숭실대학교 Soongsil University

Mar 2 – 3, 2025
Hana Square, Korea University

OMEG Origin of Matter and Evolution of Galaxies Key Research Institutes in Universities

The 3rd Korea-Japan Workshop on Hadrons and Nuclei

Enhancing Nuclear Mass Predictions with Machine Learning

Mar 2 – 3, 2025
Hana Square, Korea University

Jubin Park

Collaborators : Prof. Myung-Ki Cheoun, Dr. Myeong-Hwan Mun, Seonghyun Kim

Contents

① **Introduction & Motivation**

- Why predicting nuclear masses matters
- Limitations of traditional models (WS4, HFB-31)
- Why machine learning is a promising approach

② **Overview of Machine Learning Approaches**

- Previous ML-based nuclear mass predictions
- Supervised vs. unsupervised learning
- Key references and major contributions

③ **Our Approach: Deep Learning Models**

- Dataset: **AME2020 & feature selection**
- **Deep Neural Networks (DNN)**
- Model architectures and performance analysis

④ **Comparison with WS4, DRHBc, and ML Models**

- Accuracy comparison: Root-Mean-Square (RMS)
- Strengths and weaknesses of ML-based models
- Key observations from model performance

⑤ **Results and Analysis I (Supervised learning for nuclear mass prediction)**

- Shallow and Deep Neural Net (with 11 inputs)
- Shallow and Deep Neural Net (with 13 inputs)
- Optimized Deep Neural Net (with 12 inputs)

⑥ **Future Work & Conclusion**

- Improving generalization and feature selection
- Exploring advanced ML architectures
- Towards **high-precision nuclear mass predictions**

Motivation

◈ Why Predict Nuclear Masses?

- Fundamental quantity for understanding nuclear structure
- Essential for astrophysical nucleosynthesis models
- Important for nuclear reactor simulations

◈ Why is accuracy critical?

- **r-process nucleosynthesis** models require high-precision mass predictions.
- **Nuclear binding energy** influences reaction rates in stellar environments.
- **Nuclear reactor stability** relies on accurate fission product mass calculations.

◈ Limitations of Traditional Models

- Theoretical models (e.g., WS4, HFB-31) still have **large uncertainties (~300 keV)**
- Many models fail to capture **local nuclear structure effects**
- Need for a **data-driven approach** to improve precision

◈ Why Machine Learning?

- ML can learn **hidden patterns** from experimental data
- Can combine **global theoretical models with local corrections**
- Previous ML studies improved predictions but **generalization remains a challenge**
- Our goal: Achieve **higher accuracy (<200 keV) with better generalization**

Machine learning methods in nuclear mass predictions

Prof. Jian Li's talk in
"The 7th workshop on nuclear mass table with DRHBc theory" (@Gangneung, 2024)

- ★ **ANN:** Gazula1992NPA, Athanassopoulos2004NPA, Bayram2014ANE, Zhang2017JPG, Ming2022NST,Yuksel2021IJMPE, Li2022PRC, Zeng2024PRC
- ★ **BNN:** Utama2016PRC, Niu2018PLB, Niu2019PRC, Niu2022PRCL, Rodriguez2019EPL, Rodriguez2019JPG
- ★ **CNN:** Yang2023PRC **DNN:**ChenPRC2022, To-Chung-Yiu2024CPC
- ★ **LightGBM:** Gao2021NST
- ★ **KRR:** Wu2020PRC, Wu2021PLB,Du2023CPC,Wu2022PLB,Wu2024PRC, Wu2023Front. Phys.
- ★ **NBP:** Liu2021PRC **PUN:** Babette-DellenPLB2024
- ★ **RBF:** Wang2011PRC, Niu2013,2016PRC,2018SciB
- ★ **BGP:** Neufcourt2018,2020PRC, Neufcourt2019PRL
- ★ **SVM:** Clark2006IJMPB
- ★ **CLEAN:** Morales2010PRC **MDN:**A. E. Lovell2022PRC **PIML:** Mumpower2022PRC
- ★ **.....**

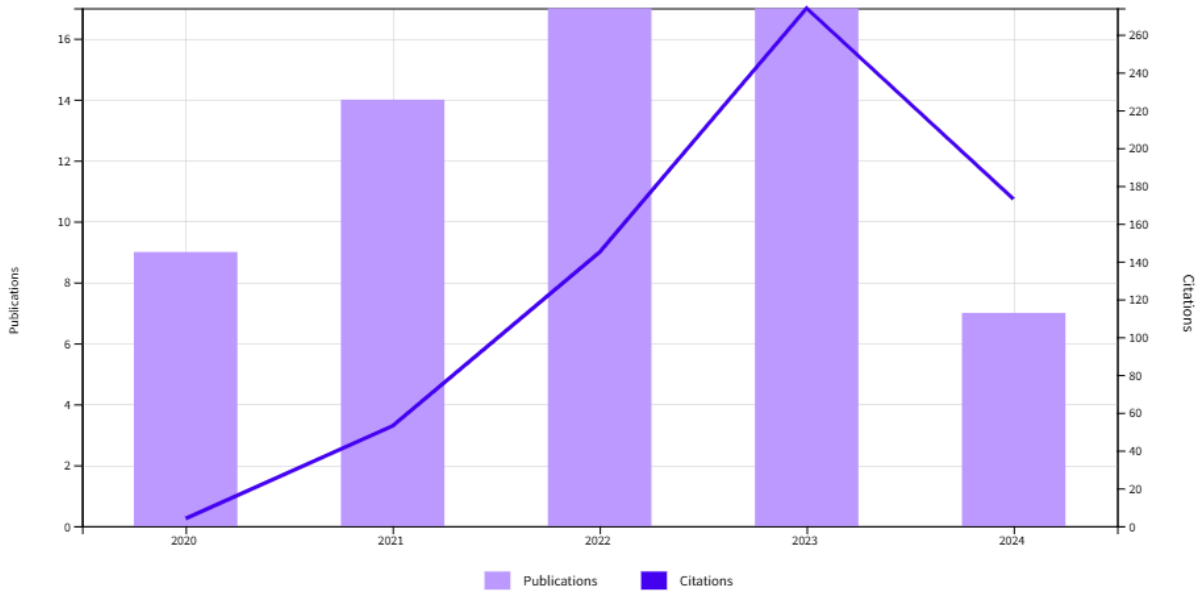
Machine learning in nuclear physics

- **(D)NN**: (Deep) Neural Network
- **BNN**: Bayesian Neural Network
- **CNN**: Convolutional Neural Network
- **MDN**: Mixture Density Network
- **(B)GP**: (Bayesian) Gaussian Processes
- **CGP**: Constrained Gaussian Processes
- **DT**: Decision Tree
- **NBP**: Naive Bayesian Probability Classifier
- **SVM**: Support Vector Machines
- **RBF**: Radial Basis Function
- **KRR**: Kernel Ridge Regression
- **CLEAN**: CLEAN Image Reconstruction
- ...

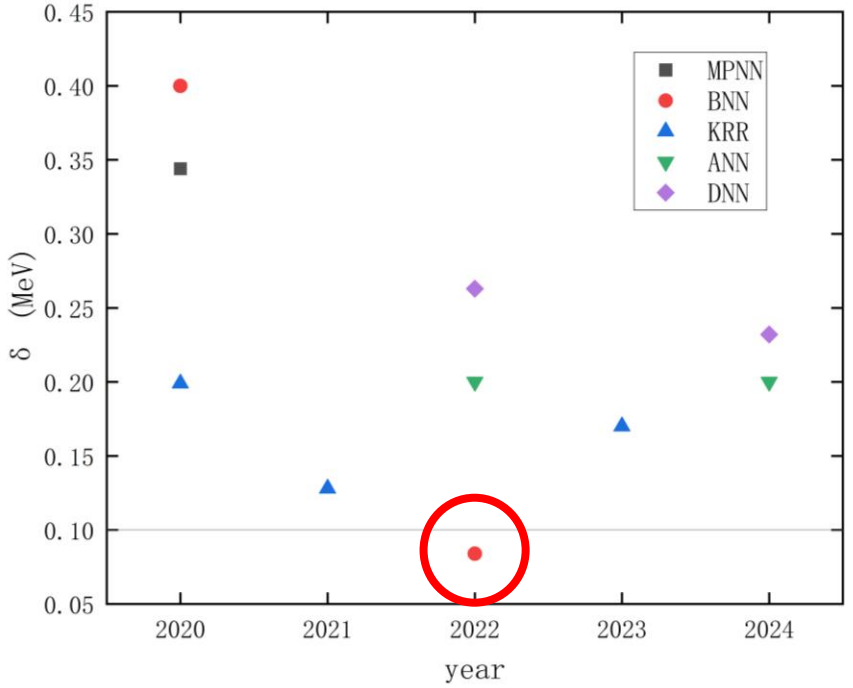
Prof. Jian Li's talk in
"The 7th workshop on nuclear mass table with DRHBc theory" (@Gangneung, 2024)

Machine learning in nuclear mass predictions

Publications/citations per year



Precision of prediction



- ✓ Although there are many studies using machine learning to predict nuclear masses, most of them achieve an accur acy of only around 200 keV.
- ✓ To overcome this bottleneck, it is necessary to consider more physics, as demonstrated by some successful studies.

Machine Learning in Nuclear Mass Predictions

ML Model	Strengths	Weaknesses	Key References
ANN (Artificial Neural Network)	Simple, fast training	Overfitting on small datasets	Gazula1992, Athanassopoulos2004, Bayram2014, Zhang2017, Ming2022, Yuksel2021, Li2022, Zeng2024
BNN (Bayesian Neural Network)	Uncertainty quantification, better generalization	Computationally expensive	Utama2016, Niu2018, Niu2019, Niu2022, Rodriguez2019
CNN (Convolutional Neural Network)	Captures local spatial correlations	Needs large datasets	Yang2023, To-Chung-Yiu2024
DNN (Deep Neural Network)	More complex feature extraction	Prone to overfitting	ChenPRC2022
LightGBM (Gradient Boosting)	Efficient on structured data	Not optimal for deep feature extraction	Gao2021
KRR (Kernel Ridge Regression)	Works well with small datasets	Computationally expensive for large datasets	Wu2020, Wu2021, Du2023, Wu2022, Wu2024, Wu2023
NBP (Naive Bayesian Classifier)	Simple, interpretable	Assumes feature independence	Liu2021
PUN (Probabilistic Uncertainty Network)	Handles uncertainty well	Requires careful tuning	Babette-DellenPLB2024
RBF (Radial Basis Function Network)	Good for function approximation	Sensitive to parameter choice	Wang2011, Niu2013, Niu2016, Niu2018
BGP (Bayesian Gaussian Processes)	Strong predictive power	High computational cost	Neufcourt2018, Neufcourt2020, Neufcourt2019
SVM (Support Vector Machine)	Works well with small data	Not effective for deep learning tasks	Clark2006
CLEAN (Image Reconstruction Algorithm)	Used for noise reduction	Limited to specific applications	Morales2010
MDN (Mixture Density Network)	Can model complex distributions	Hard to train	A.E. Lovell2022
PIML (Physics-Informed ML)	Incorporates physics-based constraints	Model-dependent	Mumpower2022

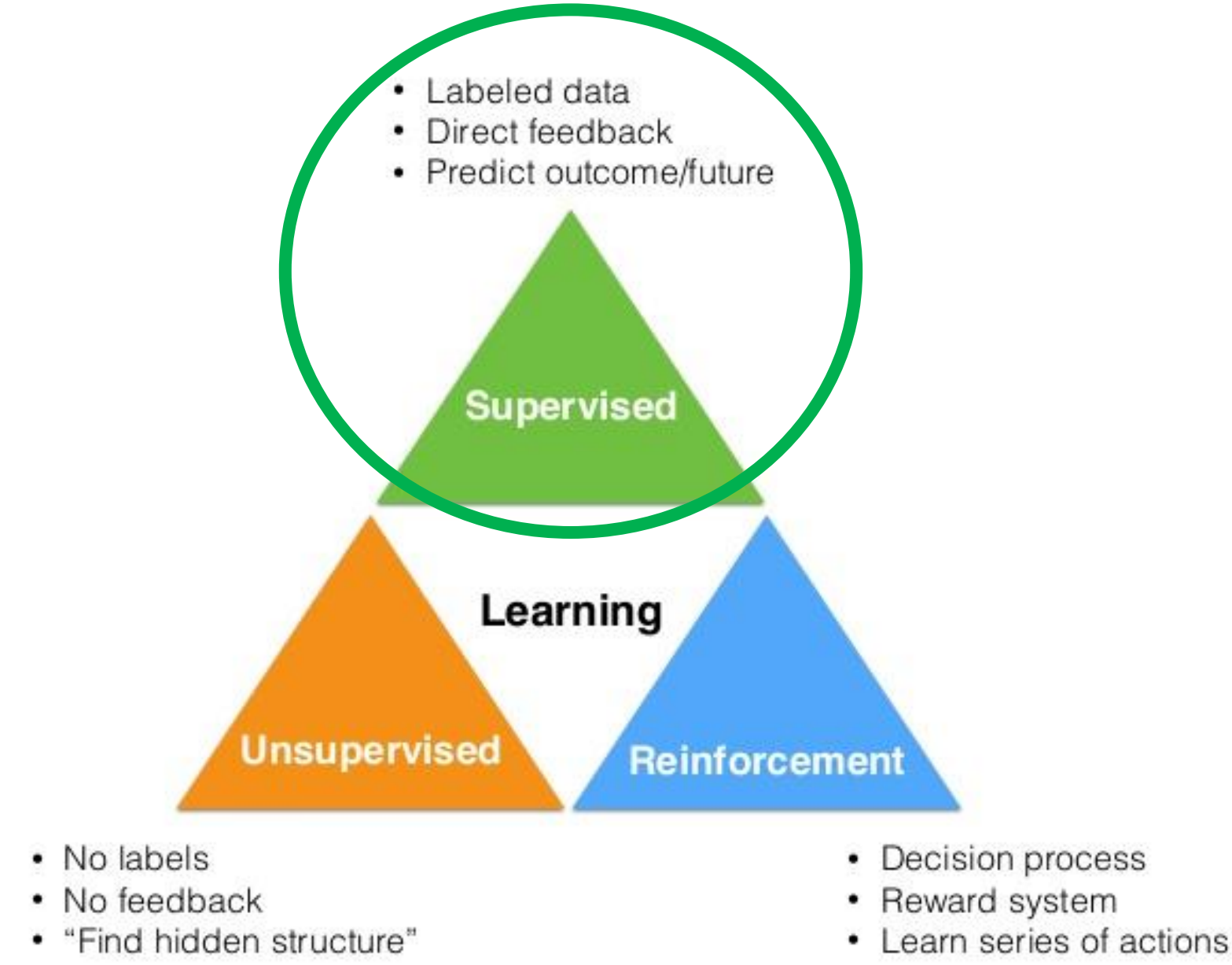
◈ Challenges in ML-based Nuclear Mass Predictions

- **Most models achieve only ~200 keV accuracy**
- **Overfitting issues**: Good performance on training data, poor generalization to validation/test data
- **Lack of physical interpretability**: Many models treat nuclear masses as purely data-driven

◈ Ways to improve performance

- **Combining ML with physics-based models (e.g., WS4, DRHBc)**
- **Feature engineering**: Selecting physical parameters that improve predictions
- **Advanced skills**: Kernel method, Bayesian, Gradient Boosting, ……..
- **Using deep learning (DNN & CNN) to capture both global and local nuclear effects**

Supervised Machine Learning



http://solarisailab.com/archives/1785, 솔라리스의 인공지능 연구실

Why choose DNN?

- Captures complex, non-linear relationships in nuclear mass data
- More flexible than traditional models (e.g., WS4, HFB-31, DRHBc)
- Can integrate physics-based features for better interpretability
- Performs well even with moderate-sized datasets
- Balances accuracy and computational efficiency
- Can incorporate physical constraints into the learning process
- Achieves high precision (~70-180 keV) in nuclear mass predictions without requiring additional techniques such as kernel methods, boosting, etc.

◈ Dataset & Feature Selection

- Training & test data from **AME2020** (2,386 nuclei)
- **75% training (1,789 nuclei) / 25% test (597 nuclei)**
- Selected input features: **Z, N, binding energy, separation energies (Sn, S2n), pairing effects**

◈ Deep Learning Architectures

- **Deep Neural Networks (DNNs)**
  - Input features: **11I, 12I, 13I (selected physical quantities)**
  - Optimized structure for nuclear mass predictions $Z, N, A, A^{2/3}, (N-Z)/A, \nu_Z, \nu_N, PF, Z_{eo}, N_{eo}, Z_{\text{shell}}, N_{\text{shell}}, \delta$

- **[Convolutional Neural Networks (CNNs)** → Not covered in this presentation.
  - Input: **3-channel (Z, N, neighbor mass)** → **4-channel (Z, N, neighbor mass, pairing effect)**
  - Captures **local nuclear interactions** better than DNN

◈ Performance & Generalization Issues

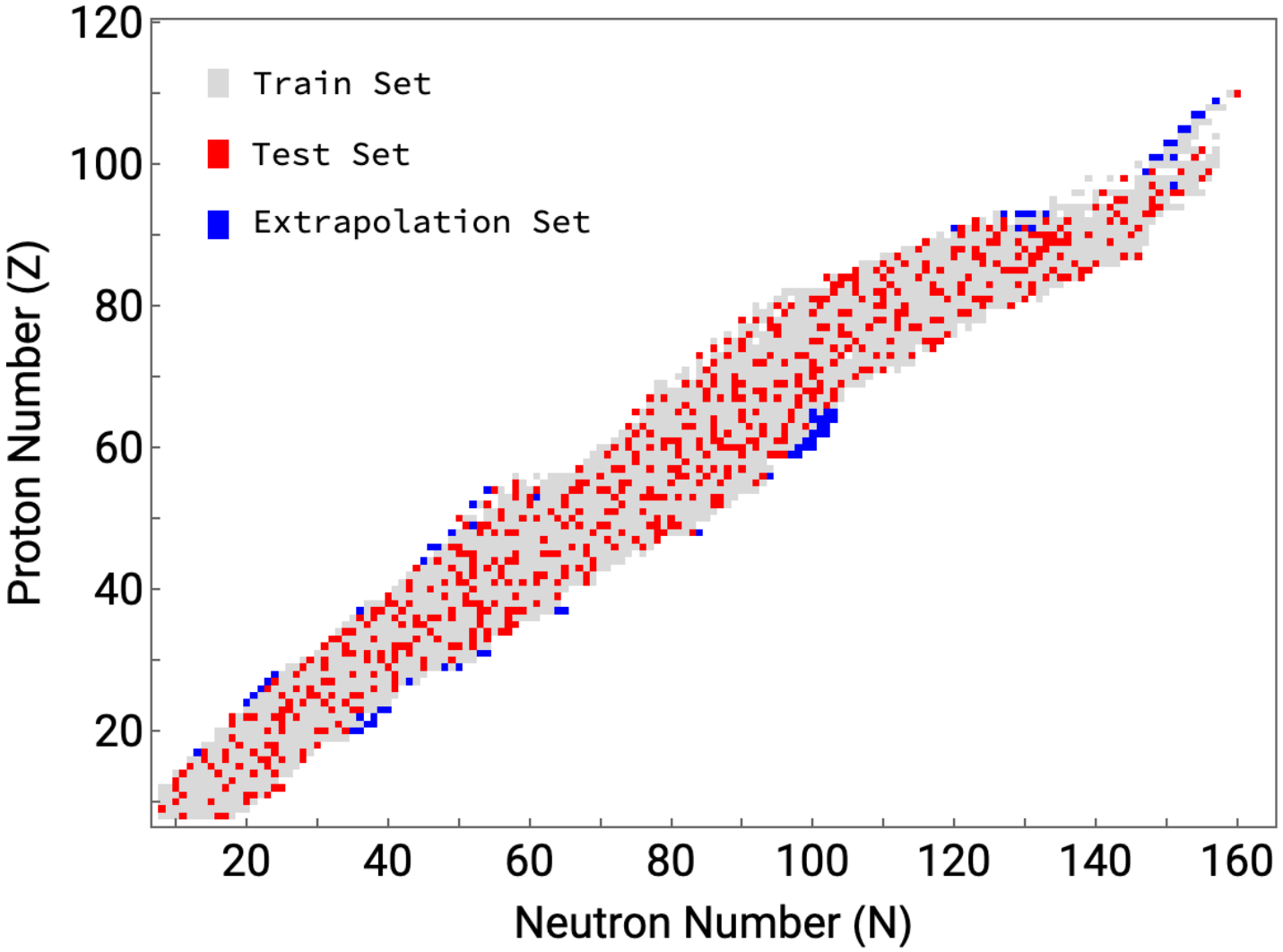
- DNN: **Best training accuracy = 63 keV, but validation = O(100~400) keV**
- *(CNN: **Higher accuracy (~70 keV), but poor generalization ← Skipped this time !)***
- To improve generalization: **Dropout, Batch Normalization, feature selection optimizations**

✅ **Dropout:** A regularization technique that randomly drops a fraction of neurons during training to prevent overfitting and improve generalization.

✅ **Batch Normalization:** A method that normalizes activations across a mini-batch to stabilize training, accelerate convergence, and reduce internal covariate shift.

✅ **Feature Selection Optimizations:** The process of selecting the most relevant input variables to enhance model interpretability, reduce overfitting, and improve performance.

AME2020: the training and test sets by 0.75:0.25



The experimental mass excess values are taken from the Atomic Mass Evaluation 2020 (AME2020) for nuclei with Z, N ≥ 8, covering 2386 nuclei. The data is split into two subsets: 75% (1789 nuclei) for training and 25% (597 nuclei) for testing, with the same division used for all calculations. Additionally, the performance of the ML models is evaluated beyond these subsets. AME2020 includes new experimental information for 71 nuclei compared to AME2016.

WS4 model

Physics Letters B 734 (2014) 215–219

Contents lists available at ScienceDirect

Physics Letters B

www.elsevier.com/locate/physletb

Surface diffuseness correction in global mass formula

Ning Wang [a,*], Min Liu [a], Xizhen Wu [b], Jie Meng [c,d]

[a] Department of Physics, Guangxi Normal University, Guilin 541004, PR China
[b] China Institute of Atomic Energy, Beijing 102413, PR China
[c] State Key Laboratory of Nuclear Physics and Technology, School of Physics, Peking University, Beijing 100871, PR China
[d] School of Physics and Nuclear Energy Engineering, Beihang University, Beijing 100191, PR China

$$E(A, Z, \beta) = E_{\mathrm{LD}}(A, Z) \prod_{k\geqslant 2} (1 + b_k \beta_k^2) + \Delta E(A, Z, \beta) + \Delta_{\mathrm{res}}.$$

$$E_{\mathrm{LD}}(A, Z) = a_{\mathrm{v}} A + a_{\mathrm{s}} A^{2/3} + a_{\mathrm{c}} \frac{Z^2}{A^{1/3}}(1 - 0.76 Z^{-2/3}) + a_{\mathrm{sym}} I^2 A f_{\mathrm{s}} + a_{\mathrm{pair}} A^{-1/3} \delta_{\mathrm{np}} + \Delta W,$$

microscopic shell correction is

$$\Delta E = c_1 f_{\mathrm{d}} E_{\mathrm{sh}} + |I| E'_{\mathrm{sh}},$$

other residual correction is expressed as

$$\Delta_{\mathrm{res}} = \Delta_M + \Delta_P + \Delta_T.$$

Table 2

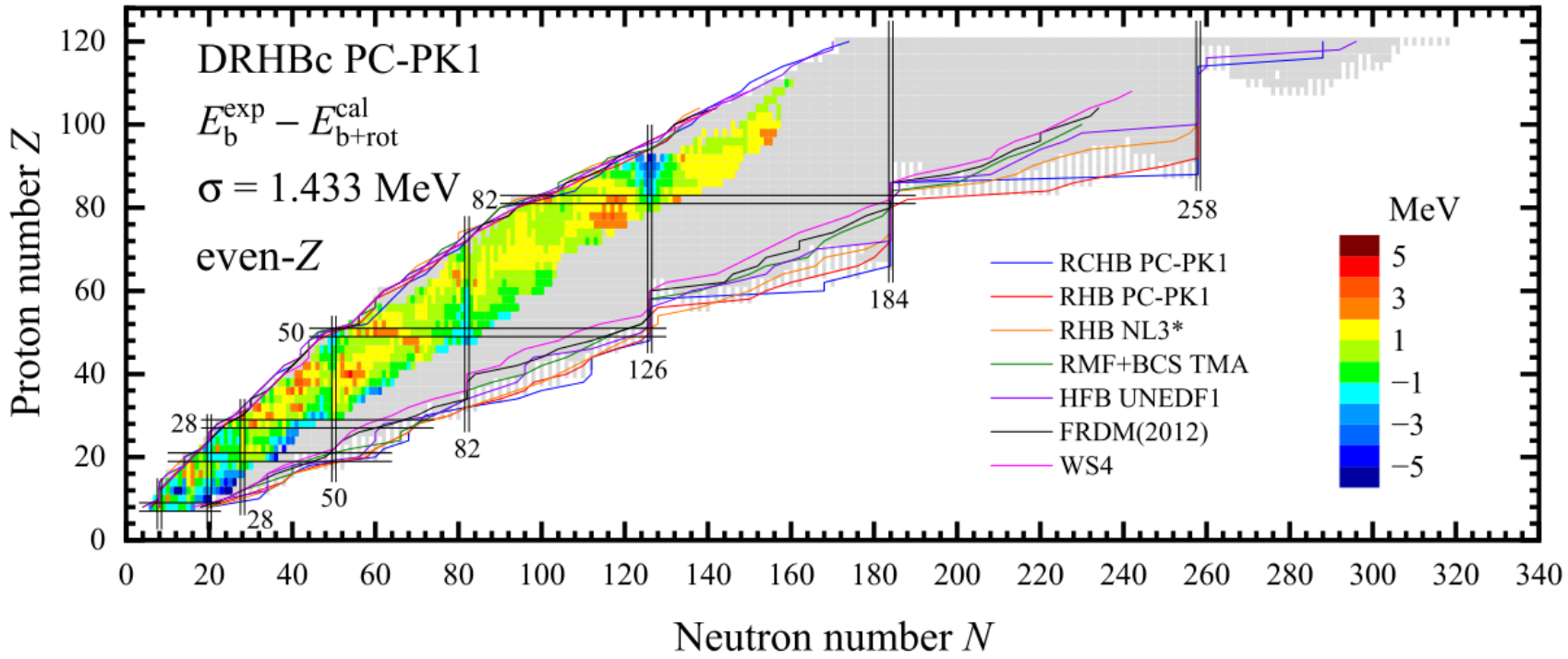
Rms deviations between data and predictions from the WS4 formula (in keV). The line $\sigma(M)$ refers to all the 2353 measured masses in AME2012, the line $\sigma(M_{\mathrm{new}})$ to the measured masses of 219 "new" nuclei in AME2012, the line $\sigma(M_{0.1})$ to the masses of 286 nuclei with $|I - I_0| > 0.1$, the line $\sigma(S_n)$ to all the 2199 measured neutron separation energies S_n, the line $\sigma(Q_\alpha)$ to the α-decay energies of 46 super-heavy nuclei ($Z \geq 106$) [14]. The corresponding results of WS3 model are also presented for comparison. WS4$^{\mathrm{RBF}}$ denotes that the radial basis function (RBF) corrections [46] are combined in the WS4 calculations.

	WS3	WS4	WS4$^{\mathrm{RBF}}$
$\sigma(M)$	335	298	170
$\sigma(M_{\mathrm{new}})$	424	346	155
$\sigma(M_{0.1})$	516	444	215
$\sigma(S_n)$	273	258	251
$\sigma(Q_\alpha)$	248	238	237

DRHBc Mass Table

From Dr. Myeong-Hwan Mun's talk

- The even-Z part of the DRHBc mass table has been completed.



- 2584 even-even nuclei and 2245 even-Z odd-N ones are predicted with $8 \leq Z \leq 120$.
- Ground-state properties, including rms radii, quadrupole deformations, d ensities, are produced.

Zhang *et al.*, (DRHBc Mass Table Collaboration) ADNDT 144, 101488 (2022)
Guo *et al.*, (DRHBc Mass Table Collaboration) ADNDT 158, 101661 (2024)

Comparison with WS4, DRHBc, and ML Models

◈ Model Performance Comparison

- **WS4 Model:** Mean absolute error (MAE) **~300 keV**
- **DRHBc Model:** MAE **~250 keV**
- **Our DNN Model:**
  - Best Training accuracy **~ 63 keV**
  - Validation accuracy **O(100~400) keV**
- **Our CNN Model:**
  - Training accuracy **~ 70 keV**
  - Validation accuracy **~O(100 ~ 1000) keV**

Model	Mean Absolute Error (MAE)	Key Features
WS4	~300 keV	Global fit, phenomenological
DRHBc	~250 keV	Density functional theory
FRDM12	~350 keV	Macroscopic-microscopic model
HFB-31	~270 keV	Self-consistent nuclear mean-field theory

◈ Key Observations

- ML models **outperform WS4 & DRHBc** but still face **generalization issues**
- CNN captures **local correlations** better but has **higher validation error**
- **Further improvements needed for better extrapolation performance**

Results and Analysis
(Supervised learning for nuclear mass prediction)

Input parameters

$$Z, N, A, A^{2/3}, (N-Z)/A, \nu_Z, \nu_N, PF, Z_{eo}, N_{eo}, Z_{\text{shell}}, N_{\text{shell}}, \delta$$

- ✅ **Z** – Proton number: Total number of protons in the nucleus.
- ✅ **N** – Neutron number: Total number of neutrons in the nucleus.
- ✅ **A** – Mass number: Total number of nucleons (A= Z + N).
- ✅ **A^(2/3)** – Mass number scaling factor: Represents the surface term in nuclear mass models, derived from the liquid-drop model.
- ✅ **(N−Z)/A** – Isospin asymmetry: A measure of the neutron-proton imbalance, which affects nuclear stability.
- ✅ **ν_Z** – Proton valence number: The number of protons outside the nearest closed shell.
- ✅ **ν_N** – Neutron valence number: The number of neutrons outside the nearest closed shell.
- ✅ **P_F** – Promiscuity factor: Defined as PF=(ν_Z * ν_N) / (ν_Z + ν_N), it quantifies the proton-neutron interactions.

$$\mathrm{PF} = \frac{\nu_Z \nu_N}{\nu_Z + \nu_N}$$

- ✅ **Z_eo** – Proton even-odd indicator: 0 if Z is even, 1 if Z is odd.
- ✅ **N_eo** – Neutron even-odd indicator: 0 if N is even, 1 if N is odd.
- ✅ **Z_shell** – Proton shell model orbital: Represents the nuclear shell level of the last proton, categorized as 0, 1, 2, 3, or 4.
- ✅ **N_shell** – Neutron shell model orbital: Represents the nuclear shell level of the last neutron, categorized similarly to Z_shell.
- ✅ **δ**– Pairing term: Accounts for additional energy corrections due to nucleon pairing effects.

Pairing effect:

$$\delta = [(-1)^N + (-1)^Z]/2$$

(Deep) Neural Nets 11 Inputs(≡ $I11$):

```
I11trainedNN2e =
 ParallelEvaluate[NetTrain[I11NN3, NNtrainingData, ValidationSet → NNtestData,
  Method → {"ADAM", "Beta1" → 0.9, "Beta2" → 0.999, "Epsilon" → 0.00001, "L2Regularization" → 0.01},
  BatchSize → 50, TargetDevice → "CPU", MaxTrainingRounds → 2 * 10 ^ 5], kernelsOnGPU1[[1]]]
Starting training.
```

병렬 평가 | 네트워크 훈련 | 검증 집합 | 방법 | 배치 사이즈 | 대상 장치 | 훈련 라운드의 최대수

NetChain

	Input	vector (size: 11)
1	LinearLayer	vector (size: 22)
2	Ramp	vector (size: 22)
3	LinearLayer	vector (size: 44)
4	Ramp	vector (size: 44)
5	LinearLayer	vector (size: 22)
6	Ramp	vector (size: 22)
7	LinearLayer	vector (size: 1)
	Output	scalar

Predicted Nucleus Mass

Parameter	Value	Description
"Beta1"	0.9	Controls the exponential decay rate of the first moment estimate (moving average of gradients).
"Beta2"	0.999	Controls the exponential decay rate of the second moment estimate (moving average of squared gradients).
"Epsilon"	0.00001	A small constant added for numerical stability to prevent division by zero.
"L2Regularization"	0.01	Applies L2 weight decay to prevent overfitting by penalizing large model parameters.

☑ ADAM (Adaptive Moment Estimation) Optimizer

- A widely used optimization algorithm that combines momentum and adaptive learning rates for efficient and stable training.
- Helps prevent vanishing gradients and accelerates convergence in deep learning models.

Feature space
$Z, N, A, A^{2/3}, (N-Z)/A, \delta$
$Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}$
11 Inputs(≡ $11I$): $Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}, Z_{\mathrm{eo}}, N_{\mathrm{eo}}$
13 Inputs(≡ $13I$): $Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}, Z_{\mathrm{eo}}, N_{\mathrm{eo}}, Z_{\mathrm{shell}}, N_{\mathrm{shell}}, \delta$

```
I11trainedNN4e =
 ParallelEvaluate[NetTrain[I11NN4, NNtrainingData, ValidationSet → NNtestData,
 병렬 평가                 네트워크 훈련                검증 집합
    Method → {"ADAM", "Beta1" → 0.9, "Beta2" → 0.999, "Epsilon" → 0.00001, "L2Regularization" → 0.01},
    방법
    BatchSize → 50, TargetDevice → "CPU", MaxTrainingRounds → 2 * 10 ^ 5], kernelsOnGPU1[[1]]]
    배치 사이즈          대상 장치          훈련 라운드의 최대수
```

```
Starting training.
```

NetChain

	Input	vector (size: 11)
1	LinearLayer	vector (size: 22)
2	Ramp	vector (size: 22)
3	LinearLayer	vector (size: 44)
4	Ramp	vector (size: 44)
5	LinearLayer	vector (size: 88)
6	Ramp	vector (size: 88)
7	LinearLayer	vector (size: 44)
8	Ramp	vector (size: 44)
9	LinearLayer	vector (size: 22)
10	Ramp	vector (size: 22)
11	LinearLayer	vector (size: 11)
12	Ramp	vector (size: 11)
13	LinearLayer	vector (size: 1)
	Output	scalar

```
NetMeasurements[I11trainedNN4e, NNtrainingData, "StandardDeviation"]
네트워크 측정
NetMeasurements[I11trainedNN4e, NNtestData, "StandardDeviation"]
네트워크 측정
```

```
172.472

424.367
```

← RMS

✅ Batch Size

- The number of training samples processed before updating the model's parameters.
- A key hyperparameter that affects training speed, memory usage, and model generalization.

Impact of Batch Size on Training

Batch Size	Characteristics	Pros	Cons
Small (e.g., 16, 32)	Uses fewer samples per update	More generalization, reduces overfitting	Noisy updates, slower convergence
Medium (e.g., 64, 128)	Balance between stability and efficiency	Stable convergence, efficient training	May require tuning for optimal performance
Large (e.g., 256, 512, 1024)	Processes more data per step	Faster training, smoother updates	Requires more memory, risk of poor generalization

11 Inputs($\equiv 11I$): $Z, N, A, A^{2/3}, (N-Z)/A, \nu_Z, \nu_N, \mathrm{PF}, Z_{\mathrm{eo}}, N_{\mathrm{eo}}, \delta$

13 Inputs($\equiv 13I$): $Z, N, A, A^{2/3}, (N-Z)/A, \nu_Z, \nu_N, \mathrm{PF}, Z_{\mathrm{eo}}, N_{\mathrm{eo}}, Z_{\mathrm{shell}}, N_{\mathrm{shell}}, \delta$

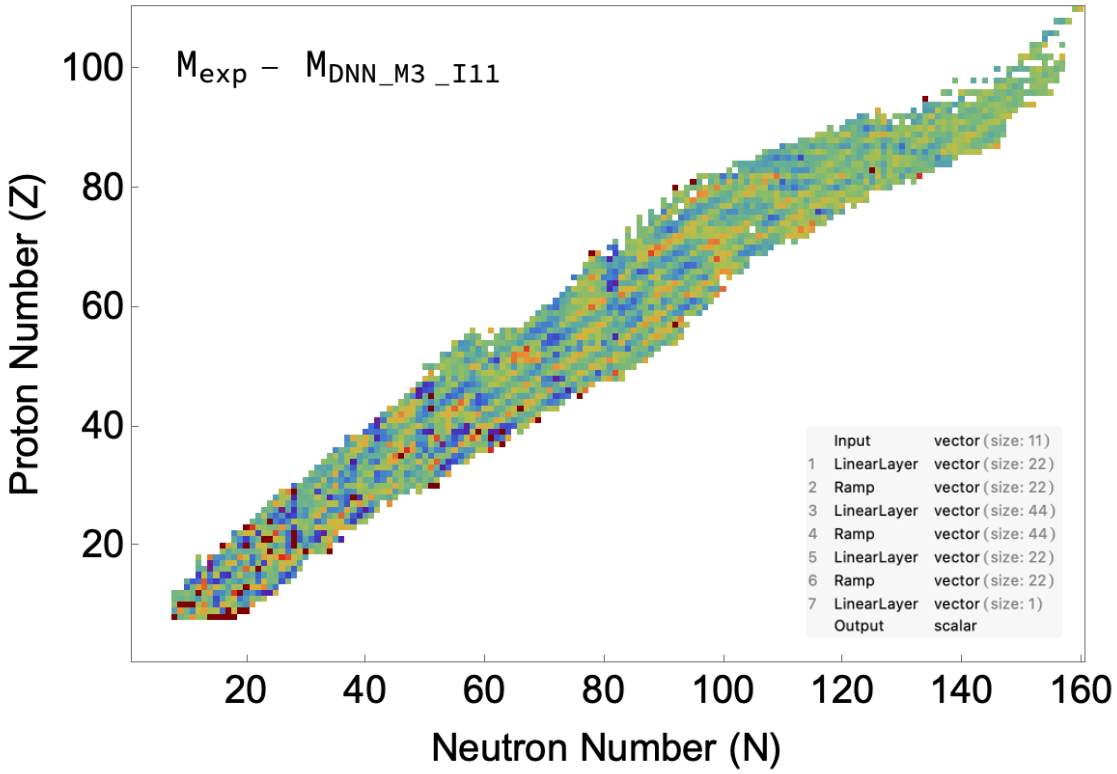
```
NetMeasurements[I11trainedNN2e, NNtrainingData, "StandardDeviation"]
```
네트워크 측정

```
NetMeasurements[I11trainedNN2e, NNtestData, "StandardDeviation"]
```
네트워크 측정

326.501

567.193

← RMS



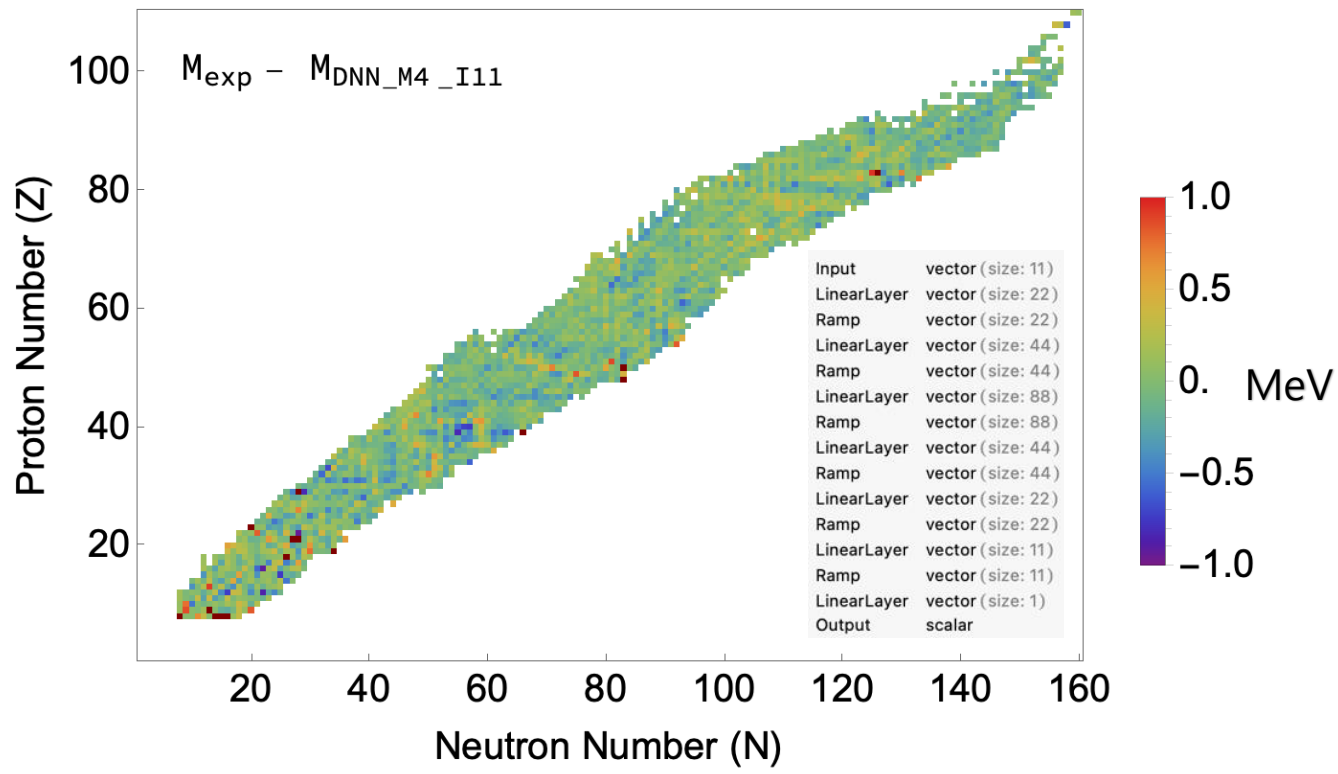

```
NetMeasurements[I11trainedNN4e, NNtrainingData, "StandardDeviation"]
```
네트워크 측정

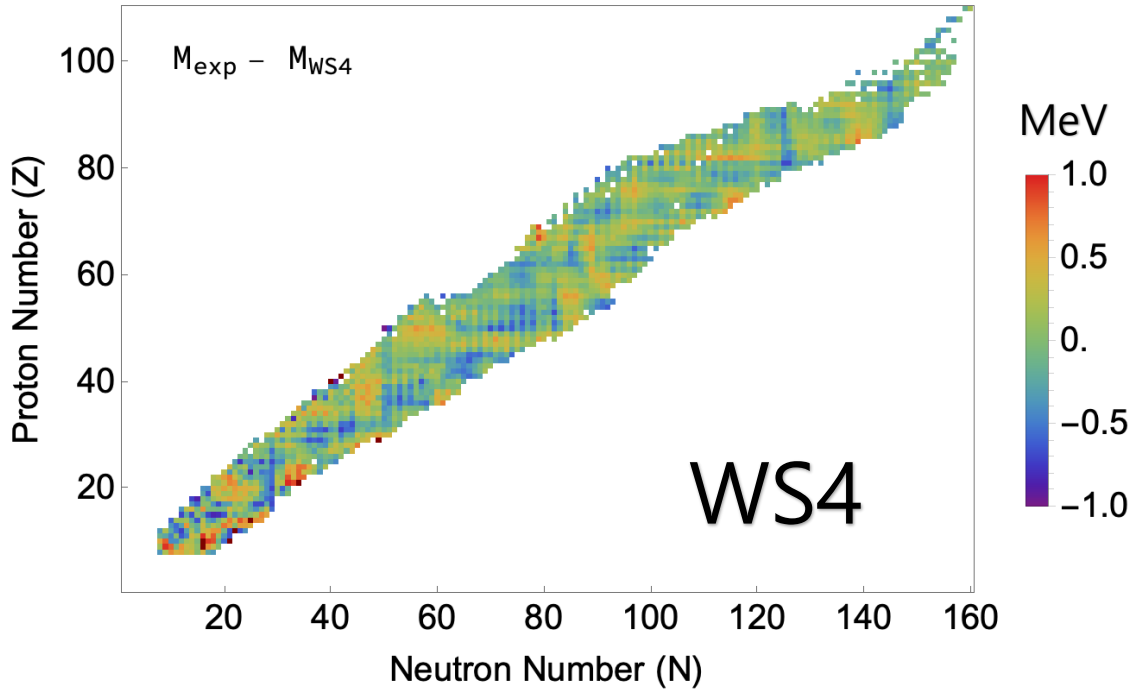
```
NetMeasurements[I11trainedNN4e, NNtestData, "StandardDeviation"]
```
네트워크 측정

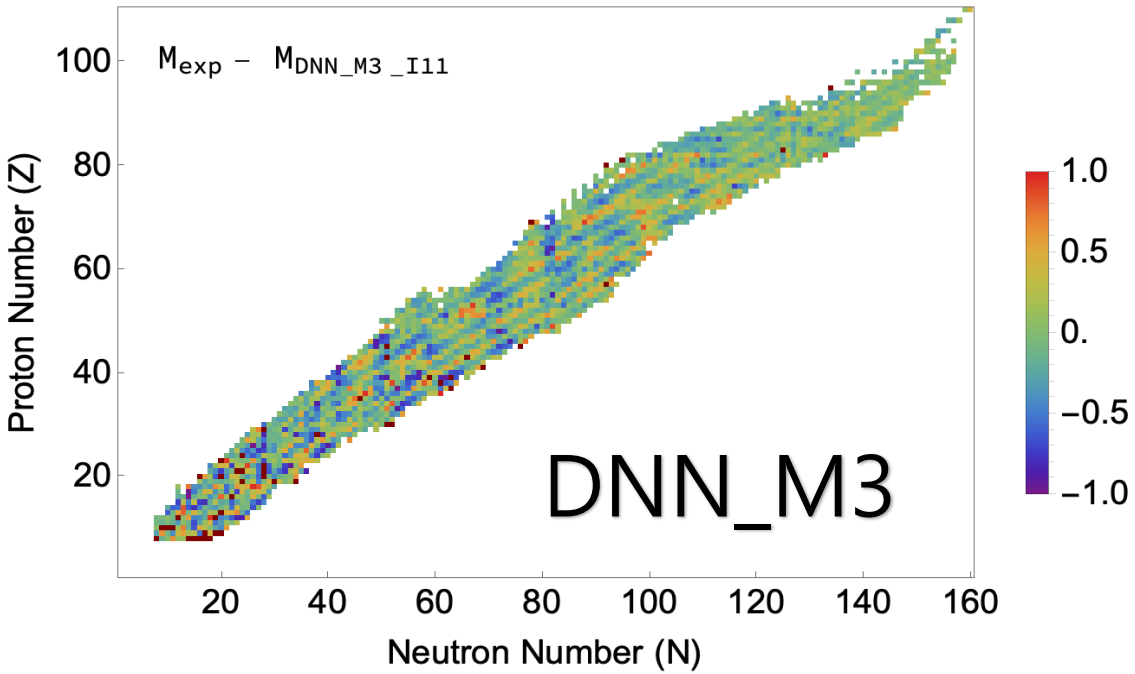
172.472

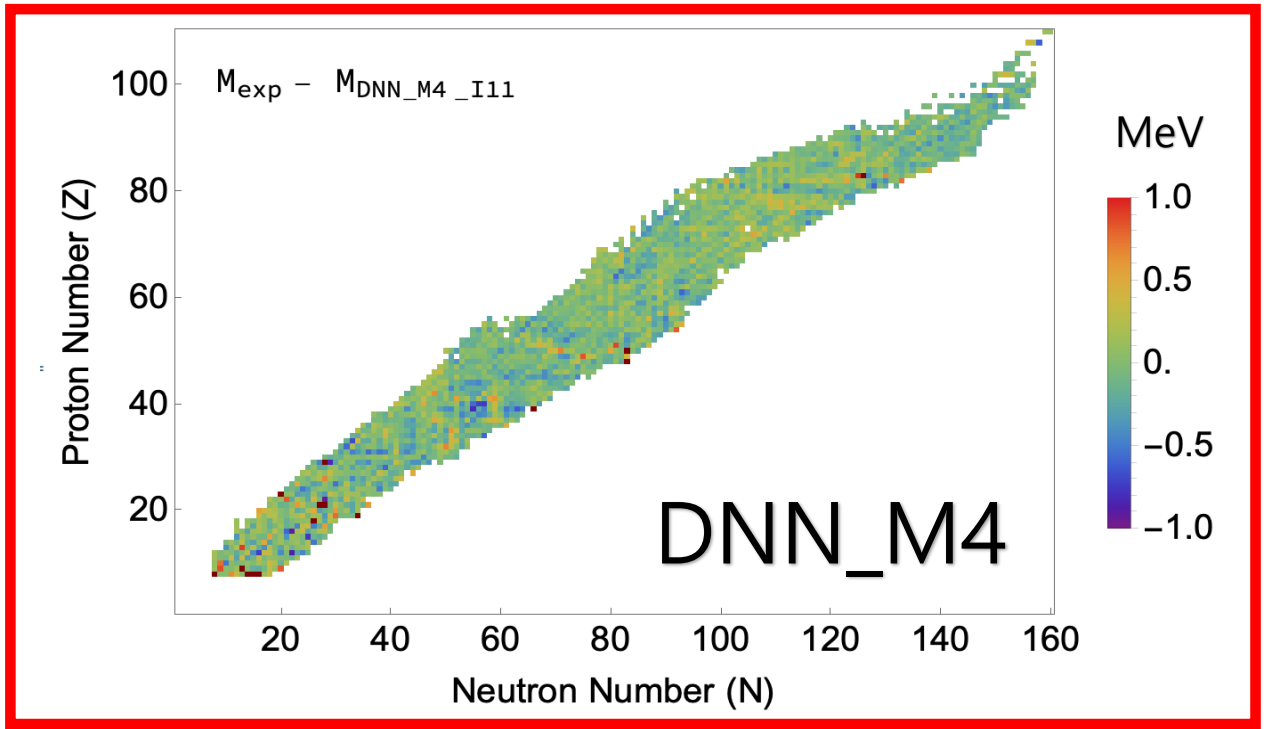
424.367

← RMS




$M_{exp} - M_{WS4}$
WS4
Proton Number (Z)
Neutron Number (N)
MeV
1.0
0.5
0.
−0.5
−1.0


$M_{exp} - M_{DNN\_M3\_I11}$
DNN_M3
Proton Number (Z)
Neutron Number (N)
1.0
0.5
0.
−0.5
−1.0


$M_{exp} - M_{DNN\_M4\_I11}$
DNN_M4
Proton Number (Z)
Neutron Number (N)
MeV
1.0
0.5
0.
−0.5
−1.0

(Deep) Neural Nets 13 Inputs(≡ $I13$):

```
I13trainedNN3e =
 ParallelEvaluate[NetTrain[I13NN3, NNtrainingData, ValidationSet → NNtestData,
 |병렬 평가              |네트워크 훈련                |검증 집합
    Method → {"ADAM", "Beta1" → 0.9, "Beta2" → 0.999, "Epsilon" → 0.00001, "L2Regularization" → 0.01},
    |방법
    BatchSize → 50, TargetDevice → "CPU", MaxTrainingRounds → 10^5], kernelsOnGPU1[[1]]]
    |배치 사이즈          |대상 장치          |훈련 라운드의 최대수
```

```
Starting training.
```

```
NetChain[
  Input          vector (size: 13)
1 LinearLayer    vector (size: 26)
2 Ramp           vector (size: 26)
3 LinearLayer    vector (size: 52)
4 Ramp           vector (size: 52)
5 LinearLayer    vector (size: 26)
6 Ramp           vector (size: 26)
7 LinearLayer    vector (size: 1)
  Output         scalar
]
```

```
NetMeasurements[I13trainedNN3e, NNtrainingData, "StandardDeviation"]
|네트워크 측정
NetMeasurements[I13trainedNN3e, NNtestData, "StandardDeviation"]
|네트워크 측정
```

```
247.4

472.429
```

← RMS

Pairing effect:

$$\delta = [(-1)^N + (-1)^Z]/2$$

$$\mathrm{PF} = \frac{\nu_Z \nu_N}{\nu_Z + \nu_N}$$

Feature space
$Z, N, A, A^{2/3}, (N-Z)/A, \delta$
$Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}, \delta$
11 Inputs(≡ $11I$): $Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}, Z_{eo}, N_{eo}, \delta$
13 Inputs(≡ $13I$): $Z, N, A, A^{2/3}, (N-Z)/A,$ $\nu_Z, \nu_N, \mathrm{PF}, Z_{eo}, N_{eo}, Z_{shell}, N_{shell}, \delta$

```
I13trainedNN4e =
 ParallelEvaluate[NetTrain[I13NN4, NNtrainingData, ValidationSet → NNtes
 병렬 평가                 네트워크 훈련                검증 집합
    Method → {"ADAM", "Beta1" → 0.9, "Beta2" → 0.999, "Epsilon" → 0.00001, "
    방법
    BatchSize → 50, TargetDevice → "CPU", MaxTrainingRounds → 2 * 10 ^ 5], ker
    배치 사이즈              대상 장치              훈련 라운드의 최대수

Starting training.
```

NetChain

	Input	vector (size: 13)
1	LinearLayer	vector (size: 26)
2	Ramp	vector (size: 26)
3	LinearLayer	vector (size: 52)
4	Ramp	vector (size: 52)
5	LinearLayer	vector (size: 104)
6	Ramp	vector (size: 104)
7	LinearLayer	vector (size: 52)
8	Ramp	vector (size: 52)
9	LinearLayer	vector (size: 26)
10	Ramp	vector (size: 26)
11	LinearLayer	vector (size: 13)
12	Ramp	vector (size: 13)
13	LinearLayer	vector (size: 1)
	Output	scalar

```
NetMeasurements[I13trainedNN4e, NNtrainingData, "StandardDeviation"]
네트워크 측정
NetMeasurements[I13trainedNN4e, NNtestData, "StandardDeviation"]
네트워크 측정
```

100.083

465.667

← RMS

```
(kernel 7) Starting training.
(kernel 7) Optimization Method: ADAM
           Beta1: 9.00*^-1
           Beta2: 9.99*^-1
           Epsilon: 1.00*^-5
           Gradient Clipping:  –
           L2 Regularization: 1.00*^-2
           Learning Rate: Automatic
           Learning Rate Schedule:  –
           Weight Clipping:  –
(kernel 7) Device: CPU
(kernel 7) Batch Size: 50
(kernel 7) Batches Per Round: 37
```

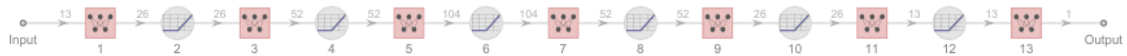


Figure 1: Illustration of the neural network architecture used in this study. The network begins on the left with an input layer of 13 nodes, representing the nuclear features. This input is processed sequentially through 13 layers. Each layer is represented by a red rectangular icon, denoting a fully connected or dense `LinearLayer` or a gray circular icon, which represents a `Ramp` activation function, also known as `ReLU` (Rectified Linear Units). The number above each arrow represents the number of nodes in that particular layer. The output layer, depicted on the right, produces a scalar value that corresponds to the predicted mass of the nucleus. The numbers below each icon indicate the layer's sequence in the network.

	time left	current loss	round loss	test loss
	3h22m32s	1.28*^+9	6.32*^+7	5.74*^+7
	3h44m05s	3.42*^+7	1.58*^+7	1.74*^+7
	4h06m26s	9.99*^+6	7.23*^+6	8.42*^+6
	4h19m37s	5.94*^+6	5.93*^+6	6.27*^+6

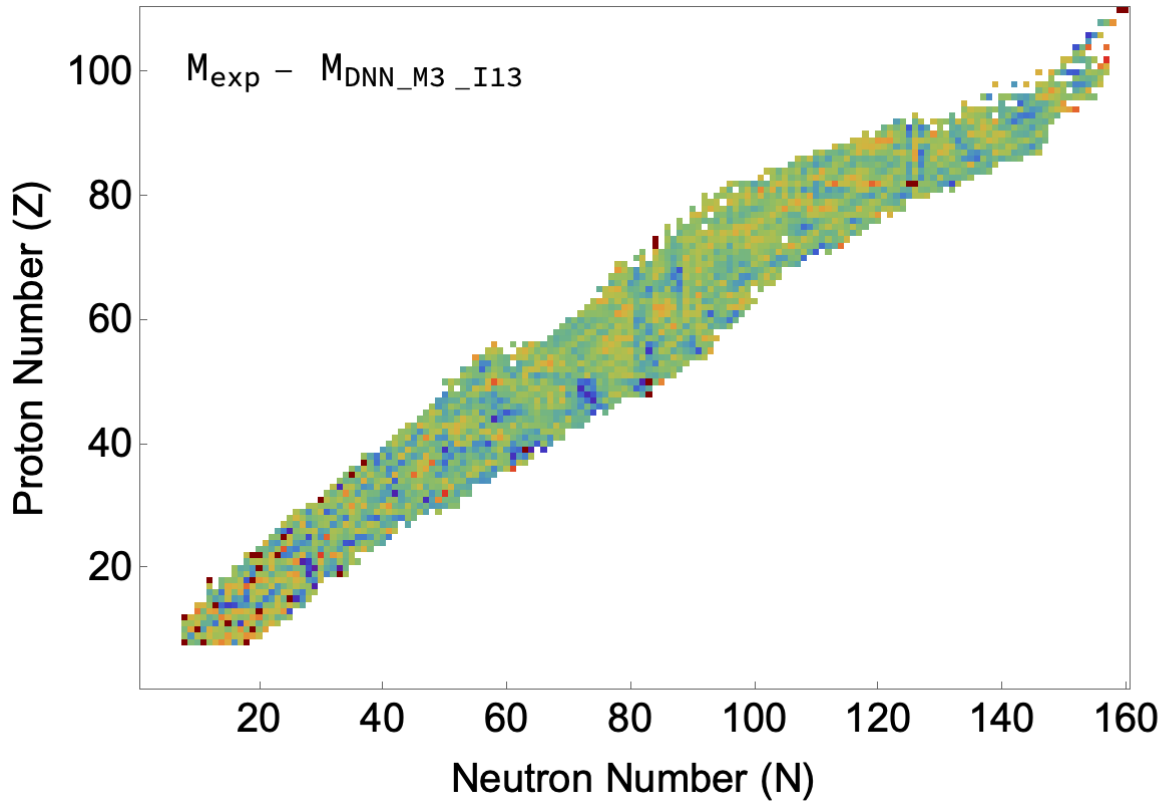
(kernel 7)	98	195915	2	362441000	19691	1.00*^-3	4h02m25s	5m03s	2.35*^+4	2.77*^+3	5.13*^+5
(kernel 7)	98	195943	25	362493950	29897	1.00*^-3	4h02m27s	5m01s	1.03*^+4	2.93*^+4	5.23*^+5
(kernel 7)	98	195972	7	362546700	23190	1.00*^-3	4h02m29s	4m59s	2.60*^+4	3.74*^+3	5.23*^+5

(kernel 7)	100	199707	12	369456700	25732	1.00*^-3	4h06m59s	22s	1.45*^+4	7.72*^+4	5.98*^+5
(kernel 7)	100	199735	32	369509500	30786	1.00*^-3	4h07m01s	20s	1.45*^+4	2.14*^+4	6.26*^+5
(kernel 7)	100	199763	37	369561550	29404	1.00*^-3	4h07m03s	18s	1.94*^+4	1.09*^+4	5.29*^+5
(kernel 7)	100	199792	2	369613450	19658	1.00*^-3	4h07m05s	16s	1.64*^+4	2.36*^+4	6.10*^+5
(kernel 7)	100	199818	1	369661500	16599	1.00*^-3	4h07m07s	14s	1.33*^+4	1.31*^+4	5.34*^+5
(kernel 7)	100	199843	12	369708300	24178	1.00*^-3	4h07m09s	12s	2.31*^+4	1.09*^+4	5.12*^+5
(kernel 7)	100	199870	20	369758650	27836	1.00*^-3	4h07m11s	10s	2.73*^+4	1.47*^+4	5.05*^+5
(kernel 7)	100	199898	6	369809750	22341	1.00*^-3	4h07m13s	8s	1.63*^+4	5.34*^+4	5.22*^+5
(kernel 7)	100	199925	37	369861250	30611	1.00*^-3	4h07m15s	6s	2.39*^+4	4.12*^+3	5.11*^+5
(kernel 7)	100	199953	3	369911350	20347	1.00*^-3	4h07m17s	4s	1.66*^+4	1.85*^+4	5.00*^+5
(kernel 7)	100	199979	2	369959400	18459	1.00*^-3	4h07m19s	2s	1.42*^+4	4.84*^+4	5.31*^+5

```
NetMeasurements[I13trainedNN3e, NNtrainingData, "StandardDeviation"]
```
네트워크 측정

```
NetMeasurements[I13trainedNN3e, NNtestData, "StandardDeviation"]
```
네트워크 측정

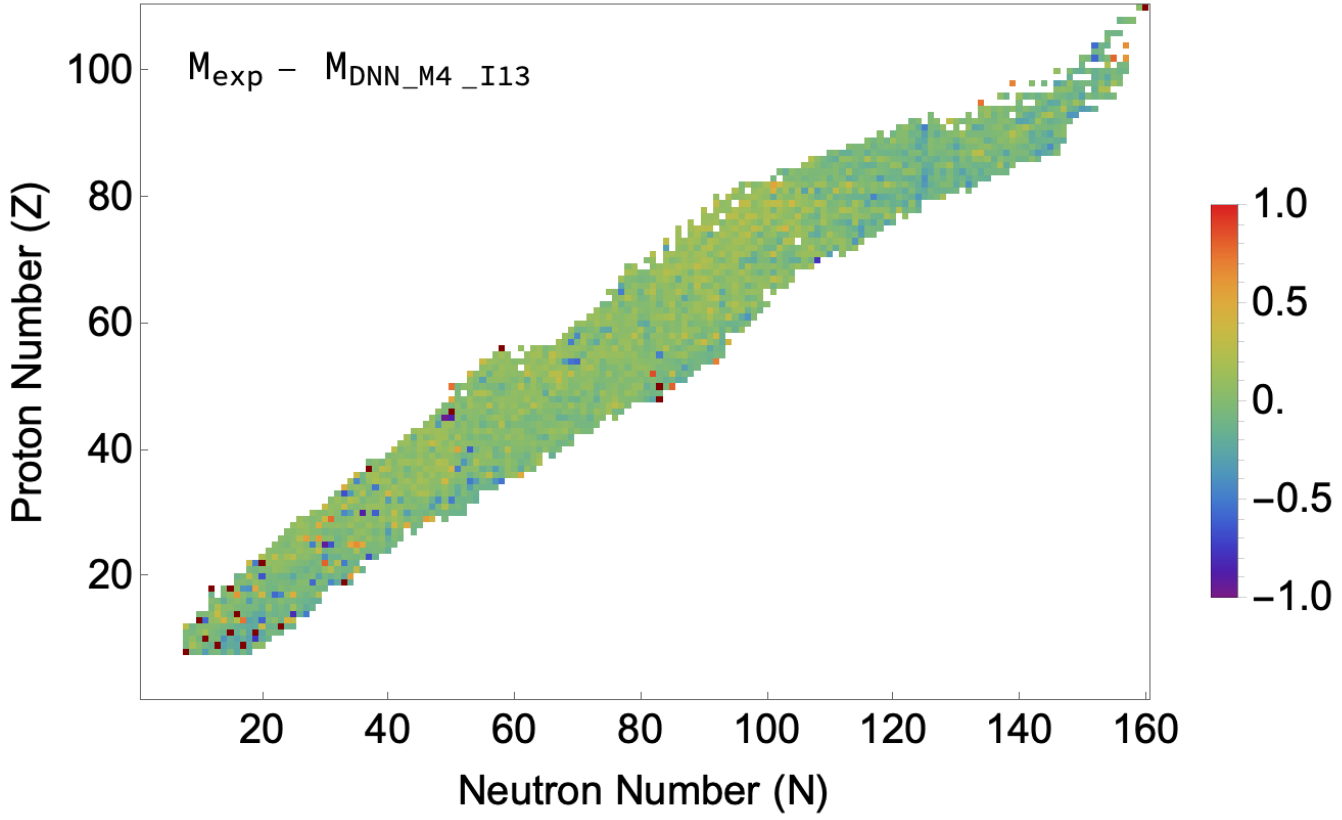
```
247.4

472.429
```
← RMS

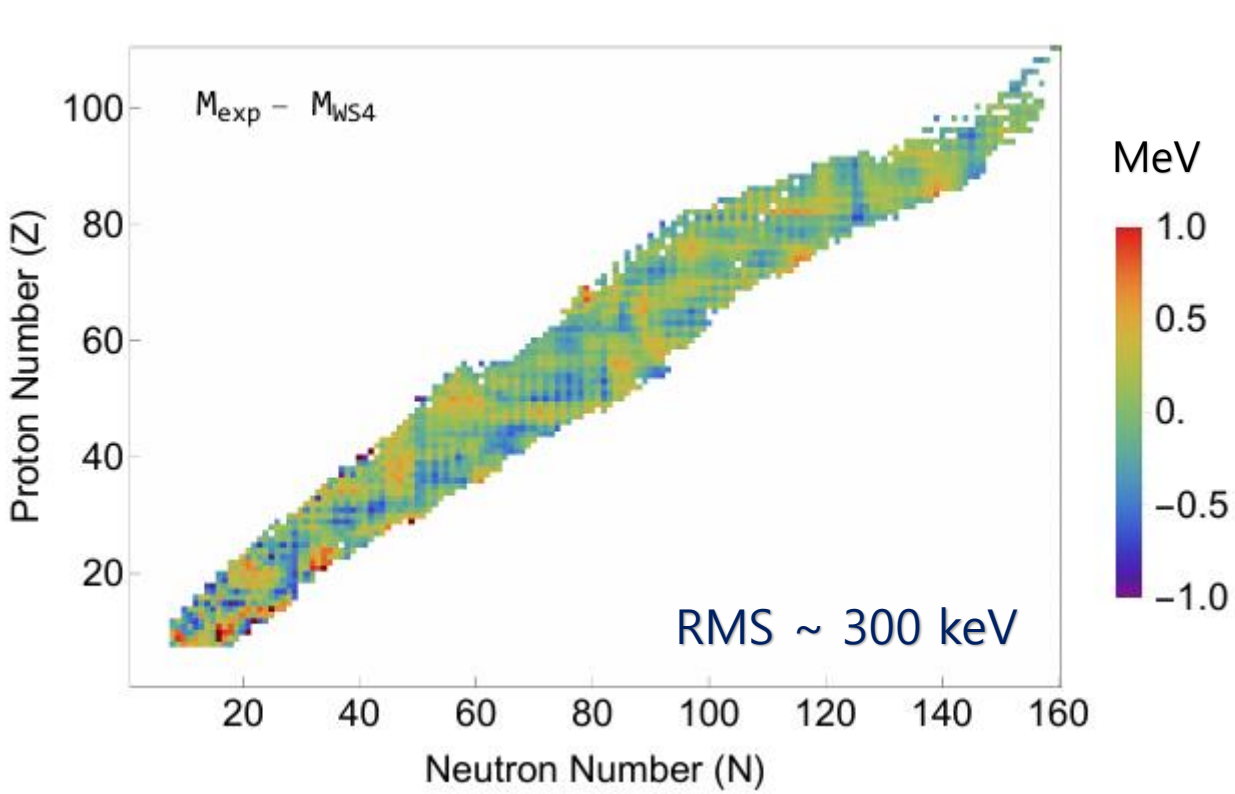


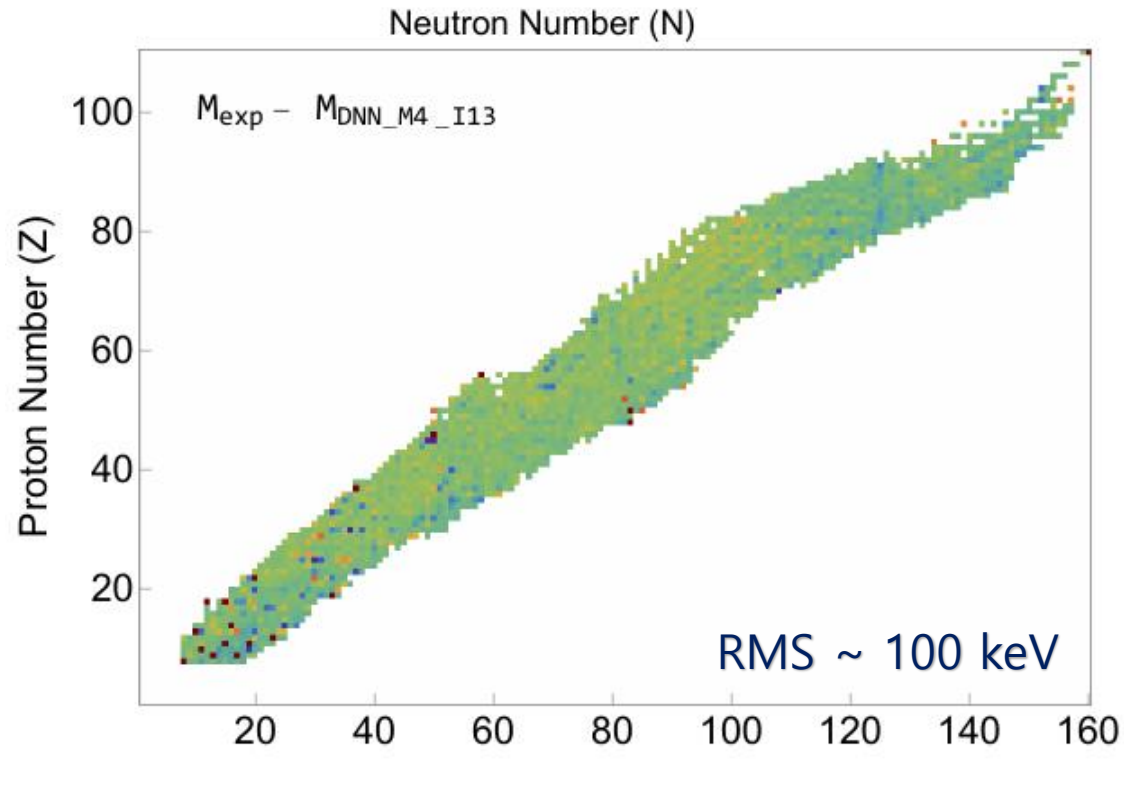

```
NetMeasurements[I13trainedNN4e, NNtrainingData, "StandardDeviation"]
```
네트워크 측정

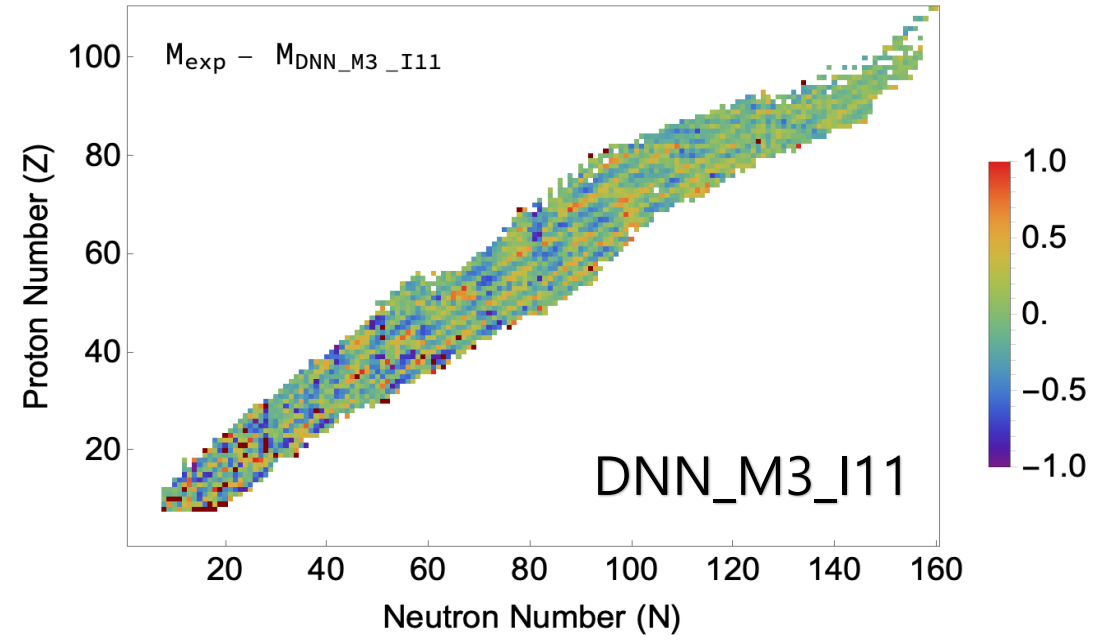
```
NetMeasurements[I13trainedNN4e, NNtestData, "StandardDeviation"]
```
네트워크 측정

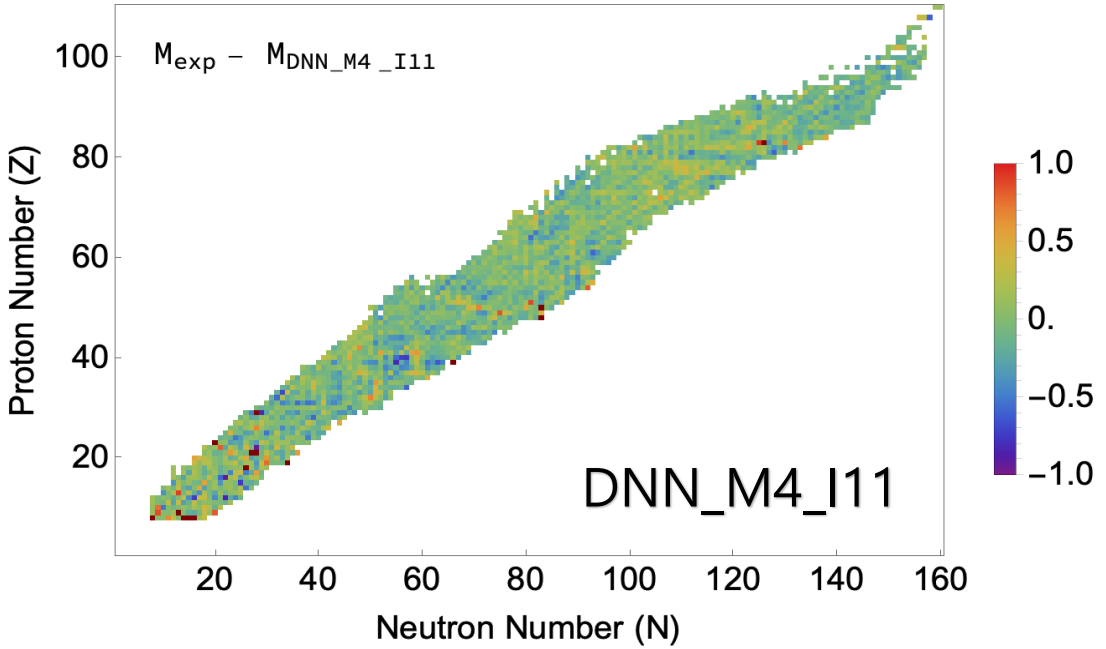
```
100.083

465.667
```
← RMS

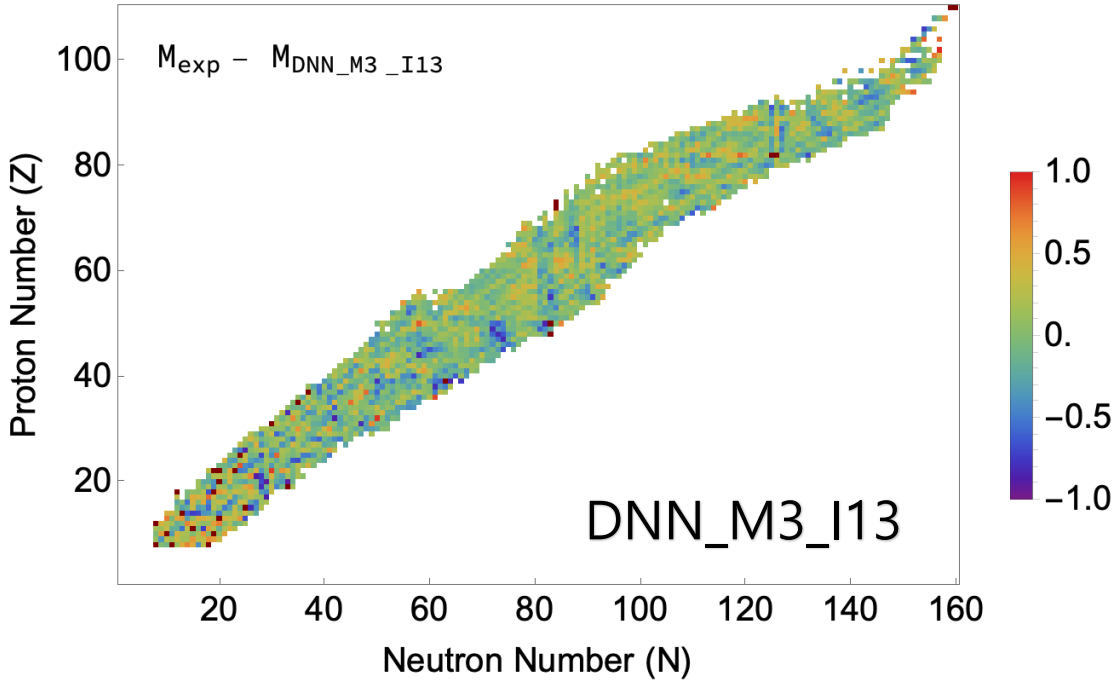


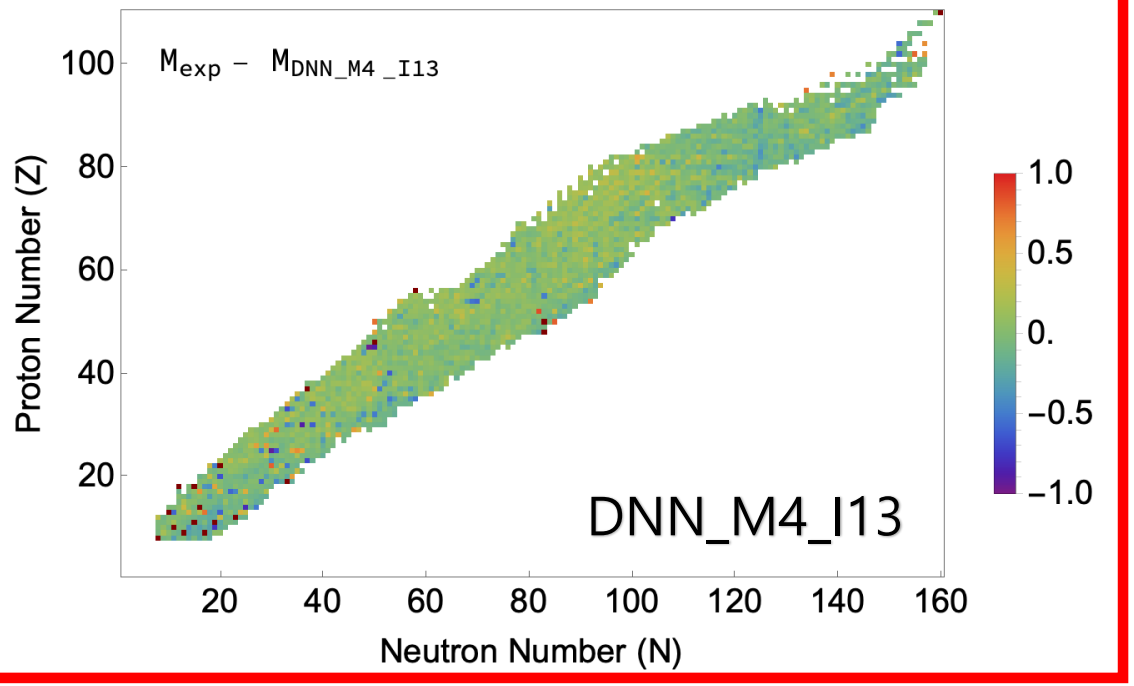

$M_{exp} - M_{WS4}$
Proton Number (Z)
Neutron Number (N)
MeV
1.0
0.5
0.
−0.5
−1.0
RMS ~ 300 keV


Neutron Number (N)
$M_{exp} - M_{DNN\_M4\_I13}$
Proton Number (Z)
RMS ~ 100 keV


$M_{exp} - M_{DNN\_M3\_I11}$
DNN_M3_I11
Proton Number (Z)
Neutron Number (N)
1.0
0.5
0.
−0.5
−1.0


$M_{exp} - M_{DNN\_M4\_I11}$
DNN_M4_I11
Proton Number (Z)
Neutron Number (N)
1.0
0.5
0.
−0.5
−1.0


$M_{exp} - M_{DNN\_M3\_I13}$
DNN_M3_I13
Proton Number (Z)
Neutron Number (N)
1.0
0.5
0.
−0.5
−1.0


$M_{exp} - M_{DNN\_M4\_I13}$
DNN_M4_I13
Proton Number (Z)
Neutron Number (N)
1.0
0.5
0.
−0.5
−1.0

DNN_I12 : Our Optimized DNN model

```
In[16]:= SetDirectoryDNN = "/Users/jubinpark/Dropbox/2024_숭실대_OMEG/2024년_핵물리_인공지능/Data";

In[30]:= I12trainedNN5e = Import[SetDirectoryDNN <> "/I12NN4_146keV.wlnet"]
```

Out[30]= NetChain

	Input	vector (size: 12)
1	LinearLayer	vector (size: 26)
2	Ramp	vector (size: 26)
3	LinearLayer	vector (size: 52)
4	Ramp	vector (size: 52)
5	LinearLayer	vector (size: 104)
6	Ramp	vector (size: 104)
7	LinearLayer	vector (size: 52)
8	Ramp	vector (size: 52)
9	LinearLayer	vector (size: 26)
10	Ramp	vector (size: 26)
11	LinearLayer	vector (size: 13)
12	Ramp	vector (size: 13)
13	LinearLayer	vector (size: 1)
	Output	scalar

```
I12NNtrainingData = Import[SetDirectoryDNN <> "/I12_146keV_NNtrainingData_NN4.mx"];

I12NNtestData = Import[SetDirectoryDNN <> "/I12_146kev_NNtestData_NN4.mx"];

NetMeasurements[I12trainedNN5e, I12NNtrainingData, "StandardDeviation"]
NetMeasurements[I12trainedNN5e, I12NNtestData, "StandardDeviation"]
```

RMS →

```
180.162

162.569
```

※ This model achieves outstanding generalization ability !

```
I12NNtrainingData[[2]]
I12NNtrainingData[[2, 2]]
I12trainedNN5e[NNtrainingData[[2, 1]]]
```

$$\left\{103, 77, 180, 31.8798, 0.144444, 21, 27, \frac{189}{16}, 1, 1, 3, 2\right\} \to -37978$$

```
-37978

-37736.6
```

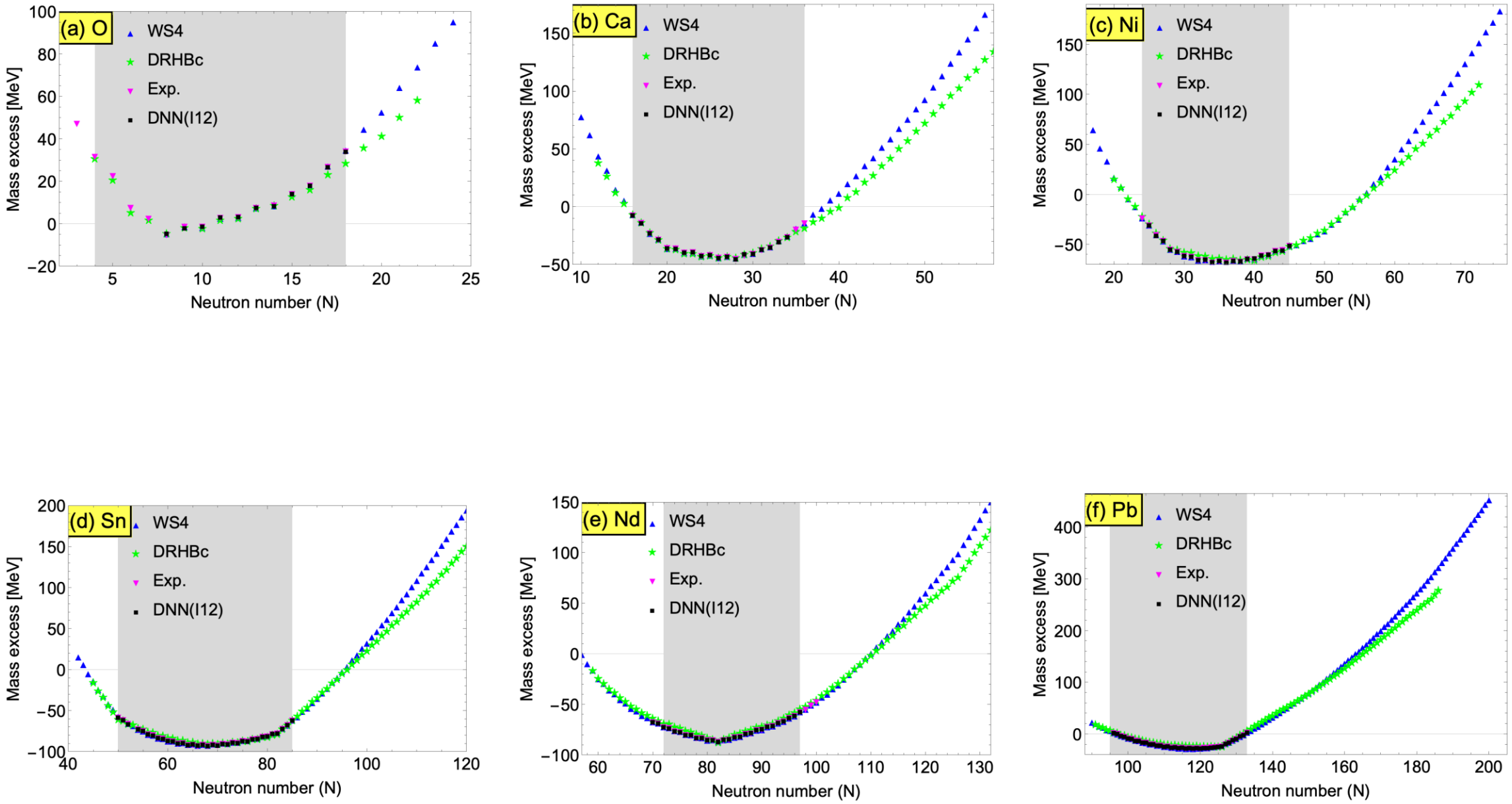
Mass deviation: 241.4 keV

```
I12NNtestData[[2]]
I12NNtestData[[2, 2]]
I12trainedNN5e[I12NNtestData[[2, 1]]]
```

$$\left\{62, 46, 108, 22.6786, 0.148148, 12, 18, \frac{36}{5}, 0, 0, 2, 1\right\} \to -89524.2$$

```
-89524.2

-89500.4
```

Mass deviation: 23.8 keV


(a) O
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)
(b) Ca
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)
(c) Ni
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)
(d) Sn
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)
(e) Nd
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)
(f) Pb
WS4
DRHBc
Exp.
DNN(I12)
Mass excess [MeV]
Neutron number (N)

Conclusion & Future Work(s)

✅ Key Achievements

- Developed an optimized **Deep Neural Network (DNN)** model for nuclear mass predictions.
- Achieved **high accuracy (~70–180 keV)**, outperforming traditional models like WS4 and DRHBc.
- Improved generalization through **feature selection, dropout, and batch normalization**.

✅ Challenges & Limitations

- **Generalization gap**: Validation accuracy still varies (100–400 keV).
- **Physical interpretability**: Need better integration of nuclear physics constraints.
- **Extrapolation issues**: Performance drops for neutron-rich nuclei.

✅ Future Improvements

- **Enhancing generalization**: Data augmentation & better feature engineering.
- **Exploring advanced ML architectures**: Transformers, Bayesian models.
- **Physics-informed ML**: Combining deep learning with nuclear theory constraints.
- **Towards high-precision mass predictions**: Reducing RMS error **below 100 keV**.

✅ Long-Term Goals

- Develop a world-class **ML-based nuclear mass model**.
- Improve **r-process nucleosynthesis predictions**.
- Expand ML applications in **nuclear astrophysics and fundamental physics**.

Thank you for your attention!