

404 A Formulation of S-MON

405 We now present the optimization formulation of S-MON. Unlike U-MON, S-MON does not require
 406 balancing between multiple objectives. Therefore, in this setting, we additionally aim to reach the
 407 goal via the fastest possible path, rather than just prioritizing information (belief) gain.

$$\begin{aligned}
 \text{[S-MON]} \min_{p_t} \max_{M^t \in \Omega} & - \sum_{t \in T} \gamma^t \sum_{i \in n} [(\beta_i^t)_0 + \Delta p_t] \\
 \text{s.t.} \quad & \|p_{t+1} - p_t\|_2 \leq \bar{d}, \quad \forall t \in \{0, \dots, \tau - 1\} \\
 & p_0 = [E/2 \quad E/2]^T \\
 & 0 \leq (p_t)_x, (p_t)_y \leq E, \quad \forall t \in T \setminus \{0\} \\
 & \alpha_i^t, \beta_i^t \in [0, 1], \quad \forall t \in T \\
 & (\alpha_i^t)_v = \sum_{u \in V} M_{uv}^i (\beta_i^{t-1})_u, \quad \forall t \in T \setminus \{0\}, v \in V \\
 & (\beta_i^t)_0 = 1 - \sum_{v \in V} (\beta_i^t)_v, \quad \forall t \in T \setminus \{0\}, v \in V \\
 & (\beta_i^0)_v = c = \begin{cases} 0, & \text{if } v = 0 \text{ or } v = \lfloor \frac{V}{2} \rfloor + 1 \\ \frac{1}{V-1}, & \text{otherwise} \end{cases} \\
 & (\beta_i^t)_v = (\alpha_i^t)_v d(p)_v, \quad \forall t \in T \setminus \{0\}, v \in V \\
 & d(p)_v = \frac{|v_x - (p_t)_x| + |v_y - (p_t)_y|}{V-1}, \quad \forall t \in T \\
 & v_x = (v \bmod E) + 0.5, \quad v_y = \lfloor v/E \rfloor + 0.5 \\
 & \Delta p_t = \|p_{t+1} - p_t\|_2, \quad \forall t \in \{0, \dots, \tau - 1\}
 \end{aligned}$$

408 B Architecture of PICNNs

409 To model the uncertainty set in a flexible and general manner, we adopt the PICNN as the score
 410 function g . Compared to traditional box or ellipsoidal uncertainty sets, which impose shape priors
 411 and thus arbitrarily restrict the form of the uncertainty, the PICNN can approximate the true uncer-
 412 tainty landscape—often non-convex—by composing multiple convex components, providing a more
 413 expressive and general representation. Using PICNNs allows us to represent complex uncertainty
 414 sets while preserving convexity with respect to a subset of the inputs. For simplicity of notation, we
 415 omit the index i . Specifically, since the PICNN architecture ensures convexity only with respect to its
 416 second input, we define $\Omega(x)$ as the sub-level set of $g(x, M)$ with x fixed and M varying, i.e.,

$$\Omega(x) := \{M \mid g(x, M) \leq p\}.$$

417 for some threshold p . This formulation enables us to capture input-dependent uncertainty sets in a
 418 principled and learnable way.

419 For PICNN [3], the score function g is defined as

$$g(x, M) = W_L \sigma_L + V_L M + b_L,$$

420 where the internal layers are computed recursively as follows:

$$\begin{aligned}
 u_{l+1} &= \text{ReLU}(R_l u_l + r_l), \quad u_0 = x, \\
 \sigma_{l+1} &= \text{ReLU}(W_l \sigma_l + V_l M + b_l), \quad \sigma_0 = 0, \\
 W_l &= \bar{W}_l \text{diag}([\hat{W}_l u_l + \omega_l]_+), \\
 V_l &= \bar{V}_l \text{diag}(\hat{V}_l u_l + v_l), \\
 b_l &= \bar{B}_l u_l + \bar{b}_l,
 \end{aligned}$$

421 for all layers $l = 0, \dots, L - 1$.

422 Therefore, the full set of parameters for the PICNN model is given by:

$$\theta^{\text{PICNN}} = \left\{ R_l, r_l, \bar{W}_l, \hat{W}_l, \omega_l, \bar{V}_l, \hat{V}_l, v_l, \bar{B}_l, \bar{b}_l \mid l = 0, \dots, L \right\}.$$

Note that, in certain cases, the inner maximization problem of the original problems with PICNN-parametrized uncertainty sets may be infeasible or unbounded ones. This problem stems from too small a chosen q (making $\Omega(x)$ empty) or $\Omega(x)$ is not compact. To address this concern, we modify PICNN architecture to ensure its sublevel sets are compact and introduce a slack variable to prevent it from being empty. Such modifications won't alter the general form of our problem.

C Proof of Proposition: Coverage with Quantile

We present a standard proof for the proposition 1 here:

Proposition 1 (coverage with quantile) *Let the dataset $\mathcal{D} = \{(x_n, M_n)\}_{n=1}^N$ to be sampled i.i.d from the implicit distribution \mathcal{P} gained during training phase. And q is set to be the $(1 - \alpha)$ -quantile for the set $\{g(x_n, M_n)\}_{n=1}^N$, then $\Omega(x)$ gains the following guarantee.*

$$1 - \alpha \leq \mathbb{P}_{x, M, \mathcal{D}}(M \in \Omega(x)) \leq 1 - \alpha + \frac{1}{N + 1}$$

proof. Let $\mathcal{X} = (x_i)_{i=1}^N$, $\mathcal{M} = (M_i)_{i=1}^N$, $g_i = g(x_i, M_i)$ for $i = 1, \dots, N$, and $\mathcal{G} = g(\mathcal{X}, \mathcal{M})$. To avoid handling ties, assume the g_i are distinct with probability 1. We begin by proving the lower bound of the inequality.

Without loss of generality, sort the calibration scores such that $g_1 \leq \dots \leq g_N$. In this case, we have that $\hat{q} = g_{\lceil (n+1)(1-\alpha) \rceil}$ when $\alpha \geq \frac{1}{n+1}$ and $\hat{q} = \infty$ otherwise, where $\lceil \cdot \rceil$ denotes the ceiling operation. Assume $\alpha \geq \frac{1}{n+1}$. Observe the equivalence of the following two events:

$$\{\mathcal{M} \in \Omega(\mathcal{X})\} = \{\mathcal{G} \leq \hat{q}\}$$

Using the definition of \hat{q} , we have:

$$\{\mathcal{M} \in \Omega(\mathcal{X})\} = \{\mathcal{G} \leq g_{\lceil (n+1)(1-\alpha) \rceil}\}$$

Then, due to the exchangeability of variables in (x_i, M_i) for $i = 1, \dots, N$, the probability of \mathcal{G} falling below a specific g_k is:

$$P(\mathcal{G} \leq g_k) = \frac{k}{N + 1}$$

In other words, \mathcal{G} is equally likely to fall in anywhere between the calibration points g_1, \dots, g_N . Note that above, the randomness is over all variables g_1, \dots, g_N . From here, we next conclude:

$$P(\mathcal{G} \leq g_{\lceil (n+1)(1-\alpha) \rceil}) = \frac{\lceil (n+1)(1-\alpha) \rceil}{N + 1} \geq 1 - \alpha$$

Thus, the lower bound is proven.

For the upper bound, we assume the conformal score distribution is continuous to avoid ties. The proof follows a similar process to the one of the lower bound. \square

D Derivation of the Dual Problems

We begin by defining the following notations: $\mathbf{0}_V$ denotes a zero vector of length V . $\mathbf{I}_{V \times V}$ and $\mathbf{0}_{V \times V}$ represents the $V \times V$ identity matrix and all-zero matrix, respectively.

The decision variable in the inner maximization problem is M_i , while p_t is treated as a constant. We first rewrite the PICNN calibration constraint here for each i and v .

$$g_i(x^i, M^i) \leq q, \forall i \in n \quad (8)$$

This can be equivalently reformulated as follows. Similarly, the indexes i and v are omitted for simplicity.

$$\begin{aligned} \sigma_l &\geq \mathbf{0}_d, \forall l = 1, \dots, L \\ \sigma_{l+1} &\geq W_l \sigma_l + V_l M + b_l, \forall l = 0, \dots, L - 1 \\ W_L \sigma_L + V_L y + b_L &\leq q \end{aligned} \quad (9)$$

454 To see why this holds, observe Eq. (9) is a relaxed form of Eq. (8), derived by replacing the
 455 equality constraints $\sigma_{l+1} = \text{ReLU}(W_l \sigma_l + V_l M + b_l)$ in the definition of the PICNN with two
 456 separate inequalities $\sigma_{l+1} \geq \mathbf{0}_d$ and $\sigma_{l+1} \geq W_l \sigma_l + V_l M + b_l$, for each $l = 0, \dots, L-1$. Conse-
 457 quently, the optimal value of the relaxed problem cannot be less than that of the one of the original
 458 problem. The relaxed constraint Eq. (9) can be expressed in the following matrix form, where
 459 $A \in \mathbb{R}^{(2Ld+1) \times (V+Ld)}$, $b \in \mathbb{R}^{2Ld+1}$.

$$A [M \quad \sigma_1 \quad \dots \quad \sigma_L]^T \leq b$$

$$A = \begin{bmatrix} & -I_d & & & \\ & & \ddots & & \\ & & & -I_d & \\ V_0 & -I_d & & & \\ \vdots & & W_1 & \ddots & \\ \vdots & & & \ddots & -I_d \\ V_L & & & & W_L \end{bmatrix}, \quad b = \begin{bmatrix} \mathbf{0}_d \\ \vdots \\ \mathbf{0}_d \\ -b_0 \\ \vdots \\ -b_{L-1} \\ q - b_L \end{bmatrix} \quad (10)$$

460 We first use the [U-MON] problem as an example and re-index M with i, v to ensure the completeness
 461 of the optimization problem. Additionally, we decompose the matrix M_i into vectors M_i^v for
 462 $v = 1, \dots, V$. We denote the vector $[M_i^v (\sigma_1)_i^v \dots (\sigma_L)_i^v]^T \in \mathbb{R}^{V+Ld}$ as k_i^v .
 463 To this end, the uncertainty sets of the two-stage robust optimization problem are transformed into
 464 linear constraints. We reformulate the optimization problem as follows:

$$\begin{aligned} & [\text{U-MON}] \min_{p_t} \max_{k_i^v} - \sum_{t \in T} \gamma^t \sum_{i \in n} (\beta_i^t)_0 \\ & s.t. \quad (1) \\ & \quad \alpha_i^t, \beta_i^t \in [0, 1], \forall i \in n, t \in T \\ & \quad T_v \alpha_i^t = (H k_i^v)^T S \beta_i^{t-1}, \forall t \in T \setminus \{0\}, i, v \\ & \quad E \beta_i^t = 1, \forall i \in n, t \in T \\ & \quad \beta_i^0 = c, \forall i \in n \\ & \quad S \beta_i^t = D^t \alpha_i^t, \forall i \in n, t \in T \\ & \quad A_i^v k_i^v \leq b_i^v, \forall i \in n, v \in V \end{aligned} \quad (11)$$

465 where

$$\begin{aligned} S &= [\mathbf{0}_V \quad \mathbf{I}_{V \times V}] \in \mathbb{R}^{V \times (V+1)} \\ E &= [1 \quad 1 \quad \dots \quad 1] \in \mathbb{R}^{1 \times (V+1)} \\ H &= [\mathbf{I}_{V \times V} \quad \mathbf{0}_{V \times Ld}] \in \mathbb{R}^{V \times (V+Ld)} \\ D^t &= \begin{bmatrix} d(p_t)_1 & & \\ & \ddots & \\ & & d(p_t)_V \end{bmatrix} \in \mathbb{R}^{V \times V} \\ (T_v)_i &= \begin{cases} 1, & \text{if } i = v \\ 0, & \text{otherwise} \end{cases}, T_v \in \mathbb{R}^{1 \times V} \end{aligned}$$

466 are constant matrices for the inner maximization problem.

467 Assuming that the problem has optimal solutions and satisfies the Slater condition, we can invoke
 468 strong duality. Let $\pi_{i,v}^t, \mu_i, \lambda_i^t, \nu_i^t$, and ξ_i^v represent the dual variables. The inner maximization prob-
 469 lem can then be reformulated as an equivalent minimization problem using the KKT conditions and
 470 Lagrangian duality. Combining this reformulated minimization problem with the outer minimization,
 471 we derive the formulation in the main paper, where $C \in \mathbb{R}^{(V+Ld) \times (V+1)}$ and $\Gamma^t \in \mathbb{R}^{1 \times (V+1)}$ represent
 472 $H^T S$ and $[\gamma^t \quad 0 \quad \dots \quad 0]_{1 \times V} S$, respectively.

473 Similarly, the dual form [S-MON*] of the [S-MON] problem can be derived as follows.

$$\begin{aligned}
\text{[S-MON*]} \quad & \min_{p, \pi, \mu, \lambda, \nu, \xi, k} \sum_{i, t, v} -c^T \mu_i - \lambda_i^t - (b_i^v)^T \xi_i^v + \Delta p_t \\
s.t. \quad & (1) - (3) \\
& \Delta p_t = \|p_{t+1} - p_t\|_2, \forall t \in T \setminus \{0\} \\
& D_t^T \nu_i^0 \leq 0, \forall i \in n \\
& D_t^T \nu_i^t - \sum_{v \in V} \pi_{i,v}^t T_v^T \leq 0, \forall i \in n, t \in T \setminus \{0\} \\
& \sum_{t \in T} \pi_{i,v}^t C \beta_i^t - (A_i^v)^T \xi_i^v \leq 0, \xi_i^v \geq 0, \forall i, v \\
& (\Gamma^t)^T + (\nu_i^t)^T S + \lambda_i^t E + G_{i,t} \geq 0, \forall i \in n, t \in T \\
& G_{i,t} = \begin{cases} \mu_i^T - \sum_{v \in V} \pi_{i,v}^0 (k_i^v)^T C, & \text{if } t = 0 \\ 0, & \text{if } t = \tau \\ - \sum_{v \in V} \pi_{i,v}^t (k_i^v)^T C, & \text{otherwise} \end{cases}
\end{aligned}$$

474 E Convergence and Optimality of NEURO

475 Convergence and Gradient Consistency

476 Network components, parameterized by θ , are trained using a composite reward signal derived from
477 both optimization-specific rewards (r_t^{task}) and non-optimization rewards (r_t^{env}). Convergence to a
478 desirable θ requires consistent gradients from these disparate sources.

479 We establish gradient consistency as follows. The gradient $\nabla_{\theta} r_t^{\text{task}}(\theta)$, originating from the convex
480 Robust Optimization (RO) problem, inherently directs parameter updates towards configurations of
481 θ that enhance the RO problem’s objective value, thus steering θ towards an optimal parameter set
482 θ^* . Similarly, the gradient $\nabla_{\theta} r_t^{\text{env}}(\theta)$ from non-optimization objectives (e.g., generation quality) is
483 engineered to guide θ towards the same θ^* . The well-behaved nature of $\nabla_{\theta} r_t^{\text{env}}(\theta)$ is maintained
484 through techniques including gradient clipping and normalization, ensuring its boundedness and local
485 consistency.

486 Since both r_t^{task} and r_t^{env} are designed to drive θ towards a common (near) globally optimal θ^* , their
487 respective gradients, $\nabla_{\theta} r_t^{\text{task}}(\theta)$ and $\nabla_{\theta} r_t^{\text{env}}(\theta)$, are expected to exhibit consistent alignment. This
488 directional alignment implies that updates suggested by each gradient component are collaborative
489 rather than contradictory, which is critical for stable and effective learning using gradient-based meth-
490 ods. For applications involving non-convex or non-linear problem structures, NEURO’s framework
491 can be extended by incorporating PICNNs to maintain output convexity, or by integrating advanced
492 gradient estimators and convex relaxations.

493 Beyond theoretical analysis, we empirically address gradient consistency in this multi-objective
494 reinforcement learning setting by employing the Goal Vector Method (GVM) to combine r_t^{task} and
495 r_t^{env} . GVM formulates each reward component as a dimension in a goal vector, and adaptively
496 projects the multi-dimensional gradient onto a unified descent direction. This projection balances task
497 and environment objectives while ensuring that parameter updates lie within a consensus direction that
498 respects both reward signals. By avoiding naive scalarization, GVM mitigates gradient interference
499 between competing objectives and facilitates more stable convergence during training.

500 Optimality Guarantees

501 NEURO’s convergence to the optimal solution of the *final, learned* RO problem is guaranteed under
502 two readily satisfied assumptions: (i) The RO problem, formulated with the uncertainty set $\mathcal{U}(\theta)$
503 generated by the converged network parameters θ^* , adheres to the structure of a Disciplined Convex
504 Program (DCP). (ii) The objective and constraint functions within this RO problem are differentiable
505 with respect to the decision variables and any parameters influenced by θ .

506 Under these assumptions, and bolstered by the gradient consistency established above (which ensures
507 the convergence of θ to a stable θ^*), the RO problem defined by $\mathcal{U}(\theta^*)$ is convex. Consequently,

standard convex optimization solvers can efficiently identify its global optimum, $x^*(\theta^*)$. NEURO is thus guaranteed to converge to the optimal solution for this specific, data-driven RO formulation.

While the learning process for θ (which defines the uncertainty set $\mathcal{U}(\theta)$) navigates a potentially non-convex landscape—meaning $\mathcal{U}(\theta^*)$ itself may not be the globally "true" uncertainty set in an absolute sense—our framework guarantees optimality for the RO problem constructed with $\mathcal{U}(\theta^*)$. Crucially, due to our calibration guarantees, the learned uncertainty set $\mathcal{U}(\theta^*)$ effectively encapsulates a $(1 - \alpha)$ confidence region for the underlying uncertain parameters. Therefore, the solution $x^*(\theta^*)$ derived from NEURO represents the optimal worst-case performance over this empirically validated, high-confidence uncertainty region.

F Scalability of NEURO

In this section, we discuss the scalability of the NEURO framework, focusing on its runtime and task performance under larger grid configurations. We first examine the performance of the standardNEURO formulation and then introduce a method to enhance its computational efficiency for larger-scale problems.

Baseline Scalability

The inference times reported in the main paper under the configuration ($E = 3, \tau = 4$) are normalized with respect to a baseline setup. For completeness, Table 5 provides the corresponding absolute inference times along with key navigation performance metrics (Success and SPL) as the environment grid size E and planning horizon τ increase. This scalability trend and absolute solving time can be easily verified by constructing optimization problems of similar scale and solving them with python library CVXPY.

Table 5: NEURO performance and inference time with increasing grid size (E) and planning horizon (τ). Inference times are in seconds. (S-MON _{$m=2$})

E	τ	Success(%)	Progress (%)	SPL(%)	Inference Time (s)
3	4	75	79	64	0.008
5	6	80	85	72	0.040
15	8	83	88	78	0.547
20	12	85	92	82	1.084

As shown in Table 5, while the absolute solving time for the optimization model remains manageable for moderately sized problems, it exhibits noticeable growth with E and τ . For typical indoor robot navigation, $E = 20$ might suffice. However, for applications demanding larger fields of view (e.g., autonomous vehicles), the optimization grid at this scale may not adequately represent real-world environments with fidelity. The computational complexity of the navigation task’s optimization component is $\mathcal{O}(E^2\tau)$, indicating a quadratic dependence on the grid dimension E . Given that E often has a more significant impact on navigation performance than τ , in practice, one might fix τ to a smaller constant, resulting in a complexity of $\mathcal{O}(E^2)$. To effectively scale to larger environments, it is crucial to mitigate this quadratic growth associated with iterating over E^2 grid cells.

Improving Scalability via Basis Function Expansion

To address the scalability challenge, we explore the use of basis function expansion and sparse kernel approximation. Specifically, we represent the optimization variables α_i^t and β_i^t (representing aspects of the utility or value functions over the grid) via a predefined or network-learned basis function decomposition $\{\phi_k(x, y)\}_{k=1}^K$:

$$\alpha_i^t = \sum_{k=1}^K a_k^t \phi_k(x, y), \quad \beta_i^t = \sum_{k=1}^K b_k^t \phi_k(x, y). \quad (12)$$

This representation subsequently modifies the original problem constraints. For instance, a constraint involving these variables, such as constraint (2b) from the main paper, can be reformulated based on

the basis functions as follows:

$$a_k^t = \sum_{l=1}^K \left(\iint M^i(u, v) \phi_l(u, v) \phi_k(u, v) dudv \right) b_l^{t-1}. \quad (13)$$

The integral term in Eq. (13), $\iint M^i(u, v) \phi_l(u, v) \phi_k(u, v) dudv$, can be computed before solving with given M^i and ϕ_k . At this point, the optimization grid size is determined by the dimensionality of basis function vector (a_k^t, b_k^t) —predicting high-dimensional vectors is computationally inexpensive for neural networks. This adjustment reduces the time complexity of the underlying optimization module (referred to as U-MON in internal development) from $\mathcal{O}(E^2\tau)$ to $\mathcal{O}(K\tau)$, where $K \ll E^2$ is the number of basis functions. Empirically, this approach achieved a $23.6\times$ speedup (e.g., 0.023s inference time at $\tau = 8$, $E = 15$) in specific configurations, enabling consideration of significantly larger effective E .

Table 6 presents the performance ofNEUROwhen employing this basis function expansion. The models were trained for 250k updates, consistent with the original experiments.

Table 6: NeuRO performance with basis function expansion. (S-MON_{m=2})

E	Success(%)	Progress (%)	SPL(%)	Inference Time (s)
20	82	85	76	0.251
50	83	85	77	0.670
100	85	87	80	1.310

The results in Table 6 demonstrate that the optimization problem’s solving time now scales almost linearly with E (effectively, with K which might grow slowly with E , or E itself if K is chosen proportional to E rather than E^2). However, we observe no substantial improvement in navigation performance for these larger E values. We attribute this to two primary factors: (i) for typical indoor navigation tasks, the limited field of view and task characteristics may mean that smaller optimization grids ($E \approx 20$) are already sufficient to capture the necessary environmental information; (ii) the newly introduced module for predicting basis function coefficients (a_k^t, b_k^t) may require dedicated training strategies, more extensive data, or further architectural refinement to fully realize its potential.

In summary, this exploration offers a promising direction for enhancing the scalability of theNEUROframework. Nevertheless, we consider the primary contribution of this work to be the introduction and validation of the coreNEUROframework. This discussion on scalability is therefore presented as supplementary material, highlighting potential avenues for future research and application to more demanding, large-scale scenarios.

G Broader Application: Power Grid Scheduling as a Case Study

We recommend adopting the NEURO framework in the following scenarios. *First*, data-scarce environments where task performance is critical, as the optimization model can effectively capture general task rules that are intuitive to humans. *Second*, networked systems characterized by multi-stage, multi-objective, multi-agent, and multi-constraint structures, where optimization-based models offer a natural advantage in globally coordinating multiple entities. In this section, we study a representative application in the power market: the capacity expansion problem.

G.1 Problem Setup.

The problem we address is a classic one in the context of power markets: capacity generation. The background of the problem is as follows: To enhance energy utilization efficiency, the power utility has deployed additional battery storage devices at each node in the grid. Our task, as the power utility, is to determine the optimal grid scheduling strategy to minimize overall electricity costs. A key challenge in this process is the uncertainty of electricity purchase prices, which introduces variability into the decision-making process.

However, we propose the following modifications to the original NEURO framework, see Fig. 5: *First*, since long-horizon planning is not required, we utilize a simpler deep learning structure instead

of deep reinforcement learning. *Second*, as scheduling decisions are only meaningful under accurate power prices, we incorporate the loss between the network's predicted values and the ground-truth values into the overall loss function.

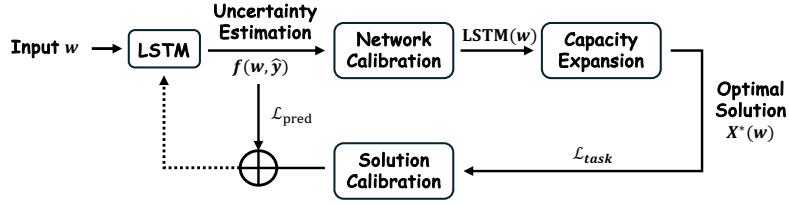


Figure 5: The workflow of solving capacity expansion problem using the NEURO framework.

Network Component

We generated a simple dataset $\mathcal{D} = \{(w_n, y_n)\}_{n=1}^M$, which contains weather data w_n and the corresponding power prices y_n over M time points. The weather data $w_n = (h_n, s_n, p_n)$ include humidity h_n , sunlight intensity s_n , and precipitation intensity p_n . The network module consists of an LSTM network capable of handling non-uniform step-size inputs, followed by a PICNN. To simplify the problem, the predicted uncertainty of power prices is represented as upper and lower bounds (i.e., box uncertainty). During the i -th training step, the network takes historical electricity prices and corresponding weather data over the past K time steps as input, $x_i = \{(w_j, y_j)\}_{j=i-K+1}^i$, and predicts electricity prices $\hat{y}_i = \{(\hat{y}_j^L, \hat{y}_j^H)\}_{j=1}^T$ for the next T time steps. \hat{y}_j^L and \hat{y}_j^H represent the upper and lower bound, respectively. The prediction loss is defined as the mean squared error (MSE) loss between the uncertainty mean and the ground truth.

$$\mathcal{L}_{pred} = \sum_{j=1}^T \text{MSE}\left(\frac{\hat{y}_j^L + \hat{y}_j^H}{2}, y_{i+j}\right) \quad (14)$$

Optimization Model

Our power grid model adopts the 3-phase lindist flow model. In the lin-dist flow power grid model, the grid structure is assumed to be a tree. We consider N nodes, where each node i and its downstream nodes form a subtree T_i . Let P_i represent the set of all paths from node i to any downstream node. As the power utility, the decision variables include the complex power s and voltage matrix v_i at each node, with the voltage represented by a voltage matrix. The objective of the optimization is to minimize the total electricity procurement cost. Given the tree structure, this is equivalent to minimizing the product of the actual power supplied by the slack bus (node 0) and the electricity price. The optimization model can be formulated as follows.

$$\min_{s, v} \max_{\hat{y}_t \in [\hat{y}_t^L, \hat{y}_t^H]} \sum_{t=1}^T \hat{y}_t \text{Re}(s_0^t) \quad (15a)$$

$$s.t. \quad \sum_{j \in N} s_j^t = 0 \quad (15b)$$

$$\lambda_{ij}^t = - \sum_{k \in T_i} s_k^t, S_{ij}^t = \gamma \cdot \text{diag}(\lambda_{ij}^t) \quad (15c)$$

$$v_j^t = v_0^t - \sum_{(j,k) \in P_i} (z_{jk} S_{jk}^{tH} + S_{jk}^t z_{jk}^H) \quad (15d)$$

$$s_j^{\min} \leq s_j^t \leq s_j^{\max} \quad (15e)$$

$$v_j^{\min} \leq \text{diag}(v_j^t) \leq v_j^{\max} \quad (15f)$$

$$\text{Re}(s_j^{b,t}) + \text{Re}(s_j^{d,t}) = \text{Re}(s_j^t) \quad (15g)$$

$$\text{SOC}_j^{t+1} = \text{SOC}_j^t + \text{Re}(s_j^t) \quad (15h)$$

$$\text{SOC}_j^0 = 0, \text{SOC}_j^{\min} \leq \text{SOC}_j^t \leq \text{SOC}_j^{\max} \quad (15i)$$

In the aforementioned model, $\text{Re}(\cdot)$ denotes the real part operator, $(\cdot)^H$ represents the Hermitian matrix operator, and $\text{diag}(\cdot)$ refers to the diagonalization operator. Eq. (15b) to Eq. (15d) define the

power flow constraints in the lindist flow model, where γ is a constant phase matrix, z_{jk} represents the impedance matrix between nodes j and k , and S_{jk}^t and λ_{jk}^t are intermediate variables. c15e and Eq. (15f) specify the operational constraints for each node. Eq. (15g) describes the composition of the power injection s_i at each node, where s_b is the dispatch power value of the battery, and s_d represents the fixed constant power consumption at the node. Finally, Eq. (15h) and Eq. (15i) impose the state-of-charge (SoC) constraints for the battery devices.

Assuming the problem has an optimal solution $z^* = (s^*, v^*)$, the task loss \mathcal{L}_{task} is defined as the optimal value of the objective function corresponding to this solution.

Table 7: Comparison on the capacity expansion task.

#	Learning Framework	Prediction MSE↓	Task Loss↓
1	ETO	275.44	810.52
2	NEURO	287.32	757.81

Experiments

We generated a set of synthetic weather data and corresponding electricity prices for training the network. In the downstream optimization model, the power grid structure is assumed to be a binary tree with seven nodes. The weights of Loss \mathcal{L}_{pred} and Loss \mathcal{L}_{task} in the NEURO framework are set to 0.8 and 0.2, respectively. As a baseline, we adopted the traditional “estimate-then-optimize” (ETO) task-based framework, where the network and the optimization model are decoupled. In the ETO method, the network is first trained to predict electricity prices, and during the testing phase, the network-predicted prices are fed into the optimization model to compute the scheduling strategy.

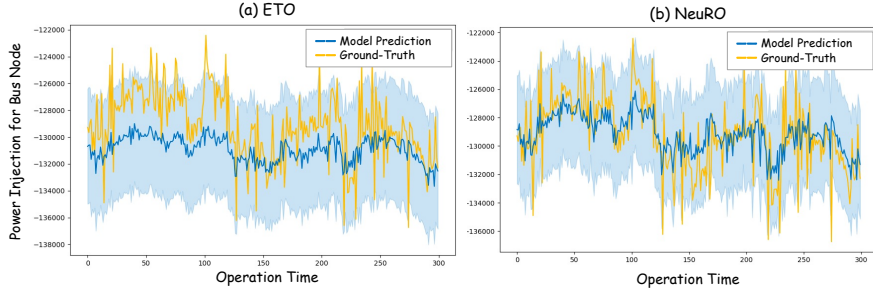


Figure 6: Visualization of prediction error on the capacity expansion problem.

The task loss and prediction loss for NEURO and ETO were recorded and are shown in Table 7, and the visualization of prediction loss can be found in Fig. 6 (the grey region is the visualization of the box uncertainty from the estimation). Table 7 indicates that NEURO’s prediction accuracy is slightly lower than the traditional approach, but its task performance improves by 7%. Therefore, we believe NEURO’s learning ability inherently involves a trade-off. However, in real-world applications, only accurate price predictions hold practical significance, making NEURO more suitable for tasks where precise recognition is not required. For example, in navigation, we care less about whether the robot forms human-like cognition and more about how it interacts with the environment to complete tasks.

H Additional Experimental Results

Agent Performance During Training

We analyzed the training progression of NEURO by comparing its learning curves against several baselines: (i) NEURO without its task-specific optimization component (see main paper Ablation Study Section for more about Task-Specific Optimization), (ii) a previous SoTA method Lyon, and (iii) our foundational baseline, OracleEgoMap. (Fig. 7) Across various navigation tasks, NEURO consistently exhibited faster convergence rates and achieved superior final task performance compared to these alternatives. These training dynamics further substantiate the benefits of integrating explicit optimization within the learning agent, leading to more efficient and effective policy acquisition.

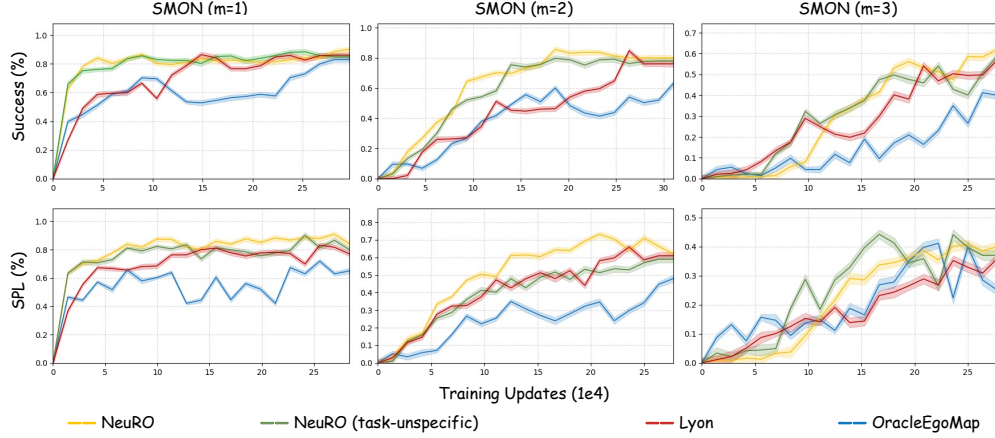


Figure 7: Evaluation metrics of agents during training with two seeds.

Generalization Across Task Variations

To assess the generalization capabilities of our agents, we conducted experiments where models trained on a specific instance of the S-MONTASK, denoted $S\text{-MON}_{m=i}$ (where m represents the number of navigation goals, and i is the specific number of sub-goals used for training), were evaluated on different instances $S\text{MON}_{m=j}$. The performance, typically measured by success rate, is reported in Table 8.

Table 8: Success scores on S-MONTASKS. Agents are trained on task $S\text{-MON}_{m=i}$ (rows) and evaluated on task $S\text{-MON}_{m=j}$ (columns). The table below indicates the relative score drop (%) compared to in-task learning performances (diagonal).

m : train \ eval	OracleEgoMap			Lyon (SoTA)			NEURO		
	1	2	3	1	2	3	1	2	3
1	83	47	28	86	61	50	90	68	57
2	79	64	32	82	76	54	88	80	60
3	77	63	37	81	75	57	88	79	62
1	0	-17	-9	0	-15	-7	0	-12	-5
2	-4	0	-5	-4	0	-3	-2	0	-2
3	-6	-1	0	-5	-1	0	-2	-1	0

The results in Table 8 indicate a common trend: agents trained on tasks with fewer navigation goals (e.g., $m = 1$) tend to struggle when generalizing to tasks requiring a larger number of goals (e.g., $m = 3$). However, the NEURO framework appears to alleviate this degradation. The average drop in performance when transferring agents trained on a specific m to other m values is comparatively lower for NEURO. We attribute this improved generalization to the embedded optimization model, which likely captures underlying task structures and common rules that are transferable across variations in the number of goals. This allows NEURO to adapt more robustly to related but unseen task configurations.

H.1 Impact of Target Object Geometry

We further investigate how the geometry of goal objects affects agent navigation performance in Table 9. We observe that agents find it easier to identify targets located near camera height (e.g., Cylinder) compared to taller objects that span the agent’s full visual field. This suggests that object shape influences navigation success. Notably, NEURO significantly narrows this performance gap, likely due to its ability to encode general task-level information while being less sensitive to low-level

visual cues such as material or shape. As a result, agents trained under the NEURO framework demonstrate improved generalization and adaptability across different task configurations.

Table 9: Navigation performance under different object geometries.

Method	Cylinder		Cube		Sphere	
	Success	SPL	Success	SPL	Success	SPL
OracleEgoMap	64	49	62	47	57	43
Lyon	76	62	74	57	73	57
NEURO	80	66	78	65	78	63

I Experimental Details

This appendix provides specific implementation details and parameter configurations for the key modules within the NEURO framework, complementing the descriptions in the main paper. The network components within the NEURO framework consist primarily of a Neural Perception Module and PICNNs.

For the Neural Perception Module, the architecture and parameters are adopted from the OracleEgoMap (Occ+Obj) configuration as described in the original MultiON work; for the PICNNs, we apply the following parameters:

Table 10: Hyperparameters for the Partially Input-Convex Neural Network (PICNN).

Parameter	Value
Input dimension	32
Convex input dimension	768
Hidden layer dimension	256
Number of layers	3
Output dimension	$= V$
Include y in output layer	True
Feasibility parameter	0.0

The ‘output_dim’ of V is subsequently reshaped into a $E \times E$ grid, followed by a softmax operation along the last dimension, to produce the final output probabilities relevant to the navigation task.

All models were trained for 250,000 updates. Training was conducted on a single NVIDIA Quadro RTX 8000 GPU.