \times 1}$, and $\mathbf{A} \in \mathbb{R}^{z \times z}$ and both $\mathbf{o}$ and $\mathbf{A}$ are learnable parameters. Following from the definition of the state-action value function $Q(s, a)$, the adjusted reward function can be substituted to yield:

$$Q^\pi(s, a) = \mathbb{E}^\pi[r_{t+1} + \gamma r_{t+2} + \dots | S_t = s, A_t = a] \tag{11}$$

$$= \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{o} + \phi_{t+1}^\top \mathbf{A} \phi_{t+1} + \gamma \phi_{t+2}^\top \mathbf{o} + \gamma \phi_{t+2}^\top \mathbf{A} \phi_{t+2} + \dots | S_t = s, A_t = a] \tag{12}$$

Dropping the conditional portion of the expectation for brevity, linearity of expectation can be used to split apart the terms containing $\mathbf{A}$ and $\mathbf{o}$. Then $\mathbf{o}$ is pulled out from the first term:

$$= \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{o} + \gamma \phi_{t+2}^\top \mathbf{o} + \dots] + \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{A} \phi_{t+1} + \gamma \phi_{t+2}^T \mathbf{A} \phi_{t+2} + \dots] \tag{13}$$

$$= \mathbb{E}^\pi[\phi_{t+1} + \gamma \phi_{t+2} + \dots]^\top \mathbf{o} + \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{A} \phi_{t+1} + \gamma \phi_{t+2}^\top \mathbf{A} \phi_{t+2} + \dots] \tag{14}$$

By recognizing the first expectation term as the successor features $\psi(s, a)$, Equation 14 can be rewritten as

$$= \psi^\pi(s, a)^\top \mathbf{o} + \mathbb{E}^\pi[\phi_{t+1}^\top \mathbf{A} \phi_{t+1} + \gamma \phi_{t+2}^\top \mathbf{A} \phi_{t+2} + \dots] \tag{15}$$

Because $\phi^\top A \phi$ results in a scalar, the trace function $tr(\cdot)$ can be used inside the right-hand term:

$$= \psi^\pi(s, a)^\top \mathbf{o} + \mathbb{E}^\pi[\mathbf{tr}(\phi_{t+1}^\top \mathbf{A} \phi_{t+1}) + \mathbf{tr}(\gamma \phi_{t+2}^\top \mathbf{A} \phi_{t+2}) + \dots] \tag{16}$$

By exploiting the fact that $\mathbf{tr}(\mathbf{AB}) = \mathbf{tr}(\mathbf{BA})$, the terms inside the trace function can be swapped to yield:

$$= \psi^\pi(s, a)^\top \mathbf{o} + \mathbb{E}^\pi[\mathbf{tr}(\mathbf{A} \phi_{t+1} \phi_{t+1}^\top) + \mathbf{tr}(\gamma \mathbf{A} \phi_{t+2} \phi_{t+2}^\top) + \dots] \tag{17}$$

Because both $tr(\cdot)$ and $\mathbf{A}$ are linear, they can be pulled out of the expectation, giving:

$$= \psi^\pi(s, a)^\top \mathbf{o} + \mathbf{tr}(\mathbb{E}^\pi[\mathbf{A} \phi_{t+1} \phi_{t+1}^\top + \gamma \mathbf{A} \phi_{t+2} \phi_{t+2}^\top + \dots]) \tag{18}$$

$$= \psi^\pi(s, a)^\top \mathbf{o} + \mathbf{tr}(\mathbf{A} \mathbb{E}^\pi[\phi_{t+1} \phi_{t+1}^\top + \gamma \phi_{t+2} \phi_{t+2}^\top + \dots]) \tag{19}$$

Finally, the remaining expectation can be expressed as a function:

$$Q^\pi(s, a) = \psi^\pi(s, a)^\top \mathbf{o} + \beta \mathbf{tr}(\mathbf{A} \Lambda^\pi(s, a)) \tag{20}$$

$\beta \in \{0, 1\}$ is a hyperparameter that controls the inclusion of the non-linear component. We define ψ^π and Λ^π as:

$$\psi^\pi(s, a) = \mathbb{E}^\pi[\phi_{t+1} + \gamma \psi(s_{t+1}, \pi(s_{t+1})) | S_t = s, A_t = a] \tag{21}$$

$$\Lambda^\pi(s, a) = \mathbb{E}^\pi[\phi_{t+1} \phi_{t+1}^\top + \gamma \Lambda(s_{t+1}, \pi(s_{t+1})) | S_t = s, A_t = a] \tag{22}$$

B A MATRIX FACTORIZATION

Similar to Λ, as the dimensionality of z increases, so does the number of parameters needed for modelling matrix $\mathbf{A} \in \mathbb{R}^{z \times z}$. Therefore, in the interest of reducing the number of parameters we use a factorization that splits the matrix $\mathbf{A} \in \mathbb{R}^{z \times z}$ into two parts with a smaller inner dimension f, $\mathbf{A} = \mathbf{L} \cdot \mathbf{R}^\top$, where $\{\mathbf{L}, \mathbf{R}\} \in \mathbb{R}^{z \times f}$. By factoring the matrix in this way, we require $2 \times z \times f$ parameters instead of $z \times z$. If we use values for f smaller than $\frac{z}{2}$, we reduce the number of parameters required by matrix $\mathbf{A}$. A similar factorization was suggested in the context of visual question answering (Yu et al., 2017; Fukui et al., 2016). The factorization of $\mathbf{A}$ was primarily done to reduce the total number of parameters in our model.

C ENVIRONMENTS

C.1 AXES

Within this environment eight separate goal locations exist split between train and test distributions. We used a modified version of the robotic model provided by Metaworld (Yu et al., 2019). An episode ends when either the agent reaches the goal or more than 25 steps have elapsed. The agent's starting location is randomly sampled from a grid of 3×3 step units, centered at $(0, 0)$.

With this state space the agent must learn a reward function that can approximate the distance between itself and the goal location, $d(a, b) = \sqrt{(b_x - a_x)^2 + (b_y - a_y)^2}$, a non-linear function.

C.2 REACHER

Similarly to Axes, the environment has predefined tasks split between training and test distributions, with the eight goals shown in Figure 2b as green and red balls, respectively. In the Reacher environment, an episode ends when 150 steps have elapsed or the agent is within 7cm of the goal. The agent receives a reward equal to the negative distance between the end-effector and the current target goal at each step.

We discretize the actions such that the agent has nine discrete actions that control the arms movements. Because the models can be used only with discrete actions, it was necessary to transform the environmental actions. Therefore, the four-dimensional continuous action space $\mathcal{A}$ was discretized using two values per dimension: the maximum positive and maximum negative torque for each actuator. An all-zero option was included that applies zero torque along all actuators, resulting in a total of nine discrete actions.

C.3 DOOM

In the Doom environment the agent must traverse between four rooms looking for a goal. The rooms are separated by doors that the agent must manually open. At each step the agent receives a small negative reward of -0.01 and upon finding the goal it receives +50. The agent perceives the state and the 4 stacked frames of RGB frames of shape (3, 84, 84), corresponding to color channels, width, and height. The agent has 4 actions available: forward, rotate left, rotate right, and activate door. We use an action repeat of 5 across all actions, which differs from the original implementation that used selective action repeat.

D EXPERIMENTS

Within both the Axes and Reacher environment, ϵ-greedy was annealed from 1.0 to a final over the first 250k steps.

In the Axes and Reacher environments the encoder and decoder each respectively contain one and two hidden layers with an embedding size equal to the double the raw state size. Initially, on the Axes environment, the models all used the raw features with no encoder such that $\phi_t = \mathbb{I}(s_t)$. We

found this led to worse performance for the linear model as it now had no chance to learn a suitable encoding of the features for reward prediction. Both ψ and Λ increase the hidden dimension z by a fixed factor before output. This factor z_{factor} depends on the environment. All environments used a discount factor of $\gamma = 0.99$, $\lambda = 0.1$, and updated the parameters every $25k$ steps.

D.1 AXES

An embedding size of $z = 8$ was used for the second-order model and $z = 24$ for the linear model in the Axes environment. The final value used in ϵ-greedy was 0.1 with a learning rate of $\alpha = 2.5e - 4$ was used. The encoder and decoder in this environment were a single fully connected layer with the same embedding size z.

D.2 REACHER

An embedding size of $z = 8$ was used for the second-order model and $z = 24$ for the linear model in the Reacher environment. The final value used in ϵ-greedy was 0.05 with a learning rate of $\alpha = 5e-4$ was used. The encoder and decoder in this environment were a single fully connected layer with the same embedding size z.

D.3 DOOM

An embedding size of $z = 512$ was used for the linear model and $z = 256$ for the second-order model. A encoder network with 3 layers of convolutional layers with hyperparameters (3c, 32o, 8s), (32c, 64o, 4s), and (64c, 64o, 3s) where c is the number of incoming "channels", o is the number of filters, and s is the filter size. The corresponding decoder had 5 layers of transposed convolutional layers: (64c, 256o, 4s), (256c, 128o, 4s), (128c, 64o, 4s), (64c, 32o, 4s), and (32c, 3o, 3s).

While the second-order variant used 2 layer convolutional encoder with: (3c, 16o, 8s), (16c, 32o, 4s). And a corresponding convolutional decoder with 2 layers: (32c, 16o, 4s) and (16c, 3o, 8s).

E TRANSFER

During transfer we reinitialize the reward specific parameters $\mathbf{A}$ and $\mathbf{o}$ to constant values of 0.1. All other model parameters are held frozen and do not change. The learning rate is increased by a factor of $2\times$ on the Axes and Reacher environments. We anneal the exploration parameter from 1 to 0.1 in Axes and to 0.05 on the Reacher environments over the first 200k steps.