Model-independent analysis of hadron-hadron scattering: a deep learning approach

Denny Lane B. Sombillo

Research Center for Nuclear Physics (RCNP), Osaka University
National Institute of Physics, University of the Philippines Diliman

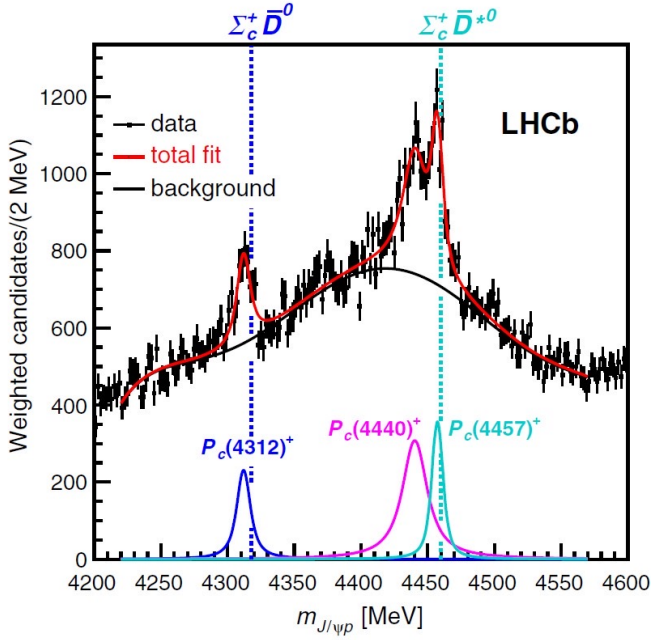
In collaboration with:
Yoichi Ikeda (Kyushu University)
Toru Sato (RCNP, Osaka University)
Atsushi Hosaka (RCNP, Osaka University and ASRC, JAEA)

arXiv:2105.04898
arXiv:2104.14182

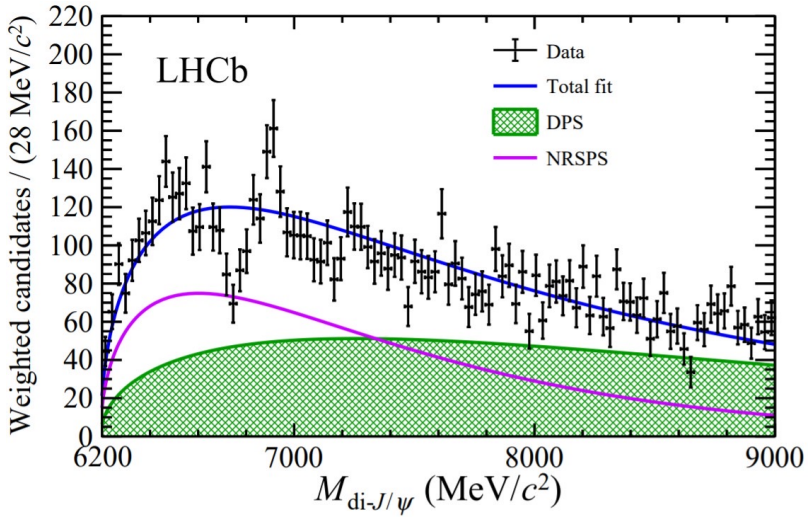
Outline

- Motivation and Overview
- Deep learning approach
  - Preparation: Generation of training dataset
  - Model development: Optimization of deep neural network (DNN) model
  - Inference stage: Application to the experimental data
- Summary and Outlook

Motivation and Overview



10.1103/PhysRevLett.122.222001;
arXiv:1904.03947



10.1016/j.scib.2020.08.032;
arXiv:2006.16957

Explanation of the observed near-threshold/ threshold structures

- Threshold cusp
- Molecular state
- Virtual state
- Compact state

How to tell if a near-threshold structure is caused by a physical state?

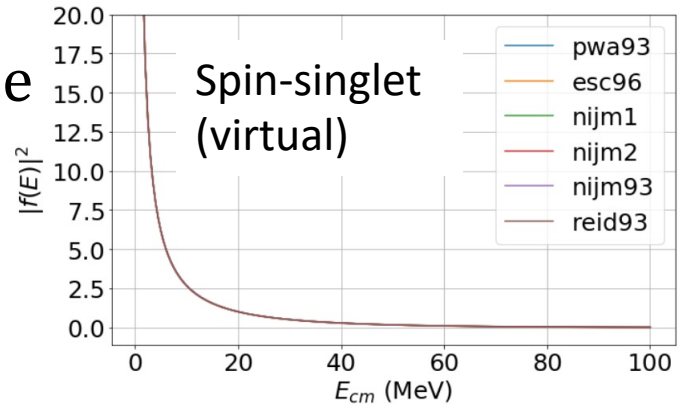
- Phrase the question as a classification problem.
- **Deep learning approach** excels in solving a classification problem.

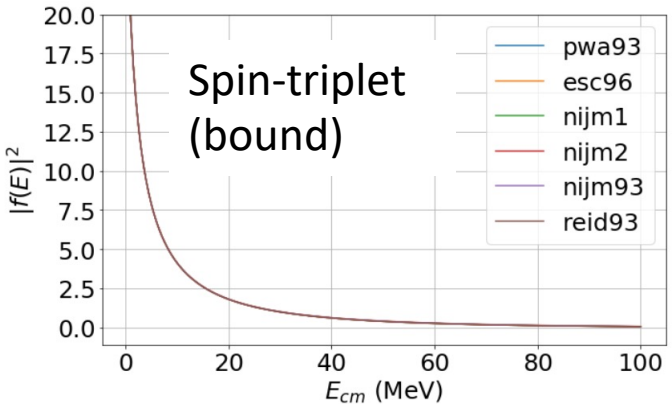
Deep learning approach: Use of **data** (simulated or real) to **improve the performance** of a **model** in accomplishing a specific **task**.

Motivation and Overview

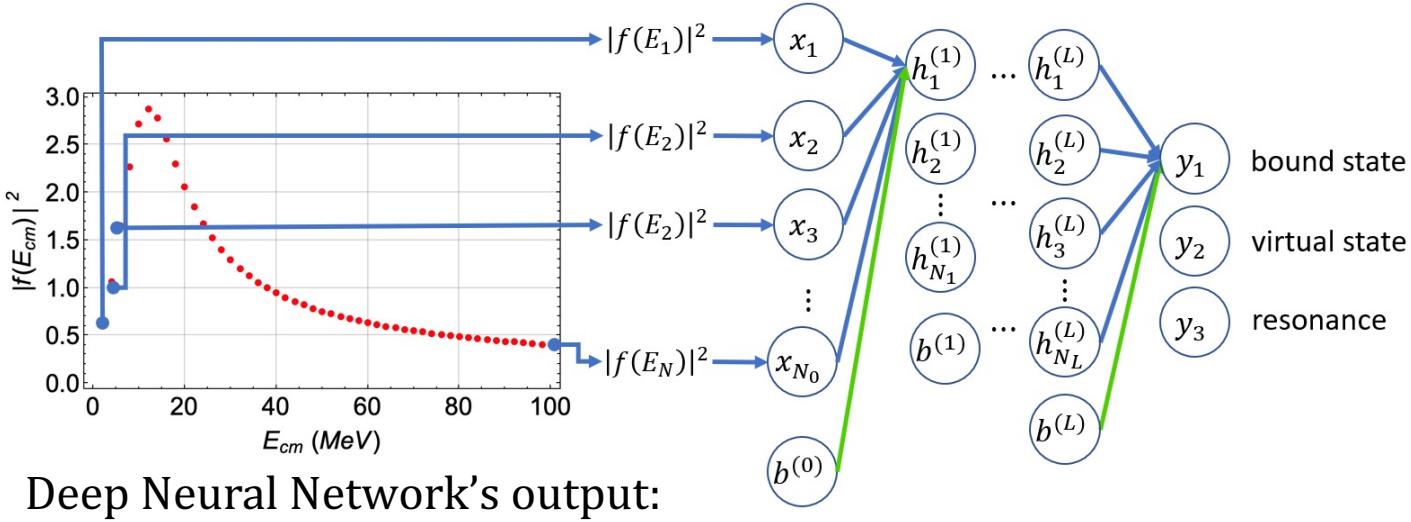
DNN to distinguish bound-virtual-resonance in single-channel cross-section

DLB Sombillo, Y Ikeda, T Sato, A Hosaka
10.1103/PhysRevD.102.016024; arXiv:2003.10770
https://github.com/sombillo/DNN-for-bound-virtual-classification





http://nn-online.org/NN/?page=nnphs2



Deep Neural Network's output:

	PWA93	ECS96	NijmI	NijmII	Nijm93	Reid93
$^1S_0$	virtual	virtual	virtual	virtual	virtual	virtual
$^3S_1$	bound	bound	bound	bound	bound	bound

Proof of principle

- Generic properties of S-matrix is sufficient to generate the training dataset.
- Threshold enhancements can be studied using data science approach.

Motivation and Overview

Extension to coupled-channel problem

Complete classification requires a model.

A. M. Badalyan et.al., Phys. Rep. 1982

Information that can be obtained without using a model:

Pole configuration: number of nearby poles in each Riemann sheet

pole-counting method

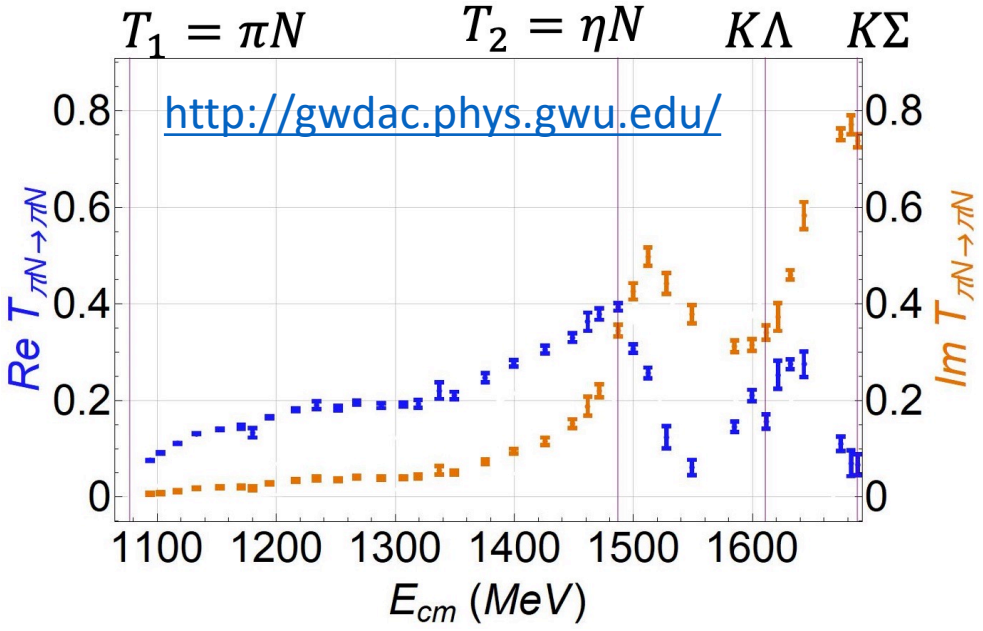
D. Morgan, Nucl. Phys. A, 543,4,1992

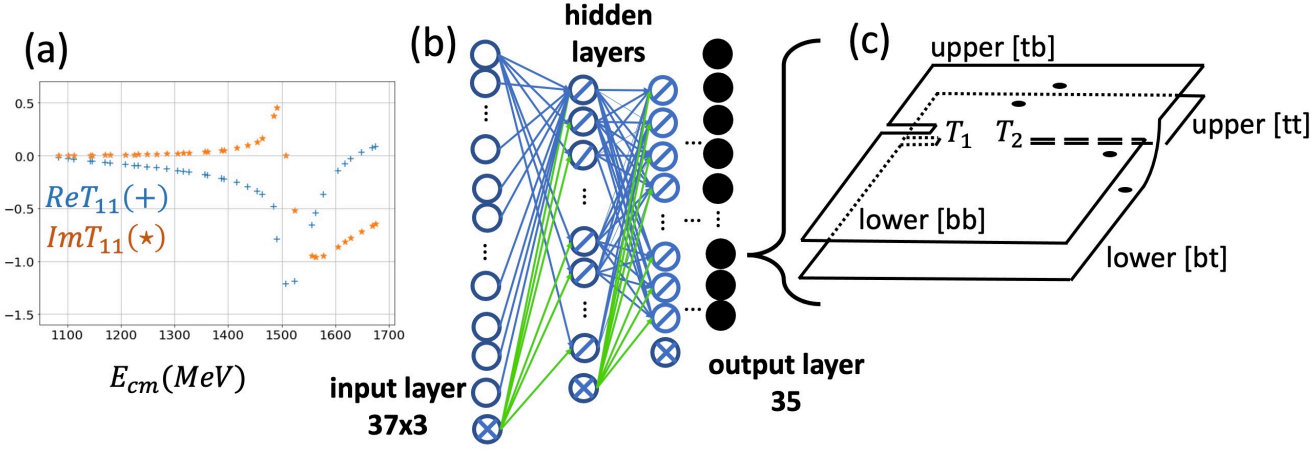
two-pole structure of $\Lambda(1405)$

PhysRevC.68.018201; arXiv:0212026

Pole configuration can be used to construct an appropriate parametrization.

Motivation and Overview





Model-independent analysis:

- No assumed functional form.
- No assumptions on poles
  - pole-shadow pair
  - Trajectory

Realizable in deep learning approach:

- DNN inference using raw data
- Poles are independent

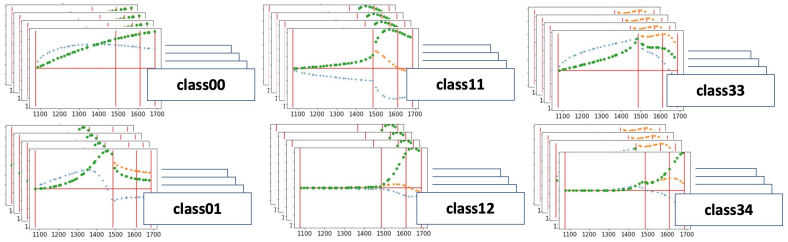
The error bars can be handled systematically during the the DNN inference stage.

Motivation and Overview

Proposed model-independent deep learning analysis:

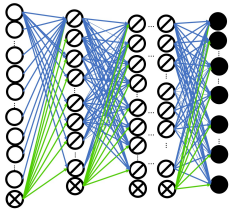
Generating the training dataset

- Use general properties of S-matrix



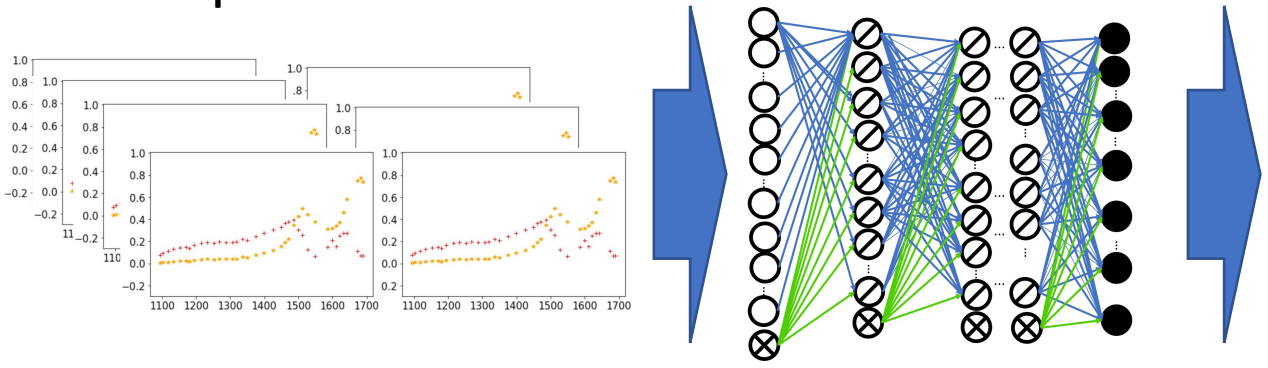
Optimization of deep neural network (DNN) model

- Include energy uncertainty



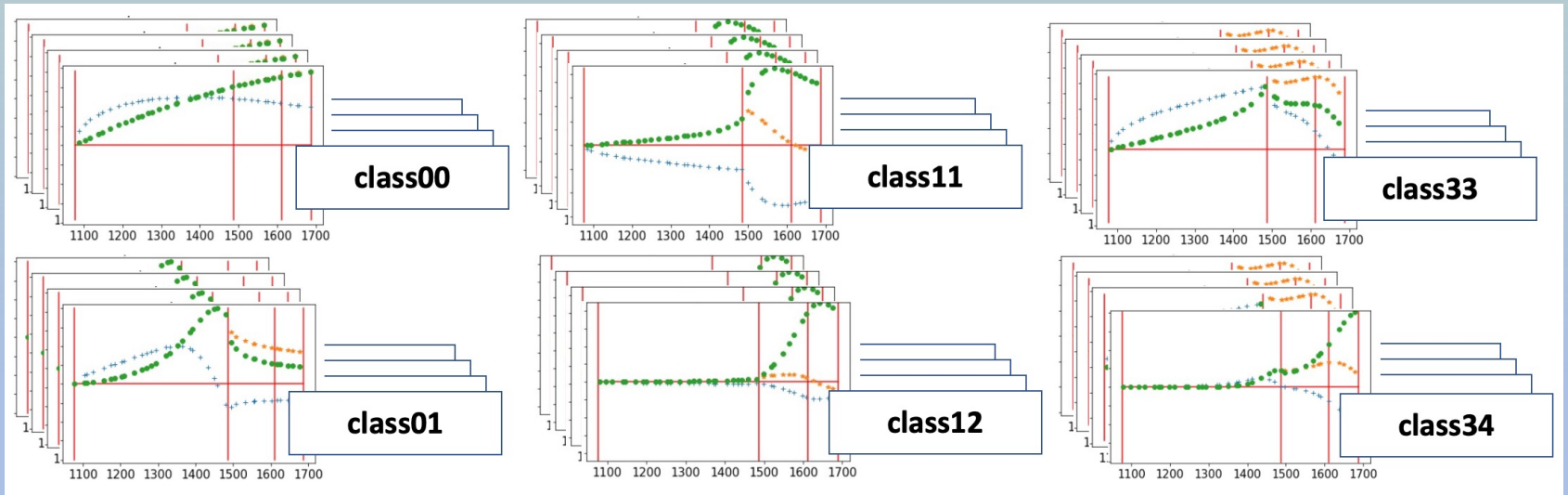
Apply trained-DNN on experimental data

- Use error bars in the data to generate inference amplitudes



Count the number of outcomes

Generation of training dataset

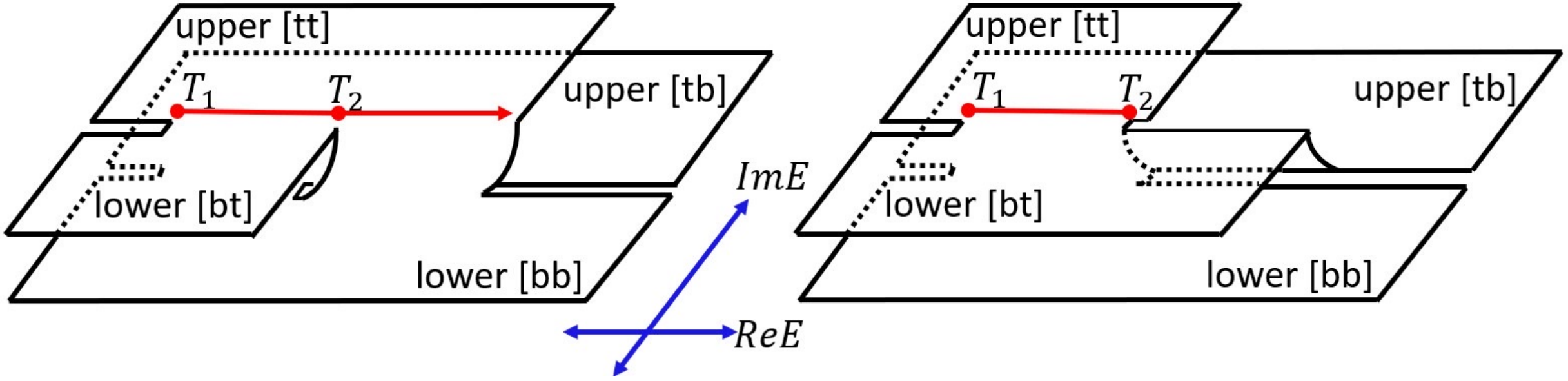


Generation of training dataset

Relevant Riemann sheets in a coupled two-channel scattering

$$E = \frac{p_1^2}{2\mu_1} + T_1 = \frac{p_2^2}{2\mu_2} + T_2$$

Riemann sheet	Notation used	$\mathrm{Im}\ p_1$	$\mathrm{Im}\ p_2$
Physical (1st) sheet	**[tt]**	$+$	$+$
2nd Riemann sheet	**[bt]**	$-$	$+$
3rd Riemann sheet	**[bb]**	$-$	$-$
4th Riemann sheet	**[tb]**	$+$	$-$



Generation of training dataset

General form of S-matrix:

- Hermitian below the lowest threshold
- Unitarity
- Analyticity

https://doi.org/10.1063/1.1703698
https://doi.org/10.1098/rspa.1960.0096

$$S_{11}(p_1,p_2)=\prod_m \frac{D_m(-p_1,p_2)}{D_m(p_1,p_2)}; \qquad S_{11}=1+2iT_{11}$$

$$S_{22}(p_1,p_2)=\prod_m \frac{D_m(p_1,-p_2)}{D_m(p_1,p_2)} \qquad S_{22}S_{11}-S_{12}^2=\prod_m \frac{D_m(-p_1,-p_2)}{D_m(p_1,p_2)}$$

The available experimental data will determine the relevant S-matrix element.

Ensure that one $D_m(p_1,p_2)$ will produce one pole (conjugate pair).

Generation of training dataset

How to control the pole configuration?

- Assign the pole position: $E_{\text{pole}}^{(m)} = E_R^{(m)} \pm iE_I^{(m)}$
- Control the Riemann sheet:

$$D_m(p_1, p_2) = \left[\left(p_1 - i\beta_1^{(m)}\right)^2 - \alpha_1^{(m)2}\right] + \lambda^{(m)}\left[\left(p_2 - i\beta_2^{(m)}\right)^2 - \alpha_2^{(m)2}\right] = 0$$

Absolute values of $\alpha_1^{(m)}, \alpha_2^{(m)}, \beta_1^{(m)}, \beta_2^{(m)}$ are determined by $E_{\text{pole}}^{(m)} = \frac{p_i^2}{2\mu_i} + T_i$

Hermiticity is automatically satisfied.

Analyticity – do not choose $\beta_1^{(m)}, \beta_2^{(m)} > 0$ simultaneously

- Use $\lambda^{(m)}$ to ensure that only one pole per $D_m(p_1, p_2)$

Generation of training dataset

$$D_m(p_1, p_2) = \left[\left(p_1 - i\beta_1^{(m)}\right)^2 - \alpha_1^{(m)2}\right] + \lambda^{(m)}\left[\left(p_2 - i\beta_2^{(m)}\right)^2 - \alpha_2^{(m)2}\right] = 0$$

$$\frac{p_1^2}{2\mu_1} + T_1 = \frac{p_2^2}{2\mu_2} + T_2$$

Quartic equation in p_1 (or in p_2)

- First 2 solutions: $E_{\text{pole}}^{(m)}$ and its conjugate partner.
- Other 2 solutions: $E_{\text{shadow}}^{(m)}$ and its conjugate partner.
  - Might mess up with causality
  - Dependent with $E_{\text{pole}}^{(m)}$
  - $\lambda^{(m)}$ can be set to push the shadow below T_1

What if there is an actual shadow in the amplitude?

Can an independent pole mimic the effect of shadow?

Generation of training dataset

Effects of shadow pole (2-channel Breit-Wigner)

$$T_{11}(p_1, p_2) = \frac{\gamma_1}{E - E_{BW} + i\gamma_1 p_1 + i\gamma_2 p_2}$$

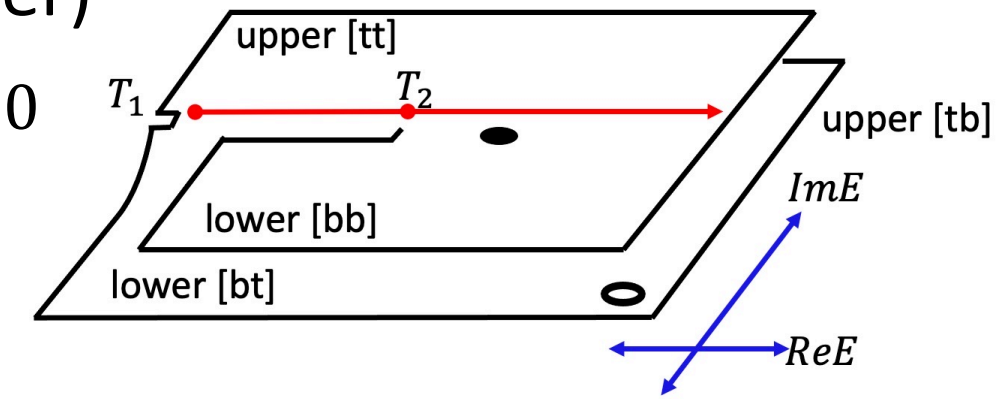
$\gamma_2 \to 0$

$$T_{11}(p_1, p_2) = \frac{\gamma_1}{E - E_{BW} + i\gamma_1 p_1}$$

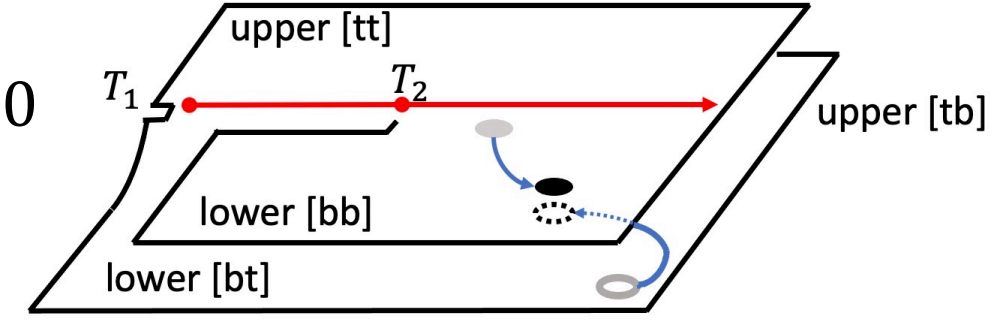
No more cusp at $E = T_2$.

No longer an explicit function of $\sqrt{2\mu_2(E - T_2)}$

$\gamma_2 \neq 0$



$\gamma_2 \to 0$

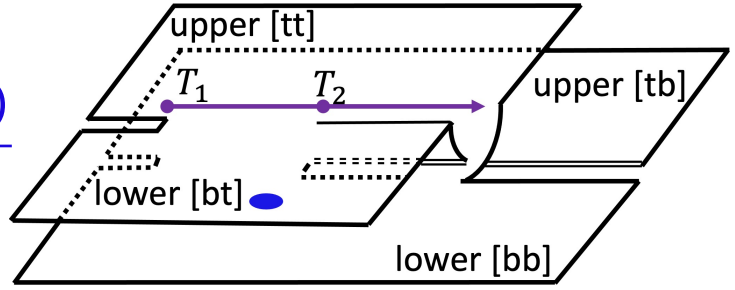


A shadow placed at the same position as the main pole will uncouple one of the channel.

Generation of training dataset

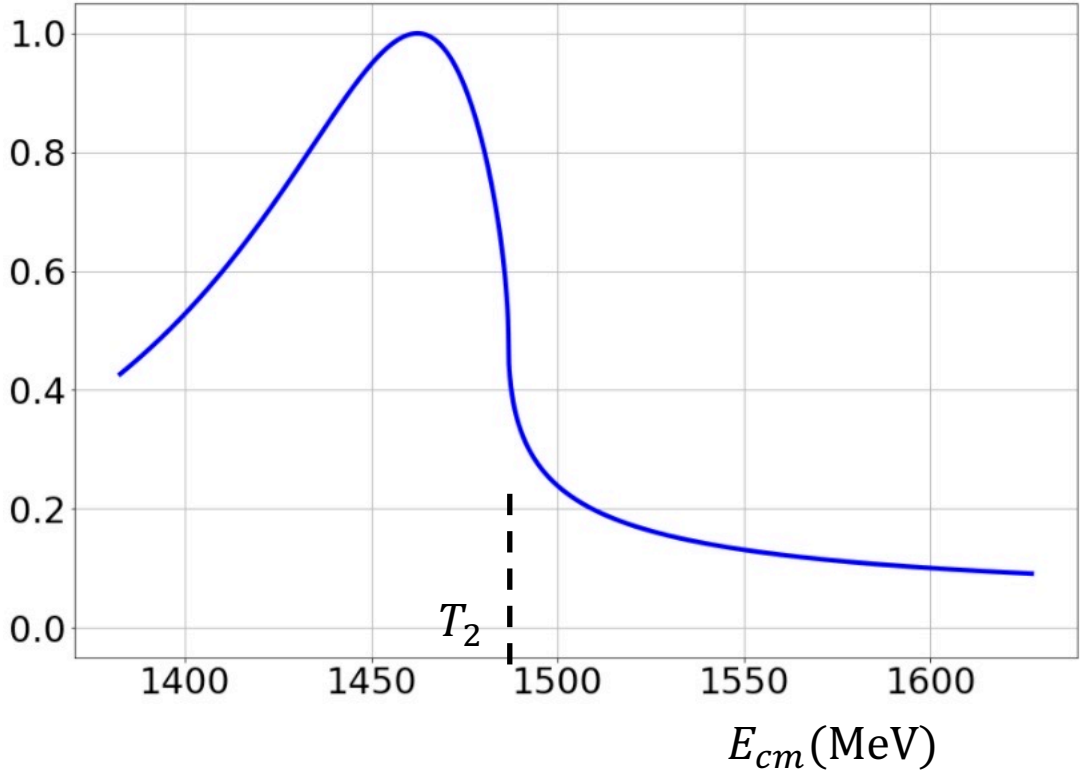
An independent pole can mimic the effect of shadow pole.

$$S_{11}(p_1,p_2) = \frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}$$



$D_{1st}(p_1,p_2) = 0$ $\quad \mathrm{Re}E_{pole} = T_2$

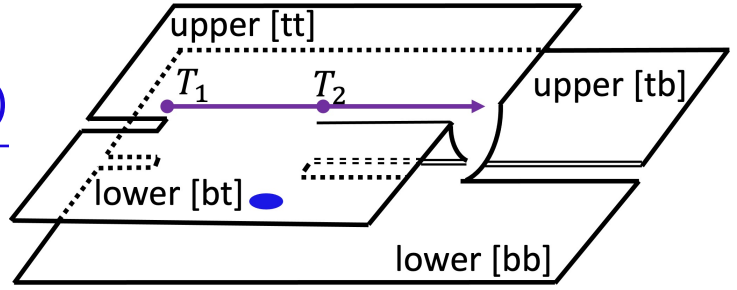
$E_{shadow1} = -16{,}000_{[tt]}$
$E_{shadow2} = -5000_{[tb]}$



Generation of training dataset

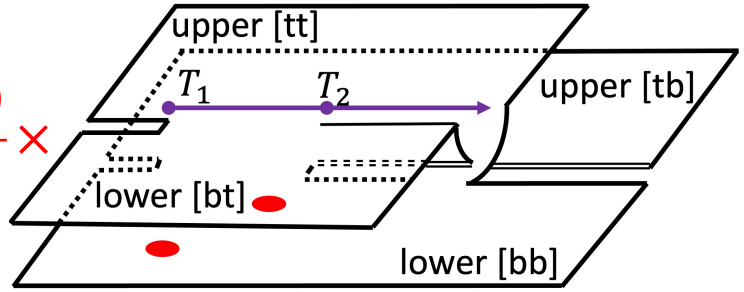
An independent pole can mimic the effect of shadow pole.

$$S_{11}(p_1,p_2)=\frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}$$

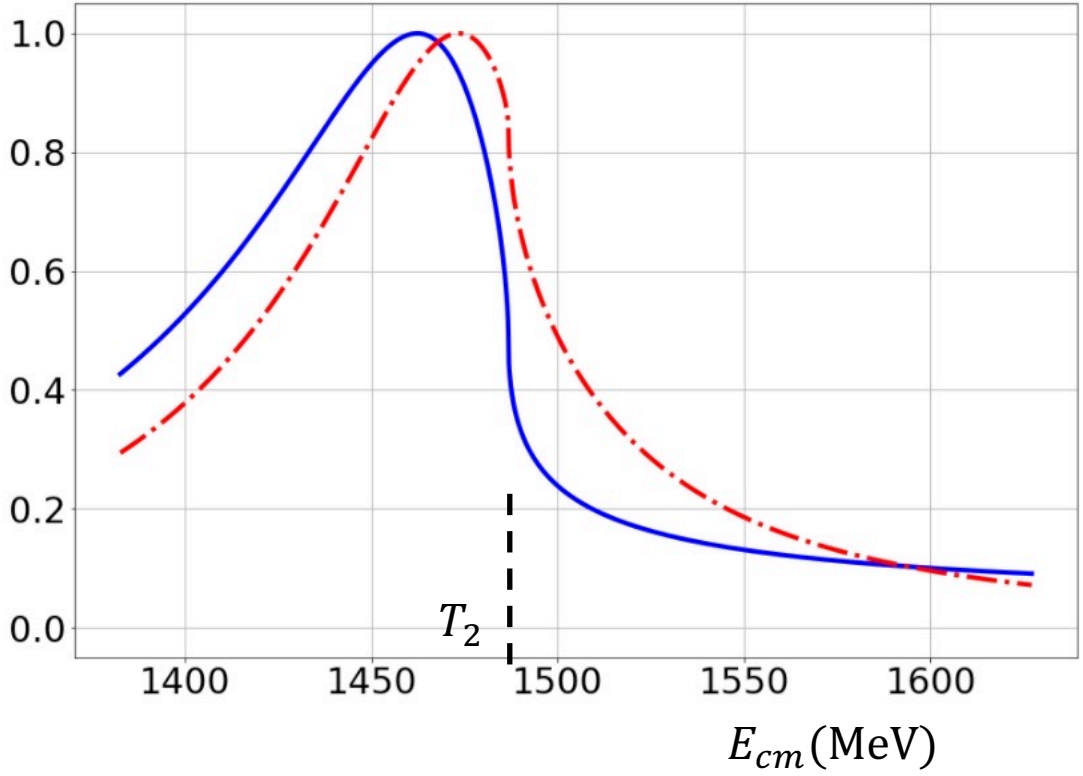


$$|T_{11}|^2$$

$$S_{11}(p_1,p_2)=\frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}\times\frac{D_{2nd}(-p_1,p_2)}{D_{2nd}(p_1,p_2)}$$



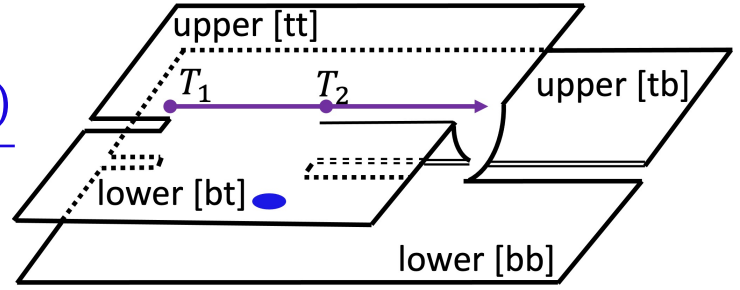
$$D_{2nd}(p_1,p_2)=0 \quad \mathrm{Re}E_{pole}<T_2$$



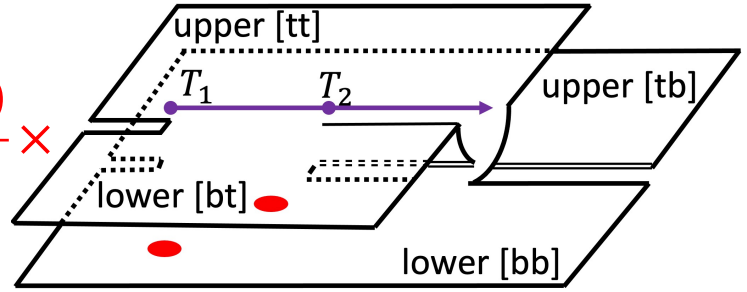
Generation of training dataset

An independent pole can mimic the effect of shadow pole.

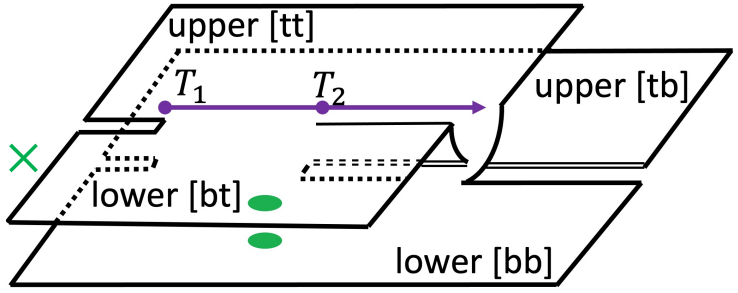
$$S_{11}(p_1,p_2) = \frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}$$



$$S_{11}(p_1,p_2) = \frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)} \times \frac{D_{2nd}(-p_1,p_2)}{D_{2nd}(p_1,p_2)}$$

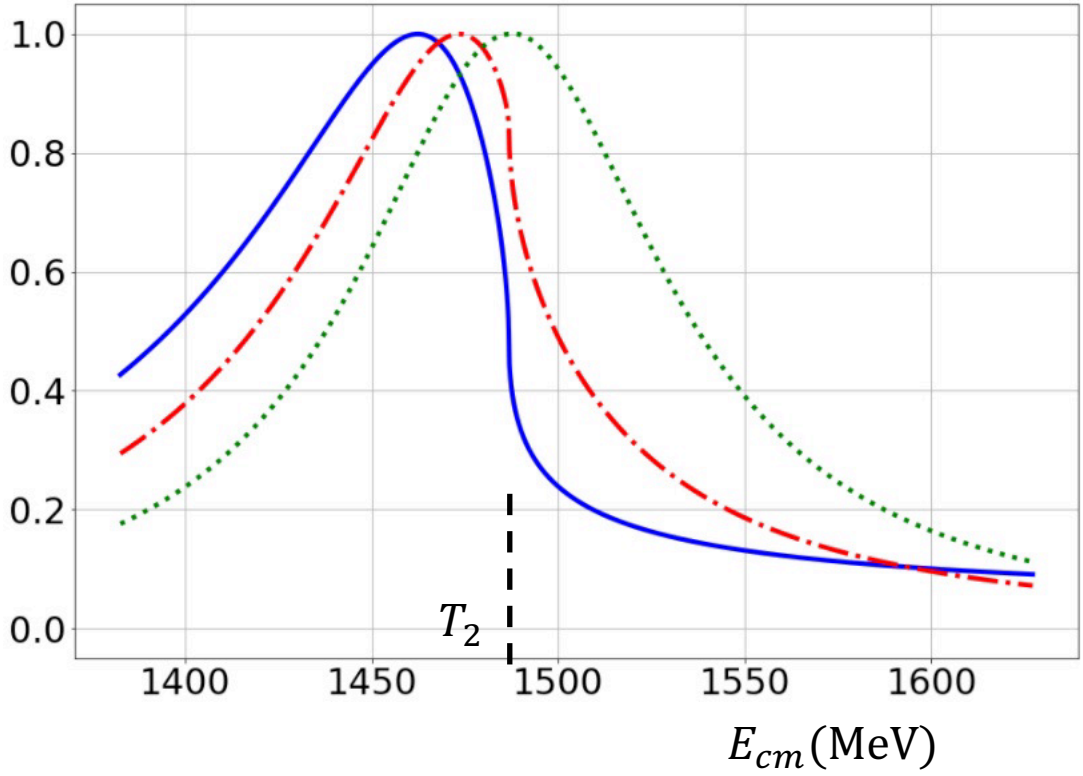


$$S_{11}(p_1,p_2) = \frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)} \times \frac{\overline{D}_{2nd}(-p_1,p_2)}{\overline{D}_{2nd}(p_1,p_2)}$$



$\overline{D}_{2nd}(p_1,p_2) = 0 \quad \mathrm{Re}E_{pole} = T_2$

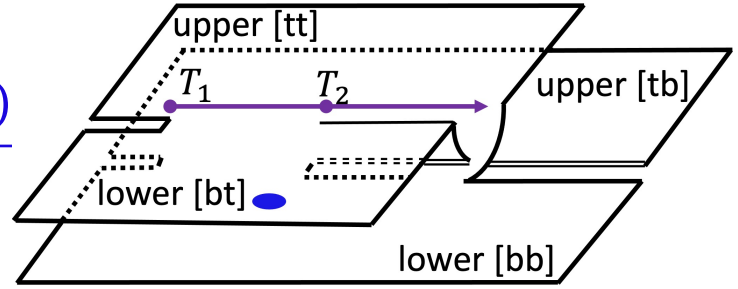
$|T_{11}|^2$



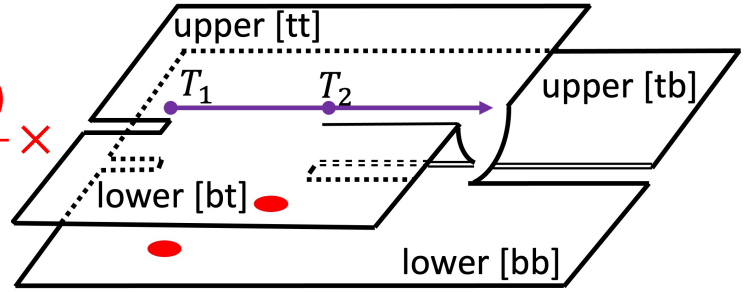
Generation of training dataset

An independent pole can mimic the effect of shadow pole.

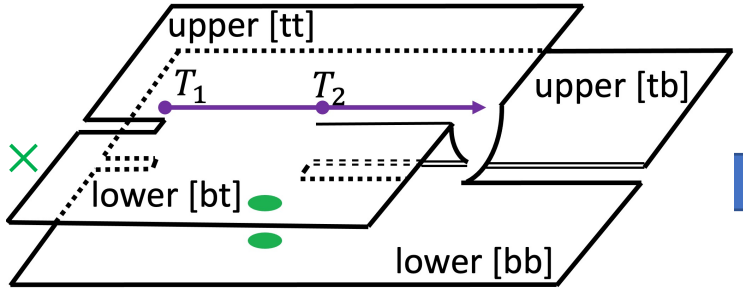
$$S_{11}(p_1,p_2)=\frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}$$



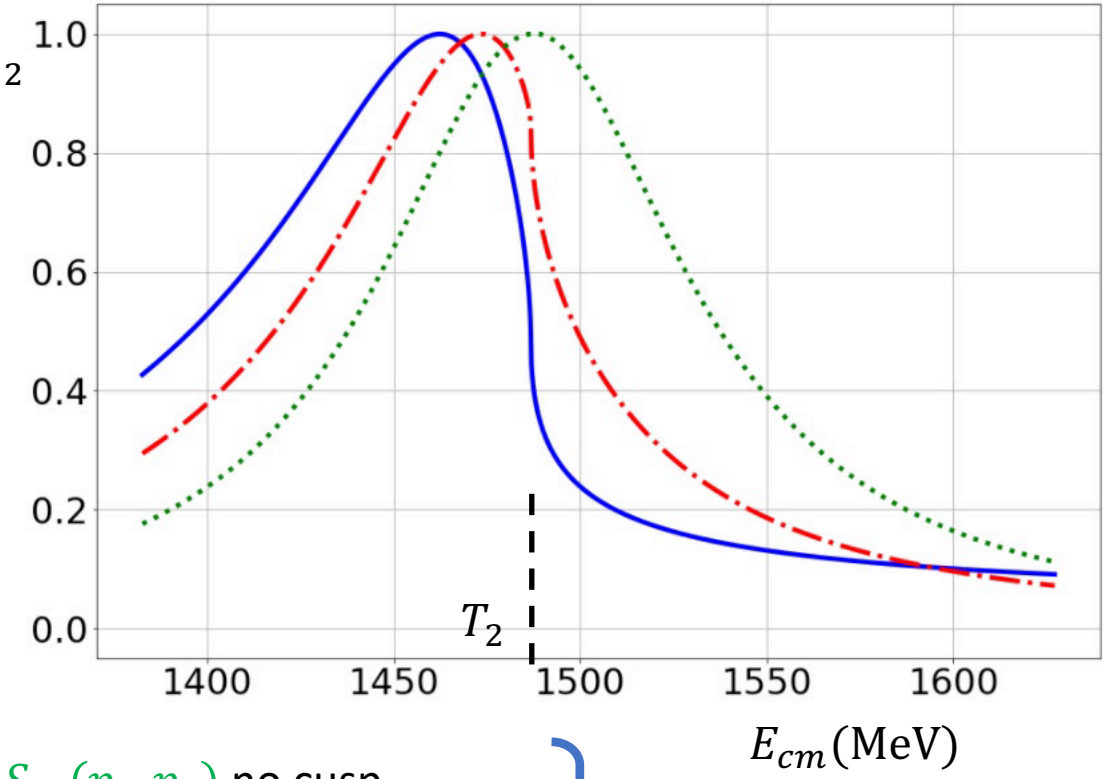
$$S_{11}(p_1,p_2)=\frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}\times\frac{D_{2nd}(-p_1,p_2)}{D_{2nd}(p_1,p_2)}$$



$$S_{11}(p_1,p_2)=\frac{D_{1st}(-p_1,p_2)}{D_{1st}(p_1,p_2)}\times\frac{\overline{D}_{2nd}(-p_1,p_2)}{\overline{D}_{2nd}(p_1,p_2)}$$



$|T_{11}|^2$



$S_{11}(p_1,p_2)$ no cusp

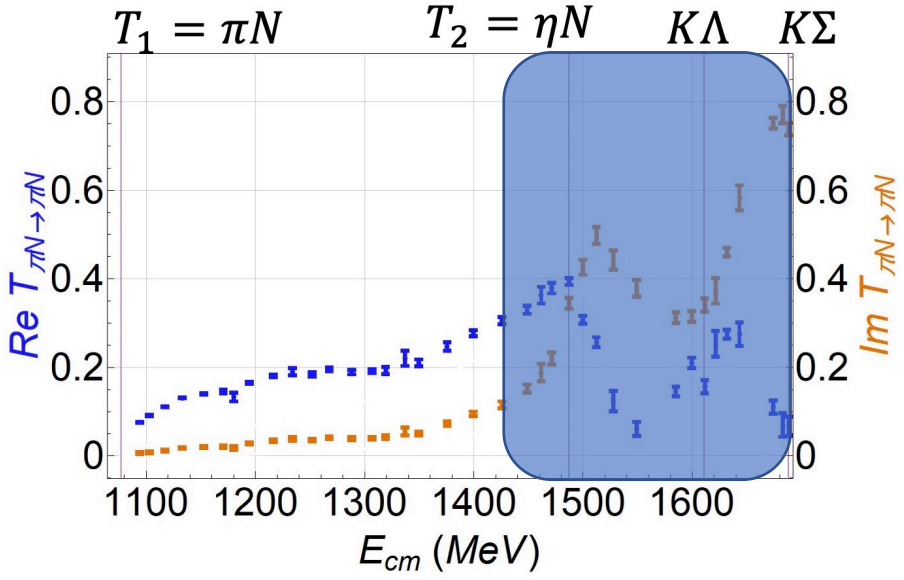
$S_{22}(p_1,p_2)=1$ unitary

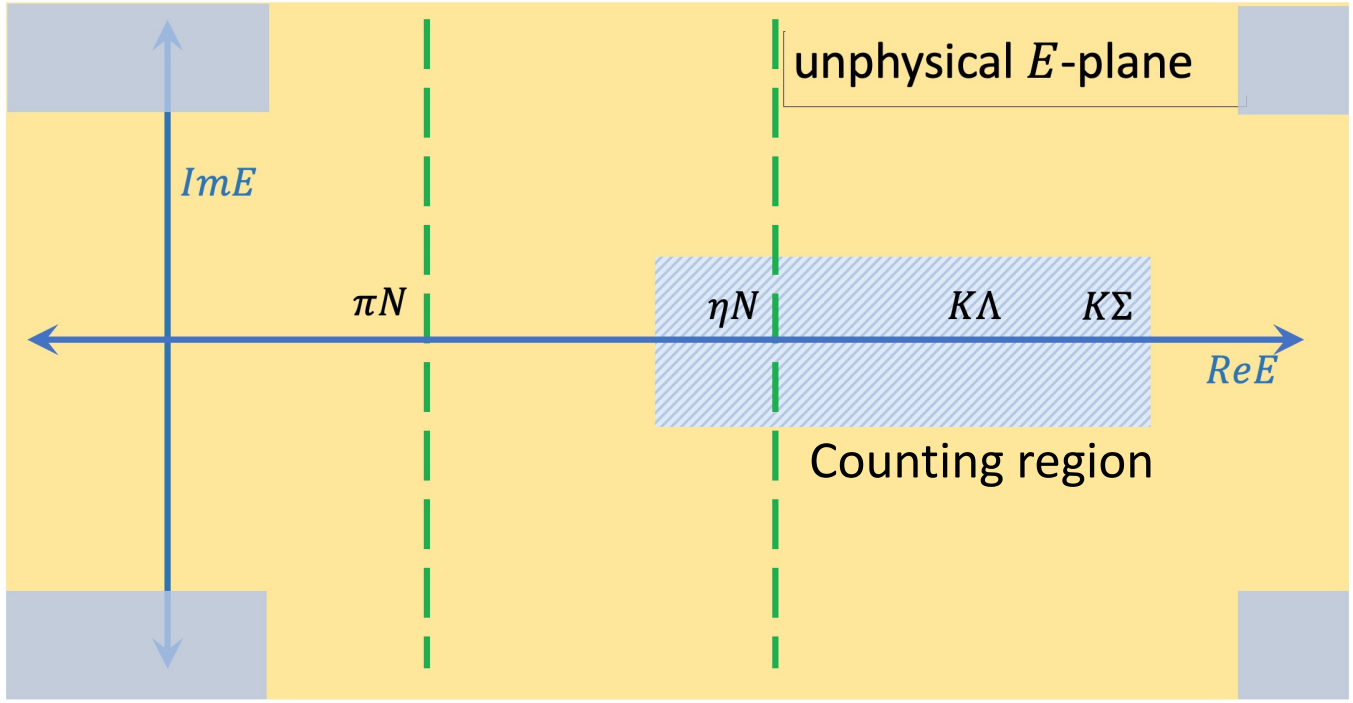
$S_{12}(p_1,p_2)=0$ uncoupled

See Appendix of arXiv:2105.04898

Generation of training dataset

We limit the region of analysis.





Region where poles are generated and counted:

$$\eta N - 50 \leq Re\ E_{pole} \leq \eta N + 200$$
$$-200 \leq Im\ E_{pole} \leq 200$$

Region where poles are generated but not counted:

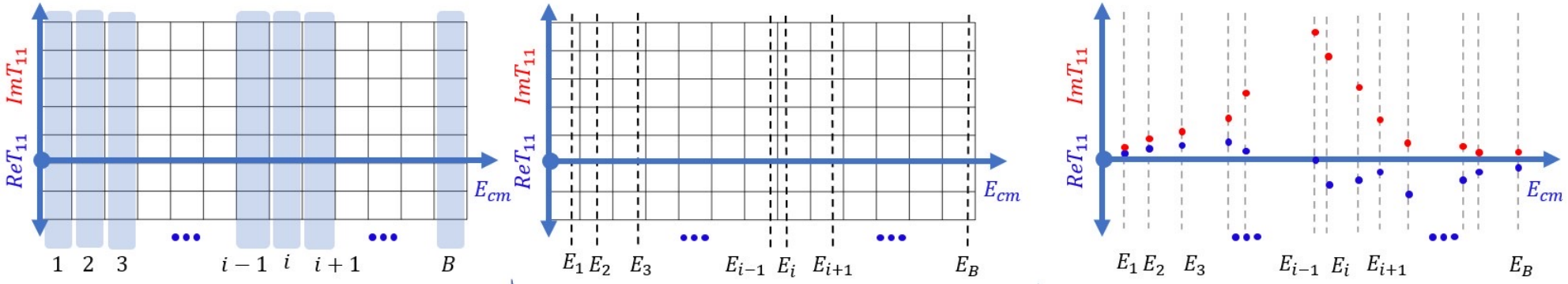
$$Re\ E_{pole} < \pi N - 100 \quad and \quad Re\ E_{pole} > \eta N + 500$$
$$700 < \left|Im\ E_{pole}\right| < 2000$$

$$S_{11}(p_1, p_2) = \prod_j \frac{D_j(-p_1, p_2)}{D_j(p_1, p_2)} \prod_n \frac{D_n(-p_1, p_2)}{D_n(p_1, p_2)}$$

We limit the maximum number of counted poles to 4.

Generation of training dataset

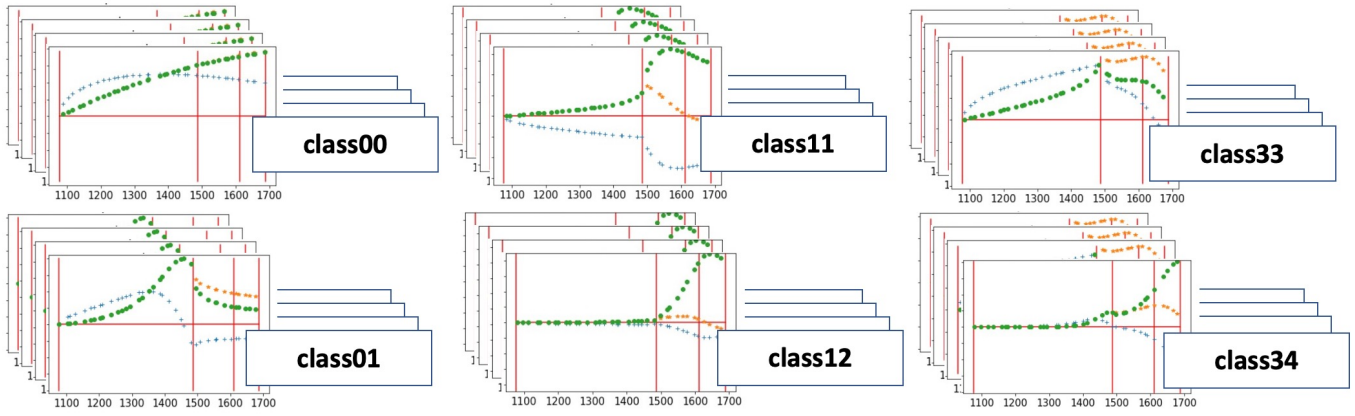
To simulate the limited energy resolution.



(1) Divide the energy range into B bins

(2) Choose a random energy value in each bin

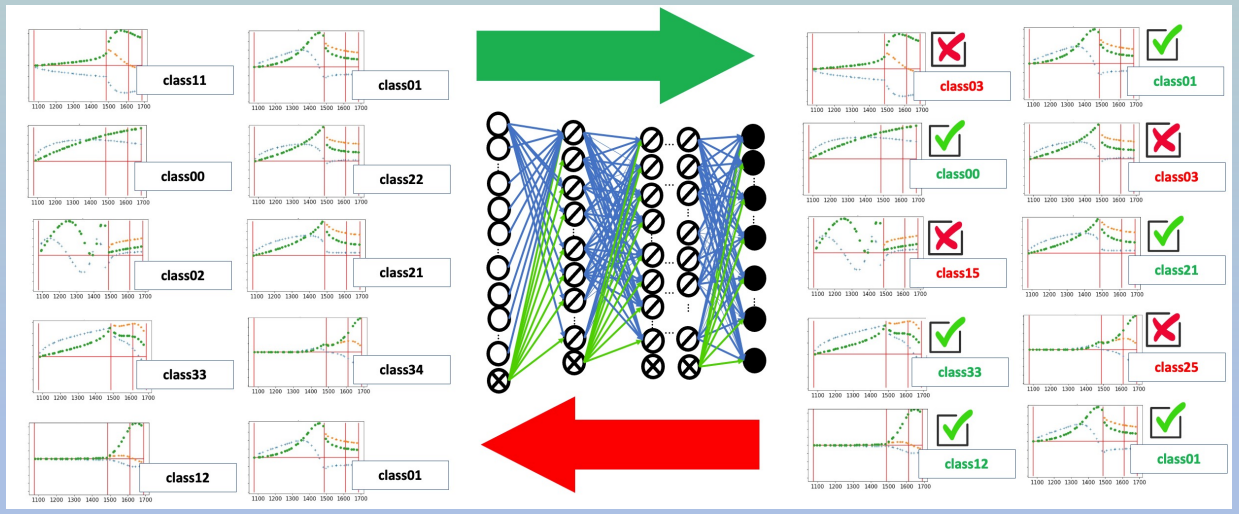
(3) Calculate the amplitude for each random energy value



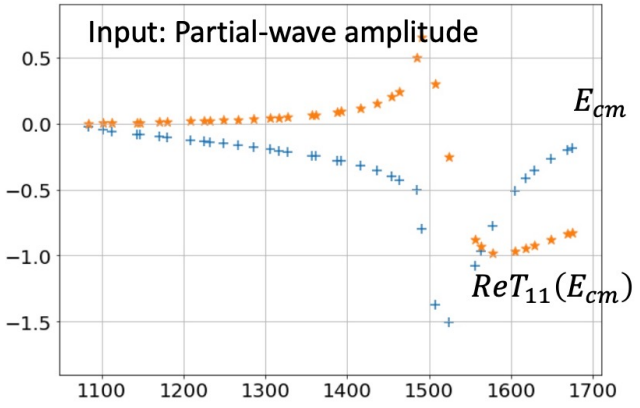
(4) Label each amplitude according to its pole-configuration

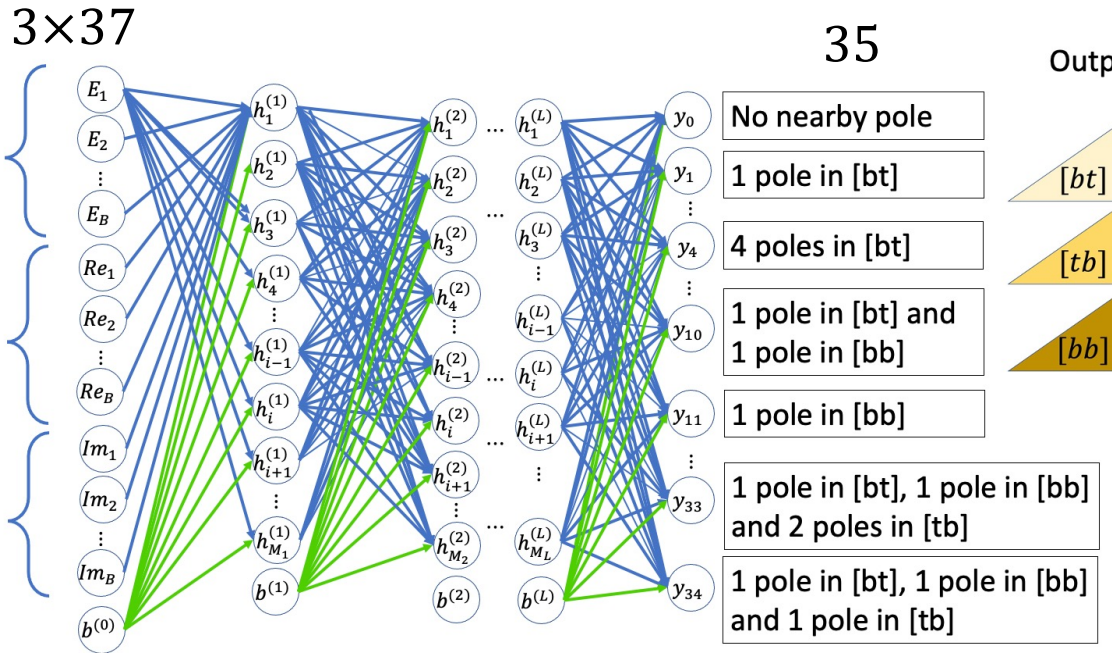
Label	S-matrix pole configuration
0	no nearby pole
1	1 pole in $[bt]$
2	2 poles in $[bt]$
⋮	⋮
32	1 pole in $[bt]$, 2 poles in $[bb]$ and 1 pole in $[tb]$
33	1 pole in $[bt]$, 1 pole in $[bb]$ and 2 poles in $[tb]$
34	1 pole in $[bt]$, 1 pole in $[bb]$ and 1 pole in $[tb]$

Optimization of DNN model

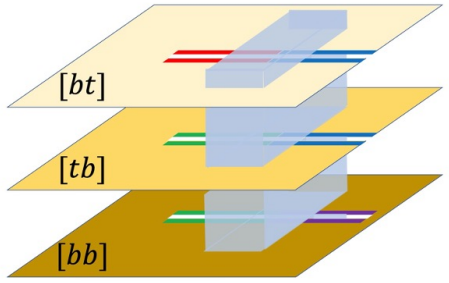


Optimization of DNN model



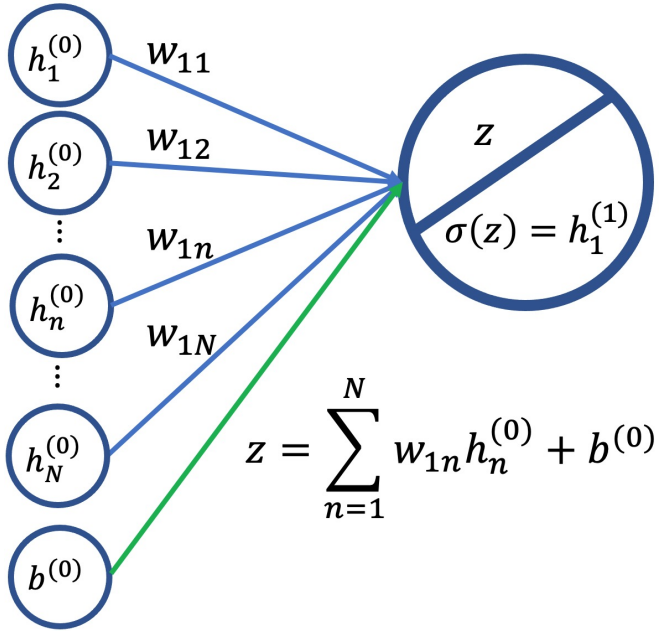


Output: Pole configuration



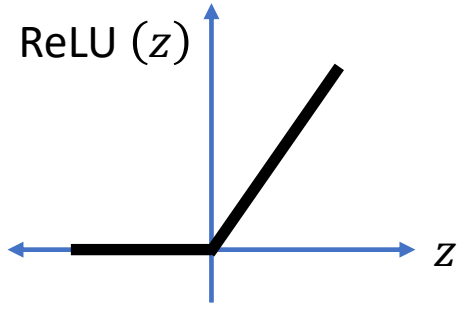
- No nearby pole
- 1 pole in [bt]
- 4 poles in [bt]
- 1 pole in [bt] and 1 pole in [bb]
- 1 pole in [bb]
- 1 pole in [bt], 1 pole in [bb] and 2 poles in [tb]
- 1 pole in [bt], 1 pole in [bb] and 1 pole in [tb]

Chainer
https://chainer.org/



$$z = \sum_{n=1}^{N} w_{1n} h_n^{(0)} + b^{(0)}$$

For hidden-layer nodes:



For output nodes:

$$\mathrm{SoftMax}\left(z_n^{\text{output−layer}}\right) = \frac{\exp\left(z_n^{\text{output−layer}}\right)}{\sum_{m=1}^{35} \exp\left(z_m^{\text{output−layer}}\right)}$$

Optimization: Find optimal values of weights and biases.

Optimization of DNN model

Define the cost-function (SoftMax cross entropy)

$$C(w,b) = \frac{1}{X}\sum_{\vec{x}} \vec{a}(\vec{x}) \cdot \log\left[\vec{y}_{w,b}(\vec{x})\right]$$

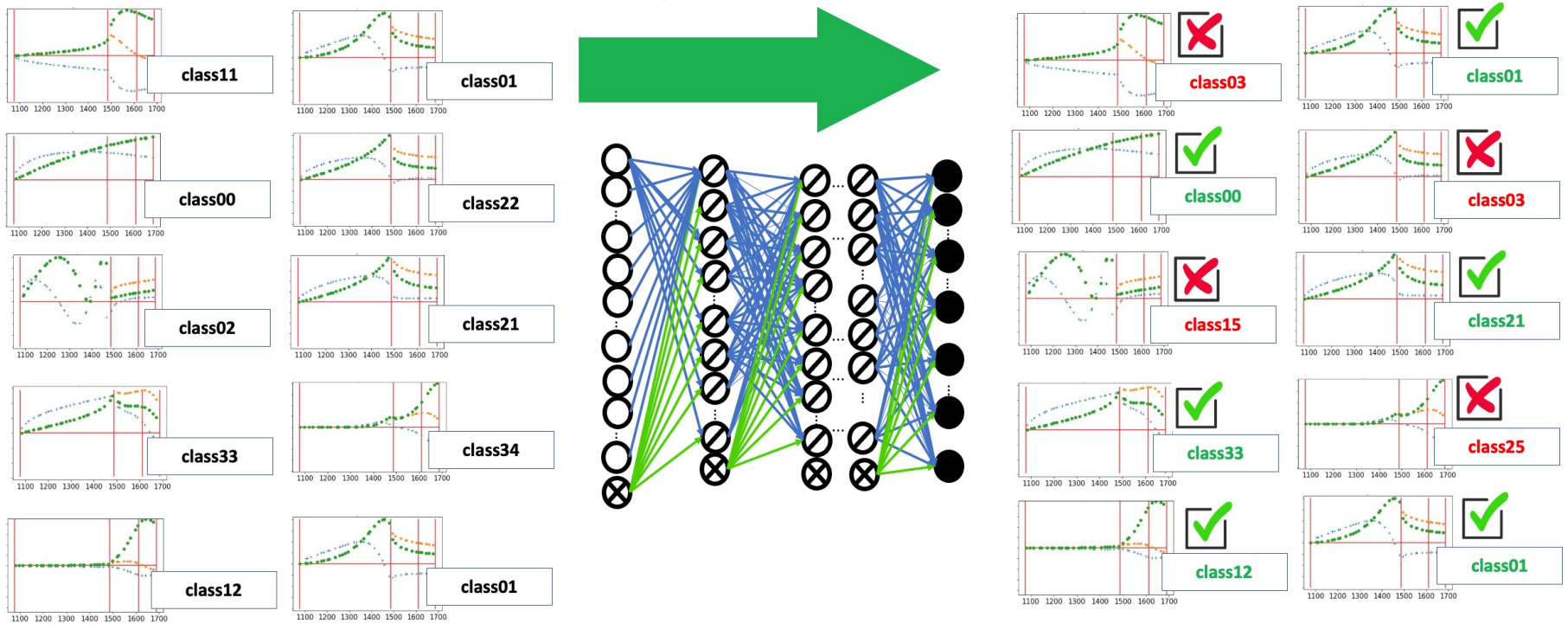
X: total number of input amplitudes in training dataset

$\vec{x}$: flattened input amplitude

$\vec{a}(\vec{x})$: true classification label of the input amplitude

$\vec{y}_{w,b}(\vec{x})$: label assigned by the DNN

Forward pass: estimate the present value of cost-function



The mis-match in the DNN labels allow us to estimate the value of $C(w,b)$ for the present state of the DNN.

Optimization of DNN model

Define the cost-function (SoftMax cross entropy)

$$C(w,b) = \frac{1}{X}\sum_{\vec{x}} \vec{a}(\vec{x}) \cdot \log\left[\vec{y}_{w,b}(\vec{x})\right]$$

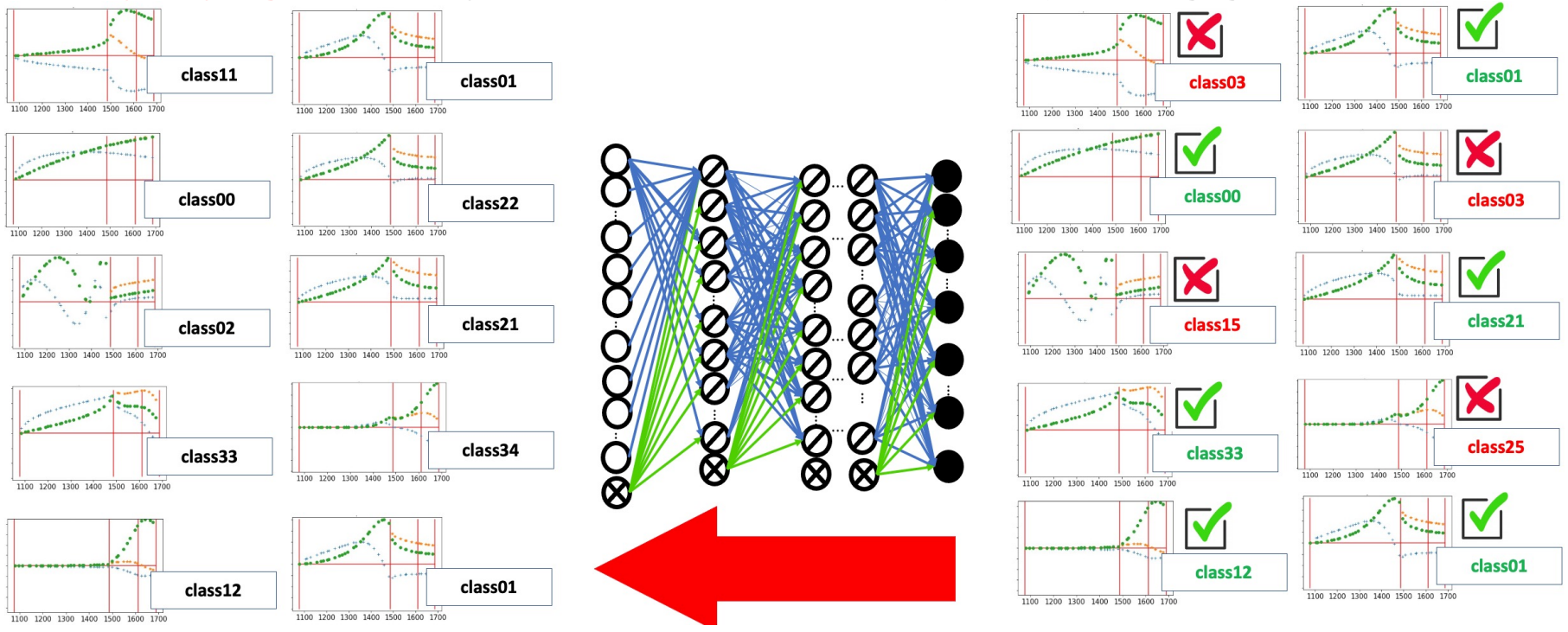
X: total number of input amplitudes in training dataset

$\vec{x}$: flattened input amplitude

$\vec{a}(\vec{x})$: true classification label of the input amplitude

$\vec{y}_{w,b}(\vec{x})$: label assigned by the DNN

Backpropagation: update the values of (w, b) using gradient descent

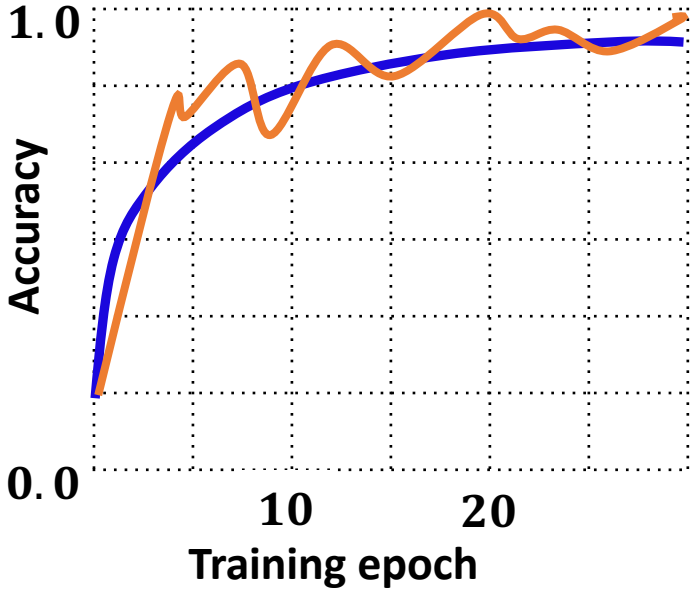


Optimization of DNN model

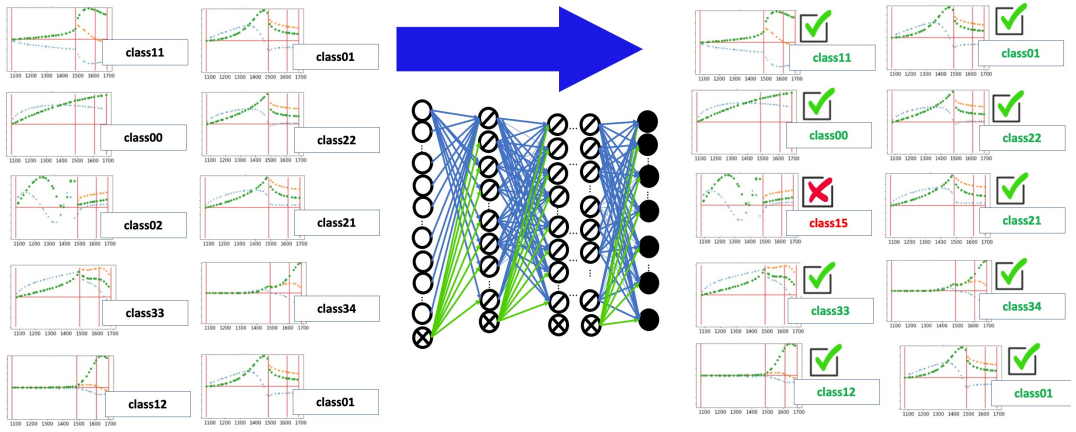
Introduce a performance metric

$$\textbf{Accuracy} = \frac{\textbf{Number of correct DNN labels}}{\textbf{Total number of labeled inputs}}$$

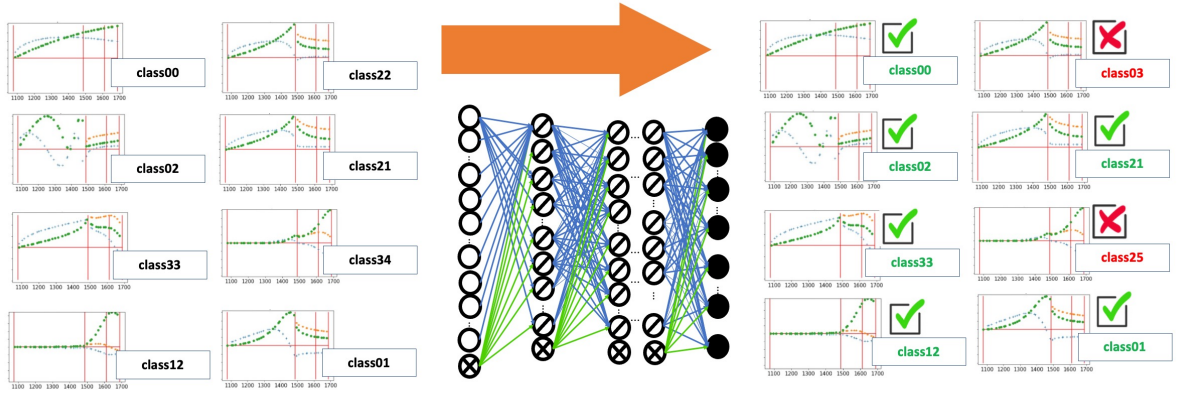
Typical performance plot



Performance on training set



Performance on an independent testing set



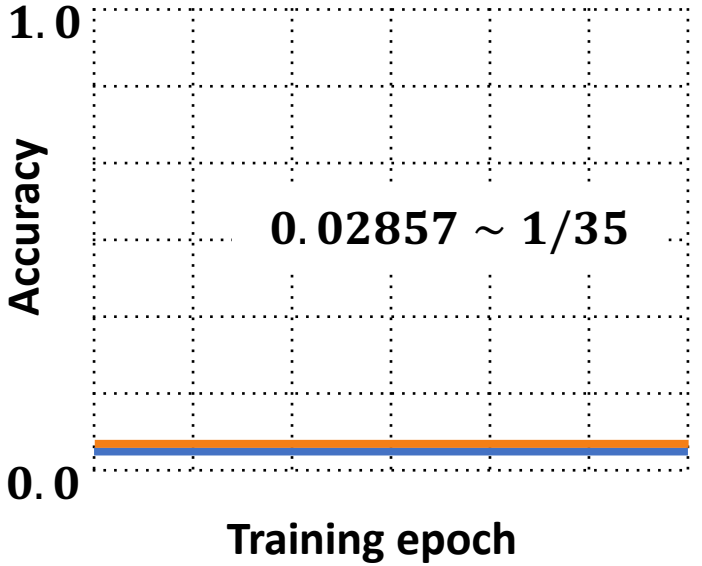
Optimization of DNN model

Input layer: 3 x 37
Output layer: 35

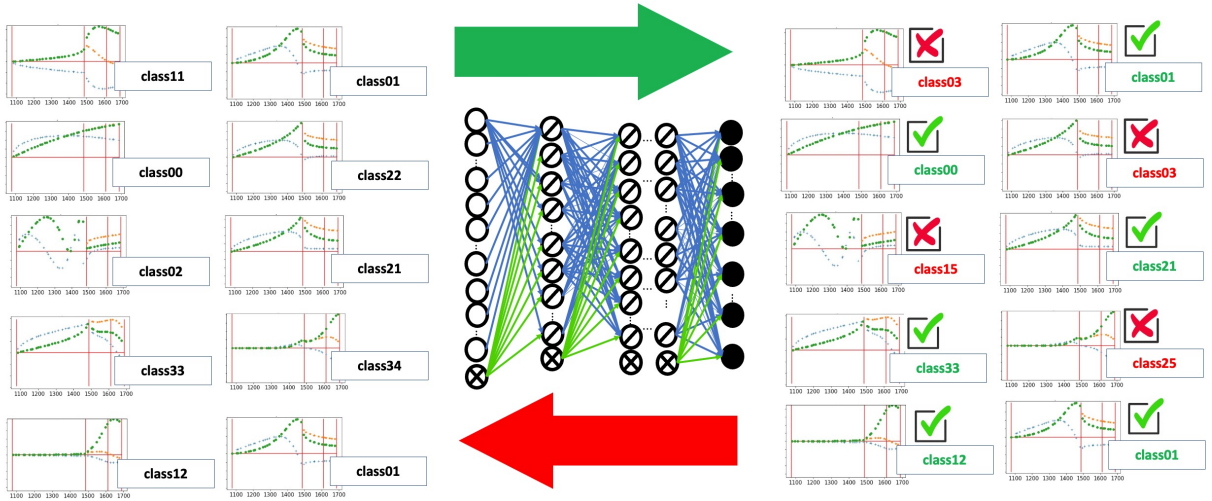
Hidden nodes: ReLU
Output nodes: softmax

Cost function:
SoftMax cross-entropy

DNN model label	Hidden layer architecture
1	[200-200]
2	[200-200-200]
3	[200-200-200-200]
4	[100-100]
5	[100-100-100]
6	[100-100-100-100]



All DNN models are just guessing.



Optimization of DNN model

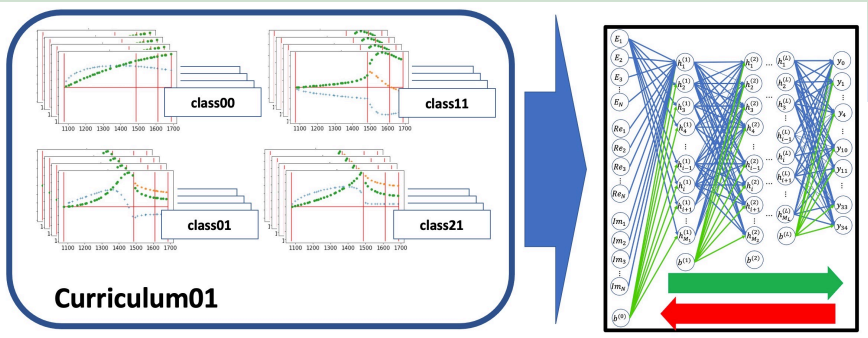
- Start with small easy examples.
  Elman 1993
- Slowly introduce new class until all examples are presented.
- Perform regular training loop

Chosen DNN architecture

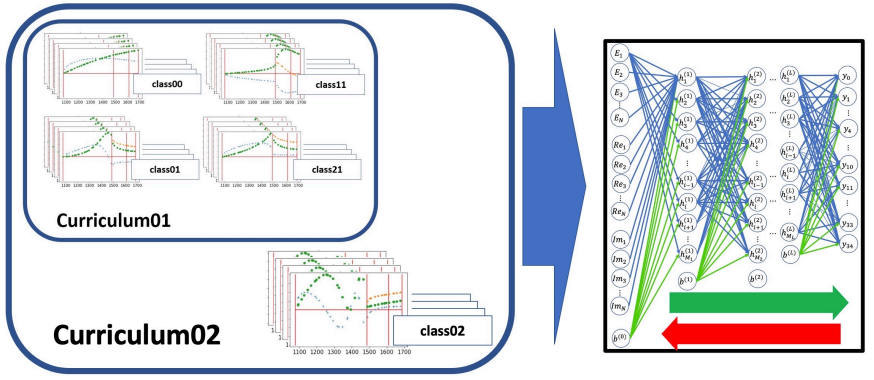
Layer	Number of nodes	Activation Function
Input	111+1	
1st	200+1	ReLU
2nd	200+1	ReLU
3rd	200+1	ReLU
Output	35	Softmax

arXiv:2105.04898
arXiv:2104.14182

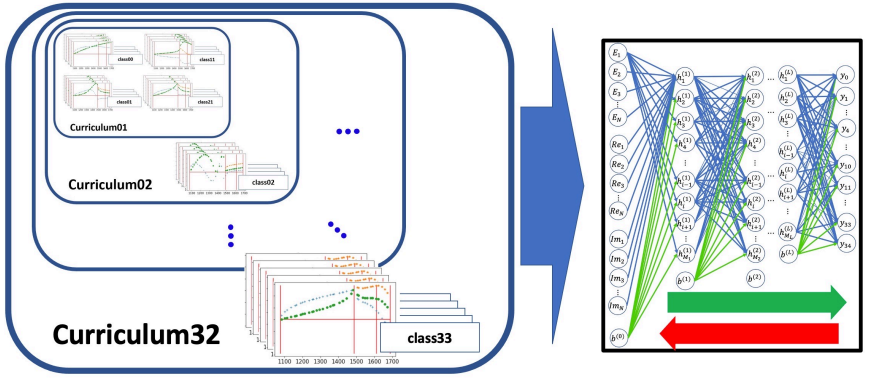
Curriculum01:
At-most-**one**-pole classification



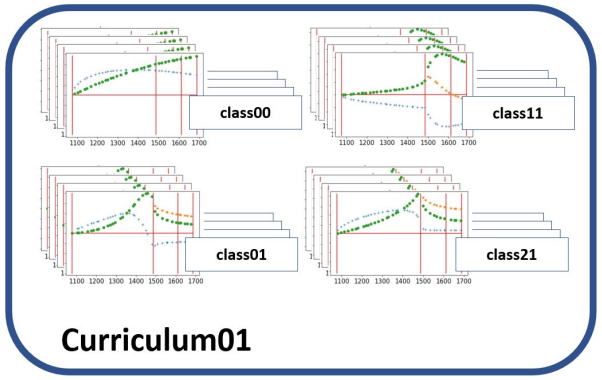
Curriculum02:
Curriculum01 +
1 new class in
the **two**-pole
classification set

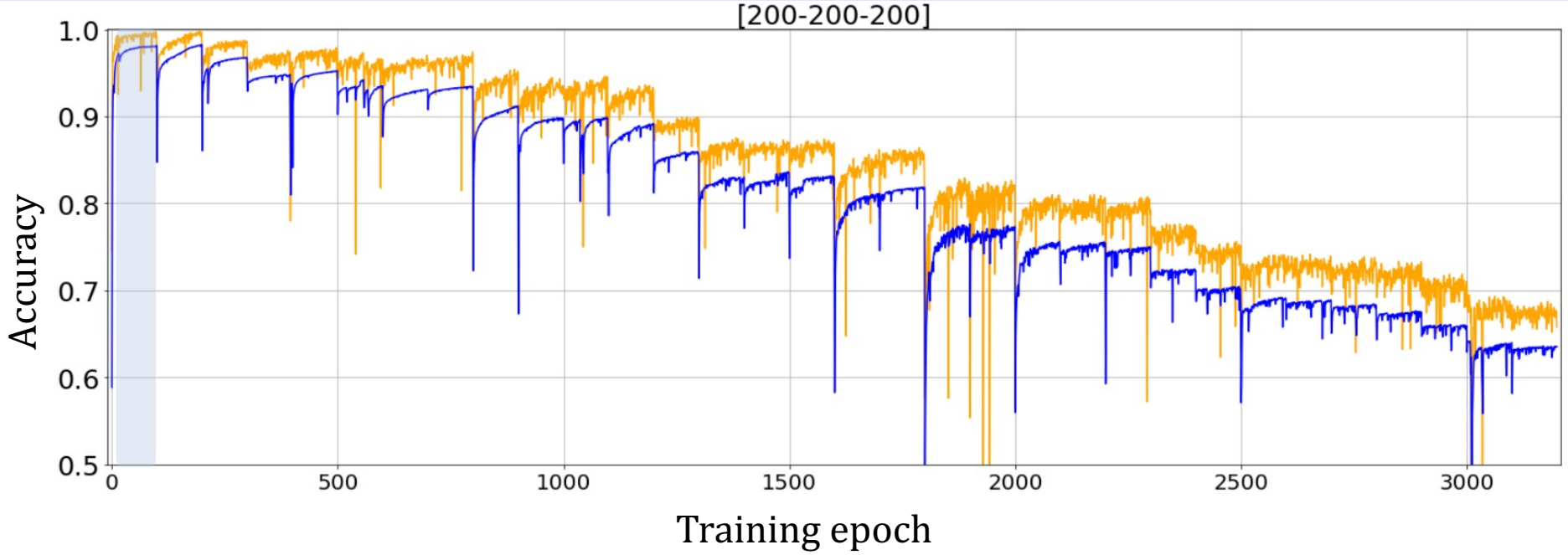


Curriculum32:
Curriculum31 +
last class in
the **four**-pole
classification set



Optimization of DNN model





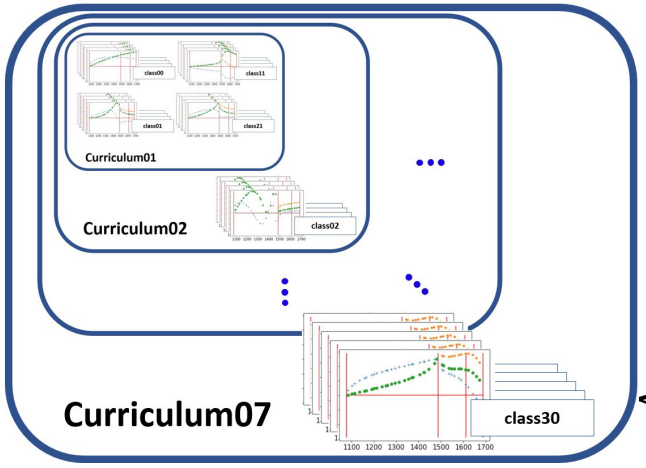
First 100 epochs (currciculum01) – most **one pole**

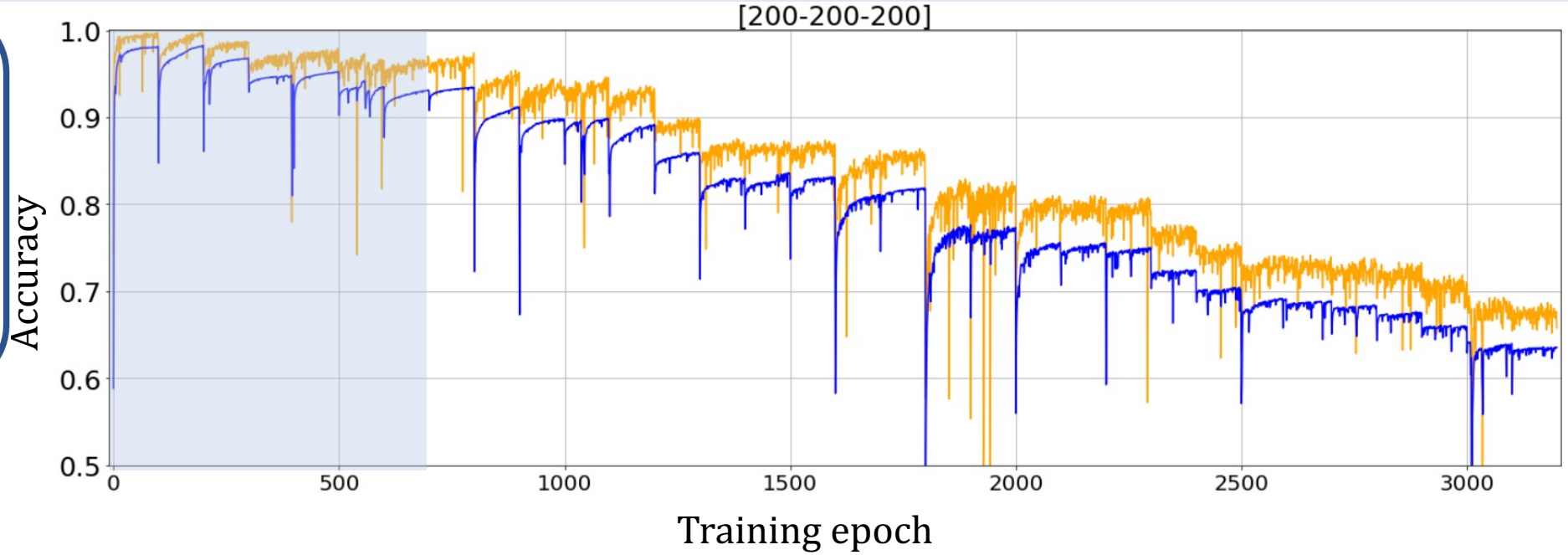
At-most-one-pole: 4 classifications (or pole configurations)

Random model accuracy (wild guessing): 25%

Easy task: accuracy is about 99%

Optimization of DNN model





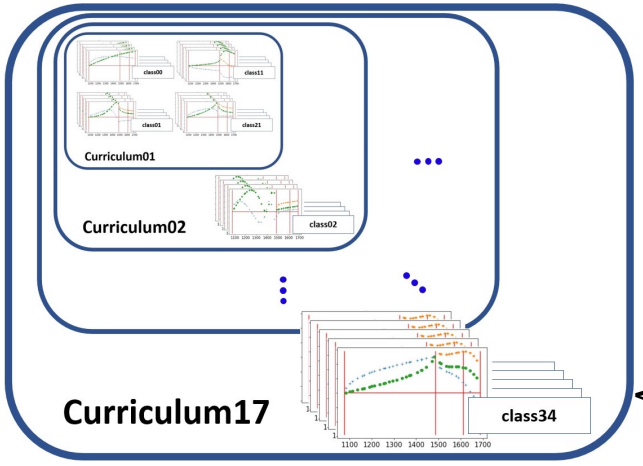
After 700 epochs (curriculum07) – most **two poles**

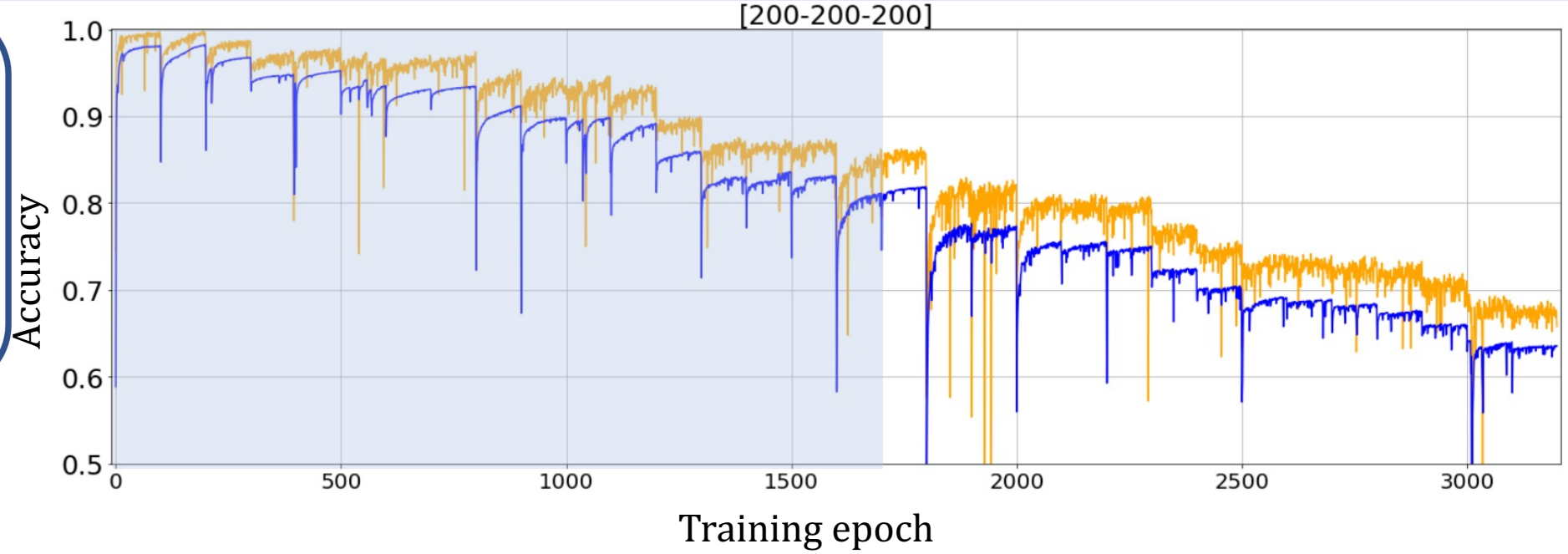
At-most-two-poles: 10 pole configurations

Random model accuracy (wild guessing): 10%

Accuracy is about 93%

Optimization of DNN model





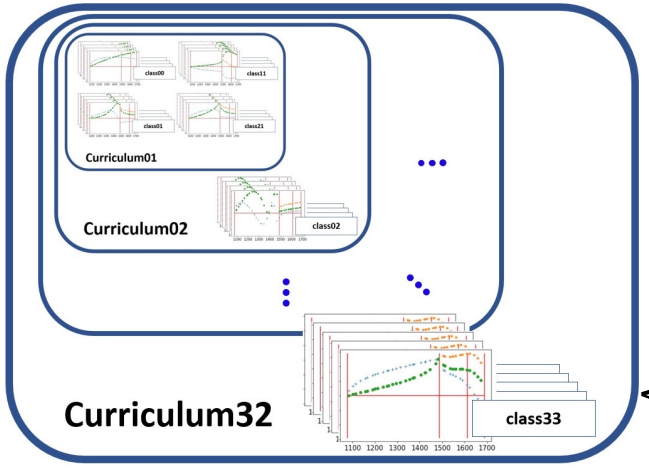
After 1700 epochs (curriculum17) – most **three poles**

At-most-three-poles: 20 pole configurations

Random model accuracy (wild guessing): 5%

Accuracy is about 81%

Optimization of DNN model



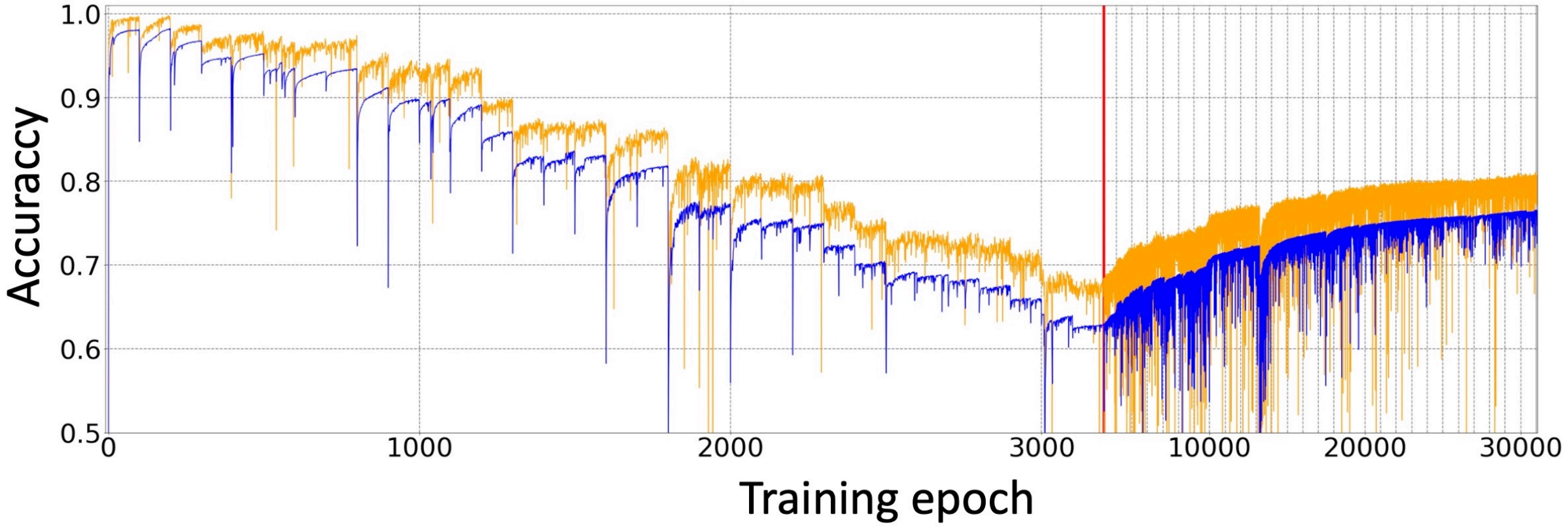
After 3200 epochs (curriculum32) – most **four poles**

At-most-four-poles: 35 pole configurations

Random model accuracy (wild guessing): 2.86%

Accuracy is about 63.5%

Optimization of DNN model

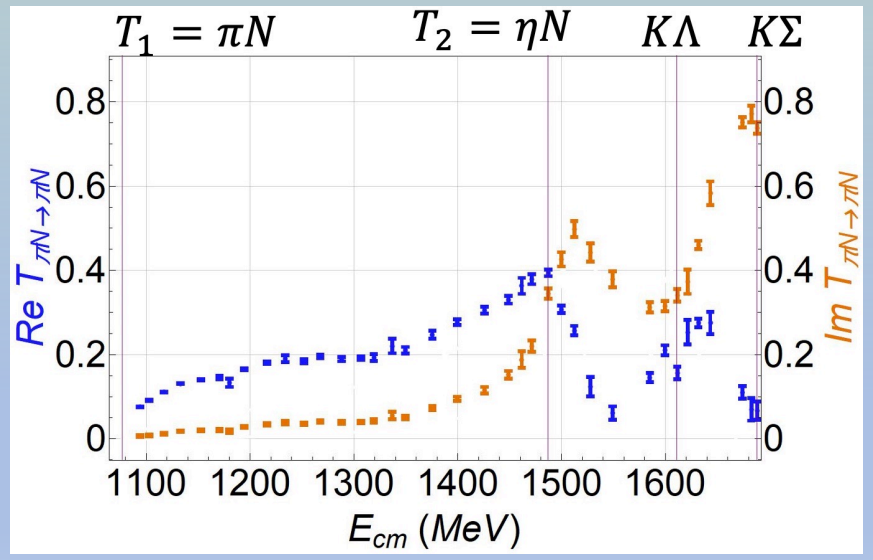


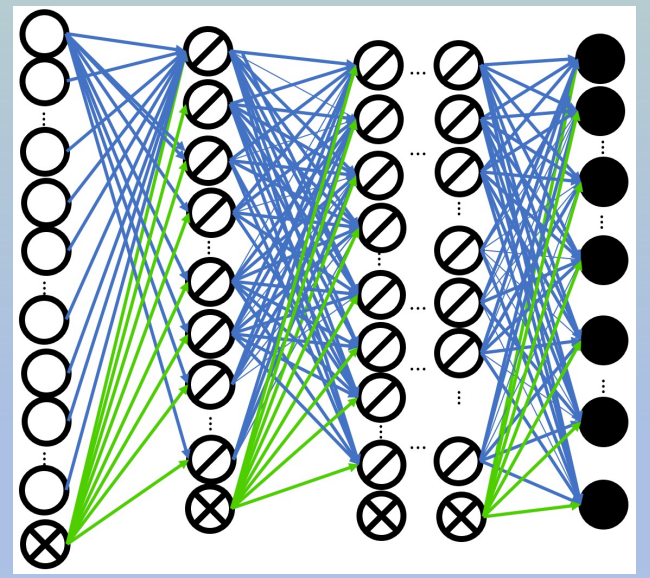
Random model accuracy (wild guessing): 2.86%

Post-curriculum training accuracy: 76.5%
Post-curriculum testing accuracy: 80.4%

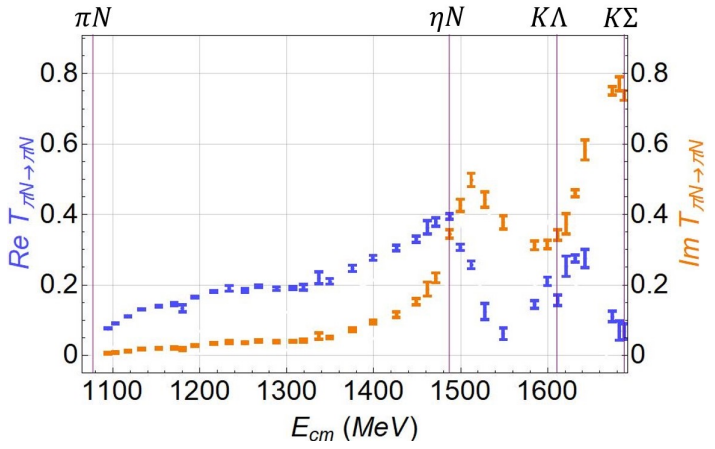
We now have a DNN that can detect up to four poles on any Riemann sheet.

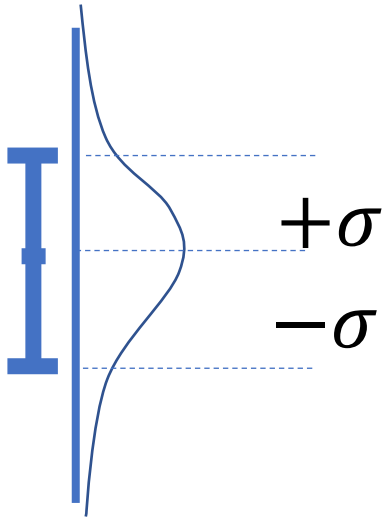
Inference stage: Application

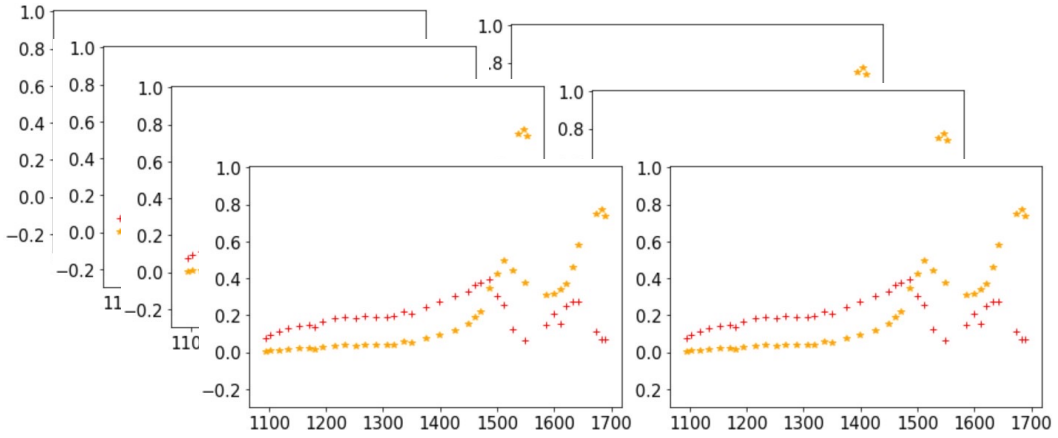




Inference stage: Application







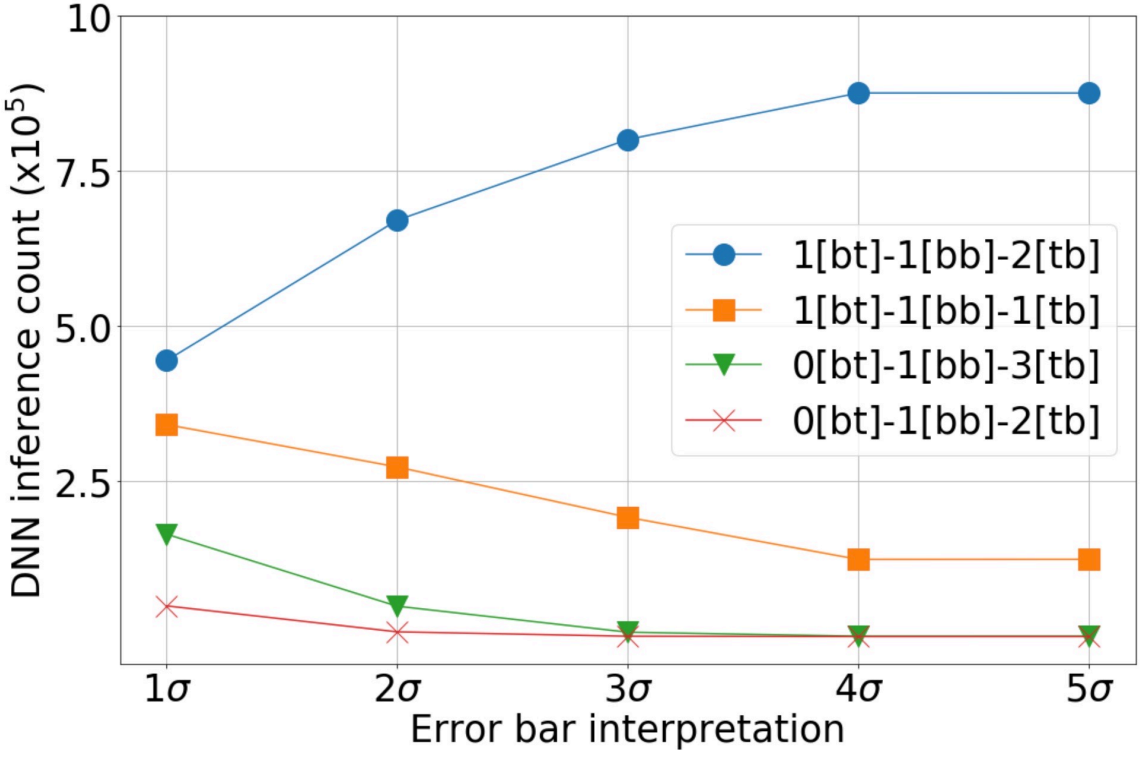
- Pick random points in each error bar
- No model-fitting step is needed
- Generate 10^6 amplitudes
- Feed to the trained DNN
- Count the number of outcomes

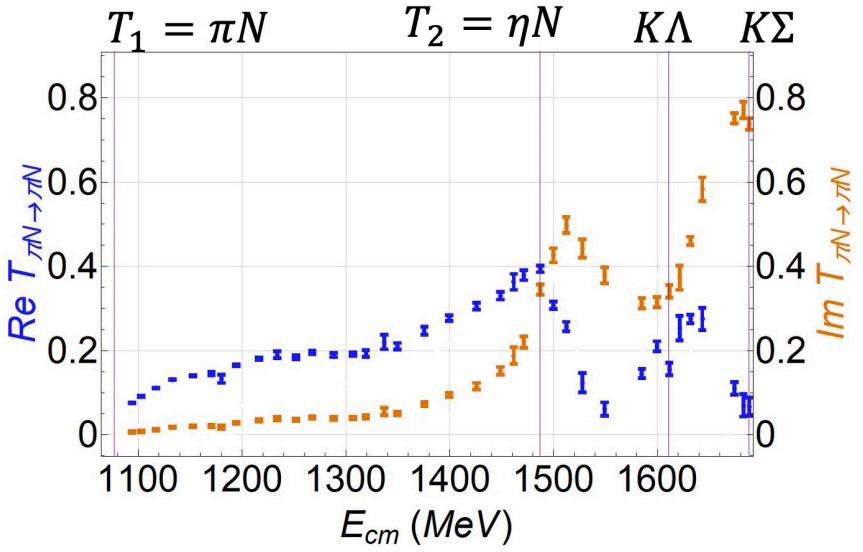
DNN inference on 10^6 amplitudes using 1 error bar = 1σ

- 44.6% 1[bt]-1[bb]-2[tb]
- 34.1% 1[bt]-1[bb]-1[tb]
- 16.4% 0[bt]-1[bb]-3[tb]
- 4.9% 0[bt]-1[bb]-2[tb]

Inference stage: Application

DNN inference on 10^6 amplitudes using Gaussian distribution for each error bar

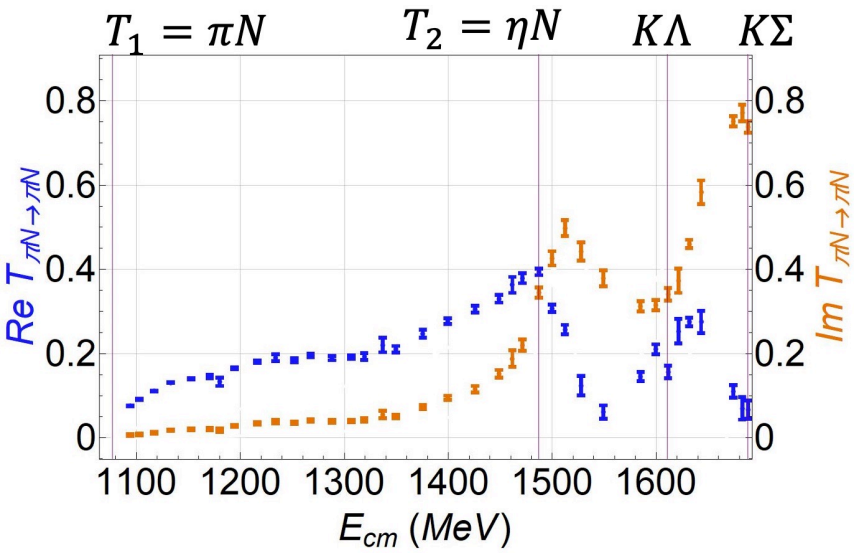




DNN inference on 10^6 amplitudes using uniform distribution for each error bar

- 60.3% 1[bt]-1[bb]-2[tb]
- 30.9% 1[bt]-1[bb]-1[tb]
- 7.5% 0[bt]-1[bb]-3[tb]
- 1.3% 0[bt]-1[bb]-2[tb]

Inference stage: Application (Discussion)



DNN inference on 10^6 amplitudes using uniform distribution for each error bar

- 60.3% 1[bt]-1[bb]-2[tb]
- 30.9% 1[bt]-1[bb]-1[tb]
- 7.5% 0[bt]-1[bb]-3[tb]
- 1.3% 0[bt]-1[bb]-2[tb]

Guide to parametrization

- Detected [bb] pole
  - Enhancement between $K\Lambda$ and $K\Sigma$ thresholds.
- Detected [bt] pole
  - Far from ηN threshold but within the counting region
  - Might be shadow of the detected [bb]
- Detected [tb] poles
  - Only poles left to explain ηN enhancement

Summary and Outlook

Summary:

- Generate training dataset using the general properties of S-matrix.
- Optimization of DNN model with noisy dataset (energy resolution).
- DNN inference stage
  - No assumed functional form for the amplitude
  - No assumptions made on the detected poles

Use the DNN result to design an appropriate parametrization

Summary and Outlook

Outlook:

- Get the relevant partial wave (given the cross-section)
- Multi-DNN analysis
  - Pole configuration
  - Pole positions
- Inclusion of models to trace the pole's origin

- DNN applicable to any two-hadron scattering
- DNN to distinguish resonance and kinematical enhancements

Thank you for listening.