A Appendix

A.1 Neural Network Architecture

A neural network is used to estimate the subgoal properties for each subgoal independently. The neural network architecture is similar to that of [3] and implemented in PyTorch [20]. The network takes as input an RGB panoramic image of size $512 \times 128 \times 3$, a *relative goal position image* of size 128×32 and a *relative subgoal position image* of size 128×32. For both *relative position images*, the value as each pixel encode the relative position to the target in the frame of an agent oriented in the direction specified by the column in which the pixel is located; we additionally divide those distance values by 100, so as to keep them of manageable size without needing to use batch norm, which would remove absolute scale.

The network begins with a convolutional encoder block architecture. Each block consists of 2 convolution operations, each followed by a Batch Norm operation (`BatchNorm2d` in PyTorch) with momentum 0.01 and a Leaky ReLU (with "leaky factor" 0.1). Each block concludes with a Max Pooling operation (`MaxPool2d` in PyTorch) of kernel size 2 and stride 2. Six such blocks are used to process the images, with output channel count: $[16, 16, 32, 32, 32, 32]$. Additionally, the relative position images are concatenated (in channel space) after application of the second convolutional encoder block, which is how knowledge of the goal and subgoal location is injected into the system; the image therefore undergoes two convolution block operations without knowledge of the location of the goal/subgoal, and then the remaining operations having received that knowledge. A single 1×1 convolution operation (followed by a single batch norm and leaky ReLU operation) is used to reduce the number of channels from 32 to 4 before flattening into a 64-element vector. Two fully-connected layers are then applied to the vector (each followed by a batch norm and a leaky ReLU) reducing the size of the vector to 32 and 16 elements respectively. A final fully connected operation is applied to map those 16 values to 3, which correspond to the *logits* of success, the expected cost of success R_S, and the expected cost of exploration (failure) R_E. We apply a logistic sigmoid to the logits, to obtain the likelihood of successfully reaching the goal P_S.

A.2 Training Details and Hyperparameters

As mentioned in Sec. 4, the objective function for training consists of three terms: $\mathcal{L}_{\text{comp}}$ (defined in full above), $\mathcal{L}_{\text{supervised}}$, and $\mathcal{L}_{\text{bounds}}$. The supervised learning term is a weighted binary cross entropy loss $\mathcal{L}_{\text{supervised}} = \beta_{\text{supervised}}\texttt{BCELoss}_w(P_S, \ell_{\text{goal}})$; for all experiments, we use a positive weight of 2.0, which approximately compensates for the dataset imbalance in the training data, in which roughly half as many subgoals lead to the goal as those that do not. As mentioned in the text, $\mathcal{L}_{\text{bounds}}$ imposes losses (either via a ReLU activation function or its smoothed variant the softplus function) to penalize negative values of the R_E and R_S subgoal properties and to impose heuristic bounds on the value of Q derived from the performance of the non-learned baseline during data collection. The bounds losses are defined as follows:

$$\begin{aligned}\mathcal{L}_{\text{bounds}} = &\sum_{R_E} \beta_{\text{neg}}\text{softplus}(-R_E) + \sum_{R_S} \beta_{\text{neg}}\text{softplus}(-R_S) \\ &+ \beta_{\text{lower}}\text{ReLU}(Q_{\text{lower}} - Q) + \beta_{\text{upper}}\text{ReLU}(Q - Q_{\text{upper}})\end{aligned} \tag{5}$$

The weights as follows: $\beta_{\text{supervised}} = [1/10$ for University Building Environment, $1/400$ for Guided Maze Environments $]$, $\beta_{\text{neg}} = \beta_{\text{lower}} = 1/50$, $\beta_{\text{upper}} = 1/500$. As discussed in Sec. 4, for high values of $\beta_{\text{supervised}}$, the agent can already perform well on the less complex Guided Maze environments and so the supervised loss weight is decreased for the Guided Maze environments so as to allow us to demonstrate the effectiveness of our training-via-explaining approach. Hyperparameters were chosen primarily via interaction with the Guided Maze environment. We sought a parameter regime in which the comparison objective $\mathcal{L}_{\text{comp}}$ was by far the largest loss term in general, yet where the other training objectives ($\mathcal{L}_{\text{supervised}}$ and $\mathcal{L}_{\text{bounds}}$) were sufficiently large so as to avoid issues with vanishing gradients and non-physical parameter values that they are included to overcome. We reiterate that the auxiliary losses exist to help stabilize training; without these losses, training is sometimes ineffective and the performance of the cannot reliably exceed the performance of the non-learned baseline. For example, preliminary experiments demonstrate that without these losses, the performance of the 4SG planner in the Guided Maze environment roughly matches the non-learned baseline. Additionally, as mentioned in Sec. 7, these losses also help to discourage the learned model from producing unrealistic values for the subgoal properties.

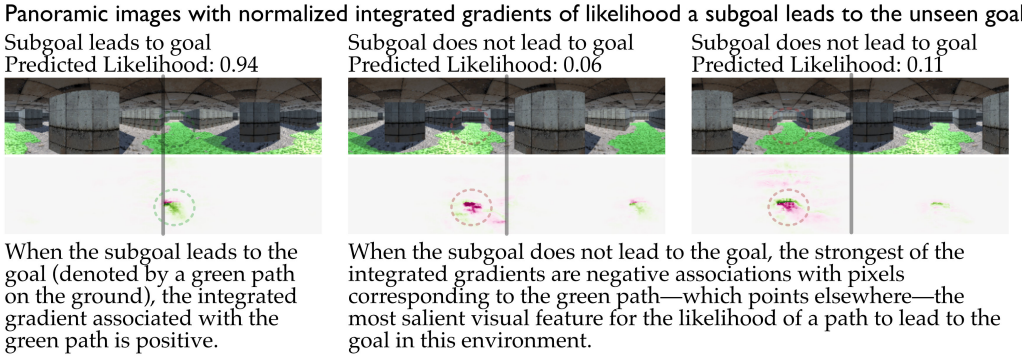

Figure 4: **Preliminary feature attribution results for P_s, the likelihood a subgoal will lead to the goal** These results show that existing tools for interpreting convolutional neural networks highlight salient features useful for informing the predictions of neural networks trained via our approach.

Training proceeds for one epoch and therefore sees each datum only a single time. We begin with a learning rate of 0.02 and decay the learning rate by half every time 1/8 of the data has been consumed. We train and evaluate on a single Nvidia 2060 SUPER GPU. For each planner, training on the Guided Maze environment takes roughly 5 hours and on the University Building environment (which has more and larger data) takes roughly 12 hours.

A.3 Natural Language Grammar

The primary algorithmic component of explanation generation produces a set of subgoal property changes $\Delta\boldsymbol{\sigma}$, from which we use a simple rule-based grammar to generate a natural language explanation. The process is relatively simple: we first order the subgoals by the importance of their most important subgoal and then order the subgoal properties within each subgoal. We loop through this newly-ordered list and for all the most important subgoal properties—those properties whose gradients were used to change the agent's behavior—and inserts them into template strings before concatenating them into an explanation. An example: if the likelihood of success P_S for Subgoal 0 decreased from 0.7 to 0.2 so that the agent would select Subgoal 2, we would output the following: *I would have preferred Subgoal 2 over Subgoal 0 if I instead believed Subgoal 0 were significantly less likely to lead to the goal (20%, down from 70%).* Fig. 1 and Fig. 3 both include example explanations generated via this process. We include the qualifiers *slightly* and *significantly* to denote when the absolute property changes are outside the range of 10% and 40% (respectively) for a change in likelihood (P_S) and 1 meter and 5 meters (respectively) for a change in cost/distance (R_S and R_E). These values are naturally somewhat context-dependent, and so what a significant change looks like may change in environments substantially different from those we include here.

We note that the grammar in its current form are unabridged representations of the underlying explanation: all of the subgoal property used during computation of our counterfactual explanation are included and their numerical values are included. However, we do acknowledge that there may be other ways of presenting this information that may be easier to digest, including not presenting some of the subgoal property changes if the changes are deemed "sufficiently small." What is implied by "sufficiently small" and the potential benefit of such a change will require additional experiments. We also anecdotally observe that human descriptions of navigation in real-world spaces often involves relationships between multiple options, something not permitted by such a simple grammar; in the future it may be possible to update our grammar to include additional language that incorporates the relationship between different subgoals: e.g., that an agent is favoring a route that leads it towards exploring two promising subgoals over another route which explores only a single somewhat promising alternative. In future work, we hope to explore how to best present this information to humans: e.g., changing the number of subgoal properties we use during masking, the grammar we use to generate the language, or the graphical interface.

A.4 Proof of concept pixel attribution for subgoal property estimation

This work focuses on explanations of *high-level* behavior for agents tasked with long-horizon planning in partially-revealed environments. Yet, as we mention in Sec. 7, while the connection between the low-level perception (via images) and the estimated subgoal properties is opaque, there exists

significant recent work devoted to interpreting convolutional neural networks to understand this relationship.

In proof-of-concept experiments, we use an implementation of *Axiomatic Attribution for Deep Networks* [31] provided via the Captum PyTorch interpretability library [13] to interpret P_S, the likelihood of a subgoal to lead to the goal. Fig. 4 shows these preliminary results for a few selected subgoals in the Guided Maze environment for predictions from our learned planner trained using 4 subgoal properties, as described in Sec. 5. These preliminary results show that across the different images—each oriented such that the subgoal of interest is at the center of the panorama—the feature attribution results correctly highlight as the most important visual feature for correctly estimated the probability. When the subgoal leads to the goal, as evidenced by the green path leading toward the subgoal at the center of the image, the attribution is positive indicating that the green path pixels are a positive predictor of for the likelihood P_S. When the subgoal does not lead to the goal, the feature attribution associated with the green path is mostly negative as the green path points elsewhere. This proof of concept experiment suggests the possibility to enhance our explanations of high-level behavior with existing tools for neural network interpretability, which could elucidate the connection between low-level perception and long-horizon behavior in this challenging application domain.