ALPhA
DAVIDSON COLLEGE

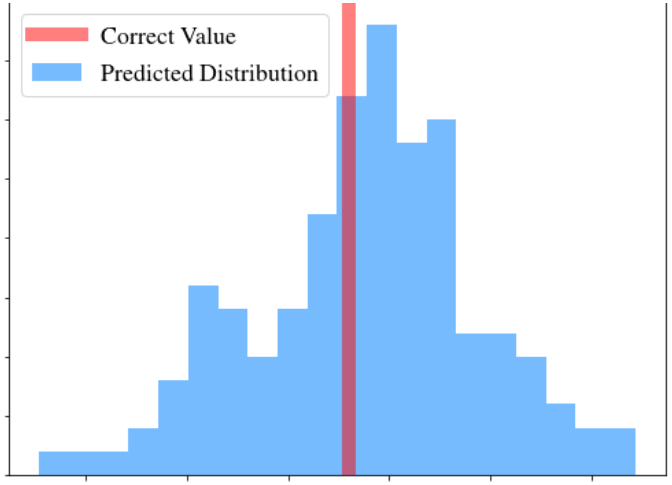
Bayesian Neural Networks for Fast Predictions from High Dimensional Theories

Braden Kronheim[1], Michelle Kuchera[1], Harrison Prosper[2], and Alexander Karbo[1]

[1]Davidson College, [2]Florida State University

Motivation

- In order to make more complete predictions from expressive Beyond the Standard Model Theories, it would be useful to have a method of generating much faster predictions.



Bayesian Neural Networks

- Deep neural networks are shown to be effective at solving regression problems.
- However, the common approaches to training these models using back propagation do not provide a reliable uncertainty estimate.
- By assigning prior distributions to the network parameters, we can obtain a posterior distribution of possible networks using Bayes' theorem given the training data.
  - This means that instead of a single prediction, we will get a distribution of predictions.
  - The distribution of these predictions measure the uncertainty in a prediction.

Bayesian Approach

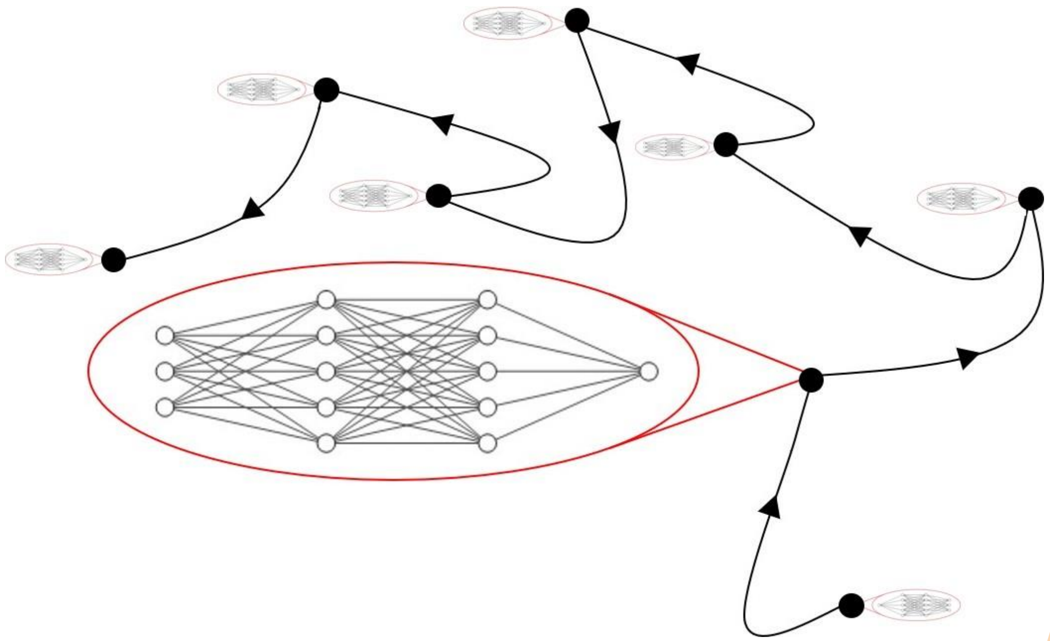
- If θ denotes the DNN parameters and D the training data, the posterior density is given by

$$P(\theta|D) = \frac{P(D|\theta)P(\theta)}{P(D)}$$

- If f represents a neural network, y is a prediction, and x a point in the parameter space of a BSM model, we can compute a distribution over y as follows:

$$P(y|x,D) = \int \delta(y - f(x,\theta))P(\theta|D)d\theta$$

- The difficulty with this approach arises from the need to integrate over all possible network parameters.
- Markov Chain Monte Carlo (MCMC) methods present a well understood way to approximate high-dimensional integrals, such as the ones that occur, for example, in lattice gauge field theory.



Hamiltonian Monte Carlo (HMC)

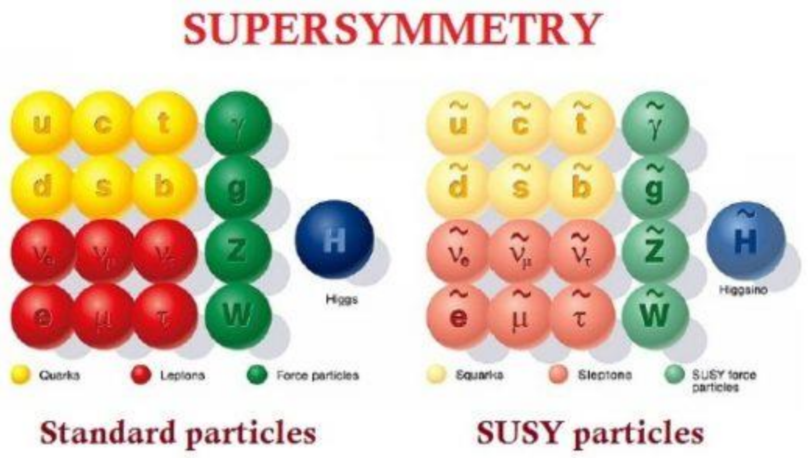
- HMC is the MCMC method of choice to deal with high dimensional integrals.
- HMC treats the negative log likelihood of the data as a (fictitious) potential energy
  - The log likelihood is $\log(P(D|\theta)P(\theta))$
- Hamilton's equations are used to traverse the network parameter space.
- The sampler proceeds by alternating between deterministic trajectories and random changes of direction.

https://arxiv.org/abs/2009.14393

https://github.com/alpha-davidson/TensorBNN

TensorBNN

- TensorFlow (TF) provides an extremely powerful back end for running machine learning tasks.
  - TF compiles training and inference tasks into efficient computational graphs and allows utilization of GPUs to speed up calculations.
- TF, however, does not provide a direct means of training Bayesian neural networks (BNN).
- Using a Hamiltonian Monte Carlo sampler from the related TensorFlow-Probability package, we built a general-purpose framework for training BNNs called TensorBNN.
  - The package makes it easy to create networks and provides several analysis tools for studying the trained networks.
- More details on the package are available here: https://arxiv.org/abs/2009.14393



Parameter	Description	Range
M_1	bino mass	$\|M_1\| \leq 4$ TeV
M_2	wino mass	$\|M_2\| \leq 4$ TeV
M_3	gluino mass	$M_3 \leq 4$ TeV
μ	higgsino mass	$\|\mu\| \leq 4$ TeV
M_A	pseudoscalar Higgs boson mass	$M_A \leq 4$ TeV
$\tan\beta$	ratio of vacuum expectation values of Higgs doublets	$1 \leq \tan\beta \leq 60$
A_t, A_b, A_τ	third generation trilinear coupling	$A \leq 7$ TeV
$m_{\tilde{q}}, m_{\tilde{u}_R}, m_{\tilde{d}_R}, m_{\tilde{l}}, m_{\tilde{e}_R}$	first/second generation sfermion mass parameters	$m \leq 4$ TeV
$m_{\tilde{Q}}, m_{\tilde{t}_R}, m_{\tilde{b}_R}, m_{\tilde{L}}, m_{\tilde{\tau}_R}$	third generation sfermion mass parameters	$m \leq 4$ TeV

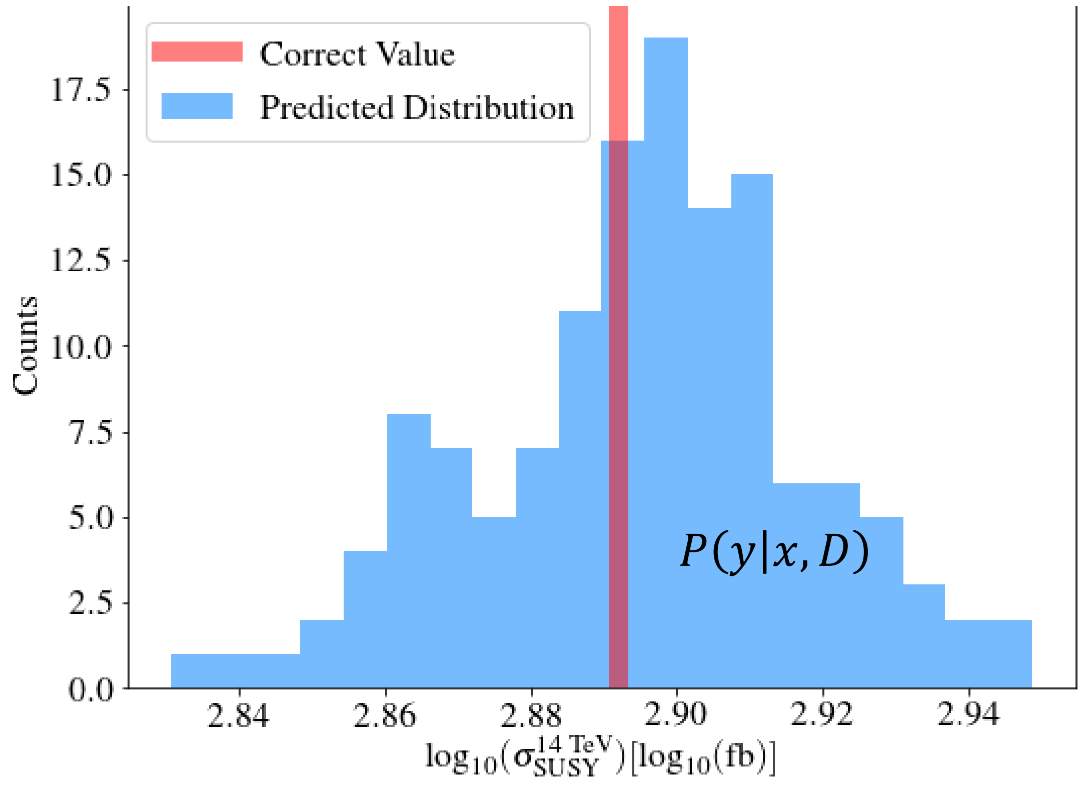
A Case Study

The model chosen for a proof-of-principle is the phenomenological Minimal Supersymmetric Standard Model (pMSSM).

1. 19 free parameters.
2. The NLO total supersymmetric cross section is the main target.
3. The standard SUSY codes, predictions take about 3.5 minutes each.
4. The training data are generated from these codes.

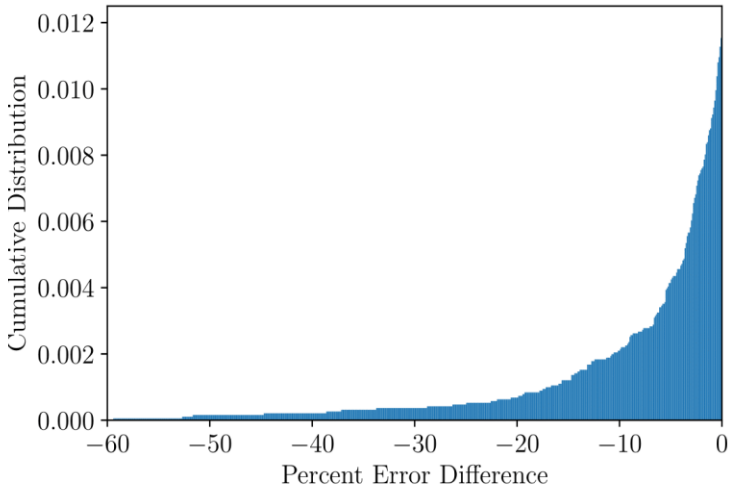
We also looked at the predicted Higgs mass, as well as whether a pMSSM parameter point is theoretically viable, as determined by the SUSY codes.

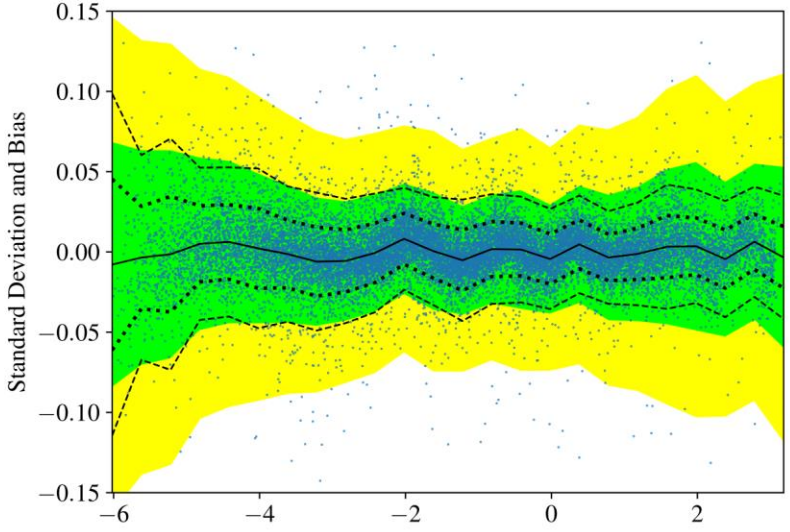
Cross section

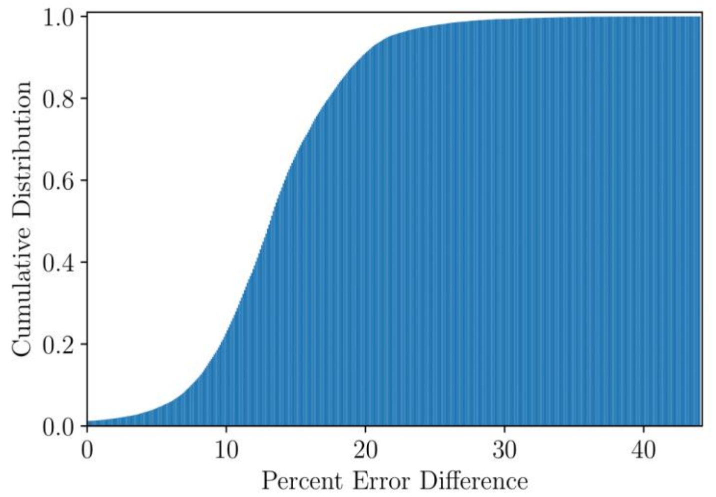


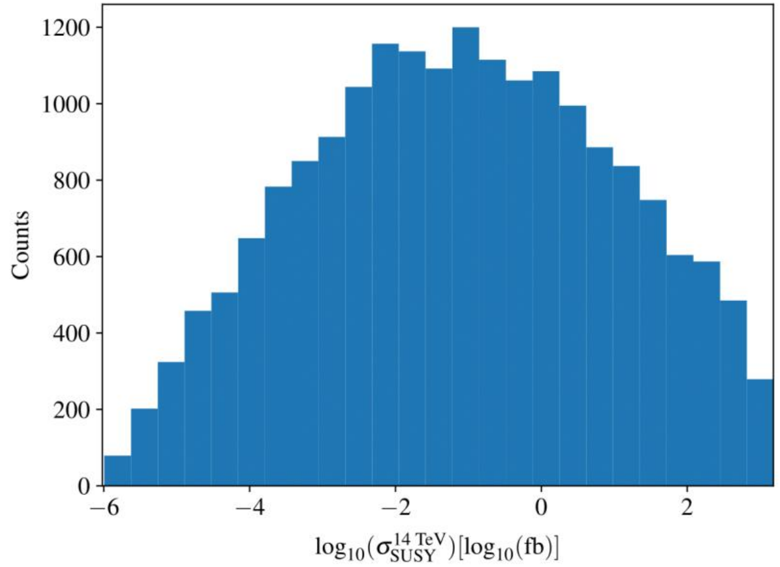
- Generated ~200,000 pMSSM points
  - ~160,000 points used to train
- Network architecture:
  - (19, 50, 50, 50, 50, 50, 1)
- After burn-in, performed 13,500 HMC sampling steps, and used 1 out of every 100 sampled networks.
- Overall, the predictions with the BNN had a percent error of 3.34%.
- If we use the $P(y|x,D)$ to construct 3 standard deviation credible intervals, but treat them as confidence intervals, the coverage is 99%.
- When run on GPUs in large batches, predictions can be made 15 million times faster than with the original SUSY codes.

Prediction Performance Plots

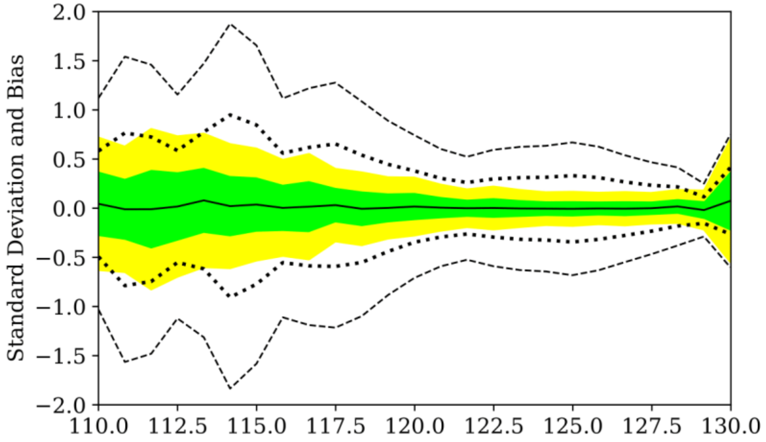


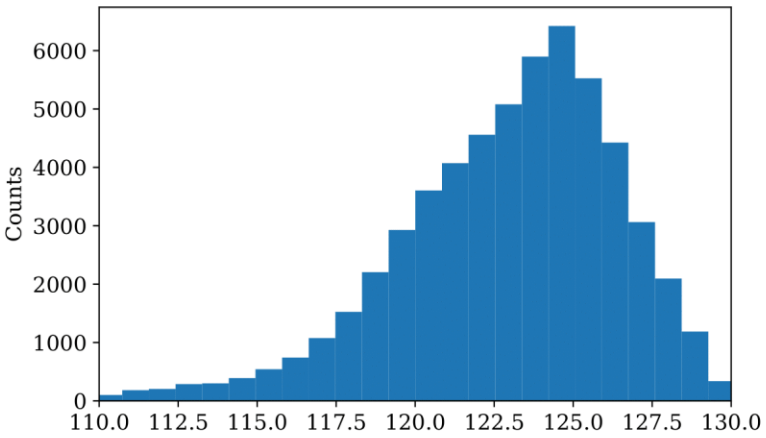






Higgs Boson Mass



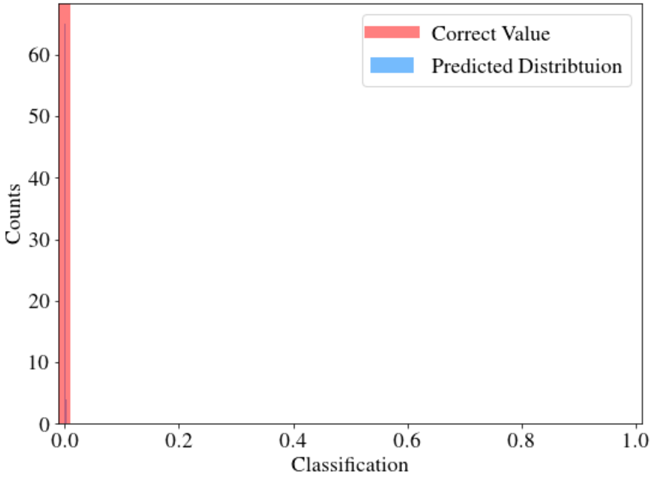


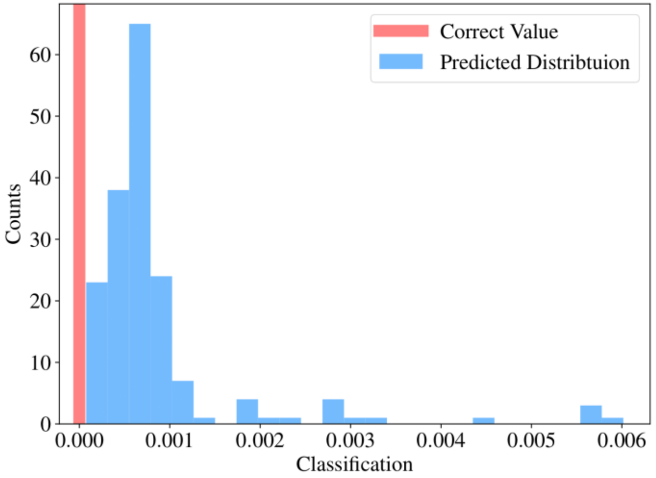
- Training dataset of ~450,000 points
- Network: (19, 50, 50, 50, 50, 50, 1)
- Performed 2,850 HMC sampling steps and used 1 out of every 10 sampled networks.
- Predictions divided into two categories based on whether or not the 3-sd credible interval overlaps the range 123-127 GeV.

Region	Percent Error	Coverage
Overlapping	0.10%	87.4 %
Non-overlapping	0.14%	86.7%

Precision	Recall	F1
0.926	0.997	0.960

Theoretical Viability





- Training dataset of ~400,000 points
- Network: (19, 50, 50, 50, 50, 50, 1)
- Performed 1,750 HMC sampling steps, used 1 out of every 10 sampled networks.

Strategy	Positive Condition	Recall	Precision	F1
Best overall performance	<NN> > 0.5	0.955	**0.953**	**0.954**
Minimize false negatives	<NN> + 3sd > 0.5	**0.982**	0.915	0.947

10.1016/j.physletb.2020.136041

Further Study

- Given the success of TensorBNN in replicating the pMSSM cross section predictions, it would be interesting to test its effectiveness on other, potentially more interesting, theoretical models.
- The package can be expanded to include other layers, such as convolutional layers
- An autocorrelation study revealed that the networks used were still highly correlated, suggesting that a greater lag is needed between used networks, and therefore longer MCMC chains, which in turn motivates the need for work to reduce computation time.
  - The package already includes an algorithm to adapt the step size and leapfrog step count automatically. But, further tuning of this algorithm may be beneficial.
  - The No-U-Turn sampler could reduce the correlation length of the sampling.
  - Other potential samplers to investigate: Riemann Manifold Hamiltonian Monte Carlo and Learning 2 HMC (l2hmc).

Questions?