Advancing Data-driven Decision Making: An Uncertainty-based Predict-and-Optimize Approach

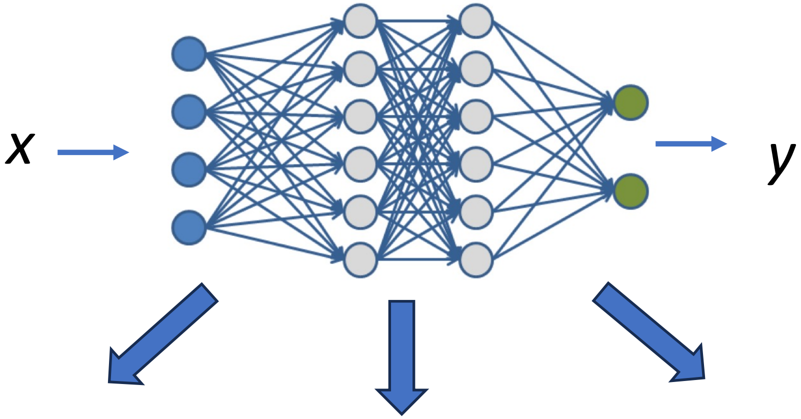
Lingkai Kong

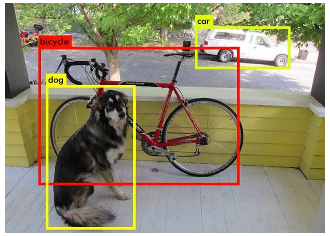
School of Computational Science and Engineering

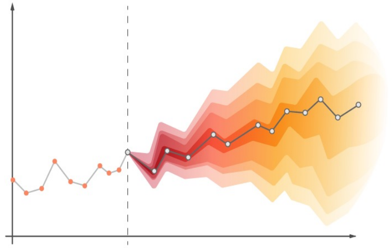
Georgia Tech

Deep Neural Networks

DNNs:



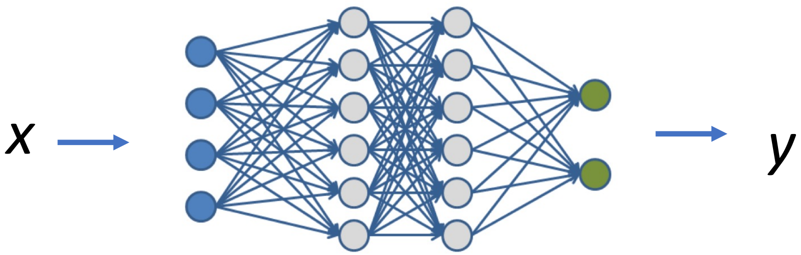


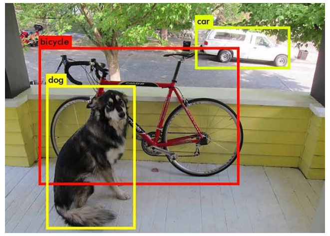


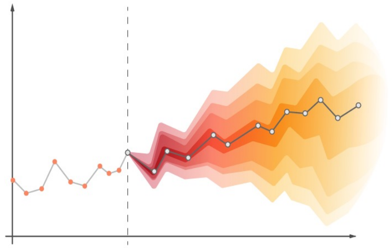
- Powerful for prediction
- Black-box, uninterpretable

Deep Neural Networks vs Optimization

DNNs:





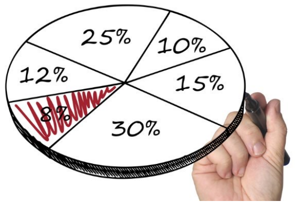


- Powerful for prediction
- Black-box, uninterpretable

Optimization:

$$a^* = \text{argmin}_a f(a; y)$$

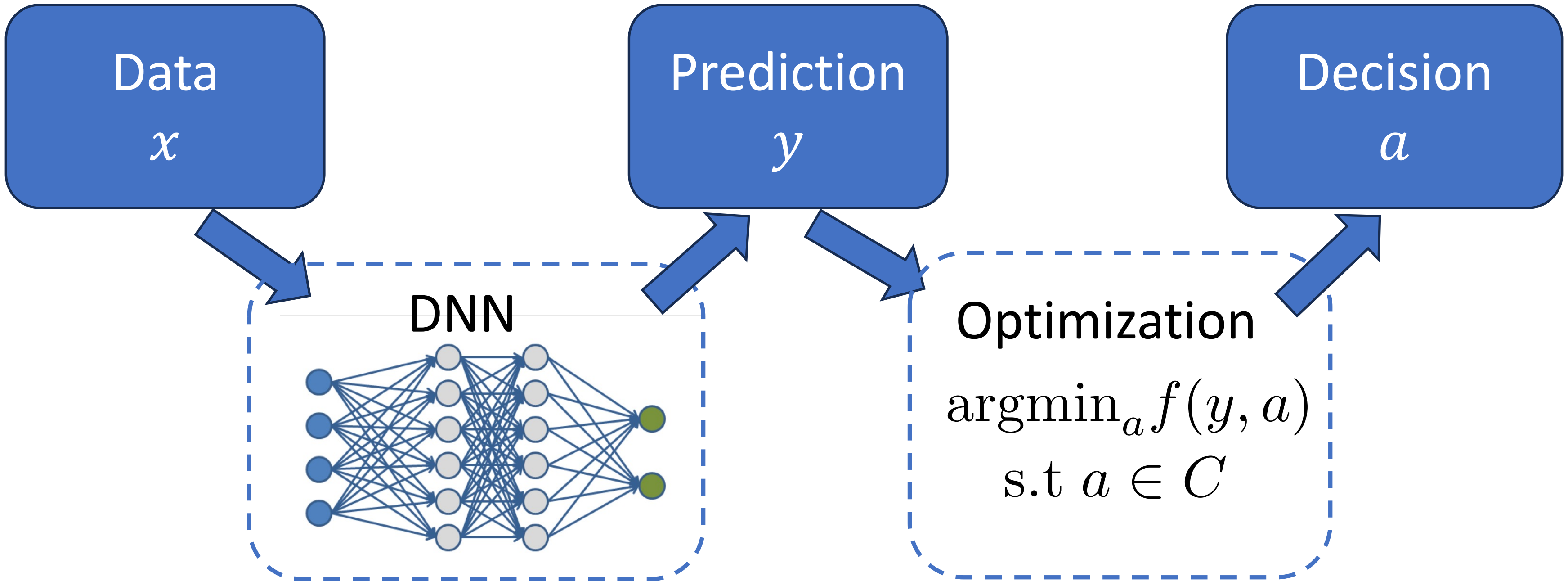
$$\text{subject to } a \in C$$



Job Scheduler

- Powerful for decision making
- Encode domain knowledge

The Data-driven Decision Making Pipeline

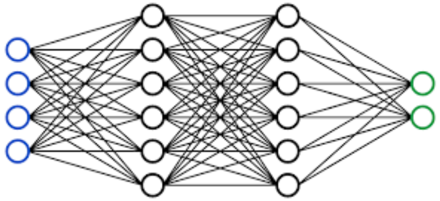


Asset Allocation for Hedge Fund

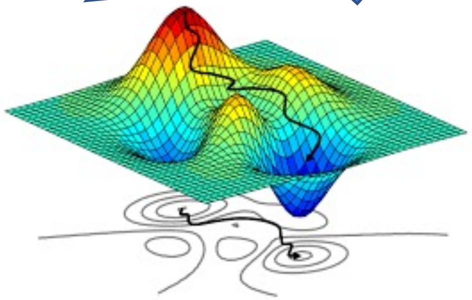
Observed features

Future stock price

Asset allocation



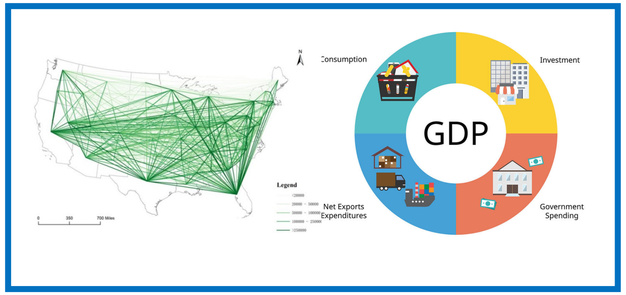
Predictive model



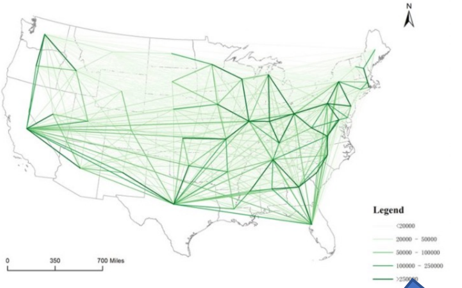
Portfolio optimization

Vaccine Distribution for COVID-19

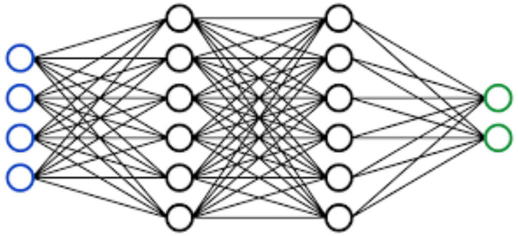
Observed features

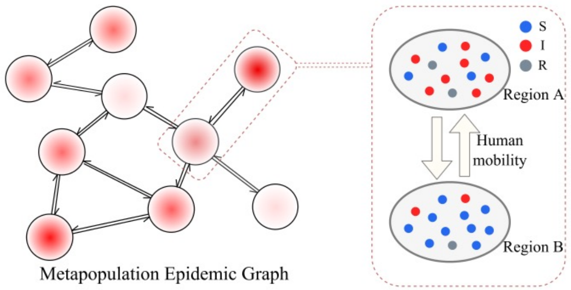


Future mobility flow



Vaccine distribution

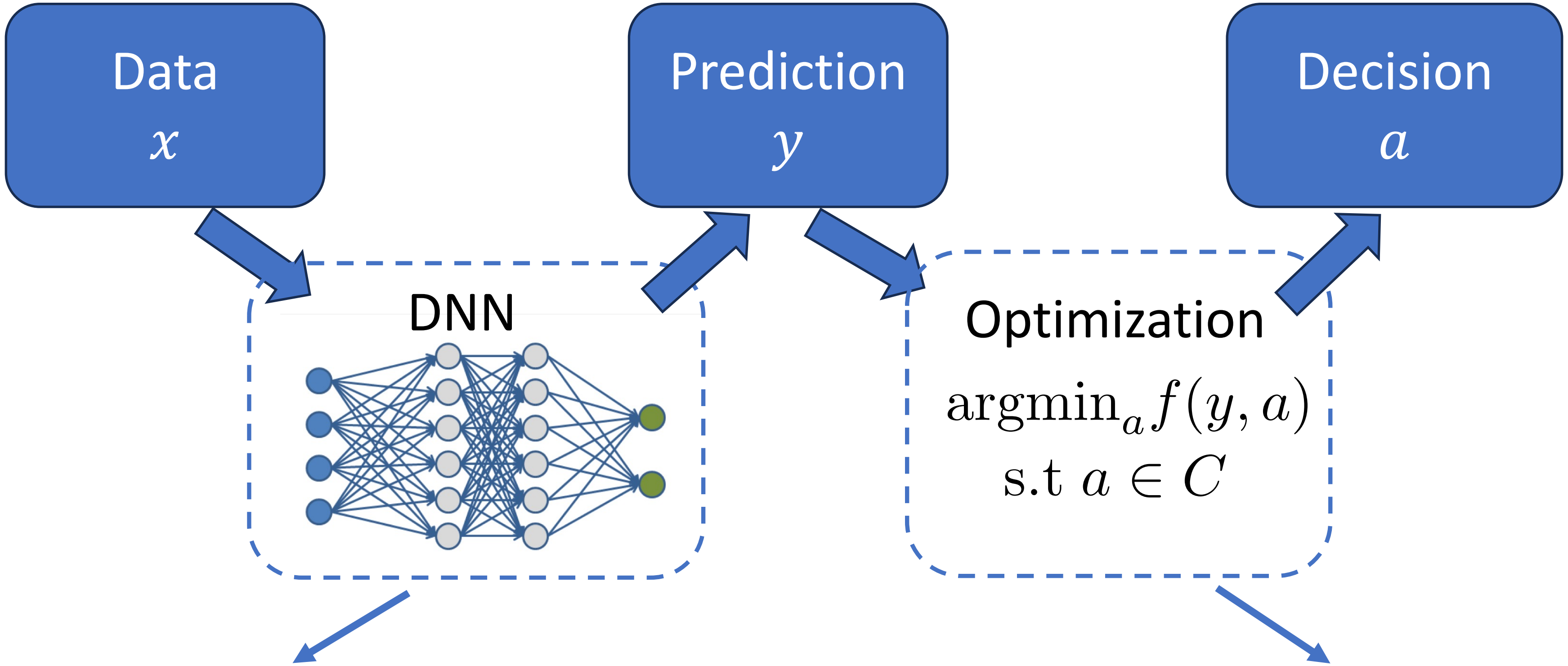




Predictive model

Compartmental simulation model

Challenges in the Data-driven Decision-Making Pipeline



I: Uncertainty quantification:

ICML 20, EMNLP 20, NeurIPS 21, WWW 21, NAACL 22

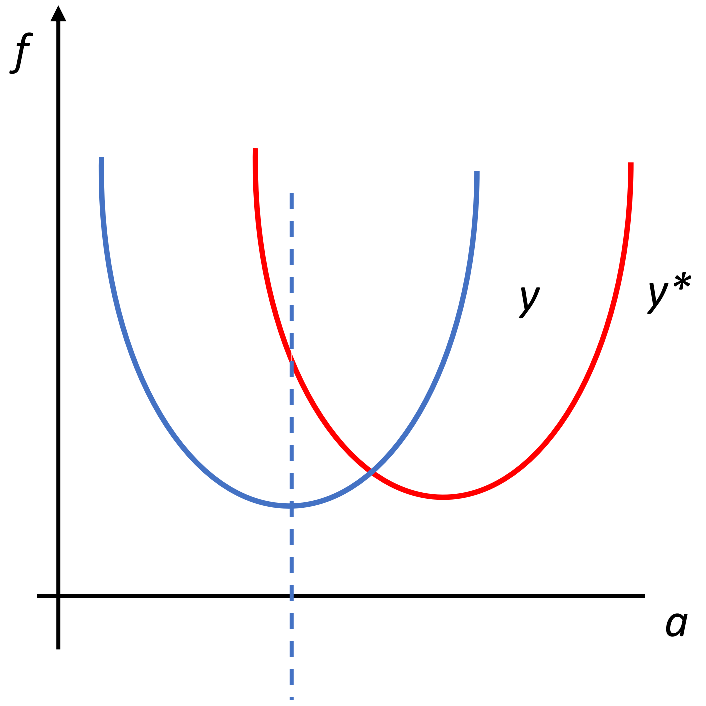
II: Integrating prediction and optimization

NeurIPS 22 (Oral), arxiv

Challenge I: Uncertainty Quantification of Predictive Model

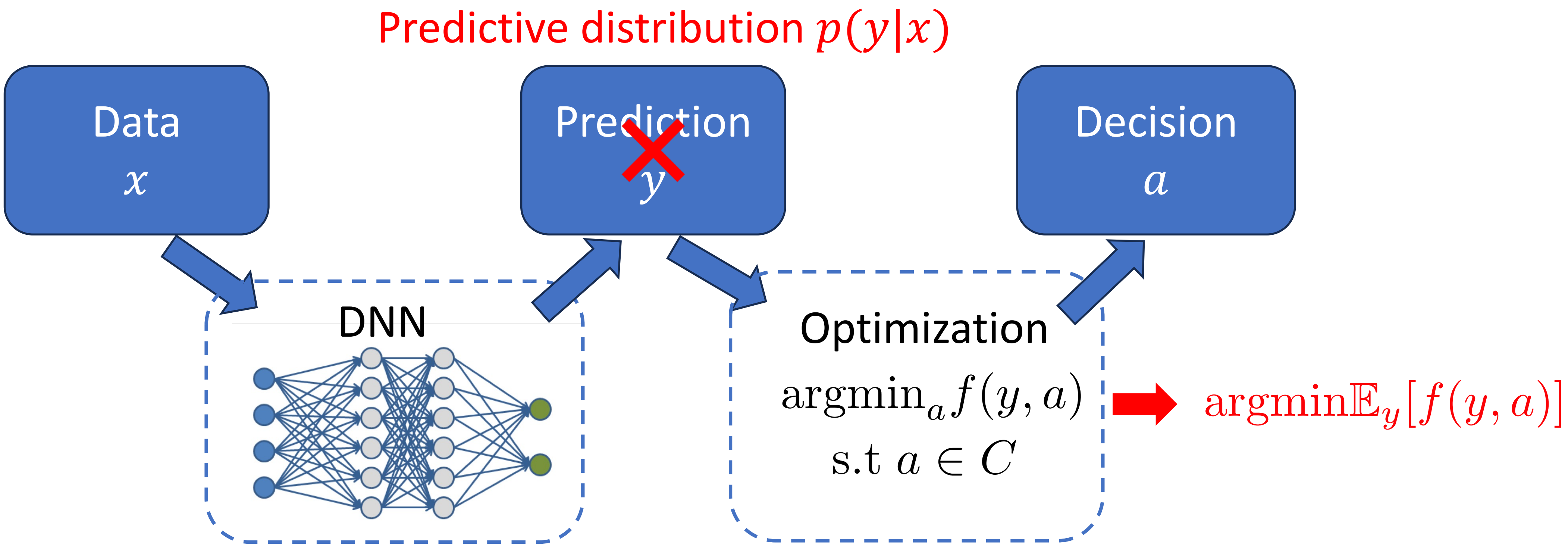
Why Should We Care about Uncertainty?

Even **small** prediction error can cause **large** decision loss



- DNN can make errors → Catastrophic for risk-sensitive domains

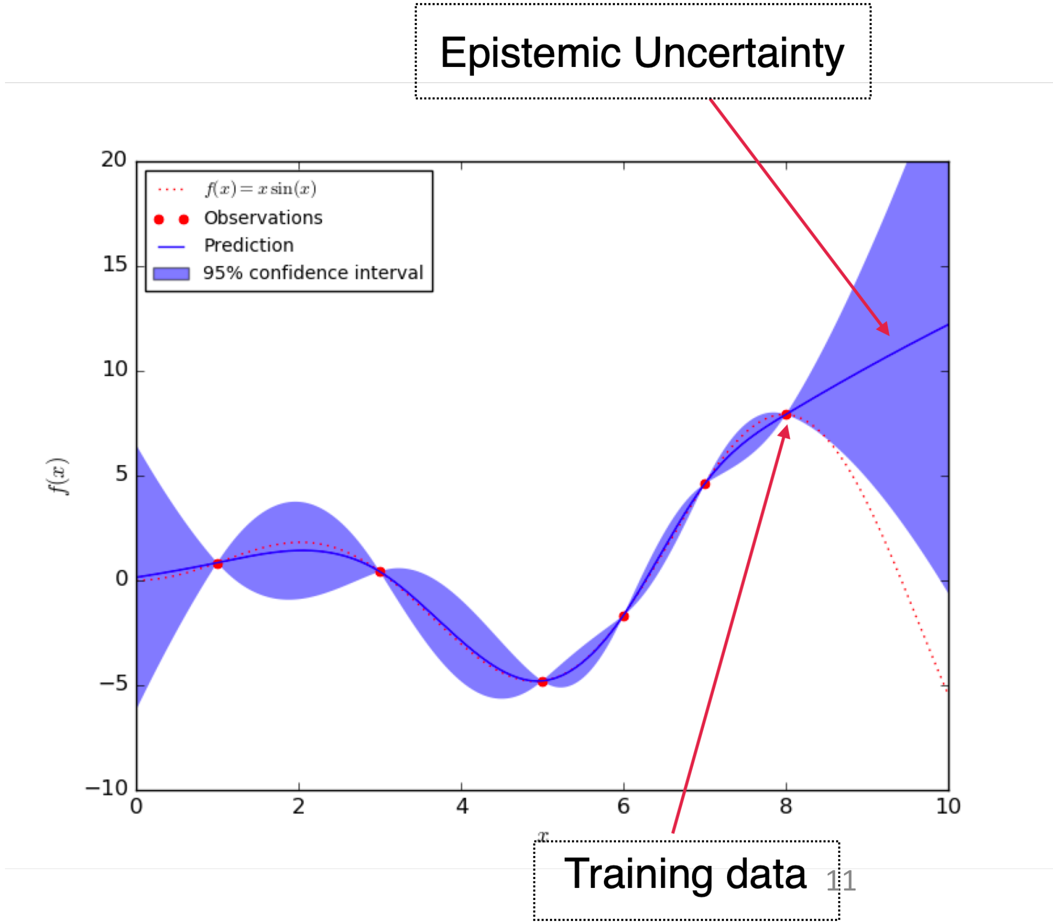
Why Should We Care about Uncertainty?



- DNNs can make errors → Catastrophic for risk-sensitive domains
- We need uncertainty to make more robust decisions

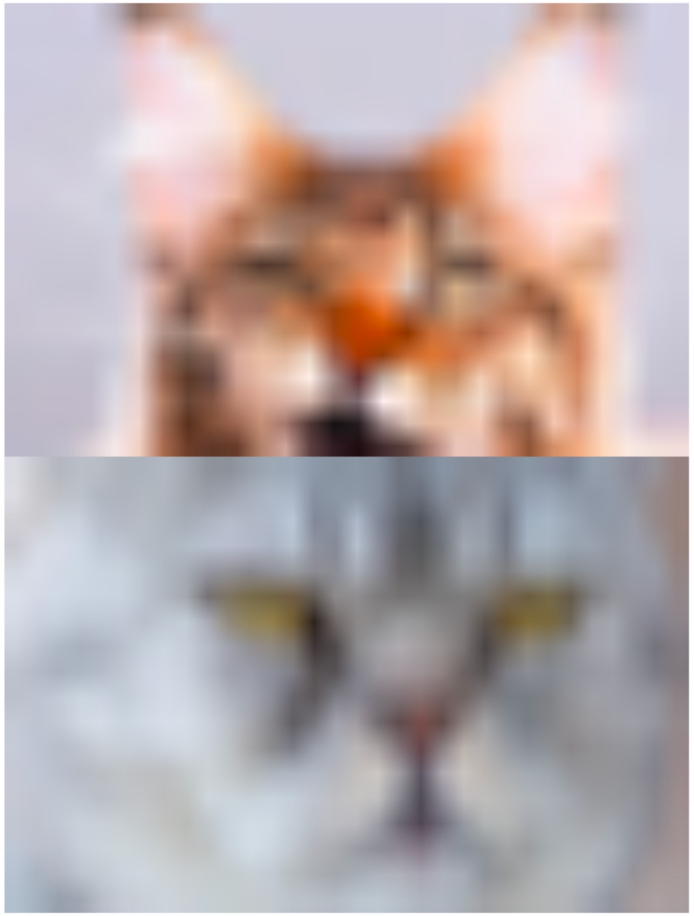
Sources of Uncertainty: Epistemic Uncertainty

- Ignorance about model caused by lack of observations
  - Insufficient data
  - Model complexity
- Also known as model uncertainty.
- **Can** be **reduced** with more training data.



Sources of Uncertainty: Aleatoric Uncertainty

- Natural randomness inherent in the task.
  - Class overlap
  - Data noise & measurement errors
  - Poor features
- Also known as data uncertainty.
- **Cannot** be **reduced** with more training data.
- **Can** be reduced with **additional features**.



Importance of the Two Uncertainties

- Model debugging
  - More training data or more features?
- Active learning
  - Query data of high epistemic uncertainty

Existing Methods

- **Bayesian Neural Networks**:
  - ✗ Difficult to specify the prior
  - ✗ Exact Bayesian inference is intractable
- **Ensemble Methods**:
  - ✗ Resource intensive
- **Post-processing Calibration**:
  - Ex: Temperature scaling
  - ✗ Conflate aleatoric uncertainty and epistemic uncertainty

How Can We Better Quantify Uncertainty for DNNs

Our solution: Neural stochastic differential equation (SDE-Net)

✓ Able to model **both aleatoric** uncertainty and **epistemic** uncertainty

✓ **Efficient** and **straightforward** to implement
- No need to specify model priors and infer posterior distributions

✓ **Generically applicable** to both classification and regression tasks

SDE-Net: Equipping Deep Neural Networks with Uncertainty Estimates, Kong et al, ICML 2020

Modeling Uncertainty via Brownian Motion

- ResNet is a discretized form of ordinary differential equation [Chen et al., 2018]

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + f(\boldsymbol{x}_t, t) \longrightarrow d\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)dt$$

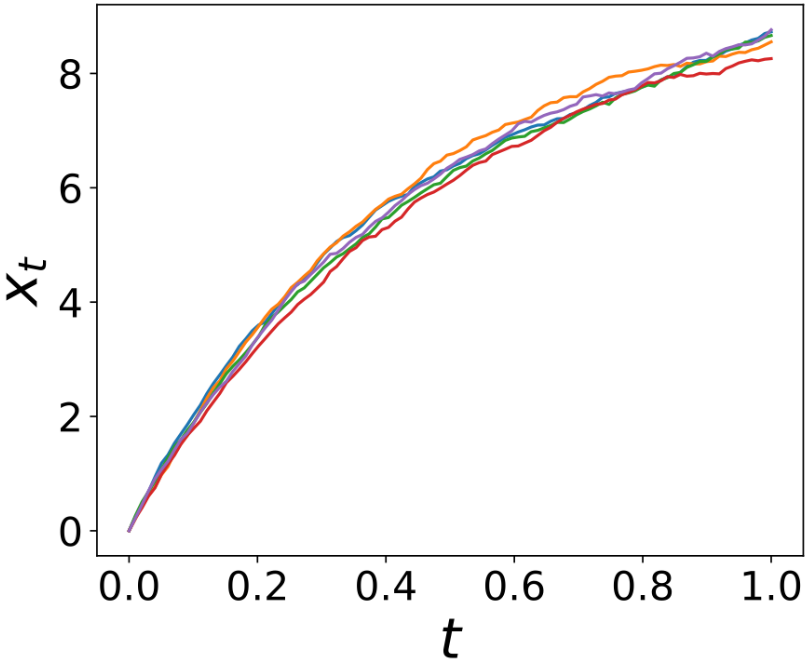
✘ Deterministic;
cannot represent uncertainty

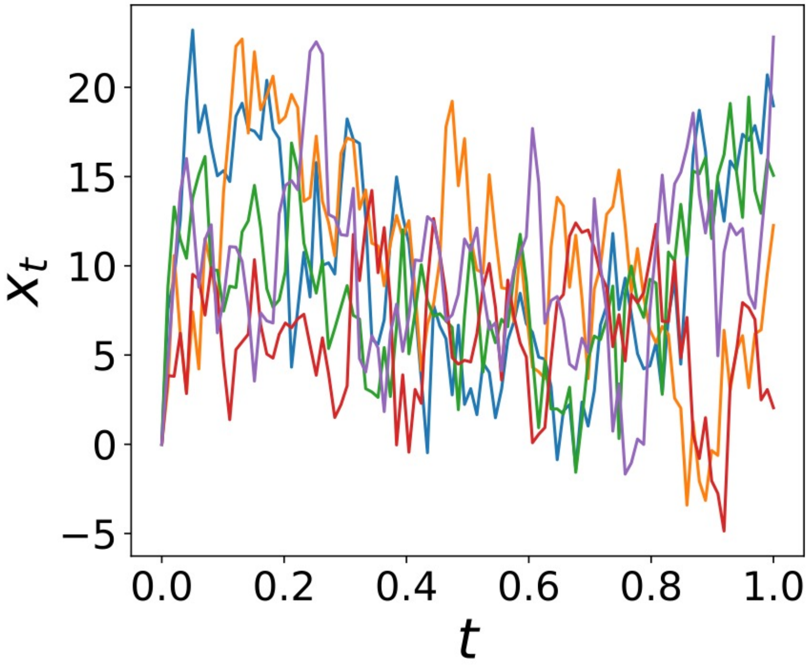
- We add a Brownian motion term to make it become a stochastic differential equation

$$d\boldsymbol{x}_t = f(\boldsymbol{x}_t, t)dt + g(\boldsymbol{x}_t, t)dW_t$$

✔ Stochastic; can represent uncertainty

Modeling Uncertainty via Brownian Motion



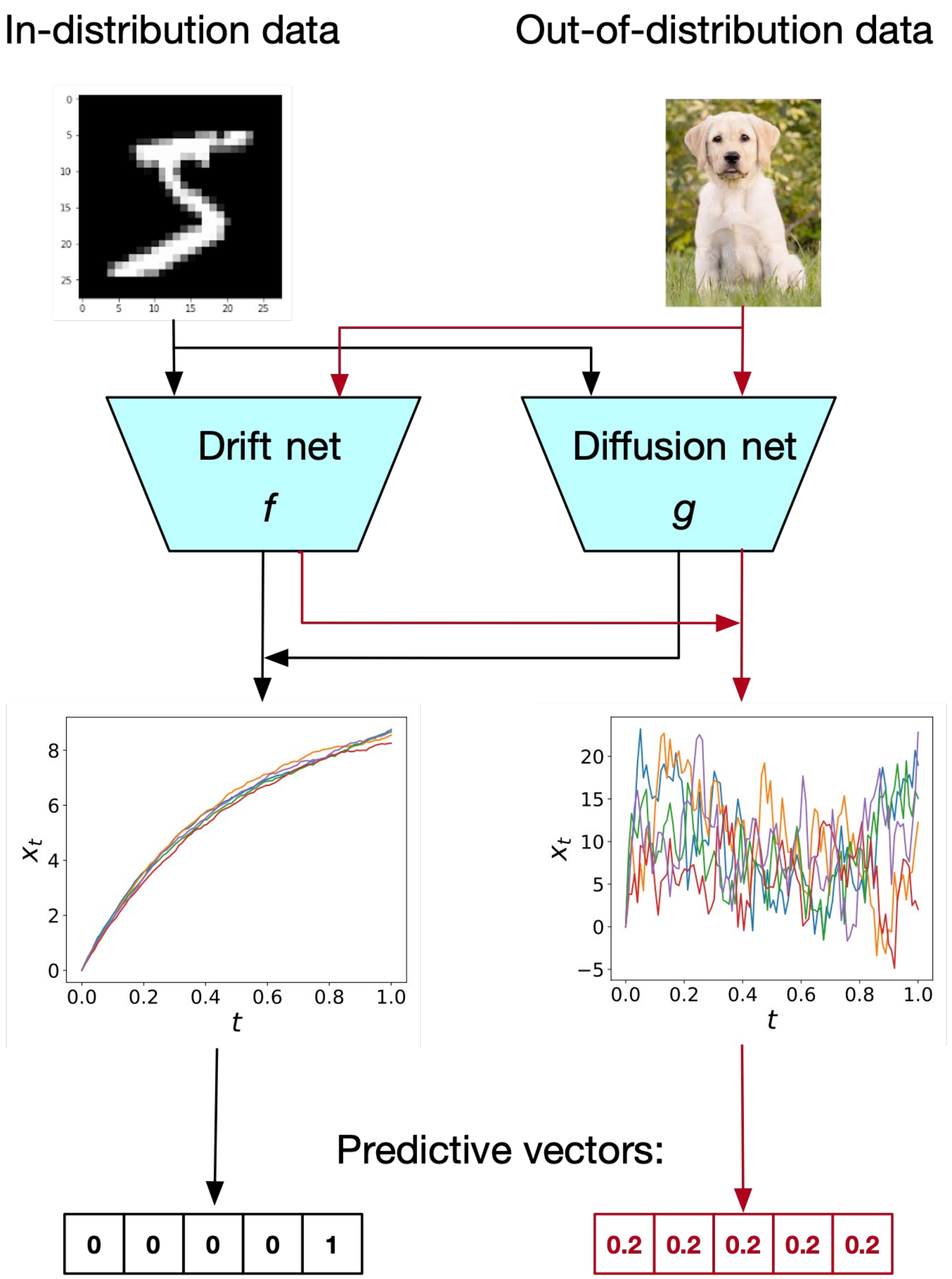


Trajectory variance → Proxy of uncertainty

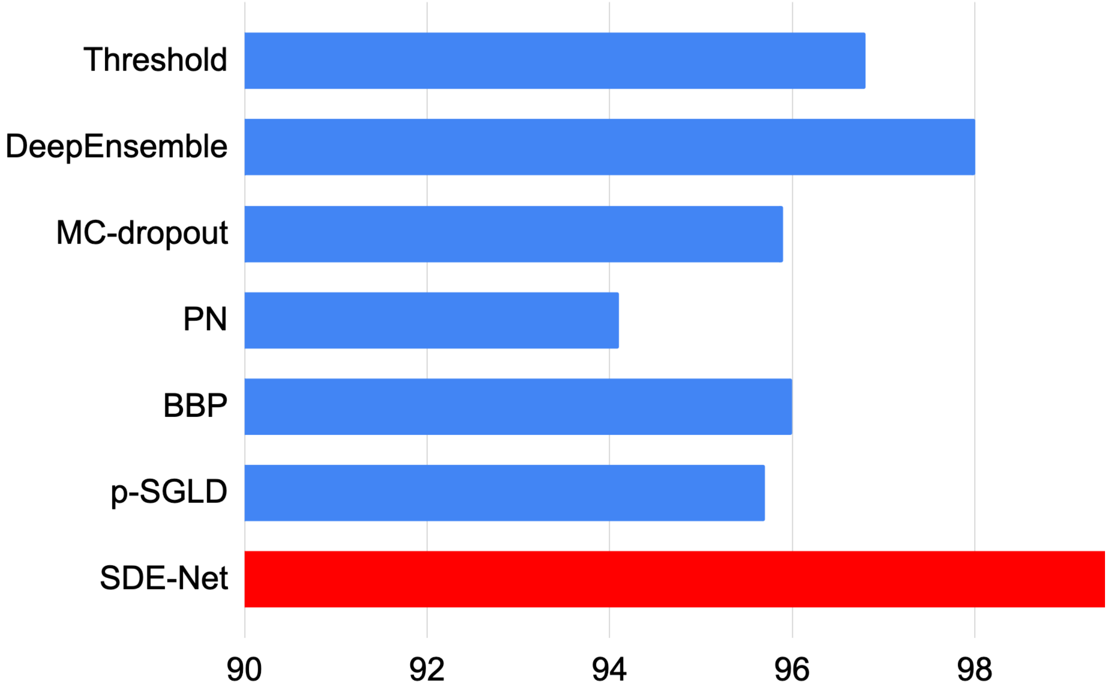
Model Components

- A Drift net *f*:
  - Control Predictive accuracy
  - Represents aleatoric uncertainty

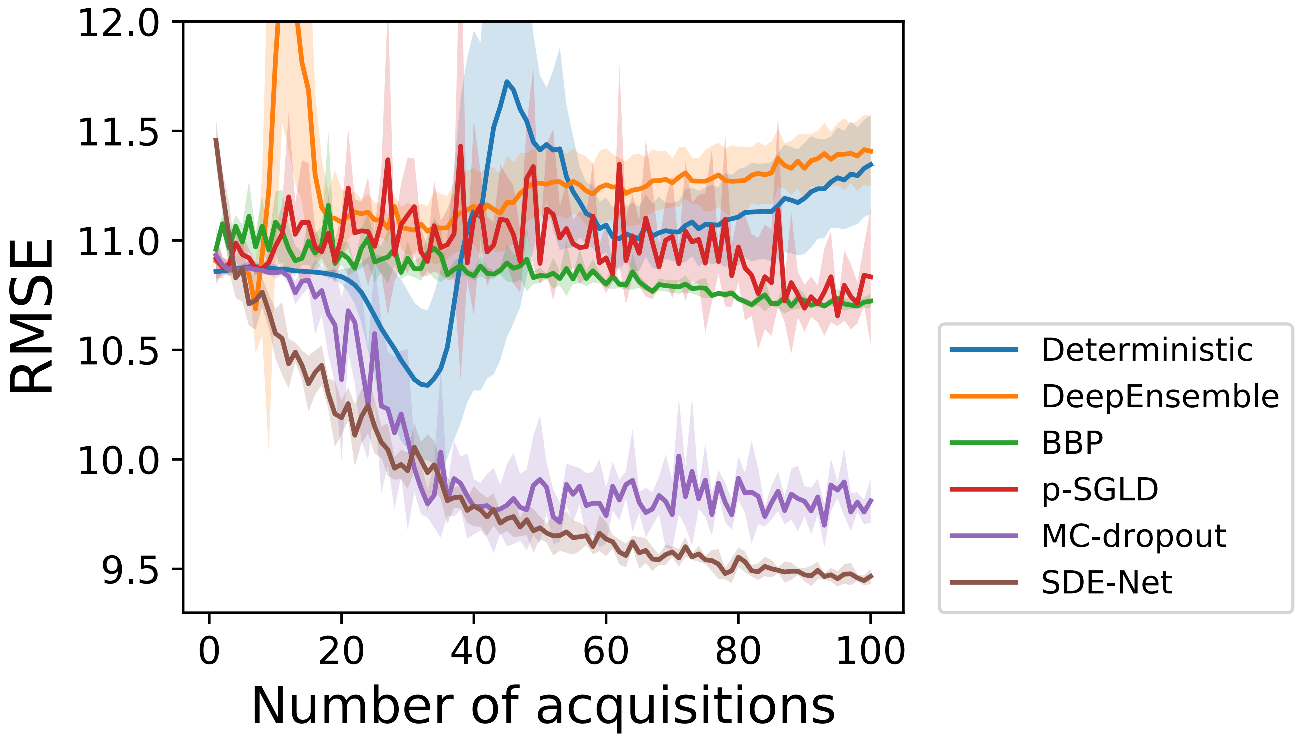
- A Diffusion net *g:*
  - Represents epistemic uncertainty



Experimental Results



OOD detection



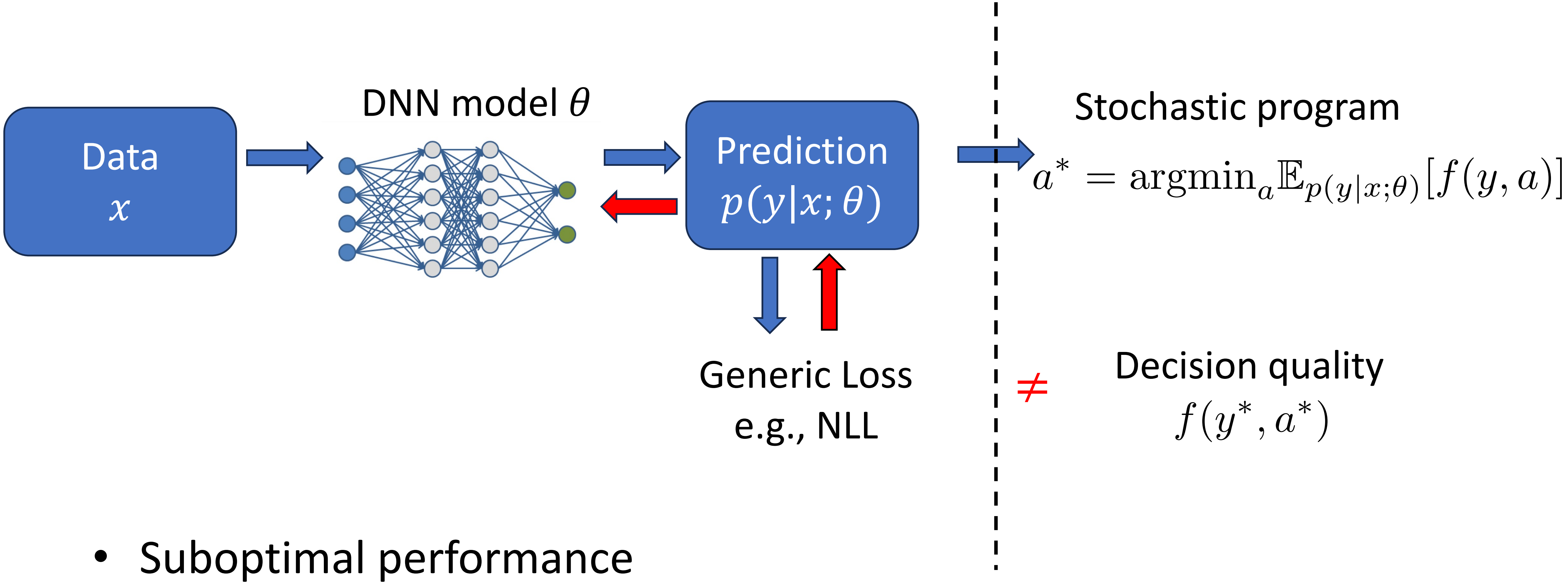
Active Learning

More Works on Uncertainty Quantification

- Language model:
  - Calibrated Language Model Fine-Tuning for In- and Out-of-Distribution Data, EMNLP 2020.
  - AcTune: Uncertainty-Aware Active Self-Training for Active Fine-Tuning of Pretrained Language Models, NAACL 2022.
- Time series forecasting:
  - When in Doubt: Neural Non-Parametric Uncertainty Quantification for Epidemic Forecasting, NeurIPS 2021.
  - CAMul: Calibrated and Accurate Multi-view Time-Series Forecasting, WWW 2021.
- Scientific applications:
  - Two Birds with One Stone: Enhancing Calibration and Interpretability with Graph Functional Neural Process, Preprint.
  - MUBen: Benchmarking the Uncertainty of Pre-Trained Models for Molecular Property Prediction, Preprint.

Challenge II: Integrating Prediction and Optimization

Two-Stage Pipeline

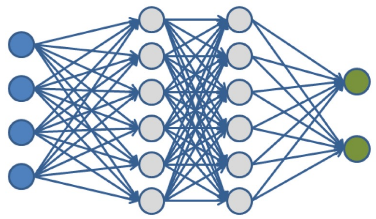


- Suboptimal performance
- Smaller predictive loss $\neq$ better decision quality

Decision-focused Learning (DFL)

Data
x

DNN model



Prediction
$p(y|x;\theta)$

Stochastic program

$$a^* = \mathrm{argmin}_a \mathbb{E}_{p(y|x;\theta)}[f(y,a)]$$

Decision quality

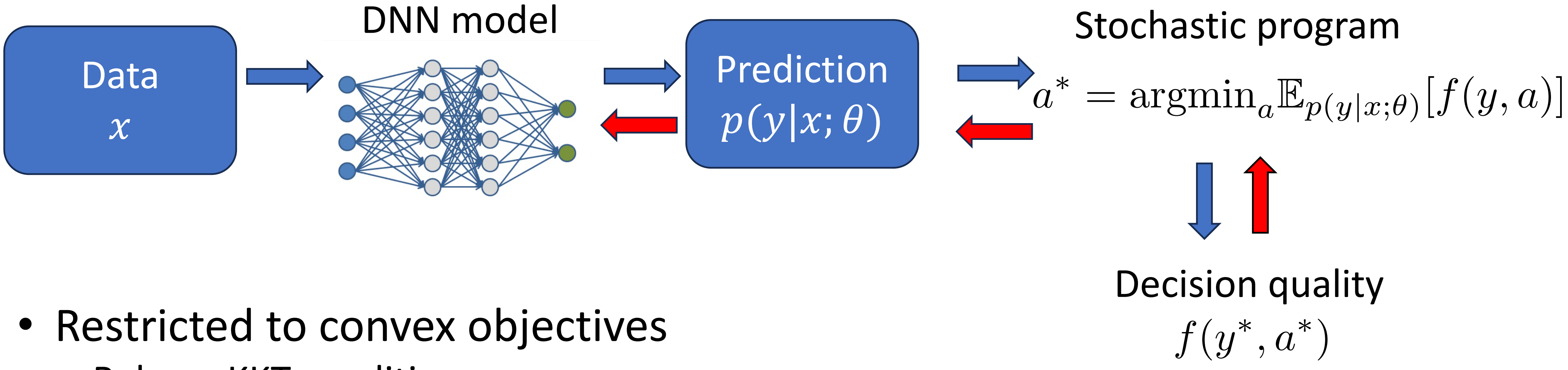
$$f(y^*, a^*)$$

$$\frac{\partial f(y^*, a^*)}{\partial \theta} = \frac{\partial f(y^*, a^*)}{\partial a^*} \boxed{\frac{\partial a^*}{\partial \theta}}$$

Compute the Jacobian through KKT conditions and implicit function theorem

- Imbue a differentiable optimization solver into the training pipeline

Decision-focused Learning (DFL)



- Restricted to convex objectives
  - Rely on KKT conditions
- Inefficient training
  - Need to solve and differentiate through the stochastic optimization problem at every training iteration

Our Solution: End-to-end Stochastic Optimization with Energy-based model (SO-EBM)

- General
  - Not restricted to convex objectives
  - Can be applied in a wide class of stochastic optimization problem

- Efficient
  - Eliminates the need of optimizing and differentiating through the optimization problem at every training iteration

End-to-end Stochastic Optimization with Energy-based model, Kong et al, NeurIPS 2022 **(oral)**

Connecting Features and Decisions with EBM

- Model the probability distribution over decisions conditioned on the features using energy-based parameterization

$$q(a|\mathbf{x};\theta) = \frac{\exp(-E(\mathbf{x},a;\theta))}{Z(\mathbf{x},\theta)}, \quad Z(\mathbf{x};\theta) = \int \exp(-E(\mathbf{x},a;\theta)da$$

Nonlinear regression function:
$E(\mathbf{x};\theta) : \mathbb{R}^D \to \mathbb{R}$

  - Direct connection between input features and decision variable
  - More tailored to downstream decision-making tasks

Connecting Features and Decisions with EBM

- Model the probability distribution over decisions conditioned on the features using energy-based parameterization

$$q(a|\mathbf{x};\theta) = \frac{\exp(-E(\mathbf{x},a;\theta))}{Z(\mathbf{x},\theta)}, \quad Z(\mathbf{x};\theta) = \int \exp(-E(\mathbf{x},a;\theta)da$$

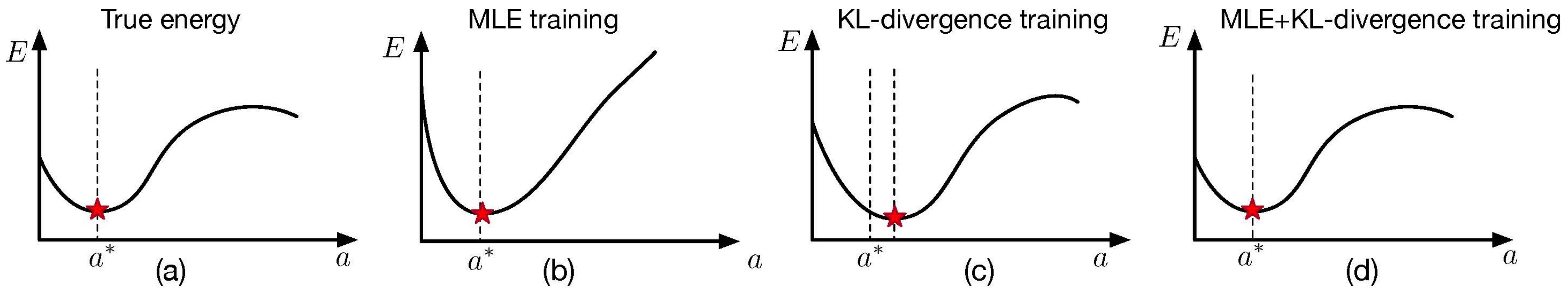
- Parameterize the energy function using the expected task loss

$$E(\mathbf{x},a;\theta) = \mathbb{E}_{p(y|\mathbf{x};\theta)} f(y,a)$$

  - Leverage prior knowledge of *f*, data efficient

Training Objective

Combination of MLE and distribution regularizer $\ell_{\text{Total}} = \ell_{\text{MLE}} + \lambda \ell_{\text{KL}}$



	(b)	(c)	(d)
Location of the optimum	✓	✗	✓
Overall energy shape	✗	✓	✓

Model Training

- How to deal with the partition function $Z(x; \theta)$?
- The gradient of the training loss:

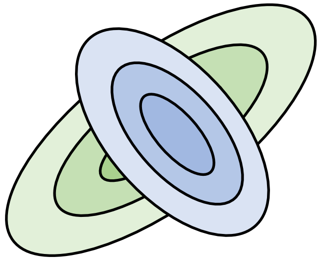
$$\begin{aligned}\frac{\partial \mathcal{L}_{\text{Total}}}{\partial \theta} =& \mathbb{E}_{(\mathbf{x},a^*)\sim\mathcal{D}_a}\left(\frac{\partial E(a^*,\mathbf{x};\theta)}{\partial\theta} - \mathbb{E}_{q(\tilde{a}|\mathbf{x};\theta)}\frac{\partial E(\tilde{a},\mathbf{x};\theta)}{\partial\theta}\right) \\ &+ \lambda\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left(\mathbb{E}_{p(\hat{a}|y)}\frac{\partial E(\hat{a},\mathbf{x};\theta)}{\partial\theta} - \mathbb{E}_{q(\tilde{a}|\mathbf{x};\theta)}\frac{\partial E(\tilde{a},\mathbf{x};\theta)}{\partial\theta}\right).\end{aligned}$$

Estimate the gradient by sampling from the model distribution $q(a|\mathbf{x};\theta)$

Model Training: Self-Normalized Importance Sampler

Sample from $q(a|\mathbf{x};\theta) = \dfrac{\exp(-E(\mathbf{x},a;\theta))}{Z(\mathbf{x};\theta)}$

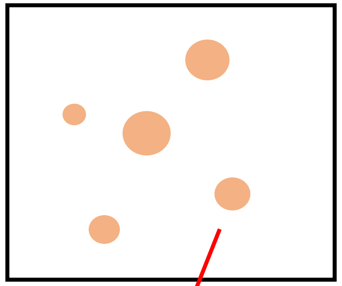
Step 1: Sample a set of *M* particle locations $\{a^m\}_{m=1}^{M}$



Proposal distribution: $\pi(a|\mathbf{x}) = \dfrac{1}{K}\sum_{i=1}^{K}\mathcal{N}(a^*;\sigma_k)$

Mixture of Gaussians located at the optimal decision

Step 2: Represent $q(a|\mathbf{x};\theta)$ as a weighted empirical distribution $w^m\delta(a-a^m)$

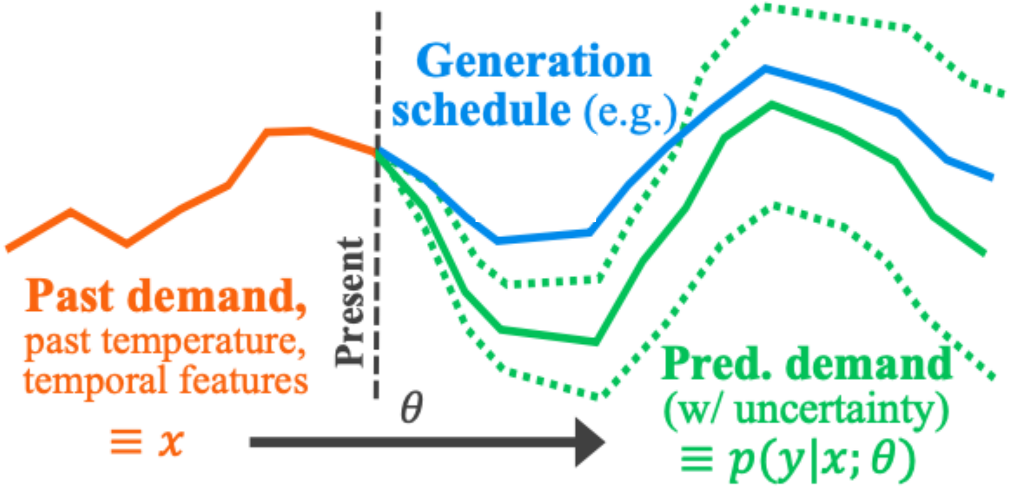


$$w^m = \frac{\exp(-E(a^m|\mathbf{x};\theta))/\pi(a^m|\mathbf{x})}{\sum_{m=1}^{M}\exp(-E(a^m|\mathbf{x};\theta))/\pi(a^m|\mathbf{x})}$$

- Fast sampling speed, only need to draw samples from a mixture of Gaussians

Experiments: Electricity Generator Scheduling

Forecasting: predict the electricity demand over the next 24 hours

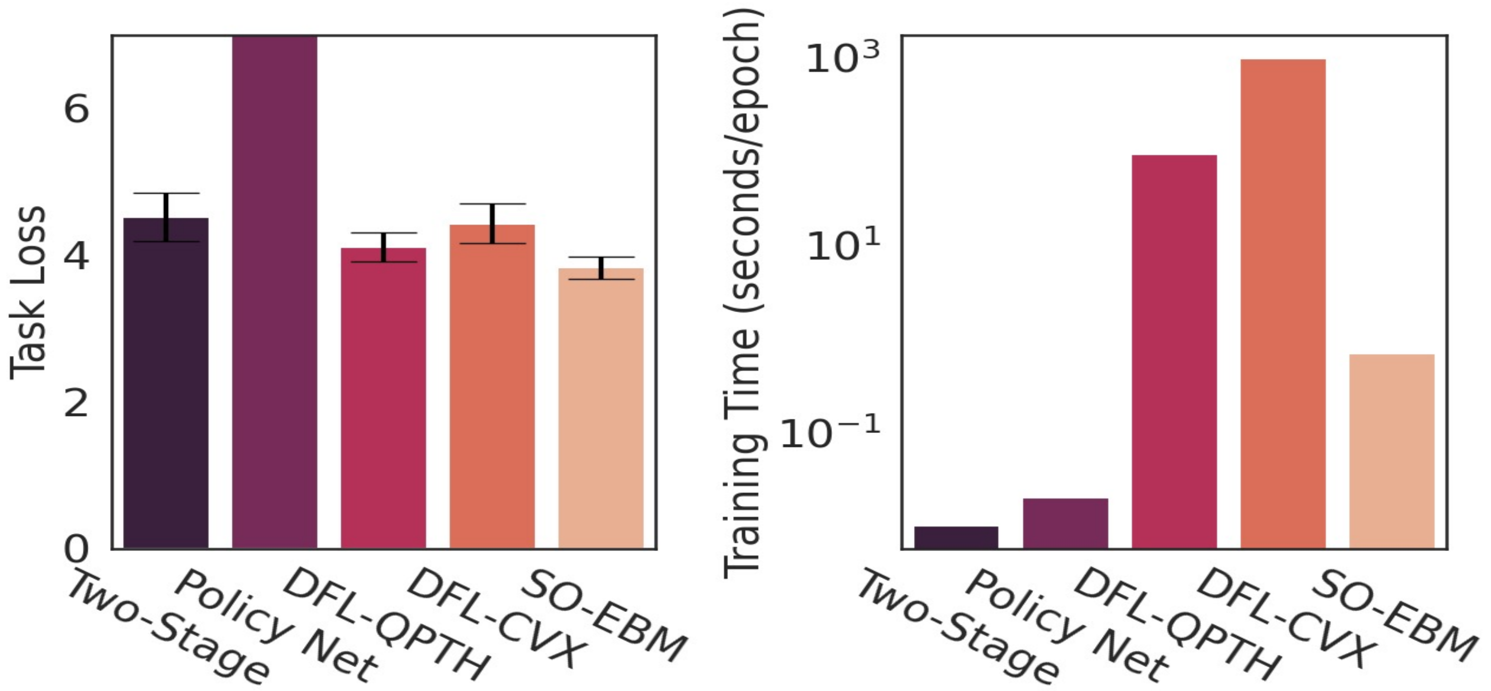


Optimization: decide how much electricity to generate

$$\text{minimize}_{a\in\mathbb{R}^{24}} \sum_{i=1}^{24} \mathbf{E}_{y\sim p(y|x;\theta)}[\gamma_s[y_i - a_i]_+ + \gamma_e[a_i - y_i]_+ + \frac{1}{2}(a_i - y_i)^2]$$

$$\text{s.t. } |a_i - a_{i-1}| \leq c_r \quad \forall i,$$

Experimental Results on Generator Scheduling

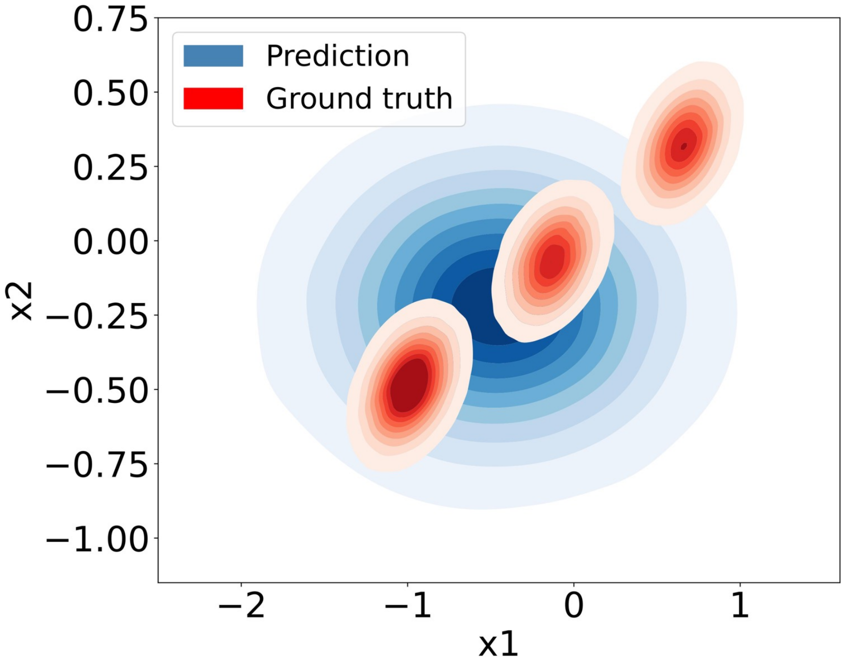


SO-EBM **improves** task loss by **7.3%** over the strongest baseline (DFL-QPTH) while being **136** times **faster** in training

Two More Bottlenecks...

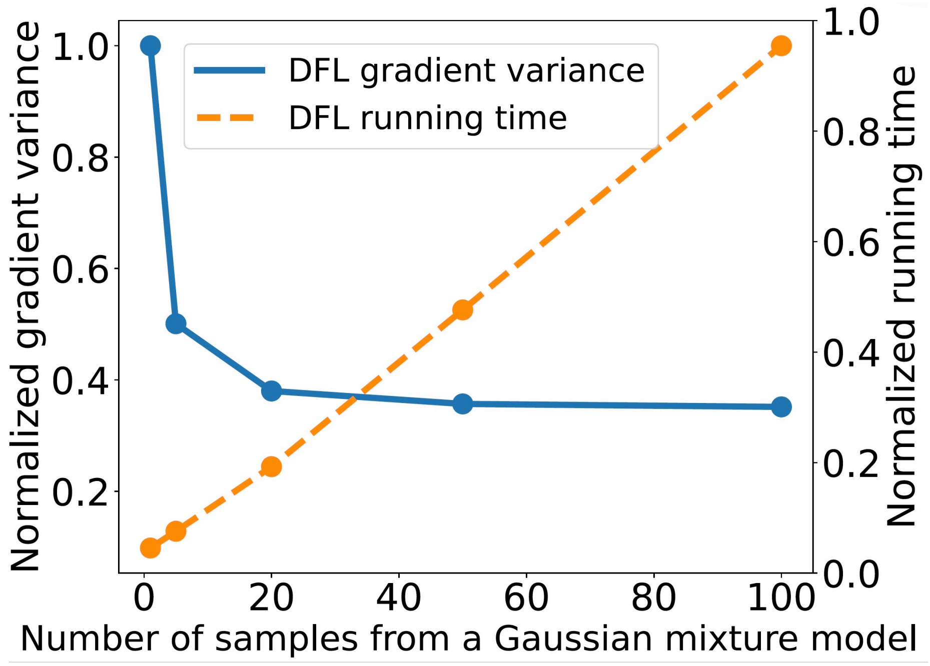
1. Model mismatach error:

$p(y|\mathbf{x})$ can be highly complicated



2. Sample average approximation error:

$$\mathbb{E}_{p(y|\mathbf{x})}[f(y,a)] \approx \frac{1}{M}\sum_{i}^{M} f(y_i,a)$$



DF2: Distribution-Free Decision-Focused Learning, Kong et al, arxiv

Distribution-Free Training Objective

- Cornerstone: $g(\mathbf{x}, a) = \mathbb{E}_{p(y|\mathbf{x})}[f(y, \mathbf{a})]$ is only a function of *x* and *a*
- Training objective:

$$g^*(\mathbf{x}, a) = \text{argmin}_g \mathbb{E}_a \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[g(\mathbf{x}, a) - f(y, a)]^2.$$

Proposition:

$$\text{MSE}_{\text{test}} = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(\mathbf{x}, a) - \mathbb{E}_{p(y|\mathbf{x})}[f(y, a)]\right)^2\right]}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(\mathbf{x}, a) - \mathbb{E}_{\mathcal{D}}[g^*_{\mathcal{D}}(\mathbf{x}, a)]\right)^2\right]}_{\text{Variance}}$$

MSE = 0 $\Longrightarrow$ $g^*(\mathbf{x}, a) = \mathbb{E}_{p(y|\mathbf{x})}[f(y, a)]$

Distribution-Free Training Objective

- Cornerstone: $g(\mathbf{x}, a) = \mathbb{E}_{p(y|\mathbf{x})}[f(y, \mathbf{a})]$ is only a function of *x* and *a*
- Training objective:

$$g^*(\mathbf{x}, a) = \mathrm{argmin}_g \mathbb{E}_a \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}[g(\mathbf{x}, a) - f(y, a)]^2.$$

Proposition:

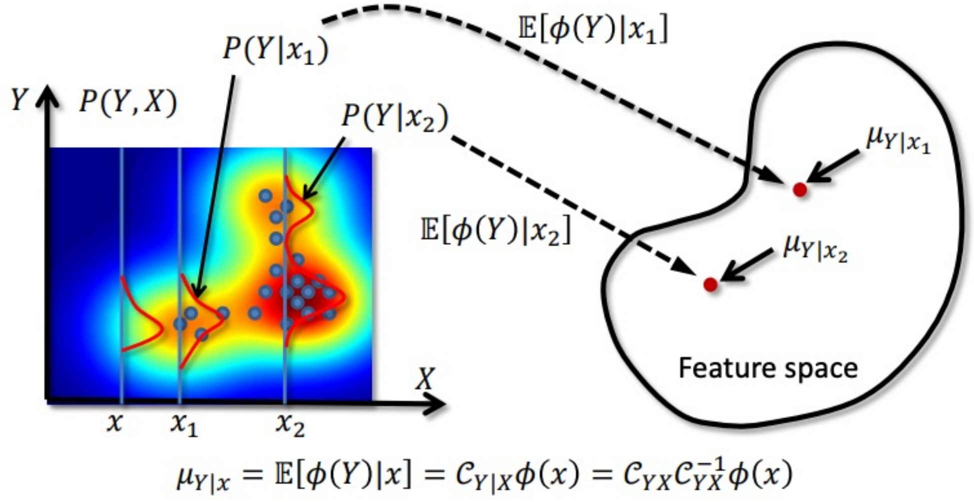
$$\mathrm{MSE}_{\mathrm{test}} = \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(\mathbf{x}, a) - \mathbb{E}_{p(y|\mathbf{x})}[f(y, a)]\right)^2\right]}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(g^*_{\mathcal{D}}(\mathbf{x}, a) - \mathbb{E}_{\mathcal{D}}[g^*_{\mathcal{D}}(\mathbf{x}, a)]\right)^2\right]}_{\text{Variance}}$$

How to reduce this term?

Reduced with more data

Distribution-based Parameterization

- Conditional Mean Embedding



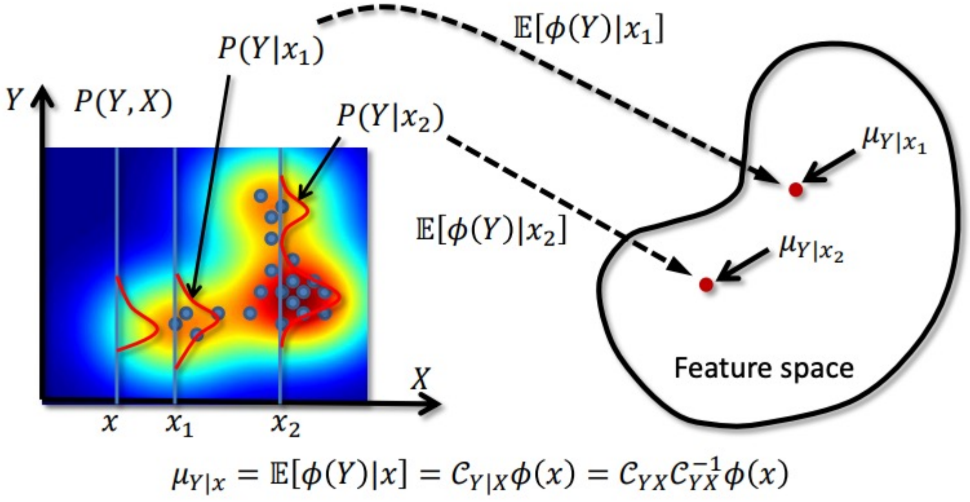
[Song et al, ICML 2013]

- Estimate the expectation of a function in RKHS
- Works well for high-dimensional problem

$$\mathbb{E}_{p(y|\mathbf{x})}[f(y,a)] = \cdots = \sum_{s=1}^{S} \underbrace{\boxed{\beta_s(\mathbf{x})}}_{\text{Real-valued weight}} f(y_s, a)$$

Distribution-based Parameterization

- Conditional Mean Embedding



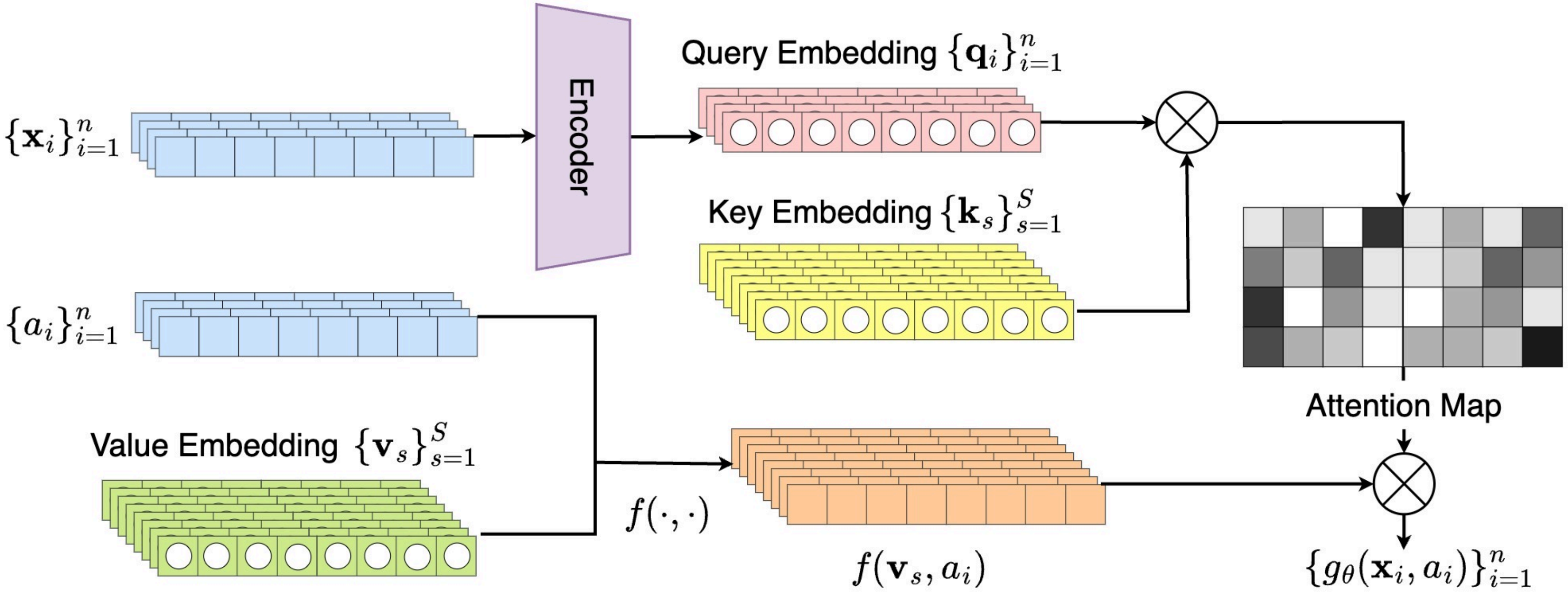
[Song et al, ICML 2013]

- Estimate the expectation in RKHS
- Works well for high-dimensional problem

$$\mathbb{E}_{p(y|\mathbf{x})}[f(y,a)] = \cdots = \sum_{s=1}^{S} \underbrace{\beta_s(\mathbf{x})}_{\text{Real-valued weight}} f(y_s, a)$$

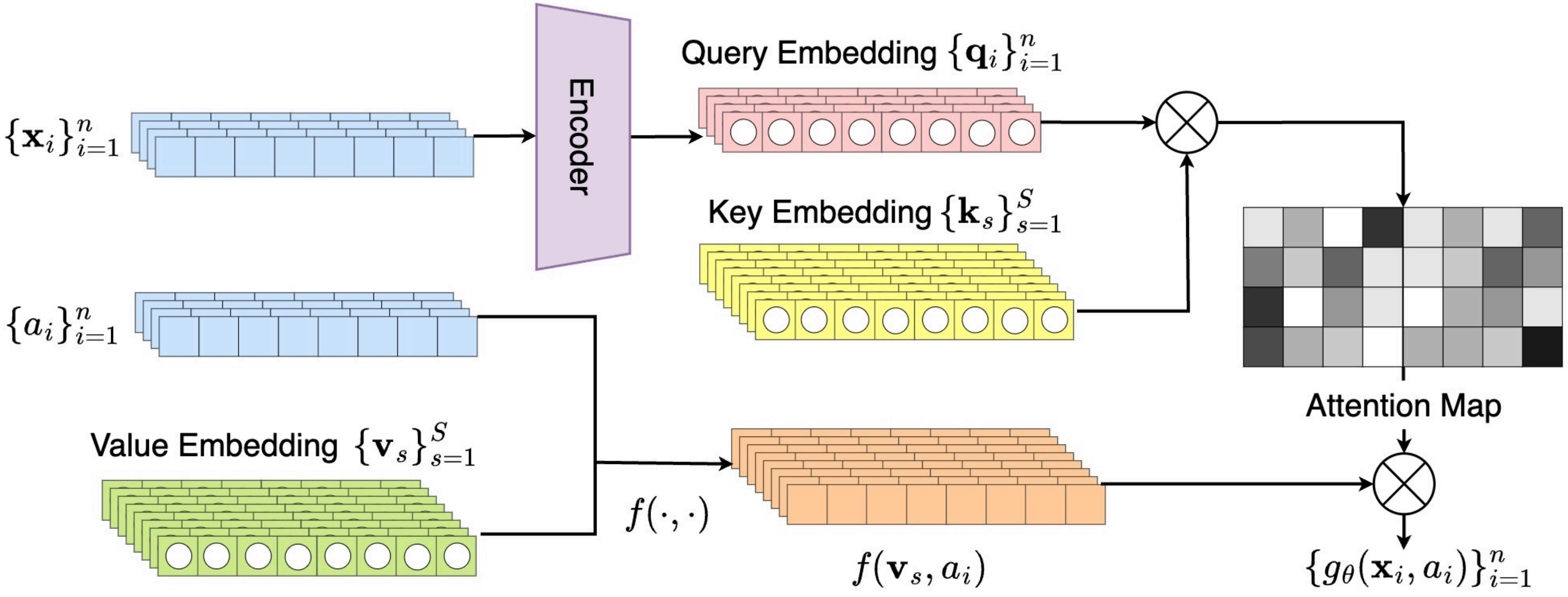
Attention

Attention-based Network Architecture



$$g(\mathbf{x}, a) = \text{Softmax}\left(\left[\frac{\mathbf{q}(\mathbf{x})^\top \mathbf{k}_1}{\sqrt{d}}, \cdots, \frac{\mathbf{q}(\mathbf{x})^\top \mathbf{k}_S}{\sqrt{d}}\right]\right)^\top [f(\mathbf{v}_1, a), \cdots, f(\mathbf{v}_S, a)]$$

Attention-based Network Architecture



Proposition:

$g(\mathbf{x}, a) = \mathbb{E}_{\hat{p}_{\mathcal{R}}(y|\mathbf{x})}[f(y, \mathbf{a})]$, where $\hat{p}_{\mathcal{R}}(y|\mathbf{x})$ is a parameterization restriction of $p(y|\mathbf{x})$

Ensure the learned function is within the true model class

Vaccine Distribution for COVID-19

Forecasting:

Historical mobility flow

Future mobility flow

Vaccine Distribution for COVID-19

Forecasting:

Historical mobility flow

Future mobility flow

Optimization:

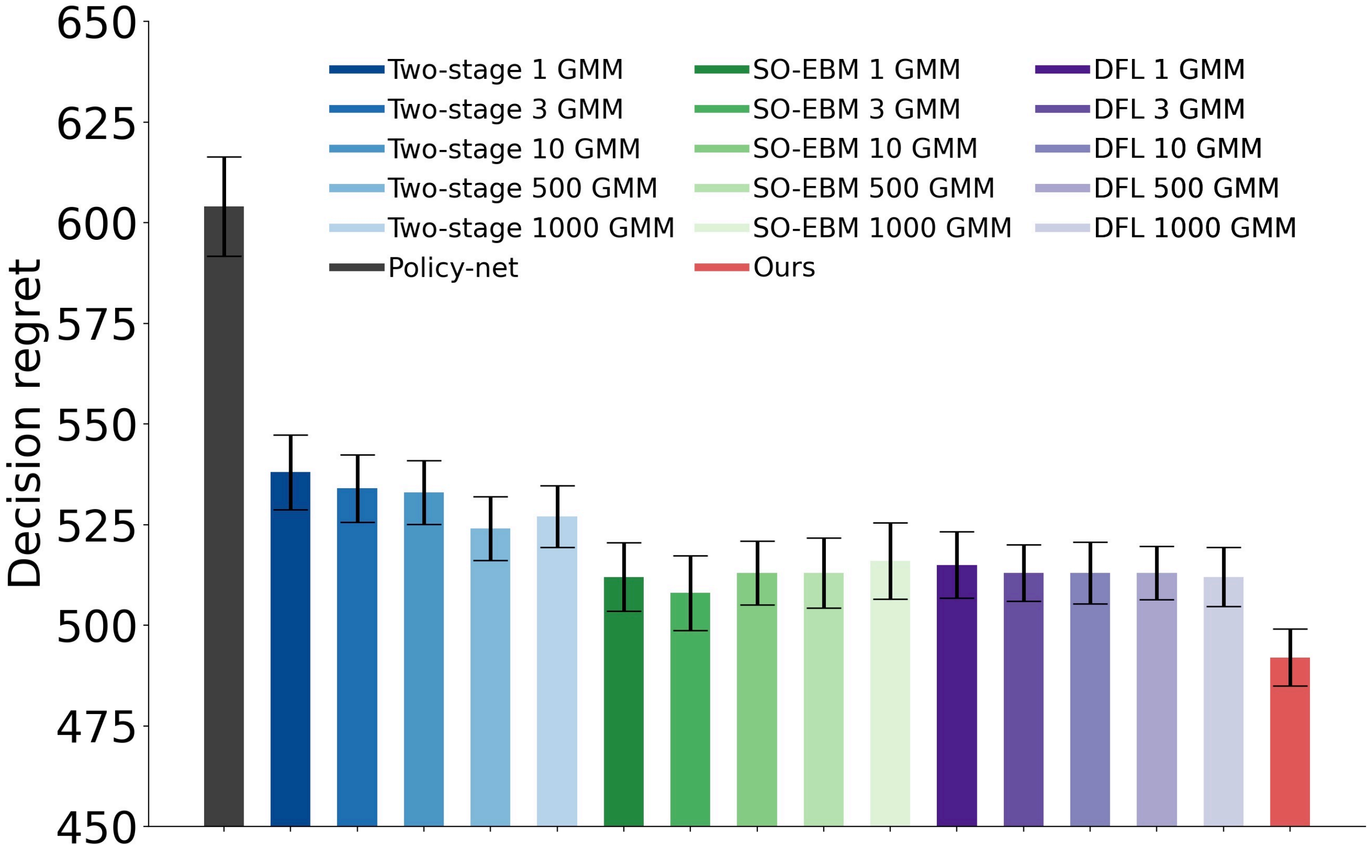
+

Meta-Population SEIR model

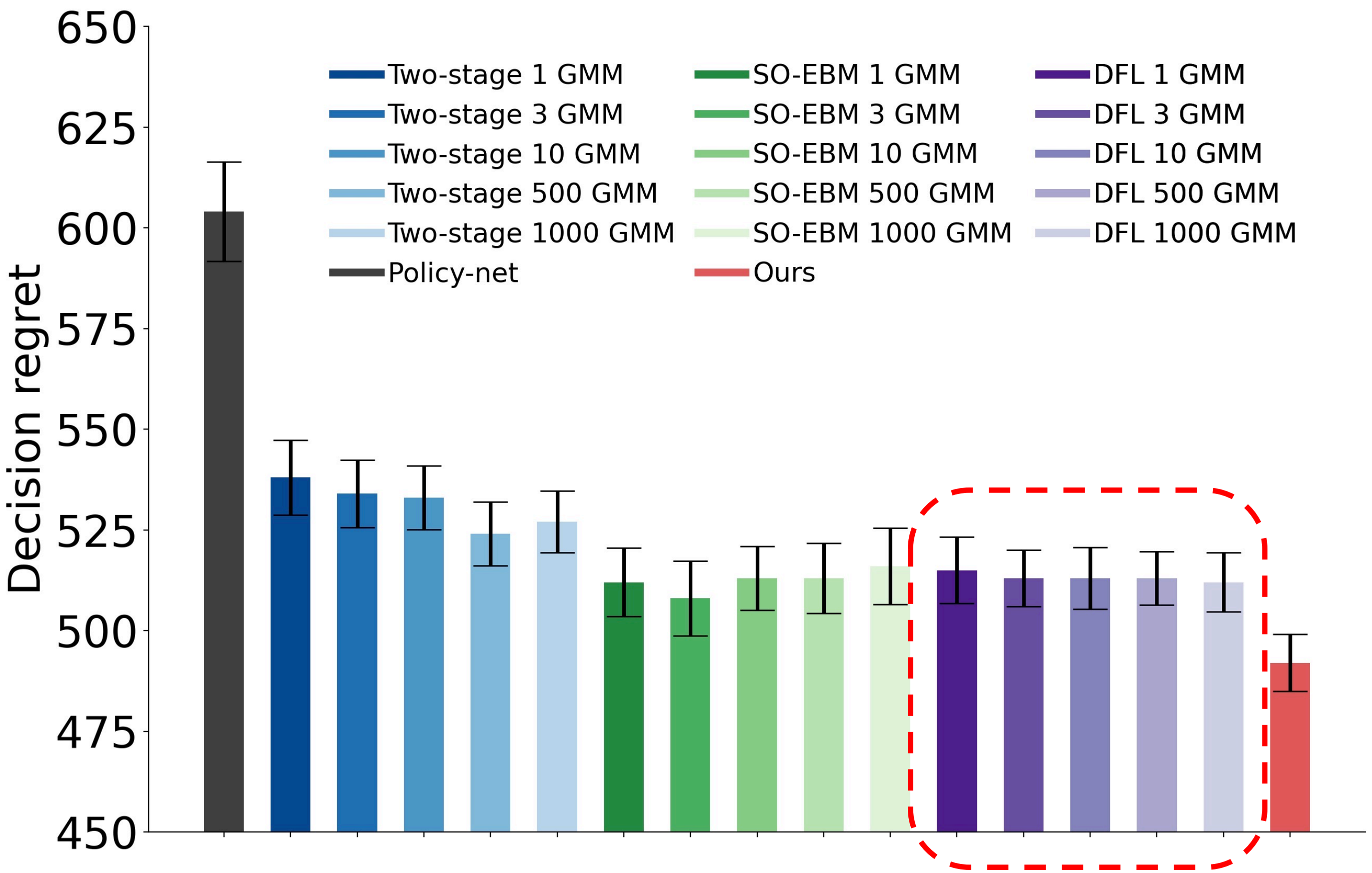
Minimize # infects

Decision variable: Vaccine distribution

Experimental Results on Vaccine Distribution



Experimental Results on Vaccine Distribution



Increasing complexity of distribution $\neq$ smaller decision regret

Thanks!

SDE-Net: Loss Function

The objective for training SDE-Net:

Low diffusion for in-distribution data

High diffusion for out-of-distribution data

$$\boxed{\min_{\boldsymbol{\theta}_f} \mathrm{E}_{\boldsymbol{x}_0 \sim P_{\text{train}}} \mathrm{E}(L(\boldsymbol{x}_T))} + \boxed{\min_{\boldsymbol{\theta}_g} \mathrm{E}_{\boldsymbol{x}_0 \sim P_{\text{train}}} g(\boldsymbol{x}_0; \boldsymbol{\theta}_g)} + \boxed{\max_{\boldsymbol{\theta}_g} \mathrm{E}_{\boldsymbol{x}_0 \sim P_{\text{OOD}}} g(\boldsymbol{x}_0; \boldsymbol{\theta}_g)}$$

$$s.t. \quad d\boldsymbol{x}_t = \underbrace{f(\boldsymbol{x}_t, t; \boldsymbol{\theta}_f)}_{\text{drift neural net}} dt + \underbrace{g(\boldsymbol{x}_0; \boldsymbol{\theta}_g)}_{\text{diffusion neural net}} dW_t,$$

L( ): loss function dependent on the task

- Generate pseudo out-of-distribution data by $\tilde{\boldsymbol{x}}_0 = \boldsymbol{x}_0 + \boldsymbol{\epsilon}$
- Parameters shared by each layer