

A APPENDIX

A.1 ARCHITECTURAL DETAILS

We implemented the NPs experts following the network architecture of CNPs and ANPs Garnelo et al. (2018a;b); Kim et al. (2019). The architecture details are shown in Figure 7 in the appendix. All MLPs in the CNPs architecture had relu activation function except the final layer. The latent encoder r_ζ estimated $\mu_z, m_z \in \mathbb{R}^{d_z}$, which parameterized $q(z|h_C) = \mathcal{N}(\mu_z, 0.1 + 0.9 * \sigma_z(m_z))$ where σ_z was the sigmoid function. In a similar way, the decoder estimated $\mu_y, m_y \in \mathbb{R}^{d_y}$, which parameterized $p(y_T|x_T, z) = \mathcal{N}(y_T|\mu_y, 0.1 + 0.9 * \sigma_y(m_y))$ where σ_y was the softplus function. For the GPs expert, we implemented standard GPs and sparse GPs using tensorflow probabilistic library Dillon et al. (2017). The biased gate function was implemented using only MLPs and softmax functions. The bias values for the GPs expert and the CNPs expert were $e^{1.4}$ and 0 respectively.

A.1.1 HYPER PARAMETERS FOR 1D EXPERIMENT

We set $d_z = 128$ for CNPs and ANPs experts and used the basic formulation of multihead cross-attention in Kim et al. (2019). For GPs expert, we exploited the commonly used kernel functions Rasmussen & Williams (2006). In training phase, we used a batch size of 1 and Adam optimizer Kingma & Ba (2015) with a fixed learning rate of $1e^{-4}$ and tensorflow default value for other hyperparameters.

A.1.2 HYPER PARAMETERS FOR 2D EXPERIMENT

The architecture was the same as the one used in 1D experiment. Different from 1D experiment, we set $d_z = 512$ for CNPs and ANPs experts and changed the final layer dimension from 2 to 6 for RGB images in decoder. The training process was the same with 1D experiment.

Table 4: Summary of kernel functions used in the experiments. The kernel functions are written either as a function of x and x' , or as a function of $r = |x - x'|$. The amplitude and length hyperparameters are abbreviated as am and ls , respectively.

Kernel function	Initial hyperparameters
squared exponential	$am(e^0), ls(e^0)$
matérn	$am(e^0), ls(e^0)$
linear	$bias(e^0), slop(e^0), shift(e^0)$
polynomial	$bias(e^0), slop(e^0), shift(e^0), exponent(10)$
rational quadratic	$am(e^0), ls(e^0), scalemixture(e^0)$
Periodic	$am(e^0), ls(e^0), period(e^0)$

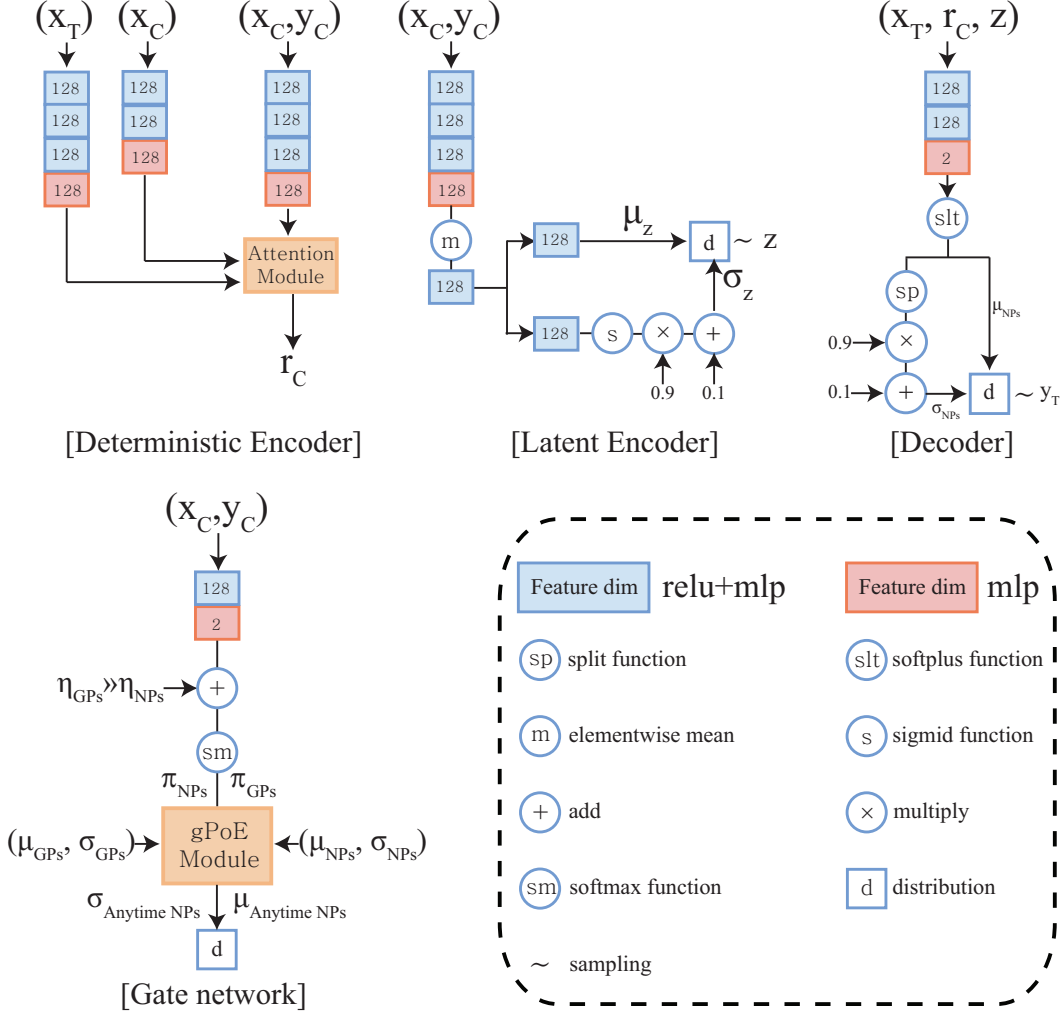


Figure 7: The network architecture for 1D experiments.

Algorithm 1: Continuous occupancy mapping.**Input** : Paired robot pose set \mathbf{p} and point cloud set \mathbf{s} .**Output:** Probabilistic occupancy global map \mathbf{M} .

```

1 for  $i = 0$  to  $|\mathbf{p}| - 1$  do
2    $(X_i, y_i) \leftarrow$  Coordinates of observed grids and corresponding occupancy using  $p_i \in \mathbf{p}$  and
    $s_i \in \mathbf{s}$ .
3   Update parameters of ATNPs using training data  $(X_i, y_i)$ .
4    $(\mu_i, \sigma_i) \leftarrow$  Predict probabilistic occupancy at target locations  $X_i^*$ .
5   Generate local map  $M_i$  using observation  $(X_i, y_i)$  and predictions  $(\mu_i, \sigma_i)$  at locations  $X_i^*$ .
6   Integrate local map  $M_i$  into global map  $\mathbf{M}$ 
7 end

```

A.2 ONLINE-LEARNING TASKS: PROBABILISTIC OCCUPANCY GRID MAP

The proposed method can be used to represent the robot’s surrounding environment. To navigate an unknown environment, robots usually generate coarse, low-resolution occupancy maps on the fly to find a possible path quickly. Detailed and high-resolution occupancy maps are then reconstructed along paths of interest. GPs have been previously used for generating the occupancy grid mapping because they can represent the continuous spatial dependency of a real environment and robustly handle sparse and noisy data O’Callaghan et al. (2009); O’Callaghan & Ramos (2012). However, the computational complexity of GP-based methods is always $\mathcal{O}(N^3)$ with a training data set of size N , which makes it impractical to use for mapping a large-scale environment. In contrast, the proposed method can achieve $\mathcal{O}(N)$ complexity after the transition from GPs to CNPs. Therefore, the proposed method can solve the scalability problem of mapping large environments.

To evaluate the proposed method, we adapted it to generate a probabilistic occupancy grid map. Following the experiment setup Yuan et al. (2018), we used a simple two-dimensional robot simulator (STDR) of a robot operating system (ROS) Quigley et al. (2009). STDR provides various types of synthetic maps, accurate odometry information, and 2D point clouds from LiDAR sensors. We used a sparse obstacle map of size 775×746 with a grid resolution of 0.02 m, as shown in Figure 8(b) in the appendix. To generate the probabilistic occupancy grid map, we extracted the location of observed grids X and their occupancy y using the robot’s position and point clouds. We updated the parameters of the ATNPs with observations (X, y) and predicted the probabilistic occupancy of target locations X^* within sensor range. The predicted probabilistic occupancy local maps were integrated into the global map, as shown in Figure 8(a). The estimated occupancy grid local map could be used on the fly for path planning in real time. However, in this experiment, we assumed the simulator provided feasible paths. We left the task of finding feasible paths as future works. The overall continuous mapping process is shown in Algorithm 1.

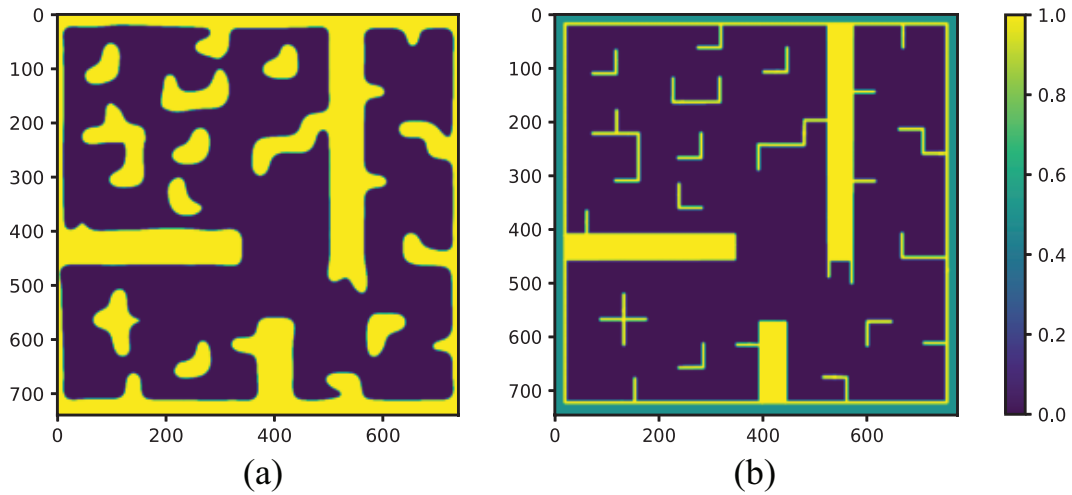


Figure 8: Qualitative results of probabilistic occupancy mapping. (a) Predictive mean of occupancy map after the convergence of the ATNPs. (b) Ground truth map.