A Appendix

We present some computational and architectural details for the proposed task-based learning model, both in the general case and for the experiments described in Section 4.

A.1 Differentiating the optimization solution to a stochastic programming problem

The issue of chief technical challenge to our approach is computing the gradient of an objective that depends upon the argmin operation $z^\star(x;\theta)$. Specifically, we need to compute the term

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial z^\star}\frac{\partial z^\star}{\partial \theta} \tag{A.1}$$

which involves the Jacobian $\frac{\partial z^\star}{\partial \theta}$. This is the Jacobian of the optimal solution with respect to the distribution parameters θ. Recent approaches have looked into similar argmin differentiations [28, 29], though the methodology we present here is more general and handles the stochasticity of the objective.

We begin by writing the KKT optimality conditions of the general stochastic programming problem (3), where all expectations are taken with respect to the modeled distribution $y \sim p(y|x;\theta)$ (for compactness, denoted here as $\mathbf{E}_{y_\theta}$). Further, assuming the problem is convex means we can replace the general equality constraints $h(z) = 0$ with the linear constraint $Az = b$. A point (z, λ, ν) is a primal-dual optimal point if it satisfies

$$\begin{aligned} \mathbf{E}_{y_\theta} g(z) &\leq 0 \\ Az &= b \\ \lambda &\geq 0 \\ \lambda \circ \mathbf{E}_{y_\theta} g(z) &= 0 \\ \nabla_z \mathbf{E}_{y_\theta} f(z) + \lambda^T \nabla_z \mathbf{E}_{y_\theta} g(z) + A^T \nu &= 0 \end{aligned} \tag{A.2}$$

where here g denotes the vector of all inequality constraints (represented as a vector-valued function), and where we wrap the dependence on x and y into the functions f and g_i themselves.

Differentiating these equations and applying the implicit function theorem gives a set of linear equations that we can solve to obtain the necessary Jacobians

$$\begin{bmatrix} \nabla_z^2 \mathbf{E}_{y_\theta} f(z) + \sum_{i=1}^{n_{ineq}} \lambda_i \nabla_z^2 \mathbf{E}_{y_\theta} g_i(z) & (\nabla_z \mathbf{E}_{y_\theta} g(z))^T & A^T \\ \operatorname{diag}(\lambda)\,(\nabla_z \mathbf{E}_{y_\theta} g(z)) & \operatorname{diag}(\mathbf{E}_{y_\theta} g(z)) & 0 \\ A & 0 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial z}{\partial \theta} \\ \frac{\partial \lambda}{\partial \theta} \\ \frac{\partial \nu}{\partial \theta} \end{bmatrix} = \begin{bmatrix} \frac{\partial \nabla_z \mathbf{E}_{y_\theta} f(z)}{\partial \theta} + \frac{\partial \sum_{i=1}^{n_{ineq}} \lambda_i \nabla_z \mathbf{E}_{y_\theta} g_i(z)}{\partial \theta} \\ \operatorname{diag}(\lambda) \frac{\partial \mathbf{E}_{y_\theta} g(z)}{\partial \theta} \\ 0 \end{bmatrix}. \tag{A.3}$$

The terms on the left side are the optimality conditions of the convex problem, and the terms on right side are the derivatives of the relevant functions at the achieved solution, with respect to the governing parameter θ. These equations will take slightly different forms depending on how the stochastic programming problem is solved, but are usually fairly straightforward to compute if the solution is solved in some "exact" manner (i.e., where second order information is used). In practice, we calculate the right side of this equation by employing sequential quadratic programming [30] to find the optimal policy $z^\star$ for the given parameters θ, using a recently-proposed approach for fast solution of argmin differentiation for QPs [31] to solve the necessary linear equations; we then take the derivatives at the optimum produced by this strategy.

A.2 Details on computation for inventory stock problem

The objective for our "conditional" variation of the classical inventory stock problem is

$$f_{stock}(y, z) = c_0 z + \frac{1}{2} q_0 z^2 + c_b [y - z]_+ + \frac{1}{2} q_b ([y - z]_+)^2 + c_h [z - y]_+ + \frac{1}{2} q_h ([z - y]_+)^2 \tag{A.4}$$

where z is the amount of product ordered; y is the stochastic electricity demand (which is affected by features x); $[v]_+ \equiv \max\{v, 0\}$; and (c_0, q_0), (c_b, q_b), and (c_h, q_h) are linear and quadratic costs on the amount of product ordered, over-orders, and under-orders, respectively. Our proxy stochastic programming problem can then be written as

$$\underset{z}{\operatorname{minimize}} \quad L(\theta) = \mathbf{E}_{y \sim p(y|x;\theta)} [f_{stock}(y, z)]. \tag{A.5}$$

To simplify the setting, we further assume that the demands are discrete, taking on values $d_1, \ldots, d_k$ with probabilities (conditional on x) $(p_\theta)_i \equiv p(y = d_i|x;\theta)$. Thus our stochastic programming problem (A.5) can be written succinctly as a joint quadratic program

$$\begin{aligned} \underset{z\in\mathbb{R}, z_b, z_h \in \mathbb{R}^k}{\text{minimize}} \quad & c_0 z + \frac{1}{2} q_0 z^2 + \sum_{i=1}^{k} (p_\theta)_i \left(c_b (z_b)_i + \frac{1}{2} q_b (z_b)_i^2 + c_h (z_h)_i + \frac{1}{2} q_h (z_h)_i^2 \right) \\ \text{subject to} \quad & d - z\mathbf{1} \leq z_b, \;\; z\mathbf{1} - d \leq z_h, \;\; z, z_h, z_b \geq 0. \end{aligned} \tag{A.6}$$

To demonstrate the explicit formula for argmin operation Jacobians for this particular case (e.g., to compute the terms in (A.3)), note that we can write the above QP in inequality form as $\text{minimize}_{\{\boldsymbol{z}:G\boldsymbol{z}\leq h\}} \frac{1}{2}\boldsymbol{z}^T Q \boldsymbol{z} + c^T \boldsymbol{z}$ with

$$\boldsymbol{z} = \begin{bmatrix} z \\ z_b \\ z_h \end{bmatrix}, \; Q = \begin{bmatrix} q_0 & 0 & 0 \\ 0 & q_b p_\theta & 0 \\ 0 & 0 & q_h p_\theta \end{bmatrix}, \; c = \begin{bmatrix} c_0 \\ c_b p_\theta \\ c_h p_\theta \end{bmatrix}, \; G = \begin{bmatrix} -1 & -I & 0 \\ 1 & 0 & -I \\ -1 & 0 & 0 \\ 0 & -I & 0 \\ 0 & 0 & -I \end{bmatrix}, \; h = \begin{bmatrix} -d \\ d \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{A.7}$$

Thus, for an optimal primal-dual solution $(\boldsymbol{z}^\star, \lambda^\star)$, we can compute the Jacobian $\frac{\partial \boldsymbol{z}^\star}{\partial p_\theta}$ (the Jacobian of the optimal solution with respect to the probability vector p_θ mentioned above), via the formula

$$\begin{bmatrix} \frac{\partial \boldsymbol{z}^\star}{\partial p_\theta} \\ \frac{\partial \lambda^\star}{\partial p_\theta} \end{bmatrix} = \begin{bmatrix} Q & G^T \\ D(\lambda^\star) G & D(G\boldsymbol{z}^\star - h) \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ q_b z_b^\star + c_b 1 \\ q_h z_h^\star + c_h 1 \\ 0 \end{bmatrix}, \tag{A.8}$$

where $D(\cdot)$ denotes a diagonal matrix for an input vector. After solving the problem and computing these Jacobians, we can compute the overall gradient with respect to the task loss $L(\theta)$ via the chain rule

$$\frac{\partial L}{\partial \theta} = \frac{\partial L}{\partial \boldsymbol{z}^\star} \frac{\partial \boldsymbol{z}^\star}{\partial p_\theta} \frac{\partial p_\theta}{\partial \theta} \tag{A.9}$$

where $\frac{\partial p_\theta}{\partial \theta}$ denotes the Jacobian of the model probabilities with respect to its parameters, which are computed in the typical manner. Note that in practice, these Jacobians need not be computed explicitly, but can be computed efficiently via backpropagation; we use a recently-developed differentiable batch QP solver [31] to both solve the optimization problem in QP form and compute its derivatives.

A.3 Details on computation for power scheduling problem

The objective for the load forecasting problem is given by

$$\begin{aligned} \underset{z\in\mathbb{R}^{24}}{\text{minimize}} \quad & \sum_{i=1}^{24} \mathbf{E}_{y\sim p(y|x;\theta)} \left[\gamma_s [y_i - z_i]_+ + \gamma_e [z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2 \right] \\ \text{subject to} \quad & |z_i - z_{i-1}| \leq c_r \; \forall i, \end{aligned} \tag{A.10}$$

where z is the generator schedule, y is the stochastic demand (which is affected by features x), $[v]_+ \equiv \max\{v, 0\}$, γ_e is an over-generation penalty, γ_s is an under-generation penalty, and c_r is a ramping constraint. Assuming that y_i is a Gaussian random variable with mean μ_i and variance σ_i^2, then this expectation has a closed form that can be computed via analytically integrating the Gaussian PDF. Specifically, this closed form is

$$\begin{aligned} & \mathbf{E}_{y\sim p(y|x;\theta)} \left[\gamma_s [y_i - z_i]_+ + \gamma_e [z_i - y_i]_+ + \frac{1}{2}(z_i - y_i)^2 \right] \\ & \quad = \underbrace{(\gamma_s + \gamma_e)(\sigma^2 p(z_i; \mu, \sigma^2) + (z_i - \mu) F(z_i; \mu, \sigma^2)) - \gamma_s (z_i - \mu)}_{\alpha(z_i)} + \frac{1}{2}((z_i - \mu_i)^2 + \sigma_i^2), \end{aligned} \tag{A.11}$$

where $p(z; \mu, \sigma^2)$ and $F(z; \mu, \sigma^2)$ denote the Gaussian PDF and CDF, respectively with the given mean and variance. This is a convex function of z (not apparent in this form, but readily established because it is an expectation of a convex function), and we can thus optimize it efficiently and compute the necessary Jacobians.

Specifically, we use sequential quadratic programming (SQP) to iteratively approximate the resultant convex objective as a quadratic objective, and iterate until convergence; specifically, we repeatedly solve

$$\begin{aligned} z^{(k+1)} = \operatorname*{argmin}_{z} \; & \frac{1}{2} z^T \mathrm{diag}\left(\frac{\partial^2 \alpha(z_i^{(k)})}{\partial z^2} + 1 \right) z + \left(\frac{\partial \alpha(z^{(k)})}{\partial z} - \mu \right)^T z \\ \text{subject to } \; & |z_i - z_{i-1}| \leq c_r \; \forall i \end{aligned} \tag{A.12}$$

until $||z^{(k+1)} - z^{(k)}|| < \delta$ for a small δ, where

$$\begin{aligned} \frac{\partial \alpha}{\partial z} &= (\gamma_s + \gamma_e) F(z; \mu, \sigma) - \gamma_s, \\ \frac{\partial^2 \alpha}{\partial z^2} &= (\gamma_s + \gamma_e) p(z; \mu, \sigma). \end{aligned} \tag{A.13}$$

We then compute the necessary Jacobians using the quadratic approximation (A.12) at the solution, which gives the correct Hessian and gradient terms. We can furthermore differentiate the gradient and Hessian with respect to the underlying model parameters μ and σ^2, again using a recently-developed batch QP solver [31].

A.4 Details on computation for battery storage problem

The objective for the battery storage problem is given by

$$\begin{aligned} \underset{z_{\text{in}}, z_{\text{out}}, z_{\text{state}} \in \mathbb{R}^{24}}{\text{minimize}} \; & \mathbf{E}_{y \sim p(y|x;\theta)} \left[\sum_{i=1}^{24} y_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2 \right] \\ \text{subject to } \; & z_{\text{state},i+1} = z_{\text{state},i} - z_{\text{out},i} + \gamma_{\text{eff}} z_{\text{in},i} \; \forall i, \;\; z_{\text{state},1} = B/2, \\ & 0 \leq z_{\text{in}} \leq c_{\text{in}}, \;\; 0 \leq z_{\text{out}} \leq c_{\text{out}}, \;\; 0 \leq z_{\text{state}} \leq B, \end{aligned} \tag{A.14}$$

where $z_{\text{in}}, z_{\text{out}}, z_{\text{state}}$ are decisions over the charge amount, discharge amount, and resultant state of the battery, respectively; y is the stochastic electricity price (which is affected by features x); B is the battery capacity; γ_{eff} is the battery charging efficiency; c_{in} and c_{out} are maximum hourly charge and discharge amounts, respectively; and λ and ϵ are hyperparameters related to flexibility and battery health, respectively.

Assuming y_i is a random variable with mean μ_i, the expectation in the objective has a closed form:

$$\begin{aligned} & \mathbf{E}_{y \sim p(y|x;\theta)} \left[\sum_{i=1}^{24} y_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2 \right] \\ & \qquad = \sum_{i=1}^{24} \mu_i (z_{\text{in}} - z_{\text{out}})_i + \lambda \left\| z_{\text{state}} - \frac{B}{2} \right\|^2 + \epsilon \|z_{\text{in}}\|^2 + \epsilon \|z_{\text{out}}\|^2. \end{aligned} \tag{A.15}$$

We can then write this expression in QP form as $\text{minimize}_{\{\boldsymbol{z}: G\boldsymbol{z} \leq h, \; A\boldsymbol{z} = b\}} \frac{1}{2} \boldsymbol{z}^T Q \boldsymbol{z} + c^T \boldsymbol{z}$ with

$$\begin{aligned} & \boldsymbol{z} = \begin{bmatrix} z_{\text{in}} \\ z_{\text{out}} \\ z_{\text{state}} \end{bmatrix}, \; Q = \begin{bmatrix} \epsilon I & 0 & 0 \\ 0 & \epsilon I & 0 \\ 0 & 0 & \lambda I \end{bmatrix}, \; c = \begin{bmatrix} \mu \\ -\mu \\ -\lambda B \mathbf{1} \end{bmatrix}, \\ & G = \begin{bmatrix} I & 0 & 0 \\ -I & 0 & 0 \\ 0 & I & 0 \\ 0 & -I & 0 \\ 0 & 0 & I \\ 0 & 0 & -I \end{bmatrix}, \; h = \begin{bmatrix} c_{\text{in}} \\ 0 \\ c_{\text{out}} \\ 0 \\ B \\ 0 \end{bmatrix}, \; A = \begin{bmatrix} 0 & 0 & 0, \ldots, 0, 1 \\ \gamma_{\text{eff}} D_1^T & -D_1^T & D_1^T - D_2^T \end{bmatrix} \; b = \begin{bmatrix} B/2 \\ 0 \end{bmatrix}, \end{aligned} \tag{A.16}$$

where $D_1 = \begin{bmatrix} I \\ 0 \end{bmatrix} \in \mathbb{R}^{24 \times 23}$ and $D_2 = \begin{bmatrix} 0 \\ I \end{bmatrix} \in \mathbb{R}^{24 \times 23}$.

For this experiment, we assume that y_i is a lognormal random variable (with mean μ_i); thus, to obtain our predictions, we predict the mean of $\log(y)$ (i.e., we predict $\log(\mu)$). After obtaining these predictions, we solve (A.4), compute the necessary Jacobians at the solution, and update the underlying model parameter μ via backpropagation, again using [31].

A.5 Implementation notes

For all linear models, we use a one-layer linear neural network with the appropriate input and output layer dimensions. For all nonlinear models, we use a two-hidden-layer neural network, where each "layer" is actually a combination of linear, batch norm [32], ReLU, and dropout ($p = 0.2$) layers with dimension 200. In both cases, we add an additional softmax layer in cases where probability distributions are being predicted.

All models are implemented using PyTorch[A.1] and employ the Adam optimizer [33]. All QPs are solved using a recently-developed differentiable batch QP solver [31], and Jacobians are also computed automatically using backpropagation via the same.

Source code for all experiments is available at `https://github.com/locuslab/e2e-model-learning`.

[A.1] `https://pytorch.org`

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE1252522, and by the Department of Energy Computational Science Graduate Fellowship.

References

[1] Stein W Wallace and Stein-Erik Fleten. Stochastic programming models in energy. *Handbooks in operations research and management science*, 10:637–677, 2003.

[2] William T Ziemba and Raymond G Vickson. *Stochastic optimization models in finance*, volume 1. World Scientific, 2006.

[3] John A Buzacott and J George Shanthikumar. *Stochastic models of manufacturing systems*, volume 4. Prentice Hall Englewood Cliffs, NJ, 1993.

[4] Alexander Shapiro and Andy Philpott. A tutorial on stochastic programming. *Manuscript. Available at* `www2.isye.gatech.edu/ashapiro/publications.html`, 17, 2007.

[5] Jeff Linderoth, Alexander Shapiro, and Stephen Wright. The empirical behavior of sampling methods for stochastic programming. *Annals of Operations Research*, 142(1):215–241, 2006.

[6] R Tyrrell Rockafellar and Roger J-B Wets. Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of operations research*, 16(1):119–147, 1991.

[7] Yann LeCun, Urs Muller, Jan Ben, Eric Cosatto, and Beat Flepp. Off-road obstacle avoidance through end-to-end learning. In *NIPS*, pages 739–746, 2005.

[8] Ryan W Thomas, Daniel H Friend, Luiz A Dasilva, and Allen B Mackenzie. Cognitive networks: adaptation and learning to achieve end-to-end performance objectives. *IEEE Communications Magazine*, 44(12):51–57, 2006.

[9] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1457–1464. IEEE, 2011.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[11] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Pattern Recognition (ICPR), 2012 21st International Conference on*, pages 3304–3308. IEEE, 2012.

[12] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*, volume 14, pages 1764–1772, 2014.

[13] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, et al. Deep speech 2: End-to-end speech recognition in english and mandarin. *arXiv preprint arXiv:1512.02595*, 2015.

[14] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

[15] Aviv Tamar, Sergey Levine, Pieter Abbeel, YI WU, and Garrett Thomas. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2146–2154, 2016.

[16] Ken Harada, Jun Sakuma, and Shigenobu Kobayashi. Local search for multiobjective function optimization: pareto descent method. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 659–666. ACM, 2006.

[17] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[18] Hossam Mossalam, Yannis M Assael, Diederik M Roijers, and Shimon Whiteson. Multi-objective deep reinforcement learning. *arXiv preprint arXiv:1610.02707*, 2016.

[19] Marco A Wiering, Maikel Withagen, and Mădălina M Drugan. Model-based multi-objective reinforcement learning. In *Adaptive Dynamic Programming and Reinforcement Learning (ADPRL), 2014 IEEE Symposium on*, pages 1–6. IEEE, 2014.

[20] Veselin Stoyanov, Alexander Ropson, and Jason Eisner. Empirical risk minimization of graphical model parameters given approximate inference, decoding, and model structure. *International Conference on Artificial Intelligence and Statistics*, 15:725–733, 2011. ISSN 15324435.

[21] Tamir Hazan, Joseph Keshet, and David A McAllester. Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems*, pages 1594–1602, 2010.

[22] Yang Song, Alexander G Schwing, Richard S Zemel, and Raquel Urtasun. Training deep neural networks via direct loss minimization. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2169–2177, 2016.

[23] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

[24] Somil Bansal, Roberto Calandra, Ted Xiao, Sergey Levine, and Claire J Tomlin. Goal-driven dynamics learning via bayesian optimization. *arXiv preprint arXiv:1703.09260*, 2017.

[25] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[26] Yoshua Bengio. Using a financial training criterion rather than a prediction criterion. *International Journal of Neural Systems*, 8(04):433–443, 1997.

[27] Adam N Elmachtoub and Paul Grigas. Smart "predict, then optimize". *arXiv preprint arXiv:1710.08005*, 2017.

[28] Stephen Gould, Basura Fernando, Anoop Cherian, Peter Anderson, Rodrigo Santa Cruz, and Edison Guo. On differentiating parameterized argmin and argmax problems with application to bi-level optimization. *arXiv preprint arXiv:1607.05447*, 2016.

[29] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. *arXiv preprint arXiv:1609.07152*, 2016.

[30] Paul T Boggs and Jon W Tolle. Sequential quadratic programming. *Acta numerica*, 4:1–51, 1995.

[31] Brandon Amos and J Zico Kolter. Optnet: Differentiable optimization as a layer in neural networks. *arXiV preprint arXiv:1703.00443*, 2017.

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[33] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.