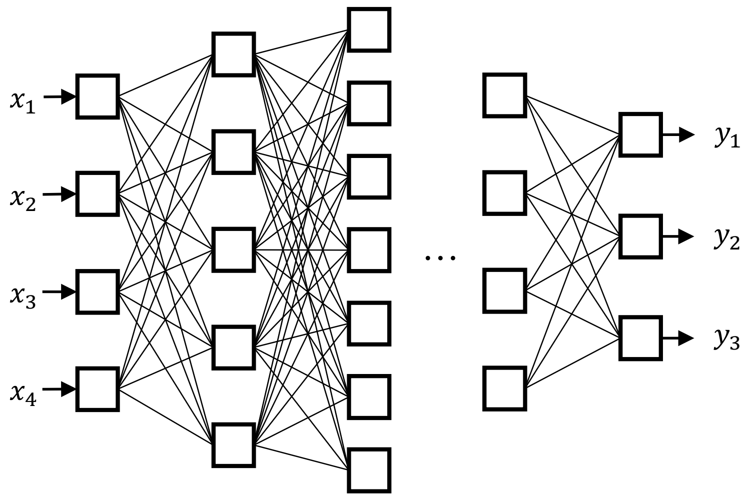
Reliable Deep Learning for Nuclear Physics

Addressing uncertainty quantification and extrapolation

Chanhee KIM (Ben)
SungKyunKwan Univ.
chkim@phys.kim

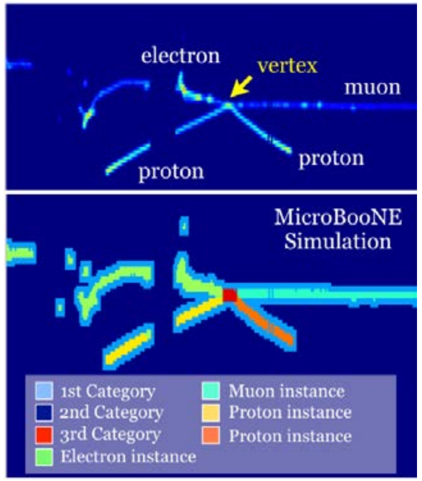
Strengths of deep learning for physics

Deep neural networks

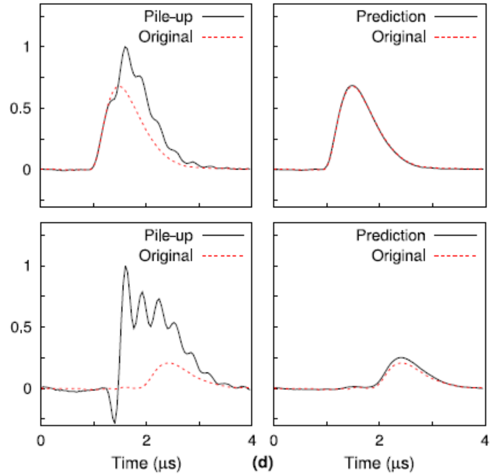


- Large number of parameters
- Complex entanglements with nonlinear functions

→ **Flexibility**
→ **Accuracy**
→ **Speed**



(2019, PRD, C. Adams)



(2023, NIMA, C. H. Kim)

σ_{pre}		LDM 2.462 ± 0.023	DZ 0.613 ± 0.007	WS4 0.302 ± 0.003	FRDM 0.599 ± 0.009
Training set	RBF by Wang [11]	–	–	0.170	–
	KRR by Wu [40]	–	–	0.199	–
	RBFs by Ma [59]	–	–	0.130	0.209
	LMNN by Zhang [39]	0.235	0.325	–	0.348
	BNN by Niu [27]	–	–	0.176	0.187
	RBFoe by Niu [41]	–	0.171	0.140	0.182
	NN by Utama [25]	0.466	0.274	–	0.342
	NN by Pastore [58]	–	0.324	–	–
	Trees by Carnini [44]	2.070	0.471	–	–
	LightGBM in this work	0.058 ± 0.011	0.066 ± 0.010	0.055 ± 0.011	0.077 ± 0.013
Test set	LMNN by Zhang	0.256	0.329	–	0.368
	BNN by Niu	–	–	0.212	0.252
	RBFoe by Niu	–	0.344	0.337	0.218
	NN by Utama	0.486	0.278	–	0.352
	NN by Pastore	–	0.358	–	–
	Trees by Carnini	2.881	0.569	–	–
	LightGBM in this work	0.234 ± 0.022	0.213 ± 0.018	0.170 ± 0.011	0.222 ± 0.016

(2021, Nucl. Sci. Tech, Z. Gao)

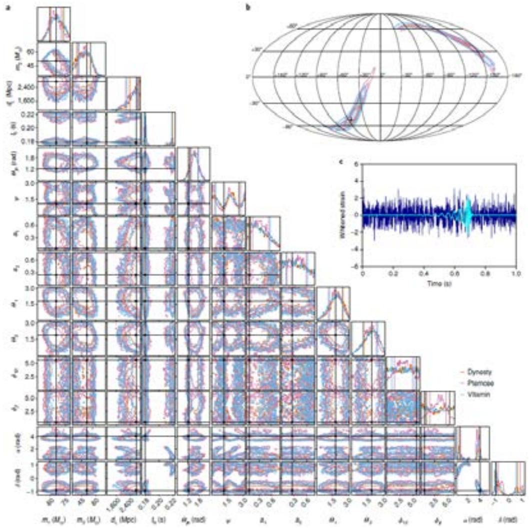


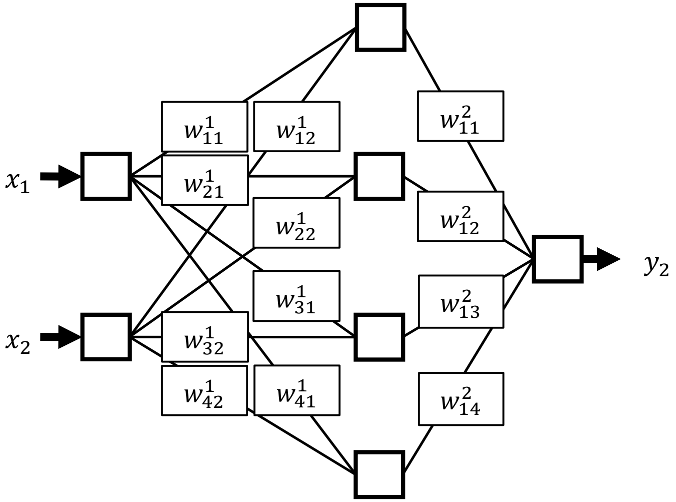
Table 2 | Durations required to produce samples from each of the sampling approaches

Sampler	Run time (s)			Ratio $\frac{\tau_{VItamin}}{\tau_x}$
	Min.	Max.	Median	
Dynesty[a7]	21,564	261,268	45,607[b]	2.2×10^{-6}
emcee[8]	16,712	39,930	19,821	5.1×10^{-6}
ptemcee[9]	2,392	501,632	41,151.0	2.4×10^{-6}
CPNest[6]	10,309	437,008	83,807	1.2×10^{-6}
VItamin[c]	1×10^{-1}			1

(2022, Nat. Phys., H. Gabbard)

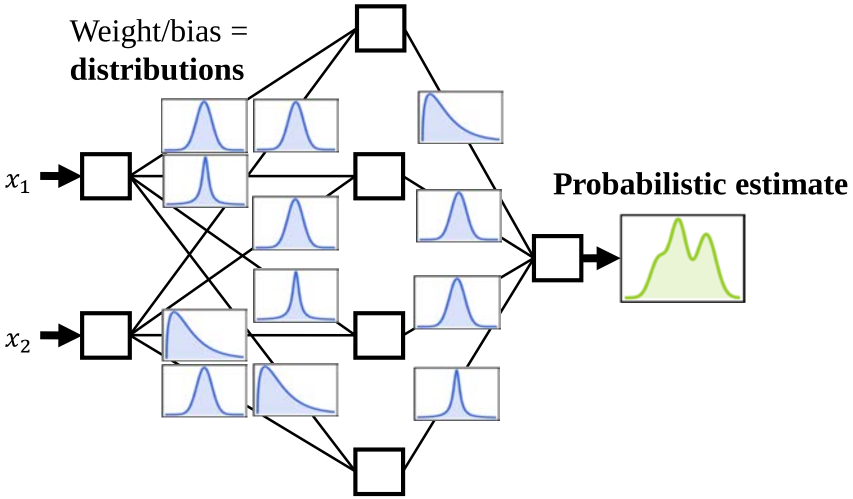
Uncertainty quantification

Conventional neural networks



VS

Probabilistic neural networks



⇒ The outputs are not expressive and only **plain single prediction**

- Test results do not reflect uncertainties in real applications
- Failure of predictions and **out-of-distribution data** cannot be detected

→ It includes **numerous possible parameter sets**

→ **Model uncertainty** from parameters can be captured

→ Expressive predictions and **confidence levels** are provided

How to obtain probabilistic expressions for parameters from training data

2022, IEEE, L. Jospin
2024, PRC, C. H. Kim

Bayesian deep learning

Bayesian inference for network parameters using training data (Training = Inference)

$$p(\boldsymbol{\theta}|\boldsymbol{D_x}, \boldsymbol{D_y}) = \frac{p(\boldsymbol{D_y}|\boldsymbol{D_x}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{D_y}|\boldsymbol{D_x})}$$

Bayesian Marginalization

<u>Conventional deep learning</u>
- Optimization
- Find single best parameter set

<u>Bayesian deep learning</u>
- Marginalization
- Consider all possible sets

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{D}) = \int_{\theta} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}')p(\boldsymbol{\theta}'|\boldsymbol{D})d\boldsymbol{\theta}'$$

Predictive distribution — Likelihood — Posterior

<u>(Markov Chain) Monte Carlo</u>

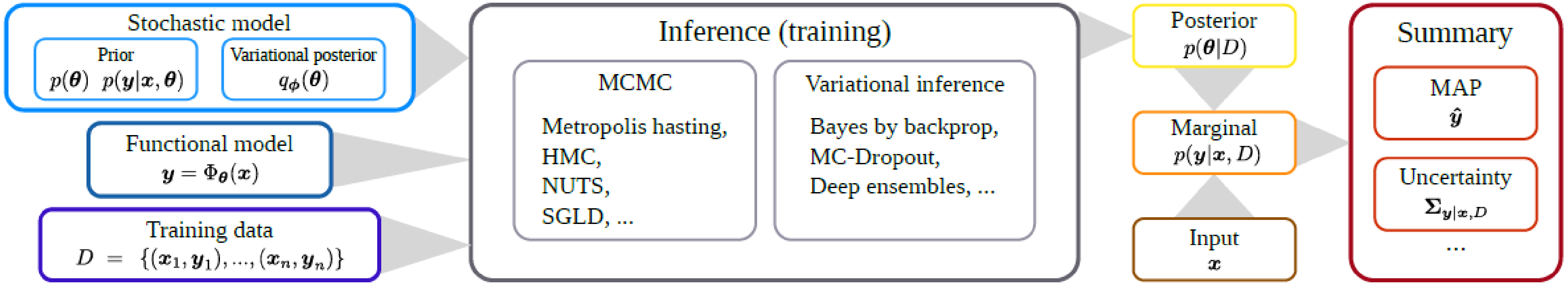
Because of countless parameters in complex structure, extremely **expensive for modern deep neural networks** (**not scalable**)

<u>**Variational inference**</u>

Use a variational distribution to approximate posterior
Suitable for modern size of neural network (**fast and scalable**)

Optimization of KL-divergence (closeness of two distributions) between approximate distribution and posterior distribution

(2022, IEEE, L. Jospin)



2020, NIPS, A. Wilson
2022, IEEE, L. Jospin

Methods for Bayesian deep learning

Monte Carlo Dropout (2016, PMLR, Y. Gal)

1. Training with **dropout** and *L2* regularization

⇒ Minimization of the loss function (NLL) ~ Optimization of KL-divergence

- Approximation distribution for posterior: Bernoulli distribution (dropout)
- Prior: Gaussian (*L2* regularization)

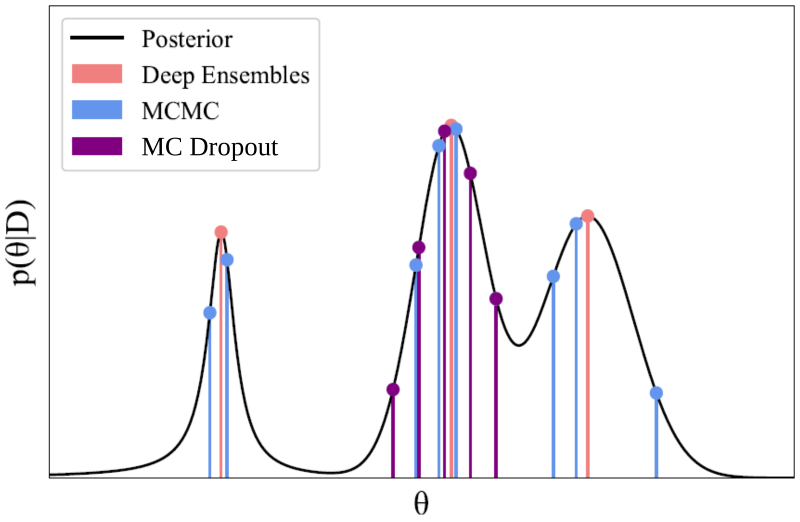
2. Even after training, **turn on dropout**

Each forward pass will sample a parameter set from the approximated posterior.

$$p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{D}) = \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}')p(\boldsymbol{\theta}'|\boldsymbol{D})d\boldsymbol{\theta}'$$

$$\approx \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}')q_{\phi}(\boldsymbol{\theta}')d\boldsymbol{\theta}'$$

$$\approx \frac{1}{N_{\mathrm{MC}}}\sum_{n=1}^{N_{\mathrm{MC}}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_n^*)$$



Deep Ensembles (2017, NIPS, B. Lakshminarayanan)

Approximation functions: mixture of delta functions

1. **Training multiples models** with *L2* with different initial model parameters

⇒ Each model will **capture a mode in the posterior**

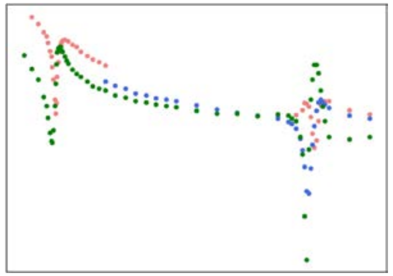
2. **Combine** the multiple model predictions

$$p(\boldsymbol{\theta}|\boldsymbol{D}) \sim \frac{1}{N_{\mathrm{model}}}\sum_{p=1}^{N_{\mathrm{model}}} \delta(\boldsymbol{\theta}=\boldsymbol{\theta}_p)$$

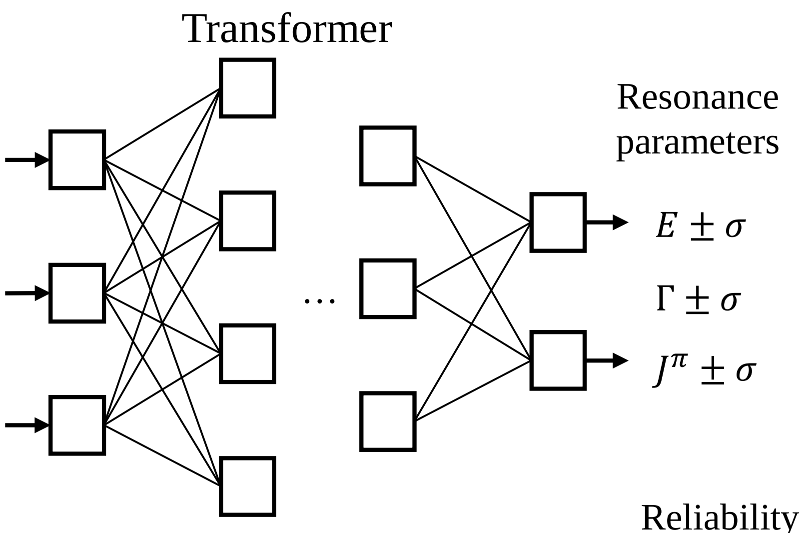
$$\Rightarrow \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}')p(\boldsymbol{\theta}'|\boldsymbol{D})d\boldsymbol{\theta}' \sim \frac{1}{N_{\mathrm{model}}}\sum_{p=1}^{N_{\mathrm{model}}} p(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{\theta}_p)$$

Deep Ensembles for *R*-matrix analysis

12C(p,p)
Cross section



with 3 resonances
(1976, ZPA, H. Meyer)

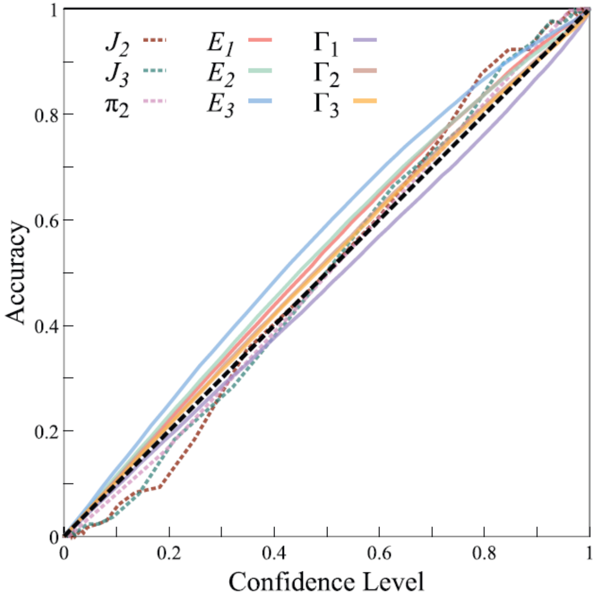


5 models trained with various 12C(p, p) data calculated using AZURE2 (2010, PRC, R. Azuma)

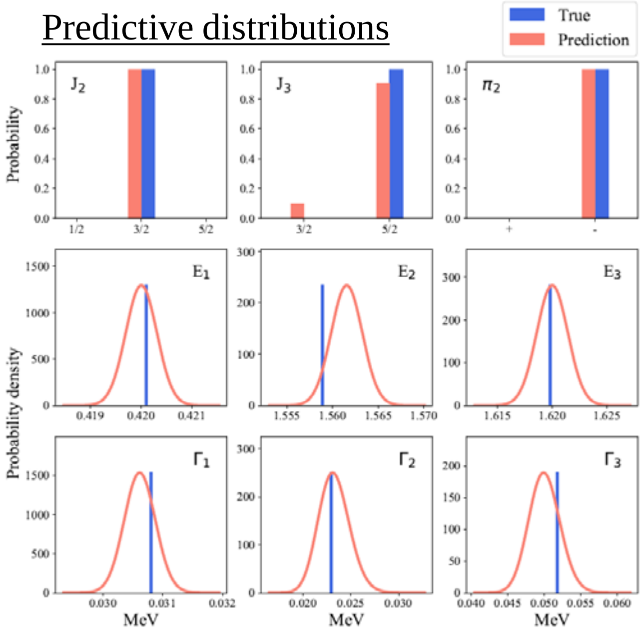
$$p(\boldsymbol{\theta}|\boldsymbol{D}) \sim \frac{1}{N_{\text{model}}} \sum_{p=1}^{N_{\text{model}}} \delta(\boldsymbol{\theta} = \boldsymbol{\theta}_p)$$

$$\Rightarrow \int_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}')p(\boldsymbol{\theta}'|\boldsymbol{D})d\boldsymbol{\theta}' \sim \frac{1}{N_{\text{model}}} \sum_{p=1}^{N_{\text{model}}} p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}_p)$$

Reliability diagram



Predictive distributions



Disentanglement of uncertainties

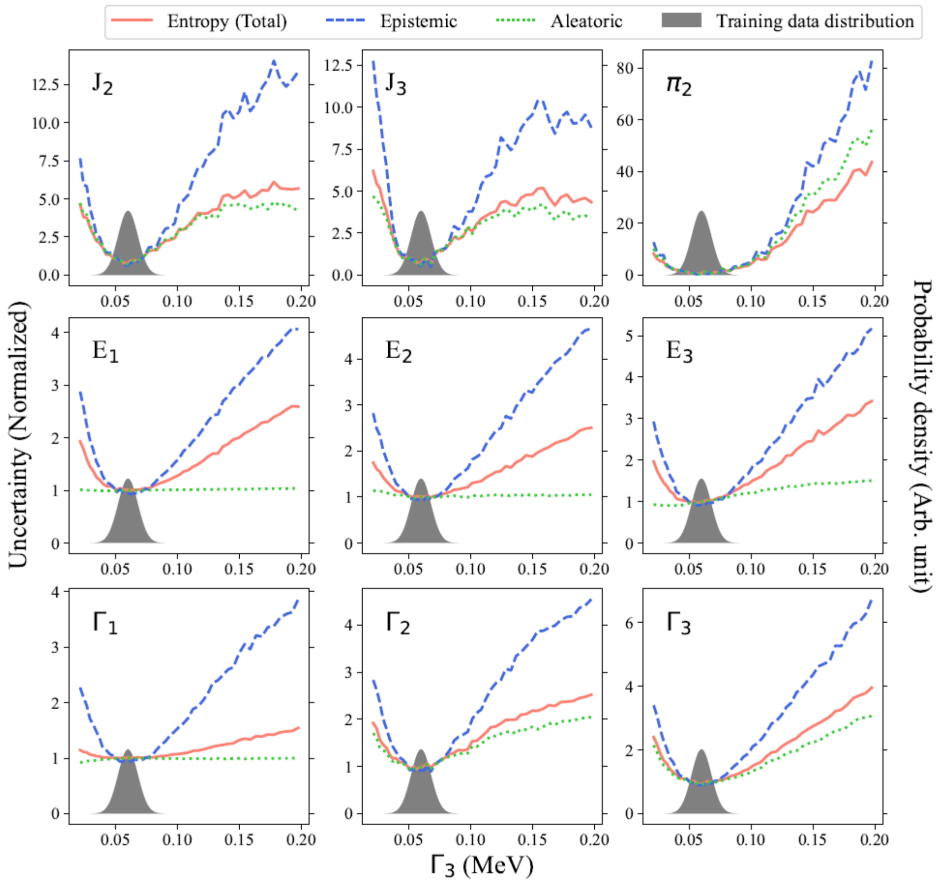
Parameters	Total (Entropy)	Aleatoric	Epistemic
J_2	4.08×10^{-3}	1.23×10^{-4}	4.46×10^{-7}
J_3	1.52×10^{-4}	1.35×10^{-5}	3.34×10^{-8}
π_2	2.82×10^{-3}	1.36×10^{-4}	1.24×10^{-6}
$\log E_1$	4.44×10^{-4}	3.57×10^{-4}	2.64×10^{-4}
$\log E_2$	7.84×10^{-4}	7.55×10^{-4}	2.11×10^{-4}
$\log E_3$	4.94×10^{-4}	3.32×10^{-4}	3.66×10^{-4}
$\log \Gamma_1$	8.05×10^{-3}	7.90×10^{-3}	1.56×10^{-3}
$\log \Gamma_2$	1.42×10^{-2}	1.38×10^{-2}	3.43×10^{-3}
$\log \Gamma_3$	1.49×10^{-2}	1.41×10^{-2}	4.65×10^{-3}

2024, PRC, C. H. Kim

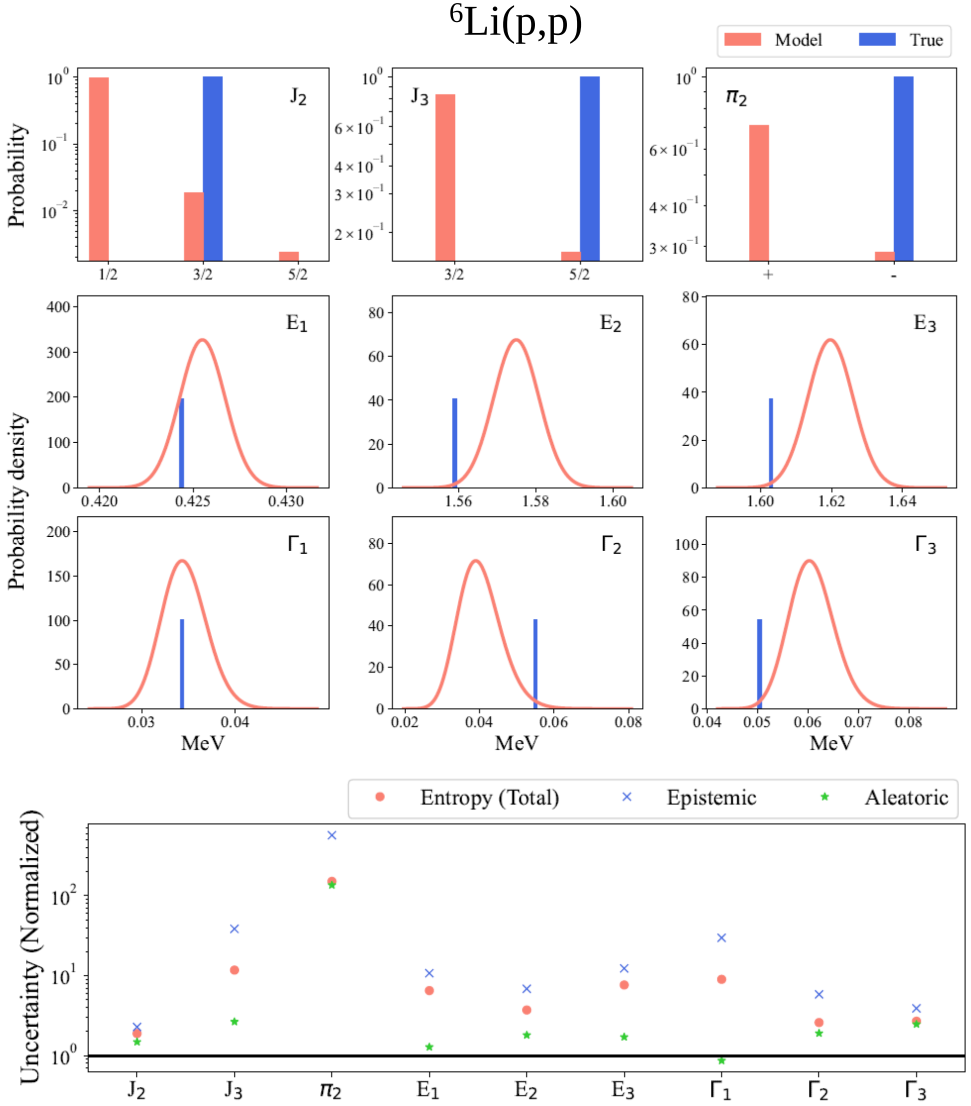
Extrapolation issues

If it is not $^{12}C(p,p)$ reaction,

If 3rd resonance width varies outside training data distribution,

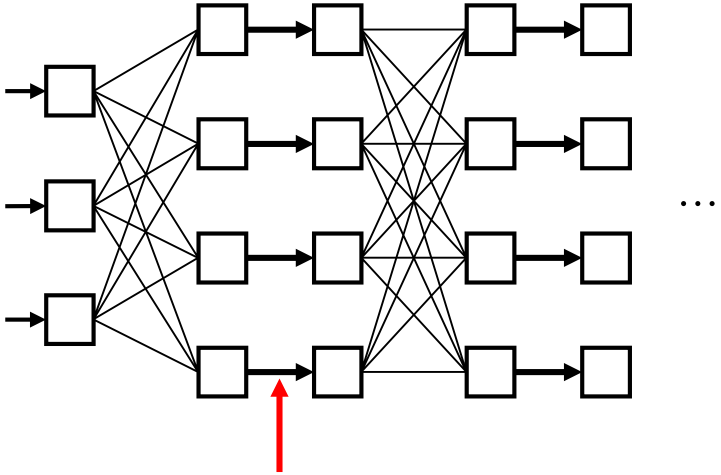


$^{6}Li(p,p)$

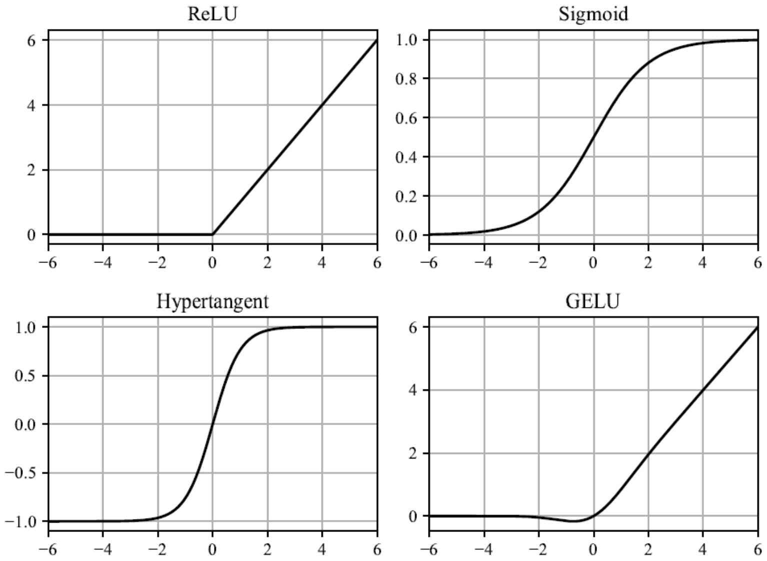


2024, PRC, C. H. Kim

Features of conventional deep neural networks



Activation functions



Countless parameters in complex structure, entangled with nonlinear functions

⇒ Prone to **over-parameterization** and **overfitting**

Reasons why extrapolation is typically poor

Empirically proven to give:

- Stable training
- Fast convergence
- High performance on computer vision, natural language processing, etc.

These features are **designed for computer science** applications

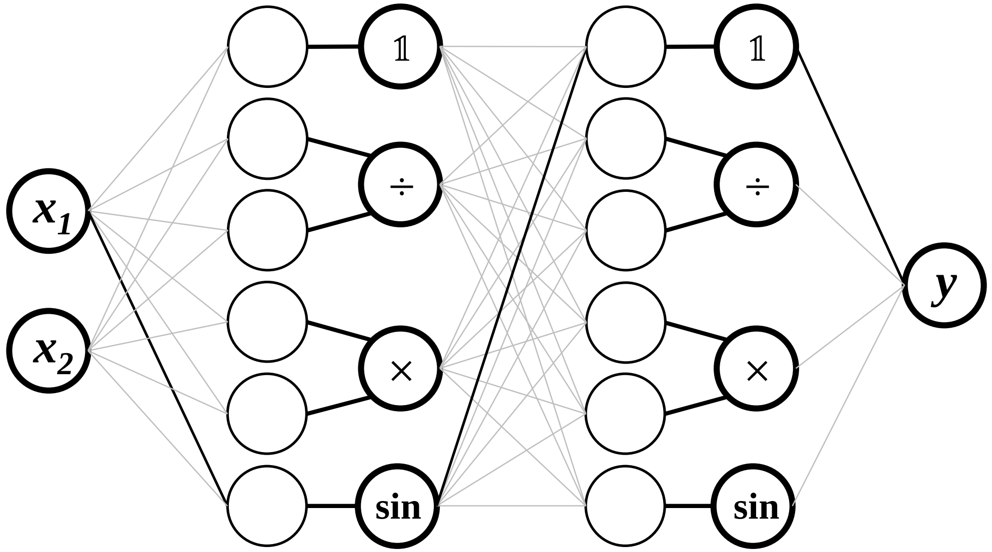
What if we **design a network for physics applications**?

2019, O'Reilly Media Inc., A. Geron
2022, MIT press, K. Murphy

Network designed for physics

Finding the **true mathematical equation** underlying the data ⇒ Extrapolation issues will be naturally solved

Equation Learning: **design a network to become** the **true mathematical equation** underlying the data

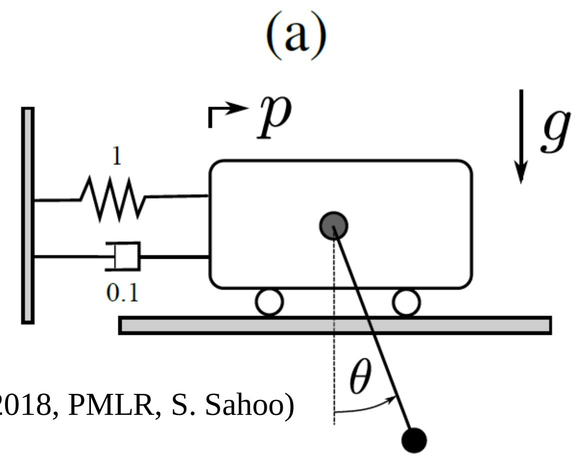


Ex) $\boldsymbol{y} = w_2\,\mathbf{sin}(w_1 \boldsymbol{x_1} + b_1) + b_2$

1st. **Replace** activation functions with **scientific activation functions**

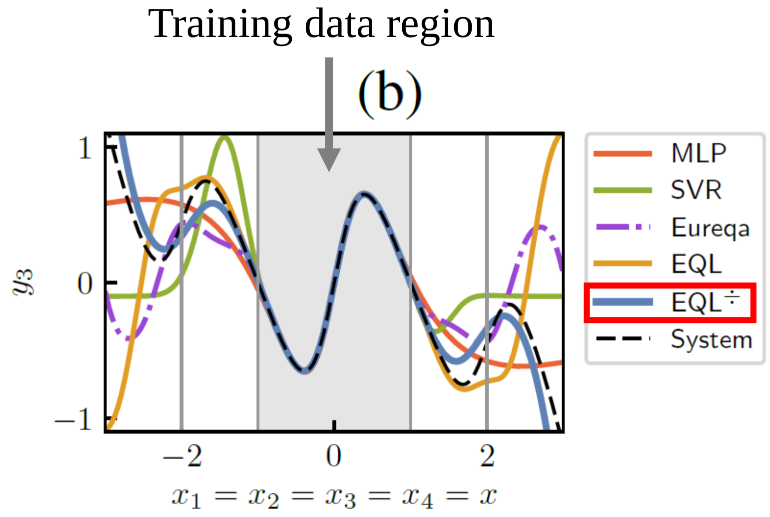
2nd. Use *L0* regularization to **prune unimportant parameters** (sparse learning)

⇒ Ideally, one can reconstruct a simple equation from the network



(2018, PMLR, S. Sahoo)

Cart-pendulum system



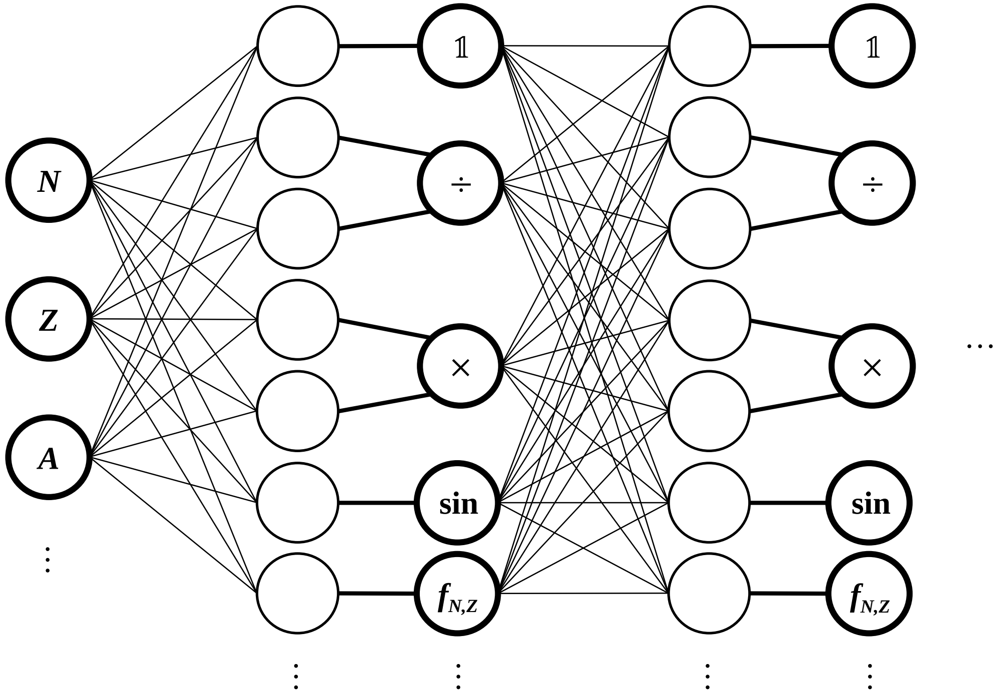
Most physics subjects might not be represented by a simple equation

Strong regularization does not give the optimal performance

2017, ICLR, G. Martius
2018, PMLR, S. Sahoo

On nuclear mass predictions

(PAF)
Network of physics-related activation functions

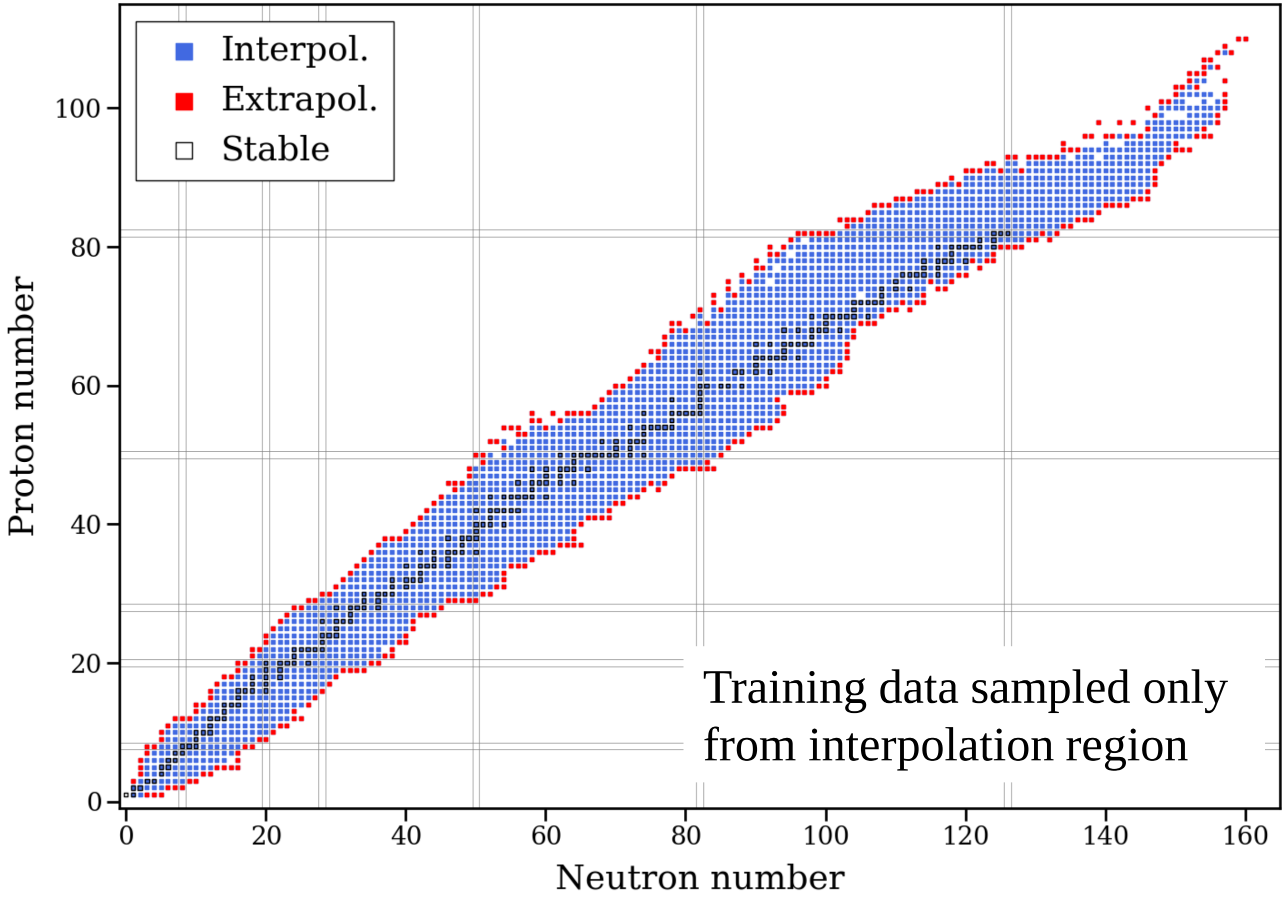


Activation functions

𝟙 (identity)	×	÷
log	sin	ReLU
N^x	Z^x	
x^N	x^Z	$((N-Z)/A)^x$

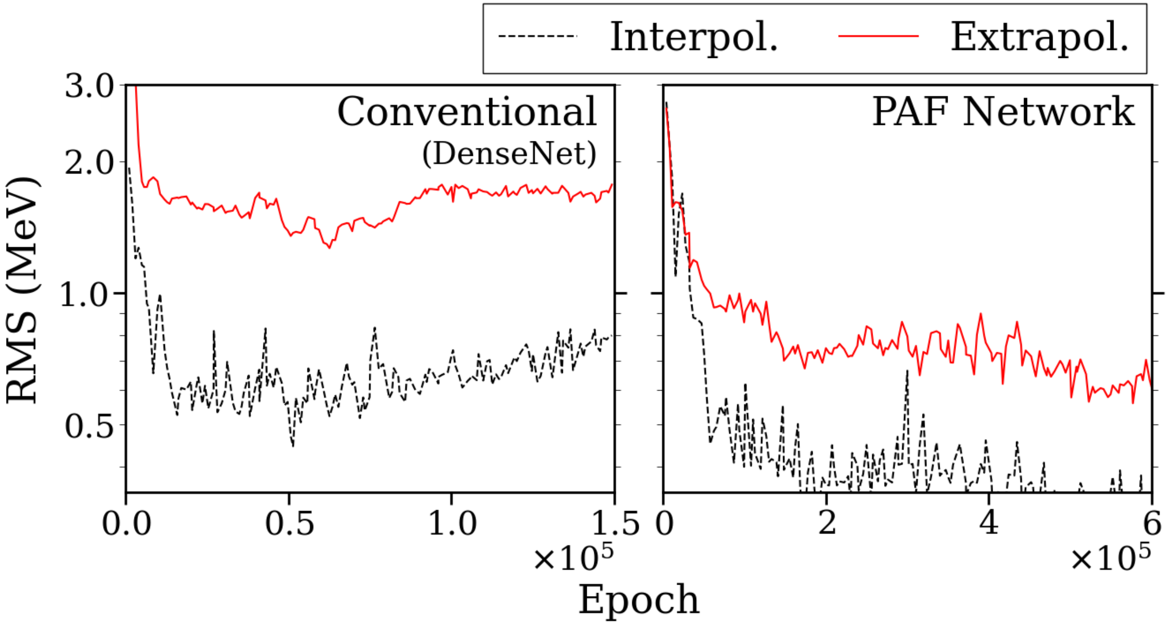
(2021, CPC, M. Wang)
AME2020

Data distribution (N, Z>0)



Improvements on extrapolation

Learning curves



Evaluation results

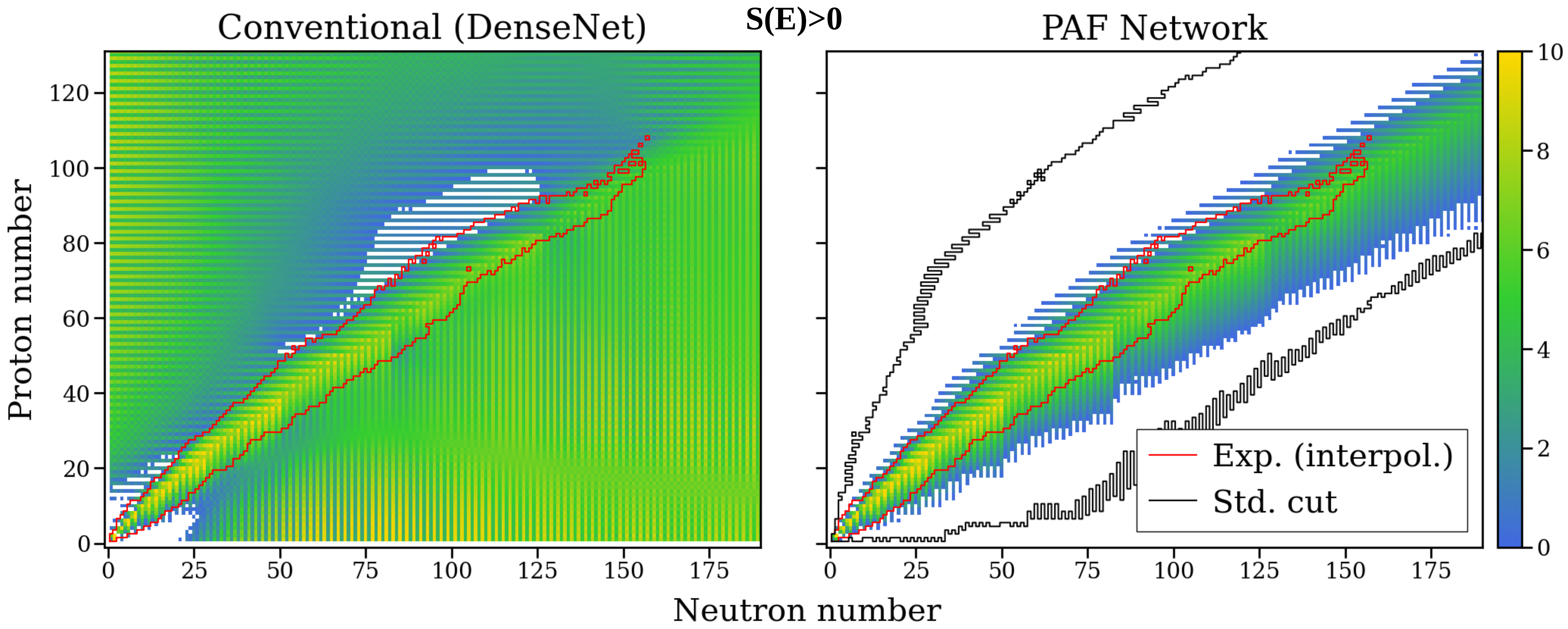
Model	RMS $(N, Z \geq 8)$ / $(N, Z > 0)$ (keV) Interpol.	Extrapol.	Total
PAF Network	207 / 255	308 / 396	258 / 328
DenseNet	494 / 532	1048 / 1173	795 / 889
ConvNet	318 / 409	531 / 658	428 / 539
Duflo-Zuker [3]	-	-	428 / -
KTUY [32]	-	-	733 / 743[1]
WS4 [5]	-	-	295 / -
FRDM [6]	-	-	606 / -

[1] $N, Z \geq 2$

(model selection done for the lowest RMS on extrapolation)

The existing global mass models fitted using the different AME versions
The total RMS error on the AME2020 data only presented

Improvements on extrapolation



Conclusion

- ✓ Representative challenges of deep learning for physics are uncertainty quantification and extrapolation
- ✓ Bayesian deep learning can be the solution for the uncertainty quantification
- ✓ We showed a deep learning application for R-matrix analysis along with uncertainty quantification
- ✓ Replacing conventional activation functions with physics-related activation functions (PAF) could be the solution for the extrapolation
- ✓ We successfully implemented the PAF network for nuclear masses, showcasing the extrapolation performance.

R-matrix application (Bayesian deep learning): C. H. Kim et al., PRC, 2024, DOI: 10.1103/PhysRevC.110.054609

Nuclear mass application (PAF network): C. H. Kim et al., arXiv:2505.15363