CALU

: **C**omplex-Valued Neural **A**rithmetic **L**ogic **U**nit with Neural Quantizer for the Abstraction of Physical Symmetries in the Nature

Wonsang Cho
(Seoul National University)

in collaboration with
Sungyeop Lee, Kayoung Ban, Chanju Park, Dongsub Lee

3rd IML Workshop
16 April 2019

Motivation

Deep Neural Networks are so powerful and being ubiquitous ...

- As a flexible approximator of patterns hidden in complicated large-scale data.

  Universal Approximation Theorem [Cybenko (89'), Hornik (91')]

- It is also great it can really be trained.

- Leading the explosion of data driven models, which can be accurate than human-knowledge based model, in almost every field of industries and sciences.

  [Vision Recognition], [Speech Recognition], [Machine Translation], ...
  [AlphaGo Lee to AlphaZero]
  [Discriminative models in search for the new physics / for jet physics / Generative models toward replacing heavy MC generators, .. in HEP]
  [Discriminative models in Astrophysics]
  [DNNs for learning quantum states, ... in Condensed Matter Physics]
  [Protein folding],
  [Medical prescription],
  .... listing up is so hard.

However, we all know that DNN requires lots of data for here and there, … to learn and build a good model toward the model in ground-truth, without bias / overfitting.

Such a strong data dependency of deep learning, may be a severe defect, if toward an AI architecture especially using the data of physical science which possesses more robust mathematical relations inside, and I believe that good ML models for physical science should be trained valid everywhere without entire data coverage, like as we believe that the physics laws which is discovered and proven to be valid here, is then valid everywhere, for a given energy or multiplicity scale.

→ **'Universality of Scientific ML models'**

One tiny failure ...
(may be BIG for the universality of DL models)

⟶ DNNs cannot even learn simple arithmetic operations in a domain-region free way.

Exp) Identity Op (x ⟶ x)

	tanh	ReLU
Interp. Error	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$
Extrap. Error	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$

Exp) Arithmetic Ops

		tanh	ReLU
Interpolation E	$x_1 + x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$
	$x_1 - x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$
	$x_1 * x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$
	x_1/x_2	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$
Extrapolation E	$x_1 + x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$
	$x_1 - x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$
	$x_1 * x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+6})$
	x_1/x_2	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+7})$

The model's universality cannot be improved by increasing model's capacity /
by training-procedure optimization.
It is a matter of architecture itself, especially on how to encode non-linearities.

Thus, it might be a stupid expectation for a DNN which is born to be a highly non-linear-ed (by activation function) multiply-accumulator, to learn the simple(?) relations, in a less data-independent way.

Our approach (to control/improve the model's universality) is to embed the structure of 'function approximation in power series' inside the DNN, in a trainable way using back-propagation, while borrowing main non-linearities from the power series, rather than from the activations.

We can handle the errors in exterior region by increasing layer's power, and also by disconnecting unnecessary connections by exactly zero-weights quantized.

Actually, power series expansion is a conventional way for many modern computers to approximate the elementary functions – which are the basic building blocks for representing the physics law.

$\longrightarrow$ Why not in DNN for scientific data?

Architecture of
CALU
(**C**omplex-valued Neural **A**rithmetic **L**ogic **U**nit)
with Neural Quantizer

Exp1) learning basic arithmetic operations
(+,-,×,÷)
Exp2) learning polynomial functions

Related Works

1) Neural Arithmetic Logic Unit (NALU)

A. Trask, F. Hill, S. Reed, J. Rae, C. Dyer, P. Blunsom [arXiv:1808.00508]

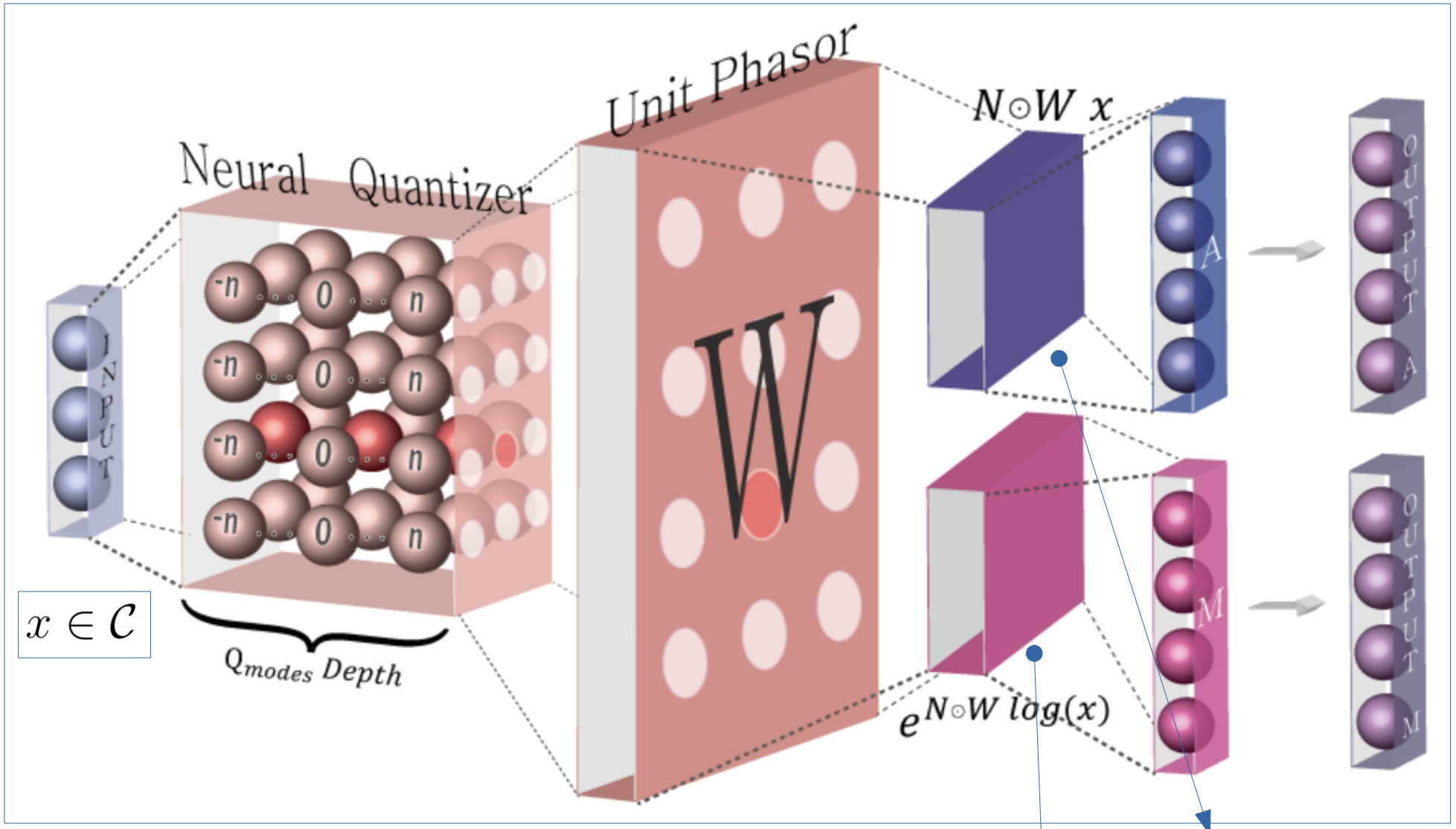
2) Binary/Ternary connected networks

M. Courbariaux, Y. Bengio, Jean-Pierre David [arXiv:1511.00363]
Zhouhan Lin, M. Courbariaux, R. Memisevie, Y. Bengio [arXiv:1510.03009]
C. Zhu, S. Han, H. Mao, W. J. Dally ['Trained Ternary Quantization']
...

3) Complex-valued neural networks

T. Kim, T. Adah ['Approximation by Fully Complex Multilayer Perceptrons']
T. Nitta ['An Extension of the BP algorithm to complex numbers']
C. Trabelsi et. al. ['Deep Complex Networks']
N. Guberman ['On Complex-Valued CNN']
....

One CALU layer in a Complex-Valued Neural Network



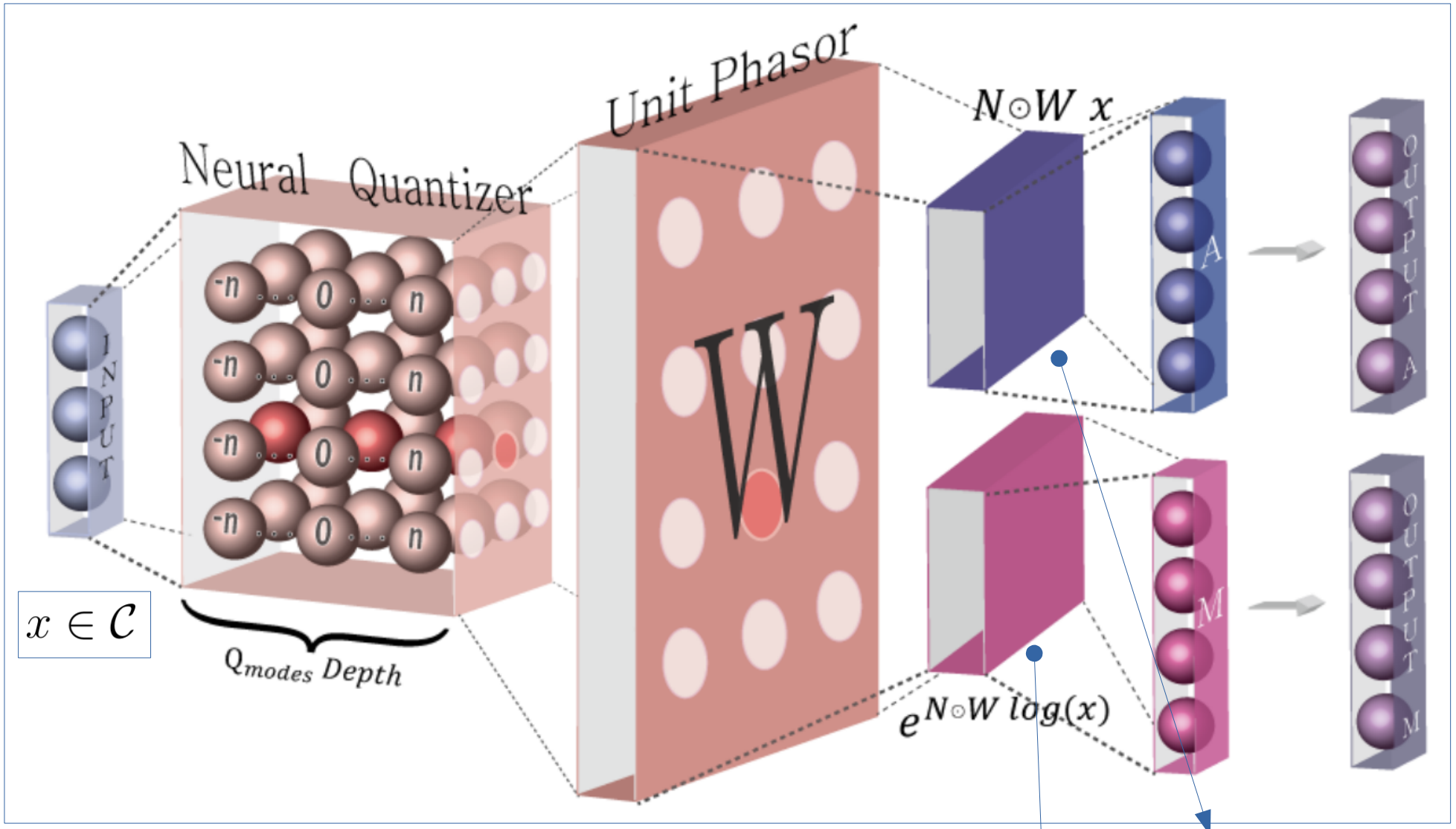
$$N = \Sigma_{n \in Q} \left[n \times \frac{\exp(N_n)}{\Sigma_l \exp(\hat{N}_l)} \right] \qquad W = e^{+i\pi\sigma(\hat{\theta})}$$

ex) $Q = \{-N_{max}, ..0, .., N_{max}\}$

CALUa :
for addition/sub. with (quantized) scaling

CALUm :
for multiplication/div. with (quantized) power op.

One CALU layer in a Complex-Valued Neural Network



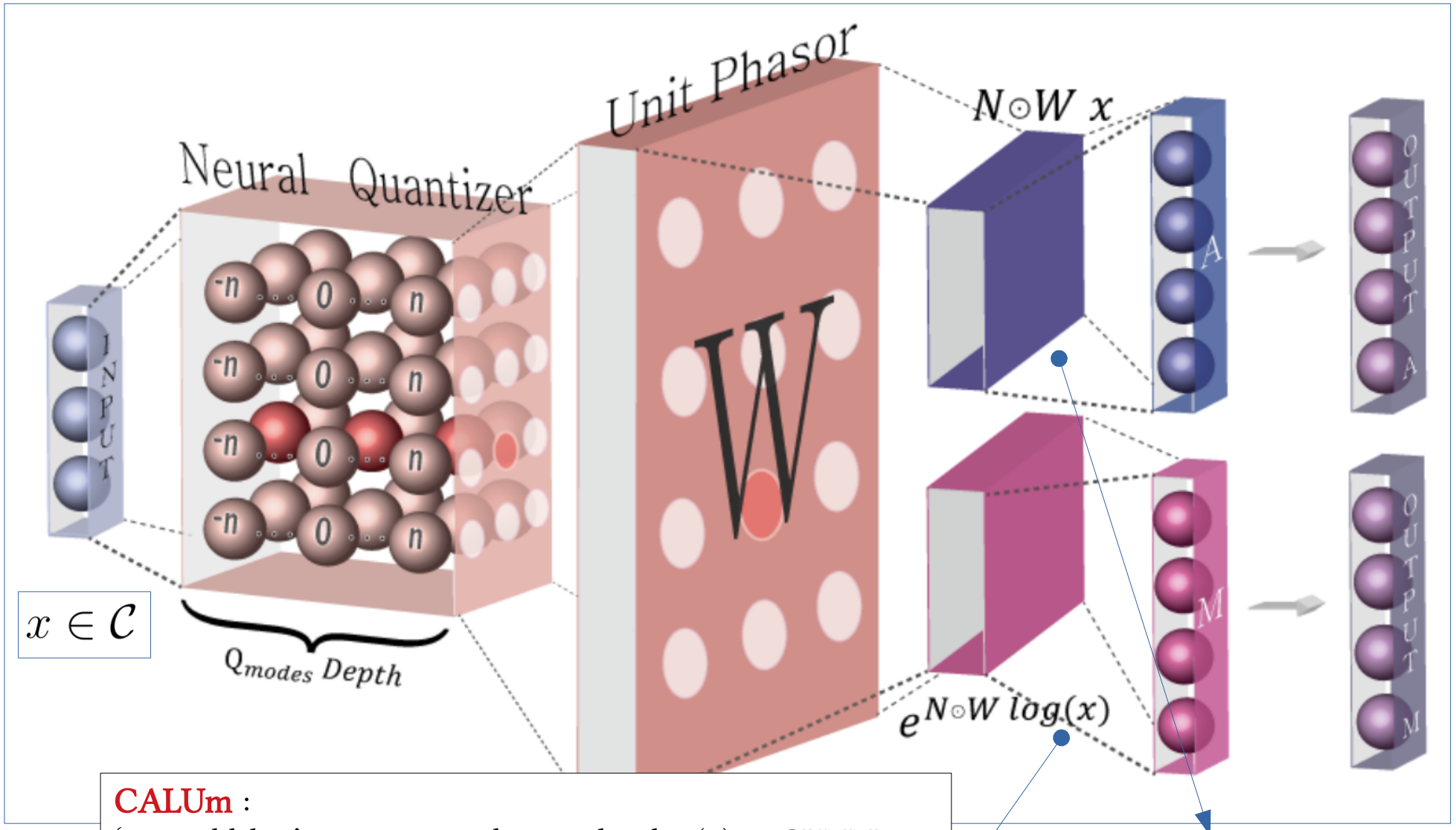
$$N = \Sigma_{n \in Q} \left[n \times \frac{\exp(N_n)}{\Sigma_l \exp(\hat{N}_l)} \right] \qquad W = e^{+i\pi\sigma(\hat{\theta})}$$

ex) $Q = \{-N_{max}, ..0, .., N_{max}\}$

CALUa :
for addition/sub. with (quantized) scaling

CALUm :
for multiplication/div. with (quantized) power op.

One CALU layer in a Complex-Valued Neural Network



CALUm :
'exp-add-log' structure with complex-log(x) in CVNN can enable us to do multiplication op. with the inputs of any phases, also with weights being consistently trainable using back-propagation of CVNN.

CALUa :
for addition/sub. with (quantized) scaling

$N =$

ex) $Q = \{-N_{max}, ..0, .., N_{max}\}$

Neural Quantizer, N

$$N_{kj} = \Sigma_{n \in Q} \left[n \times \frac{\exp(\hat{N}_{kj,n})}{\Sigma_l \exp(\hat{N}_{kj,l})} \right]$$

- Expectation value of picking a quantized value in a set Q, with each element n, weighted by the probability P(n) modeled by softmax function with learnable parameters, N-hats for each n.
- Q is a set of quantized values (in general C), which can encode any values from dynamics.

  ex) Q = {-2,-1,0,1,2} with the unit phasor W = 1 ⟶ R-valued total weights

  ex) Q = {0,1,2,…,N_{max}} with W = exp [i π σ] ⟶ C-valued total weights

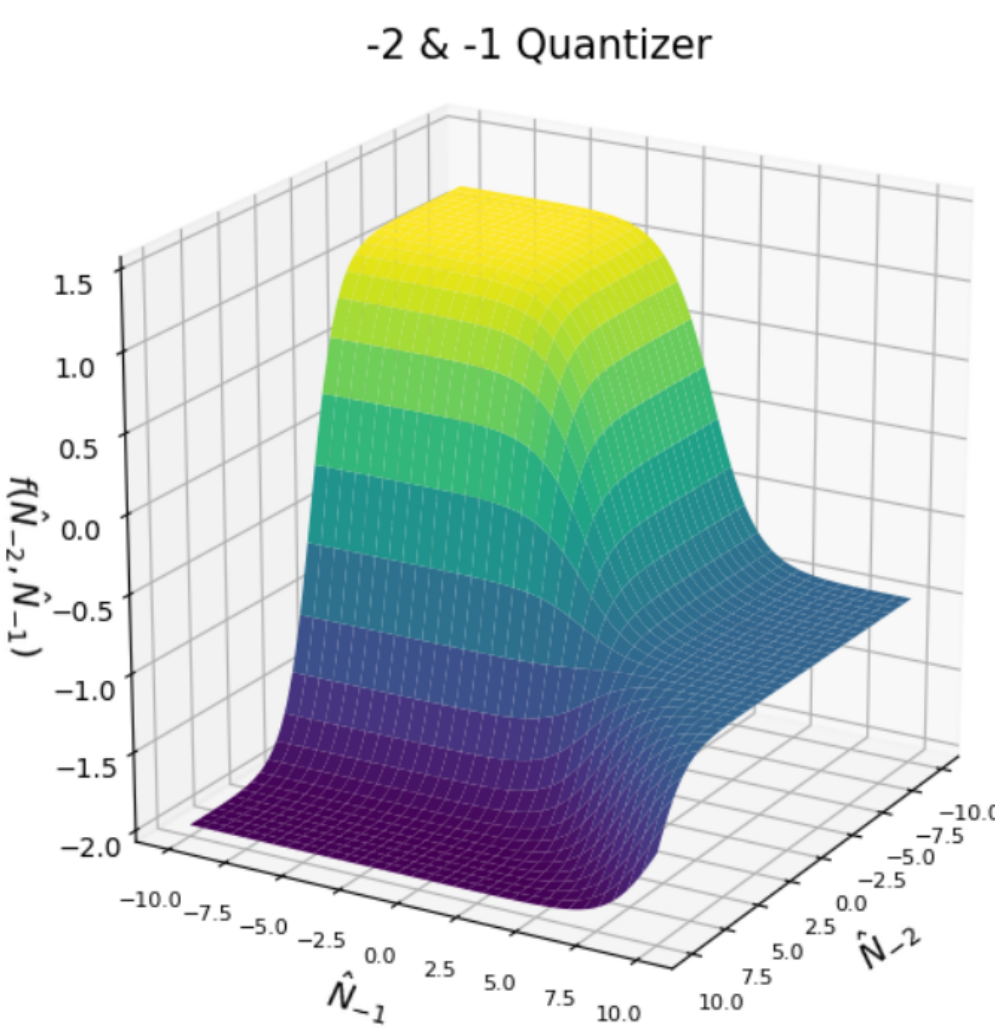
- k, j : node indexes connecting layers
- **N_{kj} is asymptotically stabilized and trained to be a Qi, in back-propagation**

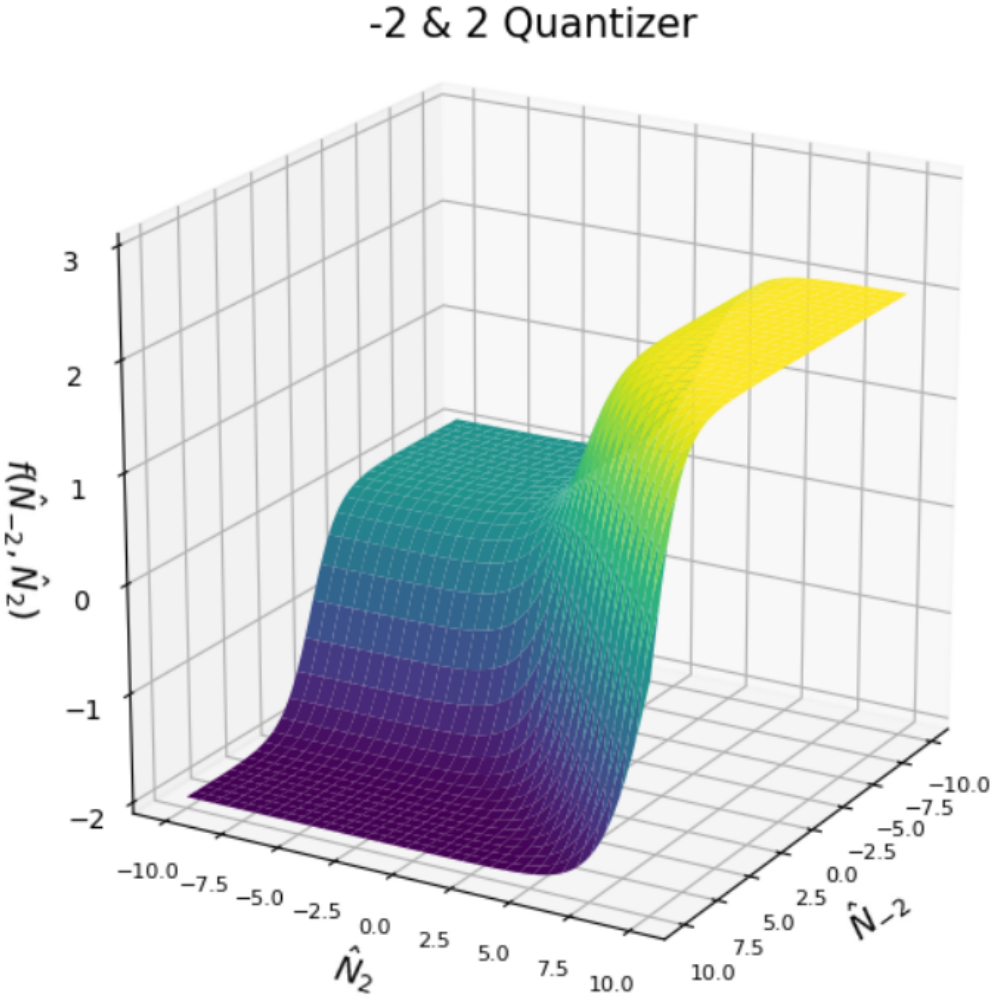
$$\frac{\partial N_{kj}}{\partial \hat{N}_{kj,n}} = -P_{kj,n}(N_{kj} - n), \quad P_{kj,n} = \frac{\exp(\hat{N}_{kj,n})}{\Sigma_l \exp(\hat{N}_{kj,l})}$$

⟶ coupled Boltzmann transport equation.

Behavior of Neural Quantizer, N

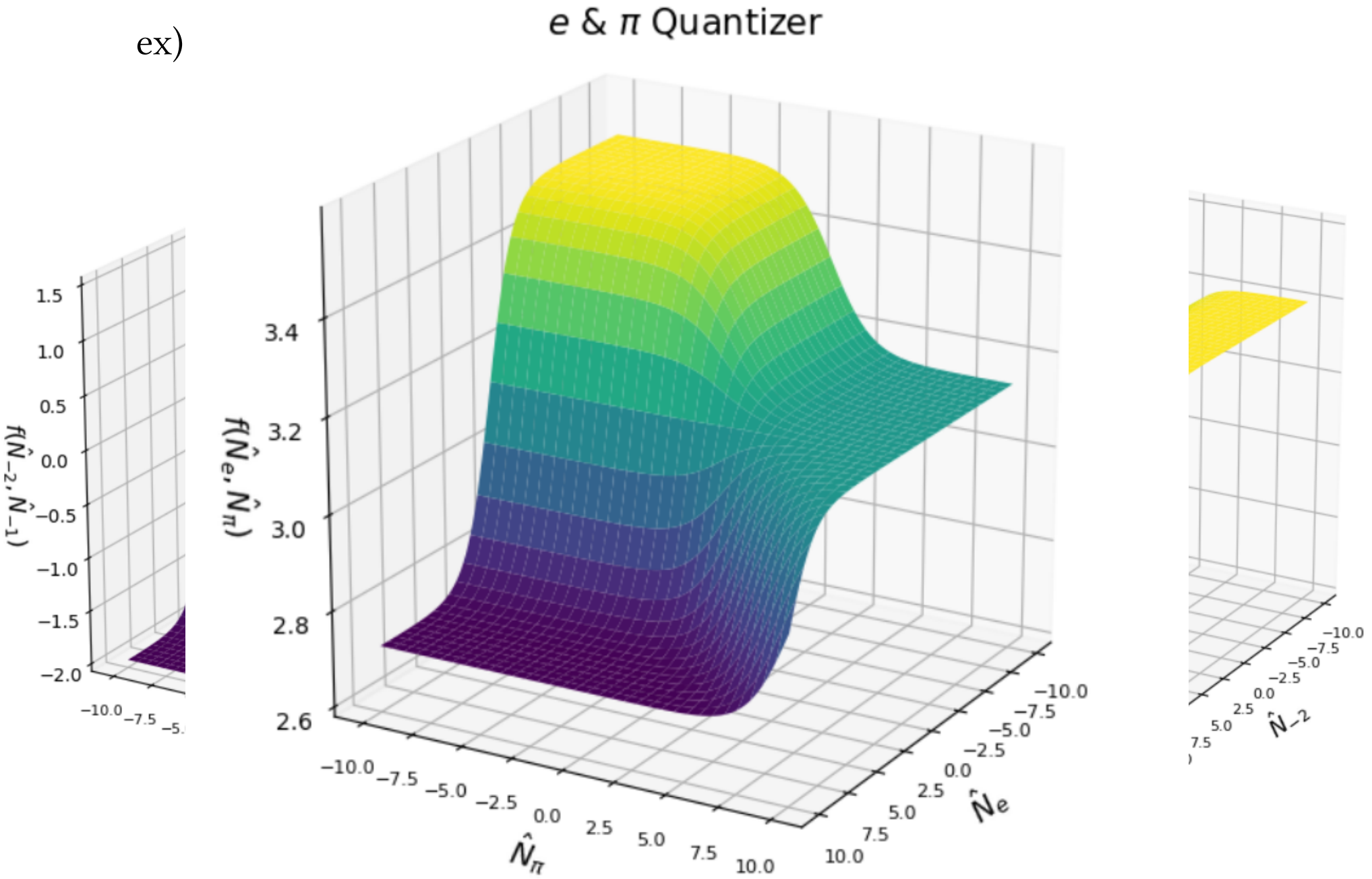
ex) Q = {-2,-1,0,1,2}





Behavior of Neural Quantizer, N

ex)



Unit Phasor

$$W_{kj} = e^{+i\pi\sigma(\hat{\theta}_{kj})}$$

- can be trained into the quantized phases ({0,pi})
- so the overall sign of weights {+1, -1} can be trained in the complex-domain, with learnable parameters, theta-hats
- trainable, if Q consists of all positive numbers, otherwise just set to 1.

all in all …

CALUa

- linear layer with scaling weights (N◦W) quantized in Q for addition and subtraction.

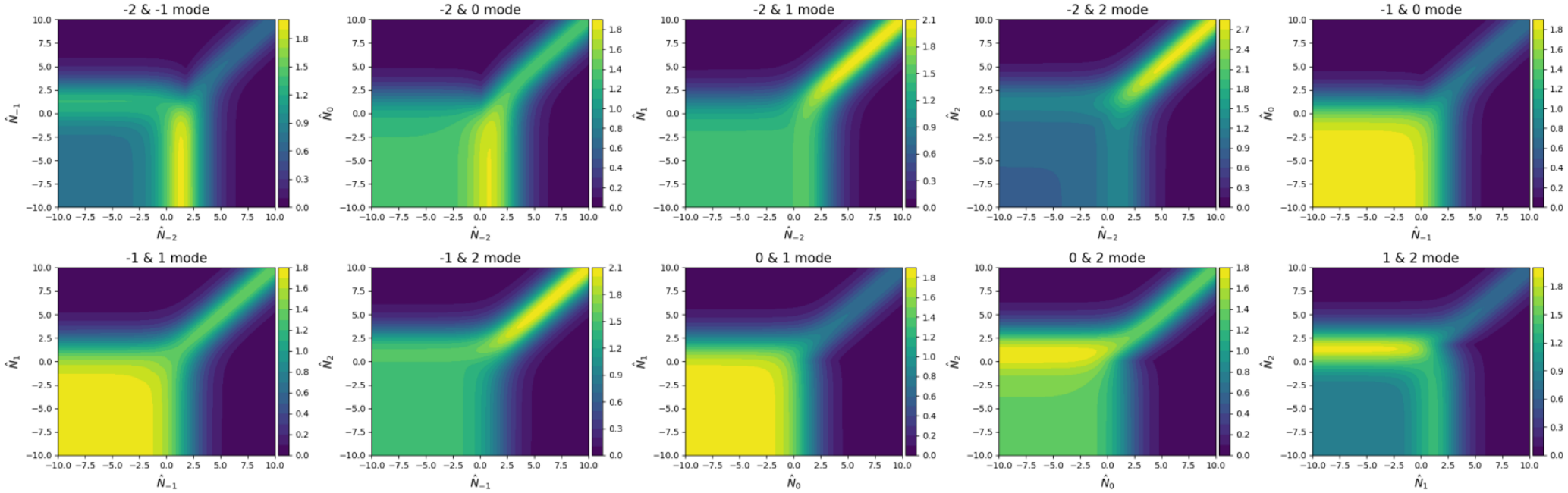
CALUm

- exp-add-log layer with powering weights (N◦W) quantized in Q for mul. and div.
- **The first NN architecture** which can learn the multiplication and division op. of input variables **in general domain, trainable using BP, in precision with universality**.

Regularization Potential

$$E_{kj} = \Sigma_{n \in Q} |N_{kj} - n| \log\left[(1 - P_{kj,n})\right]$$

- for boosting the training of neural quantizer
- $E_{kj} \longrightarrow 0$ for $N_{kj}=n$, also in preference of adjacent modes for jumping into.



Experiments

Experiment 1. Regression to a constant (1)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

NN Model	Layer Structure
tanh	DIM(X)|100:TANH|100:TANH|DIM(Y):IDENTITY
ReLU	DIM(X)|100:RELU|100:RELU|DIM(Y):IDENTITY
NALU	DIM(X)|50:NALU$_m$ |DIM(Y):NALU$_a$
CALU	DIM(X)|50:CALU$_m$ |DIM(Y):CALU$_a$

Experiment 1. Regression to a constant (1)

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

	tanh	ReLU	NALU	CALU
Interp. Error	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$\mathbf{< \mathcal{O}(10^{-25})}$
Extrap. Error	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{-4})$	$\mathbf{< \mathcal{O}(10^{-21})}$

Experiment 2. Regression as identity

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix}$$

NN Model	Layer Structure
tanh	DIM(X)|100:TANH|100:TANH|DIM(Y):IDENTITY
ReLU	DIM(X)|100:RELU|100:RELU|DIM(Y):IDENTITY
NALU	DIM(X)|50:NALU$_m$ |DIM(Y):NALU$_a$
CALU	DIM(X)|50:CALU$_m$ |DIM(Y):CALU$_a$

Experiment 2. Regression as identity

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_8 \end{bmatrix}$$

	tanh	ReLU	NALU	CALU
Interp. Error	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$\mathbf{< \mathcal{O}(10^{-25})}$
Extrap. Error	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{-4})$	$\mathbf{< \mathcal{O}(10^{-21})}$

Experiment 1. Elementary Arithmetic Operations

$$\text{Addition} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 + x_2 \end{bmatrix}$$

$$\text{Subtraction} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 - x_2 \end{bmatrix}$$

$$\text{Multiplication} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 * x2 \end{bmatrix}$$

$$\text{Division} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} x_1 / x2 \end{bmatrix}$$

Experiment 1. Elementary Arithmetic Operations

NN Model	Layer Structure
tanh	DIM(X)|100:TANH|100:TANH|DIM(Y):IDENTITY
ReLU	DIM(X)|100:RELU|100:RELU|DIM(Y):IDENTITY
NALU	DIM(X)|5:NALU|DIM(Y):NALU
CALU	DIM(X)|5:CALU|DIM(Y):CALU

		tanh	ReLU	NALU	CALU
Interpolation E	$x_1 + x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \boldsymbol{\mathcal{O}(10^{-20})}$
	$x_1 - x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \boldsymbol{\mathcal{O}(10^{-20})}$
	$x_1 * x_2$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-2})$	$< \boldsymbol{\mathcal{O}(10^{-16})}$
	x_1/x_2	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{+1})$	$< \boldsymbol{\mathcal{O}(10^{-16})}$
Extrapolation E	$x_1 + x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{-2})$	$< \boldsymbol{\mathcal{O}(10^{-17})}$
	$x_1 - x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{-2})$	$< \boldsymbol{\mathcal{O}(10^{-17})}$
	$x_1 * x_2$	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+6})$	$< \mathcal{O}(10^{+6})$	$< \boldsymbol{\mathcal{O}(10^{-14})}$
	x_1/x_2	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+7})$	$< \mathcal{O}(10^{+3})$	$< \boldsymbol{\mathcal{O}(10^{-15})}$

Experiment 2. Multinomial Expansion

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} 1 \\ (x_1 + x_2 + x_3) \\ (x_1 + x_2 + x_3)^2 \\ (x_1 + x_2 + x_3)^3 \end{bmatrix}$$

NN Model	Layer Structure
tanh	DIM(X)|100:TANH|100:TANH|DIM(Y):IDENTITY
ReLU	DIM(X)|100:RELU|100:RELU|DIM(Y):IDENTITY
NALU	DIM(X)|100:NALU$_m$ |DIM(Y):NALU$_a$
CALU	DIM(X)|100:CALU$_m$ |DIM(Y):CALU$_a$

Experiment 2. Multinomial Expansion

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} 1 \\ (x_1 + x_2 + x_3) \\ (x_1 + x_2 + x_3)^2 \\ (x_1 + x_2 + x_3)^3 \end{bmatrix}$$

	tanh	ReLU	NALU	CALU
Interp. Error	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-2})$	$\mathbf{< \mathcal{O}(10^{-25})}$
Extrap. Error	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+6})$	$< \mathcal{O}(10^{+6})$	$\mathbf{< \mathcal{O}(10^{-20})}$

Experiment 3. Learning 3 Lorentz scalar in 2 four-vectors

$$\begin{bmatrix} p_0 \\ \vdots \\ p_3 \\ q_0 \\ \vdots \\ q_3 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} p \cdot p \\ p \cdot q \\ q \cdot q \end{bmatrix}$$

NN Model	Layer Structure
tanh	DIM(X)|100:TANH|100:TANH|DIM(Y):IDENTITY
ReLU	DIM(X)|100:RELU|100:RELU|DIM(Y):IDENTITY
NALU	DIM(X)|50:NALU$_m$ |DIM(Y):NALU$_a$
CALU	DIM(X)|50:CALU$_m$ |DIM(Y):CALU$_a$

Experiment 3. Learning 3 Lorentz scalar in 2 four-vectors

$$\begin{bmatrix} p_0 \\ \vdots \\ p_3 \\ q_0 \\ \vdots \\ q_3 \end{bmatrix} \rightarrow \textbf{NN model} \rightarrow \begin{bmatrix} p \cdot p \\ p \cdot q \\ q \cdot q \end{bmatrix}$$

	tanh	ReLU	NALU	CALU
Interp. Error	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-6})$	$< \mathcal{O}(10^{-2})$	$\mathbf{< \mathcal{O}(10^{-20})}$
Extrap. Error	$< \mathcal{O}(10^{+3})$	$< \mathcal{O}(10^{+6})$	$< \mathcal{O}(10^{+6})$	$\mathbf{< \mathcal{O}(10^{-16})}$

Conclusion

- We designed a new NN architecture – **CALU**

  as a basic trainable NN block, embedding the non-linearities from power series expansion.

- **CALUm** is the first NN architecture, which can learn the multiplication and division op. of input variables **in general domain, trainable using BP, in precision with universality.**

- **Neural Quantizer** is a new way for training weights to a set of quantized levels pre-assigned freely.

- By the combination of CALUa & CALUm, a broad range of elementary functions are trainable in a more domain-region universal way.

- We, as natural scientists (elementary particle physics, …) it would be interesting to make a '**ElementaryNet**' (a.k.a '**ImageNet**') for building new DNN architectures for learning various elementary functions itself, **trainable to have the scientific ML's universality**.