GANdy: Supervised Uncertainty Prediction

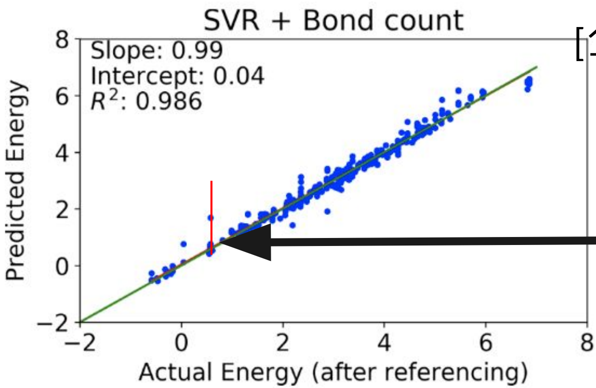
Sam Tetef, Steven Fang, Kyle Moskowitz, Yuxuan Ren, Evan Komp

eScience Institute
ADVANCING DATA-INTENSIVE DISCOVERY IN ALL FIELDS

CLEAN ENERGY INSTITUTE
UNIVERSITY of WASHINGTON

Motivation

Supervised machine learning **typically produces deterministic predictions**:



[1]

Some systems are **very sensitive to small changes** eg. chemical kinetics:

Δ(rate constant) ~10% -> Δ(concentration) ~10^N%

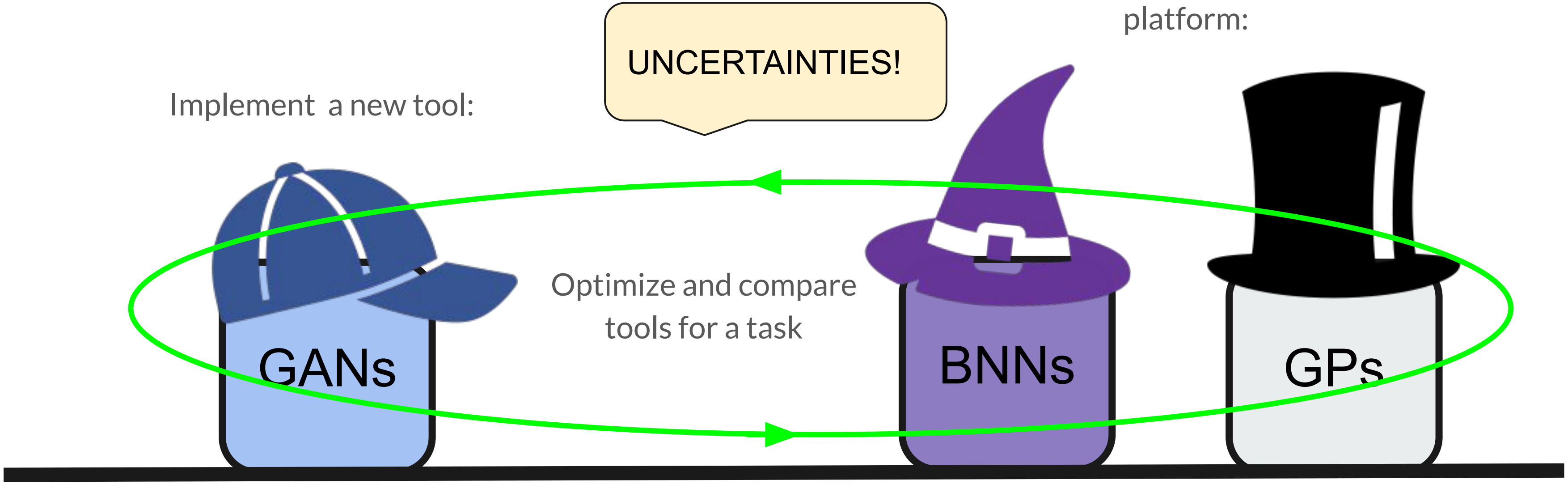
To inform design, we want **uncertainty/potential error** such that predictions can be used with confidence.

[1] A. J. Chowdhury, W. Yang, E. Walker, O. Mamun, A. Heyden, and G. A. Terejanu, “Prediction of Adsorption Energies for Chemical Species on Metal Catalyst Surfaces Using Machine Learning,” *J. Phys. Chem. C*, vol. 122, no. 49, pp. 28142–28150, 2018, doi: 10.1021/acs.jpcc.8b09284.

Package Purpose

Incorporate available tools into the same platform:

Implement a new tool:



Design Strategy

Repository:
https://github.com/GANdy-team/GANdy

We followed a fairly strict **test driven design of the package.**

Pro: Organized development, foundation for future work

Con: Not enough time to train/optimize heavy ML models

GANdy

build passing | coverage 90%

Current Functionality:

- [x] Instantialize, train, and use uncertainty models
  - [x] Gaussian Processes
  - [x] Bayesian Neural Networks
  - [x] uncertainty GANs
- [x] Judge the quality of produced uncertainties with uncertainty metrics
- [ ] Automated comparison of model structures
- [ ] Model optimization to uncertainty metrics

Package format/Code flow

`GANdy`

`models`

Uncertainty models:

- GPs
- GANs
- BNNs

Single framework for all models. Create, train, and use models of different types.

`quality_est`

Metrics:

- Traditional
- Uncertainty

Data generation

- Creation of synthesized data

Tools used to evaluate the models that we create.

`optimization`

Hyperparameter optimization:

- Grid and guided searches
- Pruning of unpromising models
- Validation schemes

Tools for searching and producing capable models.

Traditional Uncertainty Models

Gaussian Process (GP):

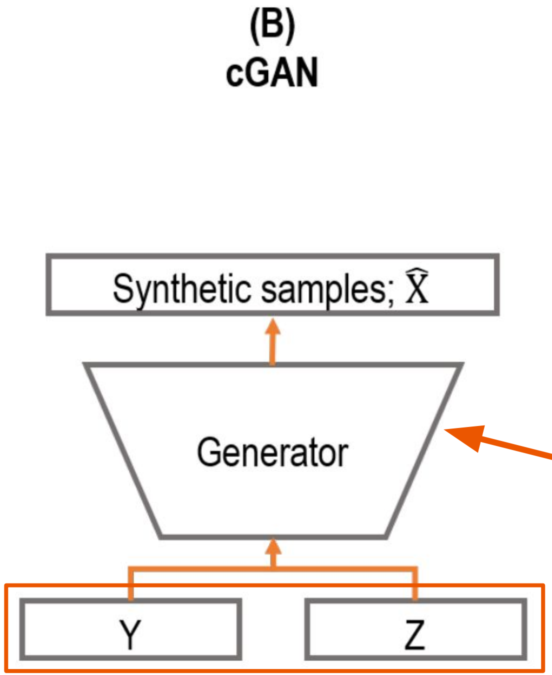
1. Pros
   a. Inherent Gaussian uncertainties -> standard
   b. Great for smooth and linear functions
2. Cons
   a. Expensive computationally: O(n^3)

Bayes Neural Network (BNN):

1. Pros
   a. More flexible for nonlinear functions
2. Cons
   a. The uncertainties aren't as good to interpret and rely on optimal network hyperparameters

Novel GANs - Basics and Advances

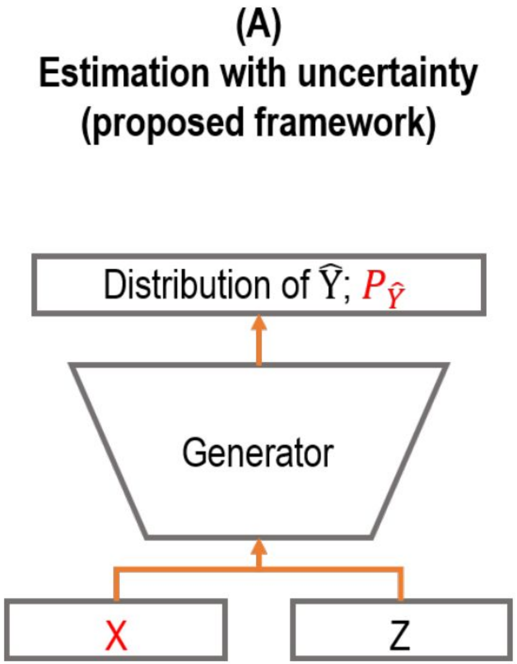
GANs



Traditional cGAN generator

[2] M. Lee and J. Seok, "Estimation with Uncertainty via Conditional Generative Adversarial Networks." ArXiv 2007.00334v1

Novel GANs - Basics and Advances

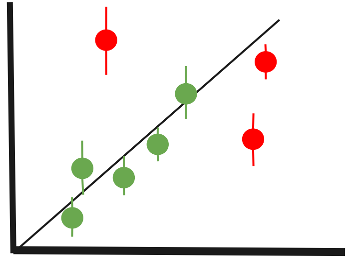


Challenges:

1. GAN loss?
2. Metric to evaluate uncertainties?

[2] M. Lee and J. Seok, “Estimation with Uncertainty via Conditional Generative Adversarial Networks.” ArXiv 2007.00334v1

Evaluating Uncertainty Predictions



__Traditional ML metrics:__

- Classification: Accuracy and F1 score
- Regression: MSE and RMSE

However, traditional metrics *do not* evaluate quality of uncertainty predictions!

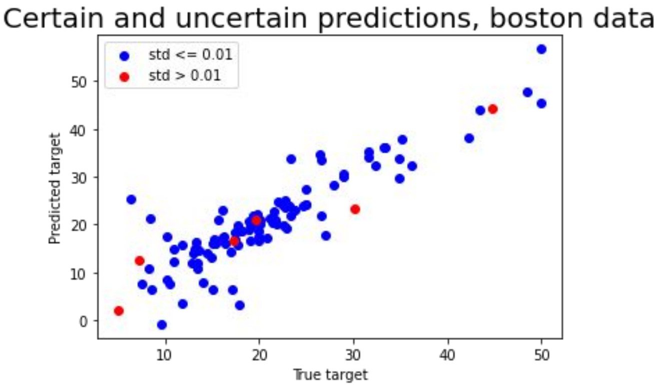
__Uncertainty Metric:__

Uncertainty Coverage Probability (UCP): Assess how often the uncertainty intervals capture the true value

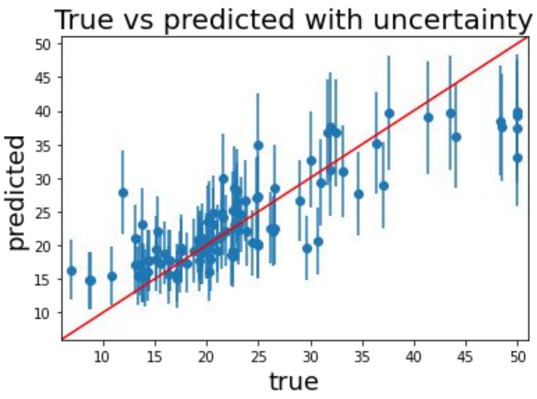
$$UCP = \frac{1}{N}\sum_{i=1}^{N} f_i \ , \qquad where\ f_i = \begin{cases} 1, & l(x_i) \le y_i \le u(x_i) \\ 0, & otherwise \end{cases}$$

Demo: Creating and training models

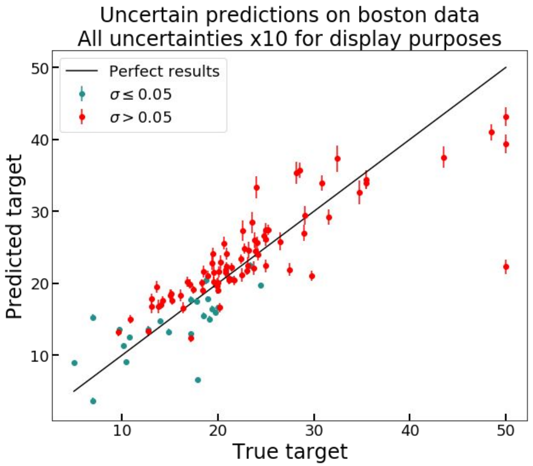
GP



BNN



GAN



Boston dataset

Synthesizing a test case

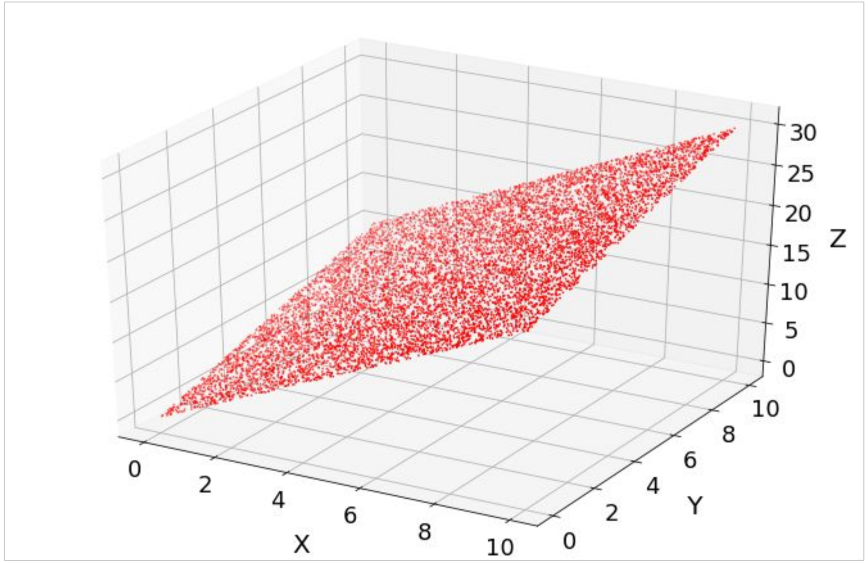
Analytical function

```
def generate_analytical_data(to_csv=True):

    x1 = np.random.uniform(0, 10, 10000)
    x2 = np.random.uniform(0, 10, 10000)
    mu = 0
    sigma = (x1 + x2) / 10

    def f(x1, x2):
        f_data = 2*x1 + x2
        return f_data

    def g(x1, x2):
        g_data = np.random.normal(mu, np.abs(sigma), 10000)
        return g_data

    noise = g(x1, x2)
    y = f(x1, x2) + noise

    gen_data = pd.DataFrame({'X1': x1, 'X2': x2, 'Y': y})

    if to_csv:
        gen_data.to_csv("analytical_data.csv", index=False, sep=',')
        # read in using gen_data = pd.read_csv("analytical_data.csv")
    return gen_data, noise
```

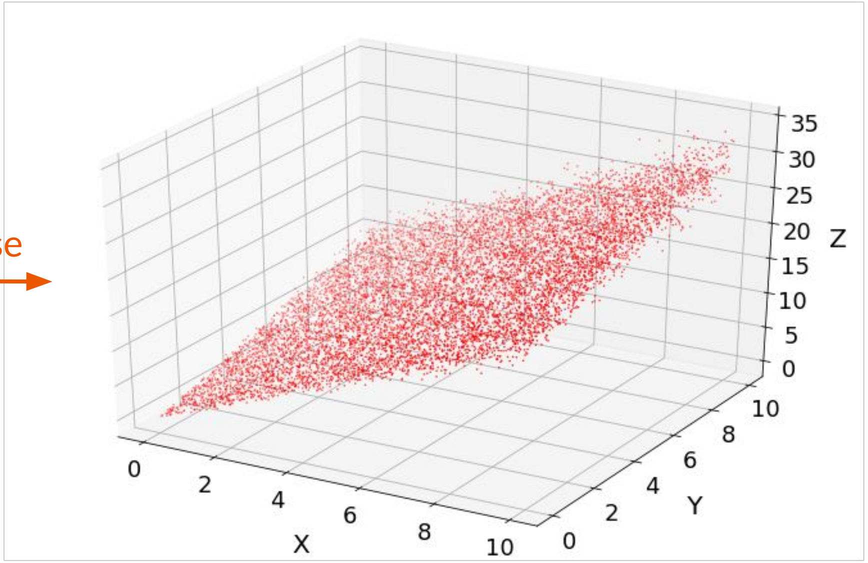
Result

	X1	X2	Y
0	855.159973	629.478184	-291.623828
1	303.518750	588.129967	717.112047
2	729.628137	604.606386	-770.744299
3	592.880184	398.744920	-1119.892617
4	740.079199	150.850121	-555.732907
...	...	...	...
995	275.839186	503.174366	-48.447220
996	885.653708	617.742444	-114.904246
997	133.715002	806.957340	-395.812813
998	293.900305	606.741015	422.247968
999	642.500546	719.364994	-44.991404

Synthesizing a test case

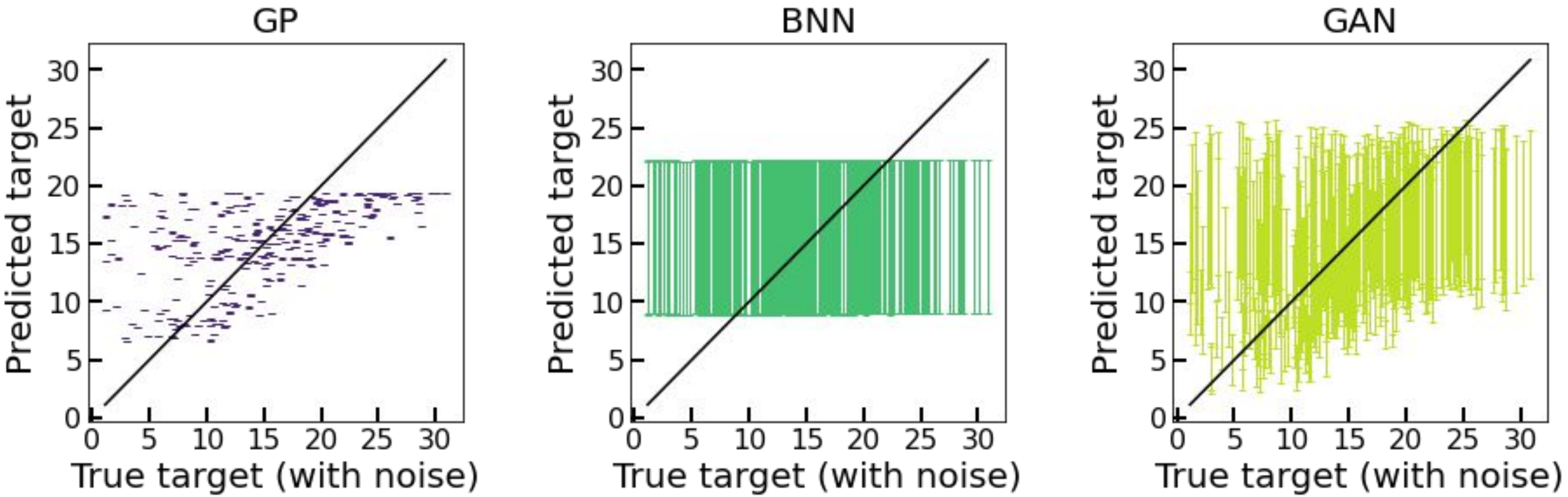


Noise



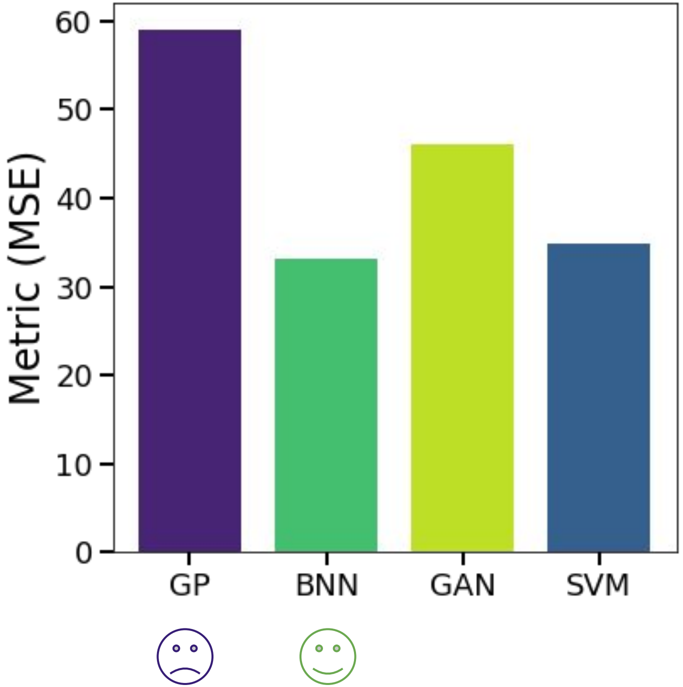
Training...Training...Training...Training...
Training...Training...Training...Training...
Training...Training...Training...Training...
Training...Training...Training...Training...
Training...Training...Training...Training...
Training...Training...Training...Training...
Training...Training...Training...Training...

Demo: Choosing a model



Demo:
Choosing a model

How good were
the predictions?



Comparing to
SVM, to serve
as a benchmark

$$UCP = \frac{1}{N}\sum_{i=1}^{N} f_i \ , \qquad where\ f_i = \begin{cases} 1, & l(x_i) \leq y_i \leq u(x_i) \\ 0, & otherwise \end{cases}$$

Demo: Choosing a model

How good were the uncertainties?

Model	UCP
GP	0.0
BNN	0.627
GAN	0.643

The GAN and BNN seem to be performing equivalently, but none of the models are predicting the uncertainties very well.

Future step: Optimize models and hyperparameters

Future Steps

1. Determine the best metric to evaluate how well the models estimate uncertainties
2. Optimize the models
   a. Hyperparameter optimization
3. Choose the best model
   a. Choose a model that best represents uncertainties using the above uncertainty metric
4. Compare model uncertainties on a real dataset (QM9)

Acknowledgments

We would like to DIRECT our thanks to the program for this opportunity along with Dave Beck, Stephanie Valleau, Sabiha Rustam, and Nisarg Joshi for their incredible teaching and advice. We would also like to thank the eScience and Clean Energy Institutes for their support of this program.

[1] A. J. Chowdhury, W. Yang, E. Walker, O. Mamun, A. Heyden, and G. A. Terejanu, "Prediction of Adsorption Energies for Chemical Species on Metal Catalyst Surfaces Using Machine Learning," J. Phys. Chem. C, vol. 122, no. 49, pp. 28142–28150, 2018, doi: 10.1021/acs.jpcc.8b09284.

[2] M. Lee and J. Seok, "Estimation with Uncertainty via Conditional Generative Adversarial Networks." ArXiv 2007.00334v1

The GAN was implemented using tensorflow with `Keras` and `deepchem`. The BNN was implemented `tensorflow` and `Keras` as well. The GP utilized `sklearn`'s GP class. The SVM regressor was implemented with `sklearn`.

The hyperparameter optimization utilized `Optuna`, which can be found at https://github.com/optuna/optuna