A 2RO PROBLEMS

A.1 ROBUST TWO-STAGE KNAPSACK

We consider the two-stage knapsack problem as defined in Arslan & Detienne (2022) with a set of n items. Each item i has a weight c_i and an uncertain profit $p_i(\boldsymbol{\xi}) = \bar{p}_i - \xi_i \hat{p}_i$, where $\bar{p}_i$ is the expected profit, $\hat{p}_i$ its maximum deviation and ξ_i the *uncertain* profit degradation factor, where the degradation happens after the first stage. In this problem we have a budgeted uncertainty set $\Xi = \{\boldsymbol{\xi} \in [0,1]^n : \sum_{i=1}^n \xi_i \leq \Gamma\}$. The first stage decision is to choose a subset of items to produce. Then in the second stage, there are three different responses to the profit degradation: (i) accept the degraded profit, (ii) repair the item by using an additional t_i units from the budget to recover the original profit $\bar{p}_i$, or (iii) outsource the item for a cost of f_i units, such that the item's profit results in $\bar{p}_i - f_i$. This gives the following problem formulation:

$$\begin{aligned}
\min_{\mathbf{x}\in\{0,1\}^n} \max_{\boldsymbol{\xi}\in\Xi} \min_{\mathbf{y}\in\{0,1\}^n, \mathbf{r}\in\{0,1\}^n} \quad & \sum_{i=1}^n (f_i - \bar{p}_i)x_i + (\hat{p}_i\xi_i - f_i)y_i - \hat{p}_i\xi_i r_i \\
\text{s.t.} \quad & \sum_{i=1}^n c_i y_i + t_i r_i \leq C \\
& r_i \leq y_i \leq x_i \qquad \forall i \in \{1,\dots,n\},
\end{aligned}$$

where x_i is the first-stage decision to produce item i. For the second-stage decisions, we have y_i and r_i: (i) $y_i = 1$ if item i is produced *without* repairing and $y_i = 0$ if the item is outsourced, and (ii) r_i is the decision for repairing item i.

A.2 CAPITAL BUDGETING

Consider the capital budgeting problem in Subramanyam et al. (2020), where a company aims to invest in a subset of n projects. For each project, i, the uncertain cost, and profit are respectively defined as

$$c_i(\boldsymbol{\xi}) = \big(1 + \boldsymbol{\Phi}_i^\intercal \boldsymbol{\xi}/2\big)\bar{c}_i \quad \text{and} \quad r_i(\boldsymbol{\xi}) = \big(1 + \boldsymbol{\Psi}_i^\intercal \boldsymbol{\xi}/2\big)\bar{r}_i, \quad \forall i \in \{1,\dots,n\},$$

where $\bar{c}_i$ and $\bar{r}_i$ are the nominal cost and nominal profit of project i. $\boldsymbol{\Phi}_i^\intercal$ and $\boldsymbol{\Psi}_i^\intercal$ are the i-th row vectors of the sensitivity matrices $\boldsymbol{\Phi}, \boldsymbol{\Psi} \in \mathbb{R}^{n\times 4}$, with $\boldsymbol{\xi} \in \Xi = [-1,1]^4$. We use the problem formulation described in 1.

B 2RO ALGORITHMS

In this section, we describe the column-and-constraint generation algorithm in more detail and the k-adaptability problem, briefly describing one of its solution methods.

B.1 COLUMN-AND-CONSTRAINT GENERATION

The CCG iterates between the *main problem* and the *adversarial problem* (AP). The MP is given as

$$\min_{\mathbf{x}\in\mathcal{X}} \max_{\boldsymbol{\xi}\in\Xi'} \min_{y\in\mathcal{Y}} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x} + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} \tag{6a}$$

$$\text{s.t.} \quad T(\boldsymbol{\xi})\mathbf{x} + W(\boldsymbol{\xi})\mathbf{y} \leq \mathbf{h}(\boldsymbol{\xi}), \tag{6b}$$

where $\Xi' \subset \Xi$ is a finite subset of scenarios. Clearly, the MP provides a lower bound on the optimal value of equation 2. To solve the MP, for each scenario in Ξ' a copy of the second-stage variables is generated. Using a level-set transformation, the problem can be formulated as

$$\min_{\mathbf{x}\in\mathcal{X}} \quad \mu \tag{7a}$$

$$\text{s.t.} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x} + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y}^{\boldsymbol{\xi}} \leq \mu \quad \forall \boldsymbol{\xi} \in \Xi' \tag{7b}$$

$$T(\boldsymbol{\xi})\mathbf{x} + W(\boldsymbol{\xi})\mathbf{y}^{\boldsymbol{\xi}} \leq \mathbf{h}(\boldsymbol{\xi}) \quad \forall \boldsymbol{\xi} \in \Xi' \tag{7c}$$

$$\mu \in \mathbb{R}, \mathbf{y}^{\boldsymbol{\xi}} \in \mathcal{Y} \quad \forall \boldsymbol{\xi} \in \Xi', \tag{7d}$$

which is a linear integer problem that state-of-the-art solvers, such as Gurobi, can solve. In each iteration of the CCG an optimal solution (x^*, μ^*) of equation 7 is calculated. Afterwards, the AP is solved, which is defined as

$$\max_{\boldsymbol{\xi}\in\Xi} \min_{\mathbf{y}\in\mathcal{Y}} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x}^\star + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} \tag{8a}$$

$$\text{s.t.} \quad W(\boldsymbol{\xi})\mathbf{y} \leq \mathbf{h}(\boldsymbol{\xi}) - T(\boldsymbol{\xi})\mathbf{x}^\star. \tag{8b}$$

Since the optimal value of the AP is the objective value of the current solution $\mathbf{x}^*$, it provides an upper bound on the optimal value of equation 2. We define the optimal value to be equal to infinity if there exists a scenario $\boldsymbol{\xi} \in \Xi$ for which no feasible second-stage solution $\mathbf{y}$ exists. If the optimal value of the AP is larger than μ^* then we add the optimal scenario ξ^* to Ξ' and start again from solving MP. Otherwise, we stop the algorithm since the upper bound is smaller or equal to the lower bound, and hence $\mathbf{x}^*$ is an optimal solution. The whole procedure is presented in Algorithm 1.

Algorithm 1 Column-and-Constraint Generation

set $ub = \infty$, $lb = -\infty$
$\Xi' = \{\xi_0\}$ for any $\xi_0 \in \Xi$
while $ub - lb > 0$ **do**
  Calculate an optimal solution x^*, μ^* of the main problem equation 7 and set $lb = \mu^*$.
  Calculate an optimal solution ξ^* (with optimal value opt^*) of the adversarial problem equation 8 where $x = x^*$.
  Set $\Xi' = \Xi' \cup \{\xi^*\}$ and $ub = \min\{ub, opt^*\}$.
end while
return x^*

CCG often fails to calculate an optimal solution in a reasonable time since both the MP and the AP are very hard to solve in the case of integer second-stage variables. In each iteration, the size of MP increases since we have to add new constraints and a copy of all integer second-stage decisions $\mathbf{y}$. This often leads to the situation that after even a small number of iterations, the MP cannot be solved to optimality anymore by classical integer optimization solvers as Gurobi.

Furthermore, solving the AP is extremely challenging for integer second-stage variables. Indeed, the problem can be formulated as a bilevel problem where the follower problem contains integer variables. In Zhao & Zeng (2012) the authors present a column-and-constraint algorithm that solves the AP if the second-stage is a mixed-integer problem. One drawback is that this method is not applicable if the second-stage does not contain continuous variables, as is the case for many problems, e.g., the capital budgeting problem. Furthermore, the method involves solving a very large mixed-integer bilinear problem, which is computationally enormously challenging. The whole procedure must be executed in each iteration of the main CCG algorithm.

B.2 k-ADAPTABILITY

The k-adaptability approach was introduced in Bertsimas & Caramanis (2010) and later studied for objective uncertainty and constraint uncertainty in Hanasusanto et al. (2015); Subramanyam et al. (2020); Ghahtarani et al. (2023); Julien et al. (2022); Kurtz (2023). The main idea of the approach is to calculate a set of k second-stage solutions already in the first-stage. Instead of choosing the best feasible second-stage solution for each scenario $\boldsymbol{\xi}$, we choose the best of the k calculated second-stage solutions. Since we restrict the number of second-stage reactions, this approach leads to feasible solutions of equation 2, which are not necessarily optimal. While for larger k the approximation guarantee gets provably better, the problem gets harder to solve at the same time. Furthermore, it was shown in Subramanyam et al. (2020) that it may happen that k has to be chosen exponentially large to guarantee optimality for equation 2. The k-adaptability problem can be formulated as

$$\min_{\mathbf{x}\in\mathcal{X}, \mathbf{y}^1,\dots,\mathbf{y}^k\in\mathcal{Y}} \max_{\boldsymbol{\xi}\in\Xi} \min_{y\in\{y^1,\dots,y^k\}} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x} + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} \tag{9a}$$

$$\text{s.t.} \quad W(\boldsymbol{\xi})\mathbf{y} + T(\boldsymbol{\xi})\mathbf{x} \leq \mathbf{h}(\boldsymbol{\xi}). \tag{9b}$$

The k-adaptability problem is very challenging to solve, especially in the constraint uncertainty case. The best-known method for this case was introduced in Subramanyam et al. (2020). The authors perform a branch-and-bound algorithm over partitions of the uncertainty set. They consider k-partitions of finite scenarios sets, which are iteratively generated, and assign each of the second-stage solutions to one of the partitions. This approach was later improved by applying machine learning methods to improve the branching decisions Julien et al. (2022). As an alternative approach in Postek & Hertog (2016), an iterative uncertainty set splitting method is presented, which converges to the exact optimal value of the two-stage robust problem.

In case of objective uncertainty, the k-adaptablity problem is easier (but still hard) to solve (Arslan et al. (2022); Ghahtarani et al. (2023)) and can be approximated if k is not too small; see Kurtz (2023).

C Detailed Formulation

This section presents the detailed $\arg\max$ formulation for equation 4. We assume that at this iteration in the MP, we have scenarios $\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_k$ and that M and L are upper and lower bounds on the prediction of the network. The complete formulation is then given by

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y},\boldsymbol{\xi}_a\in\Xi,\mathbf{p},u,\mathbf{z}\in\{0,1\}^k} \quad \mathbf{c}(\boldsymbol{\xi}_a)^\intercal\mathbf{x} + \mathbf{d}(\boldsymbol{\xi}_a)^\intercal\mathbf{y} \tag{10a}$$

$$\text{s.t.} \quad W(\boldsymbol{\xi}_a)\mathbf{y} + T(\boldsymbol{\xi}_a)\mathbf{x} \leq \mathbf{h}(\boldsymbol{\xi}_a), \tag{10b}$$

$$p_i = NN_\Theta(\mathbf{x}, \boldsymbol{\xi}_i) \qquad \forall i \in \{1, \ldots, k\} \tag{10c}$$

$$u \geq p_i \qquad \forall i \in \{1, \ldots, k\} \tag{10d}$$

$$u \leq p_i + (M - L)(1 - z_i) \qquad \forall i \in \{1, \ldots, k\} \tag{10e}$$

$$\sum_{i=1}^{k} z_i = 1 \qquad \forall i \in \{1, \ldots, k\} \tag{10f}$$

$$\boldsymbol{\xi}_a = \sum_{i=1}^{k} z_i \cdot \boldsymbol{\xi}_i \tag{10g}$$

To model the $\arg\max$, we introduce k binary variables $\mathbf{z}$ and $k+1$ continuous variables $\mathbf{p}$ and u, which are used to model big-M that ensure $\mathbf{z}$ is 1 at the index of the maximizer and 0 everywhere else. $\boldsymbol{\xi}_a$ is then given by a linear combination of the scenarios multiplied with $\mathbf{z}$.

D Extended NN Architecture

We show the extended neural network architecture used in the experiments in Figure 3.

E 2RO with fixed first-stage decision

When we compare the calculated solutions of `Neur2RO` and the baseline in our experiments, we need to calculate the objective value of a solution $\mathbf{x}^\star \in \mathcal{X}$ exactly or approximately. The former involves solving the AP equation 8 for a given solution. Solving this problem is intractable when we have uncertain parameters in the constraints. We first expand on how the adversarial would be solved in a tractable way if the uncertain parameters only appear in the objective function. Subsequently, we describe an approach to approximately solve the AP, which is based on sampling scenarios from Ξ.

E.1 Objective uncertainty

For the special case of objective uncertainty, the AP can be solved much more efficiently. In this case, the adversarial problem is given as

$$\max_{\boldsymbol{\xi}\in\Xi} \min_{\mathbf{y}\in\mathcal{Y}} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal\mathbf{x}^\star + \mathbf{d}(\boldsymbol{\xi})^\intercal\mathbf{y} \tag{11a}$$

$$\text{s.t.} \quad W\mathbf{y} \leq \mathbf{h} - T\mathbf{x}^\star, \tag{11b}$$

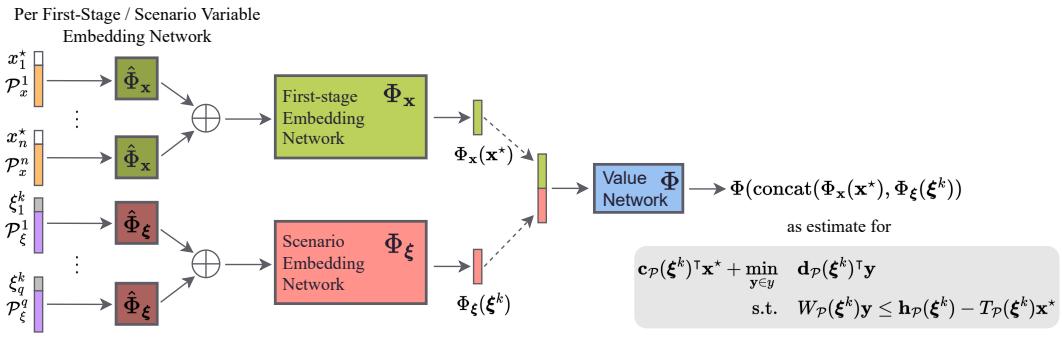

Figure 3: The extended neural network architecture for ML-based CCG. Compared to the NN architecture shown in the main text (Figure 2), this model uses the set-based method to be able to generalize across instance sizes. Let $\mathbf{x}^\star \in \mathbb{R}^n$ and $\boldsymbol{\xi} \in \mathbb{R}^q$. Then, $\hat{\Phi}_\mathbf{x}$ and $\hat{\Phi}_{\boldsymbol{\xi}}$ are the embedding networks for $x_i, i \in [n]$ and $\xi_j, j \in [q]$, respectively. The features are comprised of the single variable and single-variable specific problem specifications $\mathcal{P}^i_\mathbf{x}, i \in [n]$ and $\mathcal{P}^j_{\boldsymbol{\xi}}, j \in [q]$ for first-stage decisions and scenarios, respectively. The outputs of the $\hat{\Phi}$ networks are aggregated for $\mathbf{x}^*$ and $\boldsymbol{\xi}$ separately. These embeddings are the input of the original NN given in the main part.

which can be reformulated as

$$\max_{\boldsymbol{\xi} \in \Xi} \quad \alpha \tag{12a}$$

$$\text{s.t.} \quad \alpha \leq \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x}^\star + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} \quad \forall \mathbf{y} \in \bar{\mathcal{Y}}, \tag{12b}$$

where $\bar{\mathcal{Y}} = \{\mathbf{y} \in \mathcal{Y} : \quad W\mathbf{y} \leq \mathbf{h} - T\mathbf{x}^\star\}$. While the set $\bar{\mathcal{Y}}$ can contain an exponential number of solutions, the latter problem can be solved by iteratively generating the constraints for $y \in \bar{\mathcal{Y}}$.

E.2 Constraint uncertainty

We collect all scenarios $\xi \in \Xi$ which were generated during training and during the solution procedures of the baseline algorithm and our algorithm (including the scenarios calculated by the AP) in the set $\Xi^{samples}$. Then for the two returned solutions $\mathbf{x}^*$ and $\mathbf{x}^{baseline}$ we compare

$$\max_{\boldsymbol{\xi} \in \Xi^{samples}} \min_{\mathbf{y} \in \mathcal{Y}} \quad \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x}^\star + \mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} \tag{13a}$$

$$\text{s.t.} \quad W(\boldsymbol{\xi})\mathbf{y} \leq \mathbf{h}(\boldsymbol{\xi}) - T(\boldsymbol{\xi})\mathbf{x}^\star, \tag{13b}$$

where we replace $\mathbf{x}$ by the corresponding solution $\mathbf{x}^*$ or $\mathbf{x}^{baseline}$. The latter problem can be solved by calculating the optimal value of the second-stage problem for each scenario independently and choosing the worst-case overall optimal values.

F Convergence

In the following, we present the proof of Theorem 1.

Proof. The main idea is to show that the condition equation 5 cannot hold in infinitely many iterations. Since we stop the algorithm if 5 is not true anymore, then finite termination of the algorithm follows.

Assume the algorithm does not terminate in a finite number of iterations. Let l_t and r_t be the values of the left-hand side and right-hand side of inequality 5 in iteration t of the algorithm, i.e.,

$$l_t := \max_{\boldsymbol{\xi} \in \Xi} NN_\Theta(\mathbf{x}^t, \boldsymbol{\xi})$$

and

$$r_t := \max_{\boldsymbol{\xi} \in \Xi^t} NN_\Theta(\mathbf{x}^t, \boldsymbol{\xi}).$$

where $\mathbf{x}^t$ is the optimal solution of MP in the t-th iteration and Ξ^t the finite set of scenarios used in the MP in iteration t. Let $\mathbf{x} \in \mathcal{X}$ be a feasible first-stage solution and let $l_t(\mathbf{x})$ and $r_t(\mathbf{x})$ be the sub-sequences which contain the values of l_t and r_t only for the iterations where x is an optimal solution of the MP. Then either this sequence is finite or, if it is infinite, the sequence $\{r_t(\mathbf{x})\}_t$ is monotonous and bounded where monotony follows since $\Xi^t \subset \Xi^{t+1}$ and since the same x is used. The sequence is bounded since Ξ is a bounded set and NN_Θ a piecewise-linear function (as is known for feedforward ReLU networks (Montufar et al., 2014)) and the maximum of a piecewise linear function over a bounded set is bounded. Hence, $\{r_t(x)\}_t$ converges to a finite value $r^\star(\mathbf{x})$. Furthermore, it holds $l_t(\mathbf{x}) \leq r_{t+1}(\mathbf{x})$ since the optimal scenario of the left-hand-side is added to Ξ^t which is a subset of the set later used to evaluate $r_{t+1}(\mathbf{x})$. It follows that

$$r_t(\mathbf{x}) \leq l_t(\mathbf{x}) - \varepsilon \leq r_{t+1}(\mathbf{x}) - \varepsilon$$

for all t which contradicts the convergence of $r_t(\mathbf{x})$. Hence the sequence $r_t(\mathbf{x})$ must be finite. Since only finitely many first-stage solutions $\mathbf{x}$ exist, and the latter result holds for all of them, the number of iterations of the algorithm must be finite. □

G Distributional Results for Relative Performance

In this section, we provide distributional information for the RE for knapsack in Tables 3-4 and Figures 4-8.

Correlation Type	# items	Mean RE		Median RE		RE 1st Quartile		RE 3rd Quartile	
		Neur2RO	BP	Neur2RO	BP	Neur2RO	BP	Neur2RO	BP
Uncorrelated	20	2.005	**0.000**	1.417	**0.000**	0.541	**0.000**	2.379	**0.000**
	30	1.189	**0.000**	1.188	**0.000**	0.712	**0.000**	1.399	**0.000**
	40	2.895	**0.000**	1.614	**0.000**	1.221	**0.000**	4.042	**0.000**
	50	3.032	**0.000**	1.814	**0.000**	0.946	**0.000**	3.801	**0.000**
	60	2.099	**0.000**	1.146	**0.000**	0.577	**0.000**	2.872	**0.000**
	70	2.214	**0.000**	1.408	**0.000**	0.761	**0.000**	2.506	**0.000**
	80	1.591	**0.000**	0.968	**0.000**	0.758	**0.000**	2.063	**0.000**
Weakly Correlated	20	2.569	**0.000**	1.582	**0.000**	1.229	**0.000**	4.010	**0.000**
	30	2.664	**0.000**	2.236	**0.000**	0.616	**0.000**	4.293	**0.000**
	40	2.320	**0.000**	1.595	**0.000**	1.164	**0.000**	2.292	**0.000**
	50	2.183	**0.145**	1.757	**0.000**	0.793	**0.000**	2.674	**0.000**
	60	2.165	**0.390**	0.695	**0.000**	**0.000**	**0.000**	3.445	**0.458**
	70	0.884	**0.338**	0.165	**0.000**	**0.000**	**0.000**	0.623	**0.175**
	80	**0.392**	0.691	**0.000**	0.341	**0.000**	**0.000**	**0.165**	0.831
Almost Strongly Correlated	20	2.355	**0.000**	1.439	**0.000**	0.000	**0.000**	2.757	**0.000**
	30	1.166	**0.113**	0.782	**0.000**	0.075	**0.000**	1.911	**0.000**
	40	0.825	**0.335**	0.497	**0.000**	0.019	**0.000**	1.606	**0.000**
	50	**0.314**	0.884	0.019	**0.000**	**0.000**	**0.000**	**0.229**	1.251
	60	**0.197**	0.523	**0.000**	0.016	**0.000**	**0.000**	**0.268**	1.129
	70	**0.551**	0.615	**0.017**	0.031	**0.000**	**0.000**	**1.058**	1.227
	80	**0.388**	0.694	**0.000**	0.265	**0.000**	**0.000**	**0.554**	0.770
Strongly Correlated	20	2.387	**0.000**	1.604	**0.000**	0.905	**0.000**	3.018	**0.000**
	30	1.068	**0.121**	0.610	**0.000**	0.054	**0.000**	1.939	**0.000**
	40	0.658	**0.191**	0.443	**0.000**	0.002	**0.000**	0.888	**0.000**
	50	**0.411**	0.648	0.073	**0.000**	**0.000**	**0.000**	**0.780**	0.963
	60	**0.322**	0.367	0.042	**0.010**	**0.000**	**0.000**	**0.173**	0.693
	70	**0.389**	0.738	**0.020**	0.027	**0.000**	**0.000**	**0.535**	0.793
	80	**0.318**	0.668	**0.000**	0.179	**0.000**	**0.000**	**0.245**	0.906

Table 3: Table of distributional information for knapsack. For each row, all RE statistics are computed over 18 instances.

# items	Mean RE				Median RE				RE 1st Quartile				RE 3rd Quartile			
	Neur2RO	$k=2$	$k=5$	$k=10$	Neur2RO	$k=2$	$k=5$	$k=10$	Neur2RO	$k=2$	$k=5$	$k=10$	Neur2RO	$k=2$	$k=5$	$k=10$
10	2.558	2.849	**1.029**	1.165	1.105	1.140	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	3.534	4.349	**0.547**	1.557
20	0.423	0.304	**0.232**	0.266	**0.000**	0.196	0.112	0.064	**0.000**	0.094	0.013	0.000	0.410	0.453	**0.320**	0.362
30	0.408	0.149	0.131	**0.084**	0.109	**0.020**	0.073	0.032	0.002	**0.000**	0.003	0.000	0.337	0.182	0.212	**0.110**
40	0.234	0.114	0.098	**0.073**	**0.009**	0.074	0.011	0.019	**0.000**	0.001	**0.000**	0.002	**0.121**	0.180	0.137	0.137
50	0.090	0.107	0.090	**0.056**	**0.001**	0.033	0.039	0.020	**0.000**	0.000	**0.000**	0.002	**0.050**	0.193	0.139	0.084

Table 4: Table of distributional information for capital budgeting. For each row, all RE statistics are computed over 50 instances.

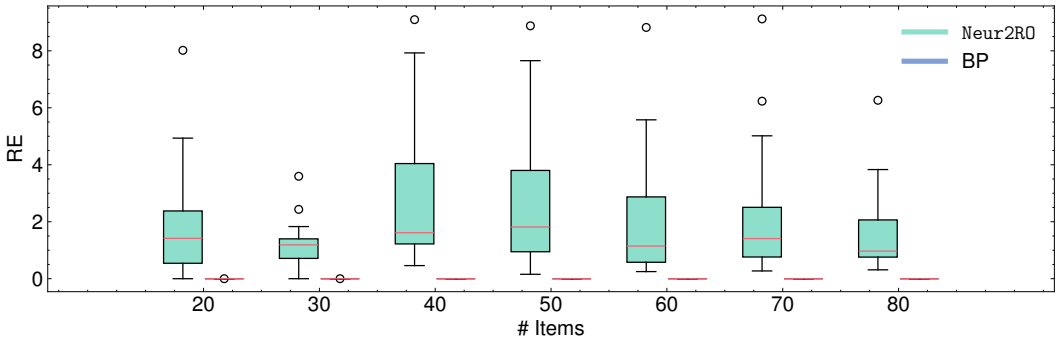

Figure 4: Boxplot of RE for baseline and `Neur2RO` on UN knapsack instances.

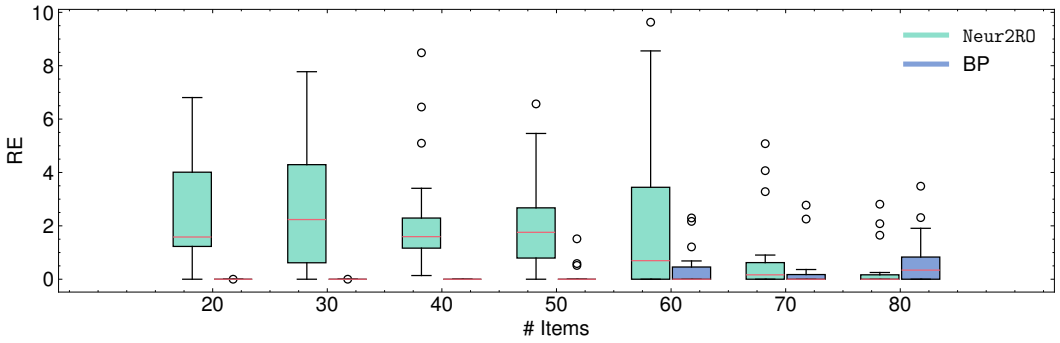

Figure 5: Boxplot of RE for baseline and `Neur2RO` on WC knapsack instances.

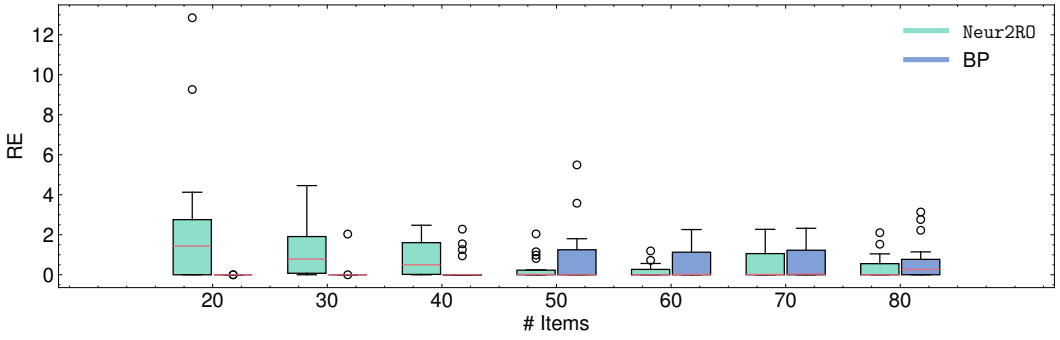

Figure 6: Boxplot of RE for baseline and `Neur2RO` on ASC knapsack instances.

H Ablation

This section presents an ablation across two aspects of `Neur2RO`, namely, the formulation of the MP and the method to obtain worst-case scenarios. Both results are presented on the knapsack instances.

H.1 Main Problem Formulation

As an alternative to the formulation using $\arg\max$ over a set of scenarios. One more straightforward formulation is to consider instead the max over all of the scenarios, which is given by

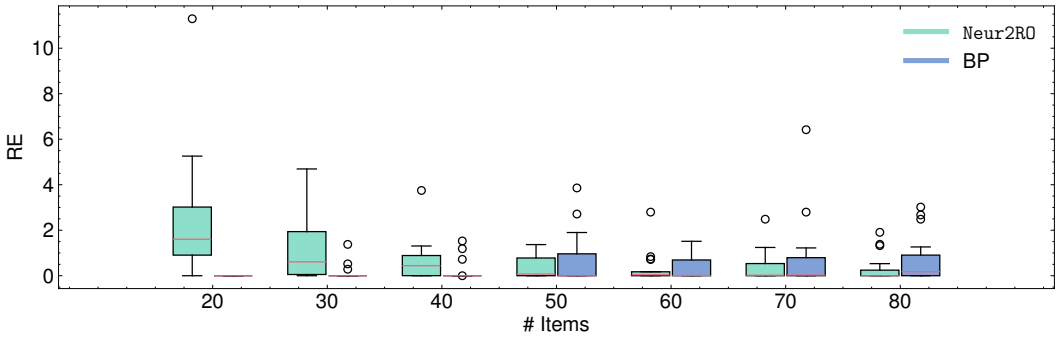

Figure 7: Boxplot of RE for baseline and Neur2RO on SC knapsack instances.

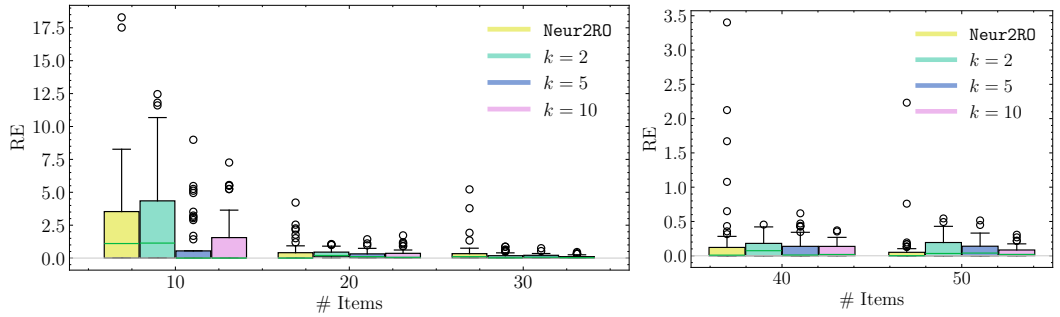

Figure 8: Box plot of RE for baselines and Neur2RO on capital budgeting instances.

$$\min_{\mathbf{x}\in\mathcal{X},\alpha} \quad \alpha \tag{14a}$$

$$\text{s.t.} \quad \alpha \geq NN_\Theta(\mathbf{x}, \boldsymbol{\xi}_i) \qquad \forall k \in \{1, \ldots, K\}. \tag{14b}$$

Table 5 reports the MRE of the $\arg\max$ and $\max$ formulations and the solving time. Table 5 demonstrates an improvement in solution quality, with $\arg\max$ obtaining a lower MRE in every case and a lower computing time in most cases.

H.2 Worst-case Scenario Acquisition

This section compares the adversarial approach for determining scenarios to a sampling and a linear programming (LP) relaxation-based approach.

H.2.1 Sampling-Based Scenario Acquisition

For sampling, as a baseline, we sample $100,000$ scenarios, and then to approximate the AP, we take the maximizer over a forward pass. Table 6 demonstrates a clear trade-off between solution quality and efficiency. Generally, sampling improves average solving time across all instances but leads to worse solution quality as the instance size increases.

H.2.2 LP relaxation-Based Scenario Acquisition

For 2RO, the uncertainty set is often polyhedral, which scenarios can be heuristically obtained via a LP relaxation. For the LP relaxation, we compare the performance of the standard MILP-based scenario acquisition (standard), i.e., solving the AP to optimality, to the relaxation (LP relaxation). For both problems, we report the RE to the baselines. Tables 7 and 8 present the knapsack and

Correlation Type	# items	Median RE		Times	
		arg max	max	arg max	max
Uncorrelated	20	**0.000**	1.167	**5**	11
	30	**0.000**	0.945	**7**	14
	40	**0.000**	1.931	**9**	24
	50	**0.000**	1.634	**10**	33
	60	**0.000**	0.452	**17**	29
	70	**0.000**	0.801	**19**	28
	80	**0.000**	2.227	**13**	35
Weakly Correlated	20	**0.000**	3.515	**6**	13
	30	**0.000**	2.405	**11**	22
	40	**0.000**	0.502	**26**	42
	50	**0.000**	0.254	**24**	39
	60	**0.000**	1.528	77	**58**
	70	**0.000**	1.769	**18**	35
	80	**0.000**	3.492	**27**	75
Almost Strongly Correlated	20	**0.000**	2.042	**5**	12
	30	**0.000**	1.433	**6**	14
	40	**0.000**	1.739	**11**	33
	50	**0.000**	3.161	**8**	20
	60	**0.000**	2.449	**15**	30
	70	**0.000**	2.497	**18**	35
	80	**0.000**	1.824	**17**	30
Strongly Correlated	20	**0.000**	1.154	**5**	11
	30	**0.000**	0.967	**7**	15
	40	**0.000**	1.928	**16**	28
	50	**0.000**	3.613	**10**	21
	60	**0.000**	2.005	**20**	26
	70	**0.000**	2.657	**16**	33
	80	**0.000**	2.051	**16**	28

Table 5: $\arg\max$ and $\max$ formulations on knapsack instances. For each row, the median RE and solving time are computed over 18 instances. All times in seconds.

capital budgeting results, respectively. In general, we can observe that the LP relaxation leads to significantly faster solving time, with an overall decreased solution quality. That being said, for capital budgeting in particular, `Neur2RO` with the LP relaxation still achieves a lower median RE than the baselines on larger instances, while being roughly five times faster than results without the relaxation.

H.3 Prediction Target

This section compares the prediction target. For capital budgeting, the coefficients of the first-stage decisions in the objective contain uncertainty. As such, this presents a choice of either predicting the sum of the first- and second-stage objectives, i.e., $\mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x} + \min_{\mathbf{y}\in\mathcal{Y}} \{\mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} : W(\boldsymbol{\xi})\mathbf{y} \leq \mathbf{h}(\boldsymbol{\xi}) - T(\boldsymbol{\xi})\mathbf{x}\}$, or only the second-stage objective, i.e., $\min_{\mathbf{y}\in\mathcal{Y}} \{\mathbf{d}(\boldsymbol{\xi})^\intercal \mathbf{y} : W(\boldsymbol{\xi})\mathbf{y} \leq \mathbf{h}(\boldsymbol{\xi}) - T(\boldsymbol{\xi})\mathbf{x}\}$. Specifically, we compare the downstream optimization performance with respect to the resulting formulations. The formulation for predicting the sum of the first- and second-stage objectives is presented in Section 3. For predicting the second-stage objective only, the MP is given by

$$\min_{\mathbf{x}\in\mathcal{X},\mathbf{y}\in\mathcal{Y},\boldsymbol{\xi}_a\in\Xi} \quad \mathbf{c}(\boldsymbol{\xi}_a)^\intercal \mathbf{x} + \mathbf{d}(\boldsymbol{\xi}_a)^\intercal \mathbf{y} \tag{15a}$$

$$\text{s.t.} \quad W(\boldsymbol{\xi}_a)\mathbf{y} + T(\boldsymbol{\xi}_a)\mathbf{x} \leq \mathbf{h}(\boldsymbol{\xi}_a), \tag{15b}$$

$$\boldsymbol{\xi}_a \in \arg\max_{\boldsymbol{\xi}\in\Xi'} \{\mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x} + NN_\Theta(\mathbf{x}, \boldsymbol{\xi})\}, \tag{15c}$$

and the AP is given by

$$\max_{\boldsymbol{\xi}\in\Xi} \mathbf{c}(\boldsymbol{\xi})^\intercal \mathbf{x}^\star + NN_\Theta(\mathbf{x}^\star, \boldsymbol{\xi}). \tag{16}$$

The main difference with this formulation is that the objective coefficients $\mathbf{c}(\boldsymbol{\xi})$ can be utilized directly rather than requiring the ML model to predict them. Table 9 compares the two approaches on the capital budgeting instances wherein the RE is computed with respect to the baselines. Empirically, we can see that predicting the sum of the first- and second-stage objectives yields significantly better solutions. On the methodological side, when only the second stage is predicted each node in the branch-and-bound tree being explored by a MIP solver will contain the exact first-stage and the predicted second-stage objectives. As such, we speculate that the LP relaxation at each node will

Correlation Type	# items	Median RE		Times	
		adversarial	sampling	adversarial	sampling
Uncorrelated	20	**0.000**	**0.000**	5	**2**
	30	**0.000**	**0.000**	7	**4**
	40	0.560	**0.000**	9	**4**
	50	0.723	**0.000**	10	**5**
	60	0.066	**0.000**	17	**6**
	70	0.150	**0.000**	19	**8**
	80	0.395	**0.000**	13	**9**
Weakly Correlated	20	**0.000**	0.074	6	**3**
	30	**0.000**	0.444	11	**4**
	40	**0.000**	0.093	26	**5**
	50	0.441	**0.000**	24	**7**
	60	0.119	**0.065**	77	**9**
	70	**0.000**	0.185	18	**8**
	80	**0.000**	0.536	27	**9**
Almost Strongly Correlated	20	**0.000**	**0.000**	**5**	5
	30	**0.000**	**0.000**	**6**	6
	40	**0.000**	**0.000**	11	**10**
	50	**0.000**	**0.000**	8	**7**
	60	**0.000**	**0.000**	15	**14**
	70	**0.000**	**0.000**	18	**13**
	80	**0.000**	**0.000**	17	**12**
Strongly Correlated	20	**0.000**	**0.000**	5	**5**
	30	**0.000**	**0.000**	7	**7**
	40	**0.000**	**0.000**	16	**11**
	50	**0.000**	**0.000**	10	**8**
	60	**0.000**	**0.000**	20	**13**
	70	**0.000**	**0.000**	16	**14**
	80	**0.000**	**0.000**	16	**13**

Table 6: Adversarial and sampling-based approaches for worst-case scenario acquisition on knapsack instances. For each row, the median RE and solving time are computed over 50 instances. All times in seconds.

Correlation Type	# items	Median RE		Times	
		standard	LP relaxation	standard	LP relaxation
Uncorrelated	20	**1.417**	1.673	4	**1**
	30	1.188	**1.167**	6	**1**
	40	1.614	**1.387**	9	**2**
	50	1.814	**1.660**	9	**2**
	60	**1.146**	**1.146**	14	**1**
	70	1.408	**1.166**	16	**2**
	80	0.986	**0.970**	11	**2**
Weakly Correlated	20	1.582	**1.454**	5	**1**
	30	2.236	**2.034**	11	**1**
	40	**1.595**	2.733	20	**2**
	50	1.757	**1.126**	19	**2**
	60	**0.695**	0.729	77	**3**
	70	**0.165**	0.243	15	**3**
	80	**0.000**	0.316	21	**9**

Correlation Type	# items	Median RE		Times	
		standard	LP relaxation	standard	LP relaxation
Almost Strongly Correlated	20	1.439	**1.211**	5	**1**
	30	0.782	**0.665**	6	**1**
	40	**0.497**	0.927	10	**2**
	50	**0.019**	1.884	7	**2**
	60	**0.000**	1.079	14	**2**
	70	**0.017**	0.025	13	**4**
	80	**0.000**	1.775	12	**4**
Strongly Correlated	20	1.604	**1.368**	5	**1**
	30	**0.610**	0.796	7	**2**
	40	**0.443**	1.375	11	**3**
	50	**0.073**	2.333	9	**2**
	60	**0.042**	0.510	11	**4**
	70	**0.020**	0.623	16	**3**
	80	**0.000**	1.097	13	**3**

Table 7: Median RE and solving times for knapsack instances with LP relaxation. For each row, the median RE and average solving time are computed over 18 instances. All times in seconds. The smallest (best) values in each row/metric are in bold.

consist of two components that are on entirely different scales. Specifically, the first-stage objective will be tight as it is being represented exactly while the second-stage objective requires the relaxation of the prediction model which will not be tight due to the big-M constraints. This means that the maximization problem in the AP favors the second stage. This mismatch could lead to inaccurate scenarios and undesirable downstream effects within branch-and-bound.

H.4 Baseline Solution Quality at Neur2RO Termination Time

In this section, we report the objective quality, i.e., the median relative error, for k-adaptability baseline at the termination time of Neur2RO in Table 10. From the table, we can see that the performance is median RE of Neur2RO is marginally better than when k-adaptability is given 3 hours, except $n = 20, 40$. Note that these tables are only be reproduced for capital budgeting as

# items	Median RE		Times	
	standard	LP relaxation	standard	LP relaxation
10	**1.105**	2.663	59	**4**
20	**0.000**	0.060	324	**142**
30	0.109	**0.071**	602	**141**
40	0.009	**0.007**	739	**226**
50	**0.001**	0.001	1,032	**231**

Table 8: Median RE and solving times for capital budgeting instances with LP relaxation. For each row, the median RE and average solving time are computed over 50 instances. All times in seconds. The smallest (best) values in each row/metric are in bold.

# items	Median RE		Times	
	sum	second only	sum	second-only
10	**1.105**	2.424	**20**	233
20	**0.000**	0.192	**324**	1,823
30	**0.109**	0.151	**602**	3,823
40	**0.009**	0.010	**739**	4,062
50	**0.001**	0.005	**1,032**	7,424

Table 9: Sum and second-stage only predictions for capital budgeting instances. For each row, the median RE and solving time are computed over 50 instances. Note that in these results, the RE is calculated with respect to the k-adaptability and each respective ML-approach. All times in seconds.

we do not have the knapsack results throughout the solving process, given only the final objective values are reported in Arslan & Detienne (2022).

# items	Median RE at 3 hours				Median RE at `Neur2RO` termination			
	`Neur2RO`	$k=2$	$k=5$	$k=10$	`Neur2RO`	$k=2$	$k=5$	$k=10$
10	1.105	1.140	**0.000**	**0.000**	0.809	1.559	**0.267**	0.359
20	**0.000**	0.196	0.112	0.064	**0.011**	0.240	0.098	0.084
30	0.109	**0.020**	0.073	0.032	0.102	0.067	0.093	**0.029**
40	**0.009**	0.074	0.011	0.019	**0.013**	0.079	0.058	0.019
50	**0.001**	0.033	0.039	0.020	**0.002**	0.035	0.006	0.008

Table 10: Median RE for capital budgeting at 3 hour time limit and `Neur2RO` termination time. For each row, the median RE and average solving time are computed over 50 instances. All times in seconds. The smallest (best) values in each row/metric are in bold.

I Machine Learning Model Details

I.1 Features

Here we provide the features for each of the problems. In both cases, set-based architectures Zaheer et al. (2017) with parameter sharing utilized, so we report the features for a single dimension of the first-stage decision and scenario accordingly. Table 11 reports all of the features for each instance.

I.2 Model Hyperparameters

This section reports the hyperparameters for the neural networks for each problem. For both problems, we have the same architecture with slightly different hyperparameters. As the objective of `Neur2RO` is to enable efficient optimization, we train small networks that can achieve a low mean absolute error value to ensure that the main and adversarial problems are tractable. For this reason, no systematic hyperparameter tuning was done. Hyperparameter optimization would likely only further improve the already strong numerical results. For both problems, we train a model for 500

Problem	First-Stage Features	Scenario Features
Knapsack	$x_i, f_i, \bar{p}_i, \hat{p}_i, r_i, c_i, t_i, C$	$\xi_i, f_i, \bar{p}_i, \hat{p}_i, r_i, c_i, t_i, C$
Capital budgeting	x_i, r_i, c_i	$\left(1 + \mathbf{\Phi}_i^\intercal \boldsymbol{\xi}/2\right)_i, \left(1 + \mathbf{\Psi}_i^\intercal \boldsymbol{\xi}/2\right)_i, r_i, c_i$

Table 11: Features for first-stage decision and scenario embedding networks.

epochs and compute the mean absolute error on a validation set every 10 epochs. We then use the model with the lowest reported mean absolute validation error during training for evaluation.

Table 12 reports the hyperparameters for each model. As our model generalizes across instances, which requires invariance to the order and number of decision variables, both the first-stage and scenario embedding networks are set-based architectures (Zaheer et al., 2017). We refer to Figure 3 for a refresher on the overall architecture which has the following hyperparameters. The hyperparameters "$\hat{\Phi}_{\mathbf{x}}$ dimensions" and "$\Phi_{\mathbf{x}}$ dimensions" correspond to the hidden and embedding dimensions of the first-stage embedding network. Specifically, "$\hat{\Phi}_{\mathbf{x}}$ dimensions" corresponds to the network with shared parameters that embed the representation for each first-stage decision. The last dimension of "$\hat{\Phi}_{\mathbf{x}}$ dimensions" is that of the aggregated vector. The hyperparameter "$\Phi_{\mathbf{x}}$ dimensions" corresponds to the network that takes the aggregated first-stage embedding vector as input. The last dimension of "$\Phi_{\mathbf{x}}$ dimensions" specifies the embedding dimension of the first-stage embedding network. "$\hat{\Phi}_{\boldsymbol{\xi}}$ dimensions" and "$\Phi_{\boldsymbol{\xi}}$ dimensions" are analogous for the scenario embedding network. "Φ dimensions" correspond to the hidden dimensions of the value network. Finally, "aggregation type" specifies the type of aggregation that combines the first-stage/scenario embeddings.

Hyperparameter	Knapsack	Capital budgeting
Feature scaling	min-max	min-max
Label scaling	min-max	min-max
# epochs	500	500
Batch size	256	256
Learning rate	0.001	0.001
Dropout	0	0
Loss function	MSELoss	MSELoss
Optimizer	Adam	Adam
$\hat{\Phi}_{\mathbf{x}}$ dimensions	[32, 16]	[16, 4]
$\Phi_{\mathbf{x}}$ dimensions	[64, 8]	[32, 8]
$\hat{\Phi}_{\boldsymbol{\xi}}$ dimensions	[32, 16]	[16, 4]
$\Phi_{\boldsymbol{\xi}}$ dimensions	[64, 8]	[32, 8]
Φ dimensions	[8]	[8]
Aggregation type	sum	sum

Table 12: Hyperparameters for neural networks.

I.3 Training Curves

Figures 9-10 plot the mean absolute error at every 10 epochs during training for the training and validation data. Generally, the training and validation mean absolute error is very close, and in both problems, a relatively low mean absolute error is achieved.

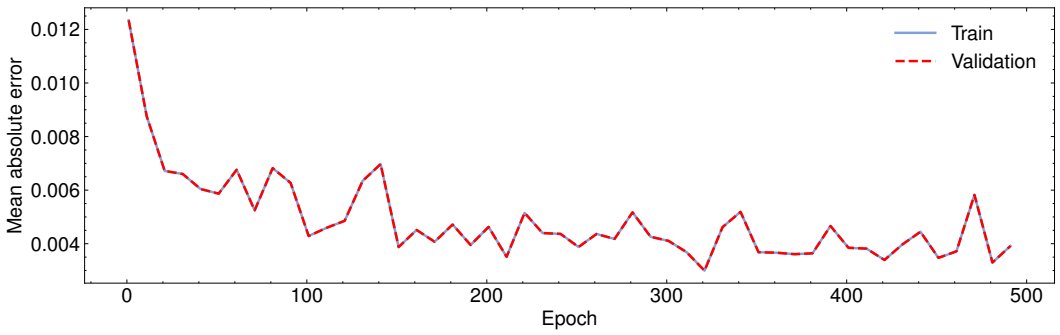

Figure 9: Training curve for knapsack.

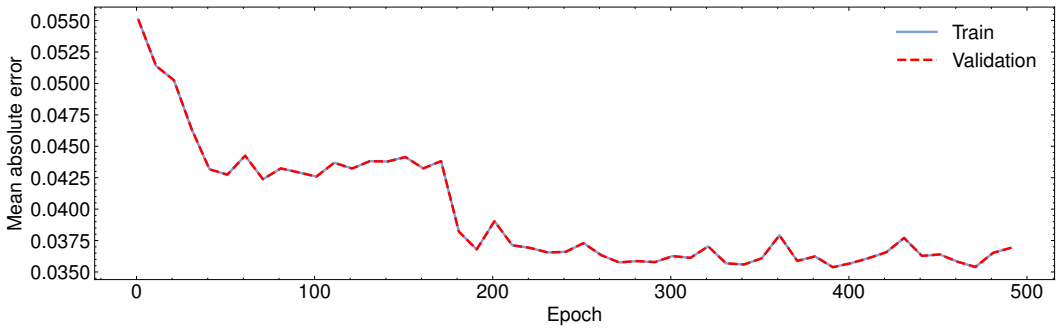

Figure 10: Training curve for capital budgeting.