Physics-constrained GAN for amplitude extraction

Glòria Montaña Faiget

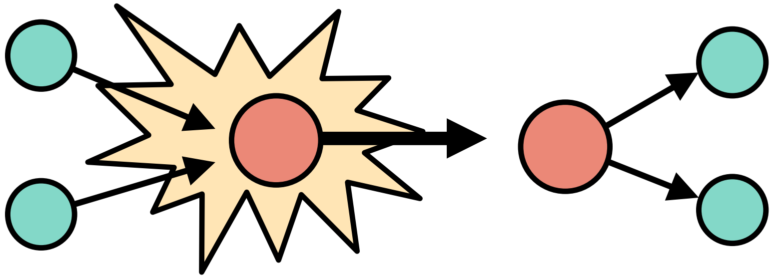
Jefferson Lab

In collaboration with M. Battaglieri, A. Pilloni, Y. Li and others

Digital Twins for Nuclear and Particle Physics – NPTwins 2024
Museo Diocesano di Genova
December 16-18, 2024

Jefferson Lab JSA A SURA Company A(i)DAPT AI for Data Analysis and PreservaTion JPAC EXOHAD Exotic Hadrons Topical Collaboration

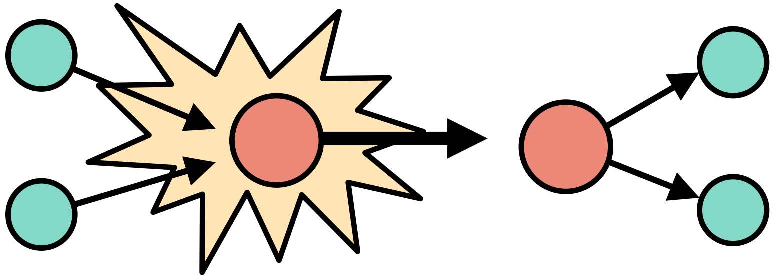
MOTIVATION



The **scattering amplitude** $\mathcal{A}$ is the fundamental quantity of interest in hadron reactions:

- Encodes the underlying dynamics of the interaction
- Crucial for understanding resonance production, decays…
- Is a complex quantity: magnitude + phase

MOTIVATION



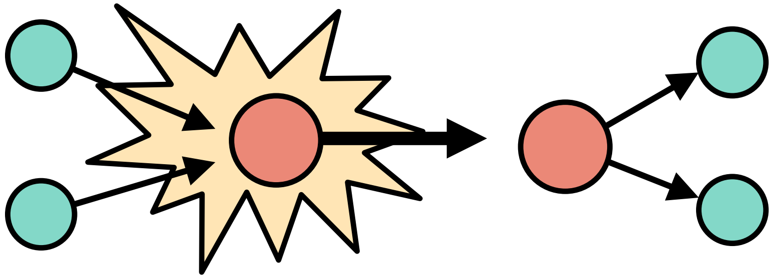
The **scattering amplitude** $\mathcal{A}$ is the fundamental quantity of interest in hadron reactions:

- Encodes the underlying dynamics of the interaction
- Crucial for understanding resonance production, decays…
- Is a complex quantity: magnitude + phase

The **cross section** σ is an experimentally observable quantity:

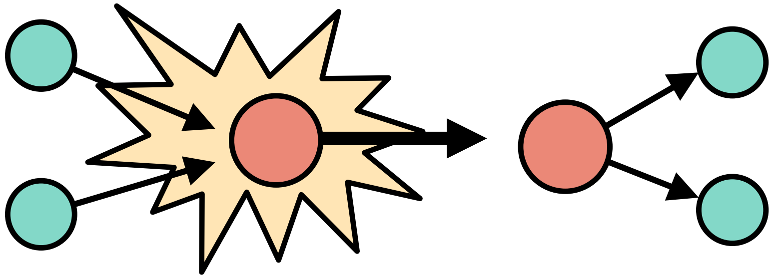
- Related to $|\mathcal{A}|^2$
- The information about the phase is lost

MOTIVATION



The reconstruction of the amplitude from the differential cross section is hard, even for the simplest elastic $2 \to 2$ scattering.

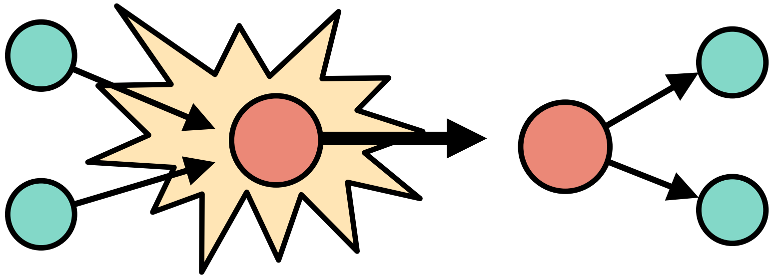
MOTIVATION



The reconstruction of the amplitude from the differential cross section is hard, even for the simplest elastic $2 \to 2$ scattering.

Can we use modern machine-learning techniques to recover the scattering amplitude from experimental data of cross sections?

MOTIVATION



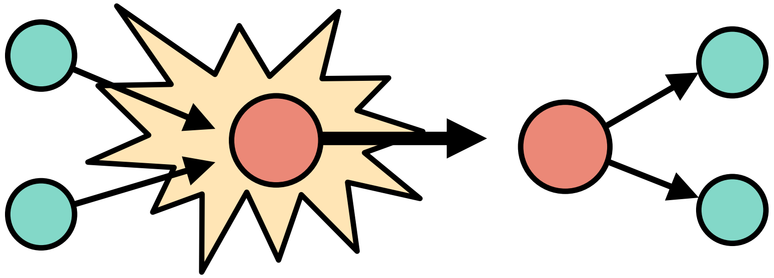
The reconstruction of the amplitude from the differential cross section is hard, even for the simplest elastic 2 → 2 scattering.

Can we use modern machine-learning techniques to recover the scattering amplitude from experimental data of cross sections?

Why physics-constrained **GANs**?

- Learn distributions and patterns of the (pseudo) data
- Incorporate physics constraints

MOTIVATION



The reconstruction of the amplitude from the differential cross section is hard, even for the simplest elastic 2 → 2 scattering.

Can we use modern machine-learning techniques to recover the scattering amplitude from experimental data of cross sections?

Why physics-constrained **GANs**?

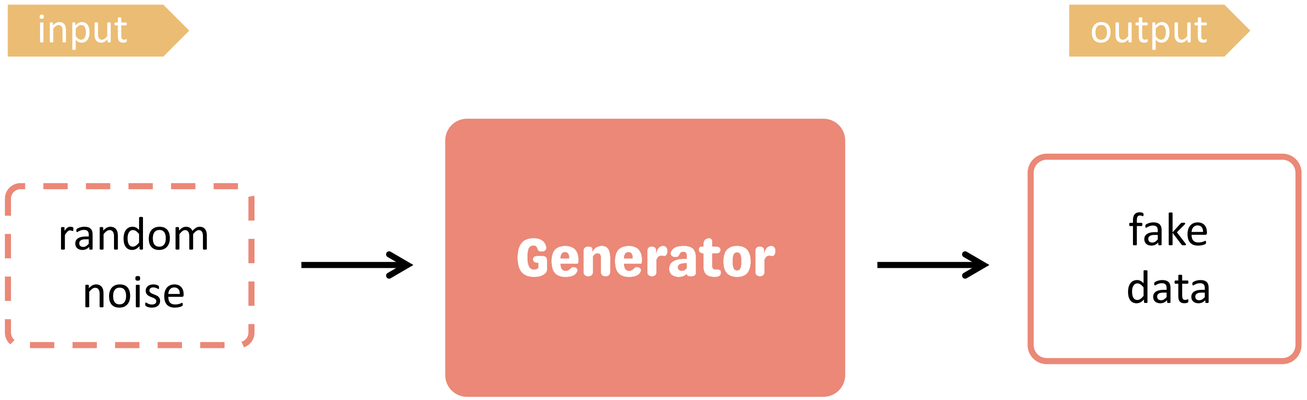
- Learn distributions and patterns of the (pseudo) data
- Incorporate physics constraints

The modulus and phase of the scattering amplitude are related by the **unitarity relation**.

INTRODUCTION TO GANs

Two neural networks:

- The **generator** needs to capture the data distribution

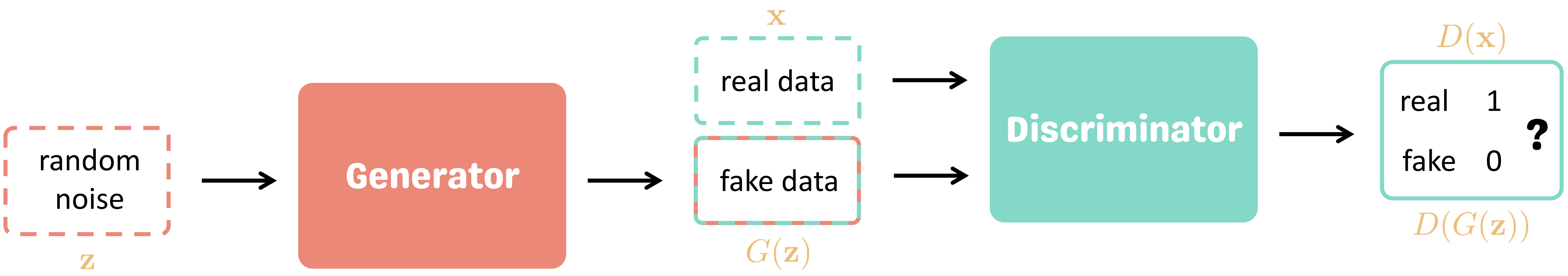


INTRODUCTION TO GANs

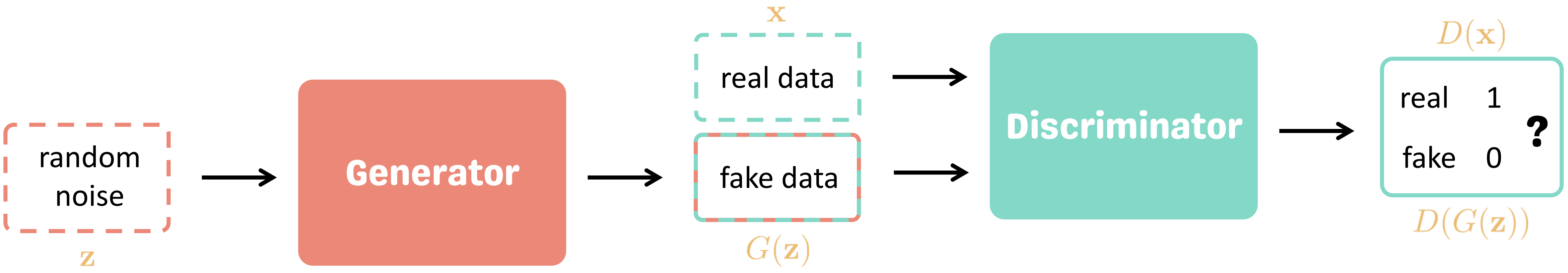
Two neural networks:

- The **generator** needs to capture the data distribution
- The **discriminator** estimates the probability that a sample comes from the training data rather than from the generator

INTRODUCTION TO GANs



INTRODUCTION TO GANs

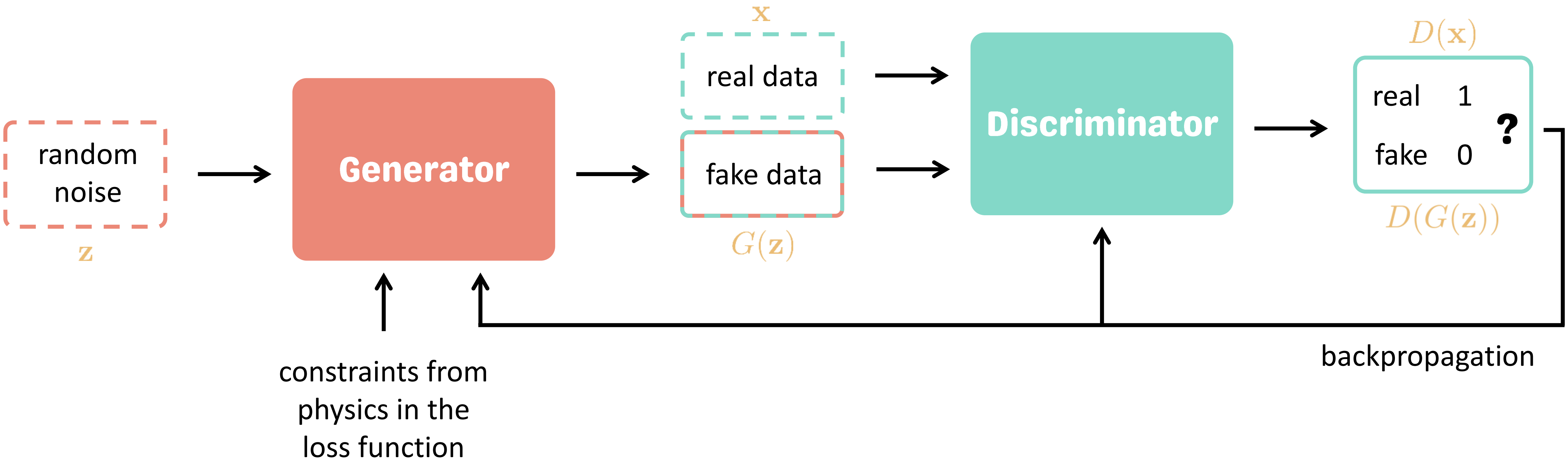


GAN Minmax Game

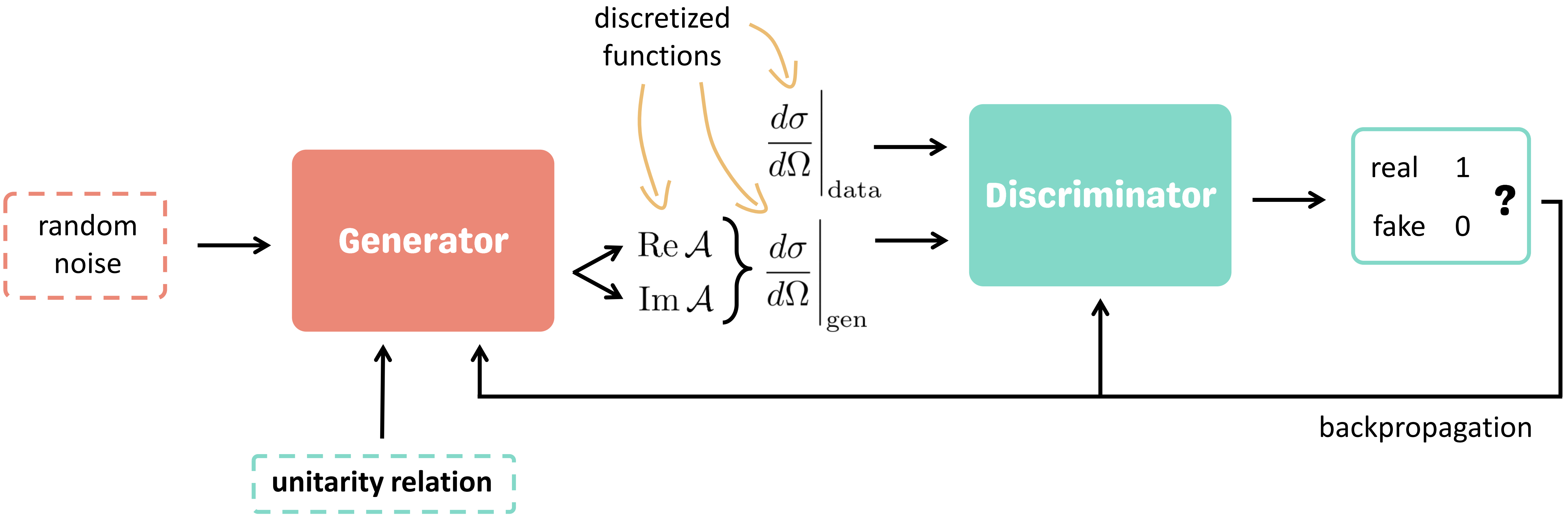
$$\min_G \max_D V(D,G) = \underbrace{\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}[\log D(\mathbf{x})]}_{\text{real data}} + \underbrace{\mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[\log(1 - D(G(\mathbf{z})))]}_{\text{fake data}}$$

	real data	fake data
G minimizes ↓	N/A	*D* classifies as 0
D maximizes ↑	*D* classifies as 1	*D* classifies as 0

INTRODUCTION TO GANs



INTRODUCTION TO GANs



PHYSICS PROBLEM

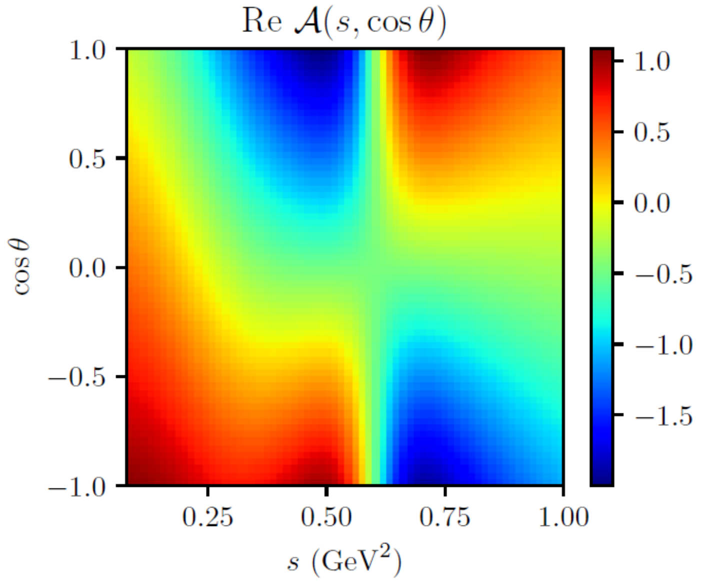
Elastic scattering $\pi^+\pi^- \to \pi^+\pi^-$

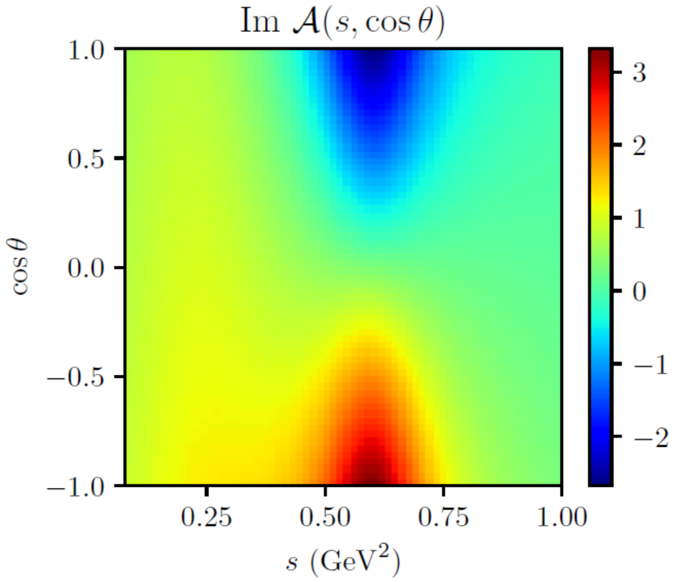
➔ dominated by $f_0(500)$ and $\rho(770)$ resonances

$$\mathcal{A}(s, \cos\theta) = \sum_{\ell=0}^{n} (2\ell+1) f_\ell(s) P_\ell(\cos\theta)$$

Partial-wave decomposition of the amplitude truncated to $n = 1$ and Breit-Wigner type partial waves:

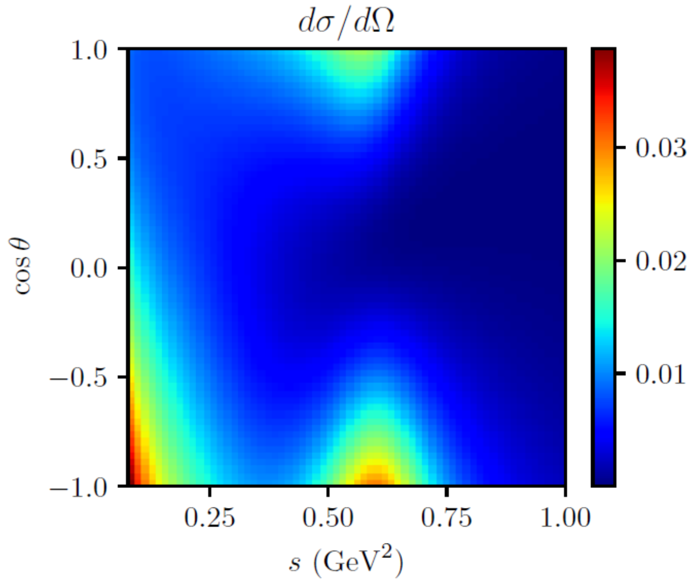
$$\mathcal{A}(s, \cos\theta) = f_0(s) + 3 f_1(s) \cos\theta \qquad f_\ell = \frac{m_\ell \Gamma_\ell}{m_\ell^2 - s - i m_\ell \Gamma_\ell}$$





Differential cross section

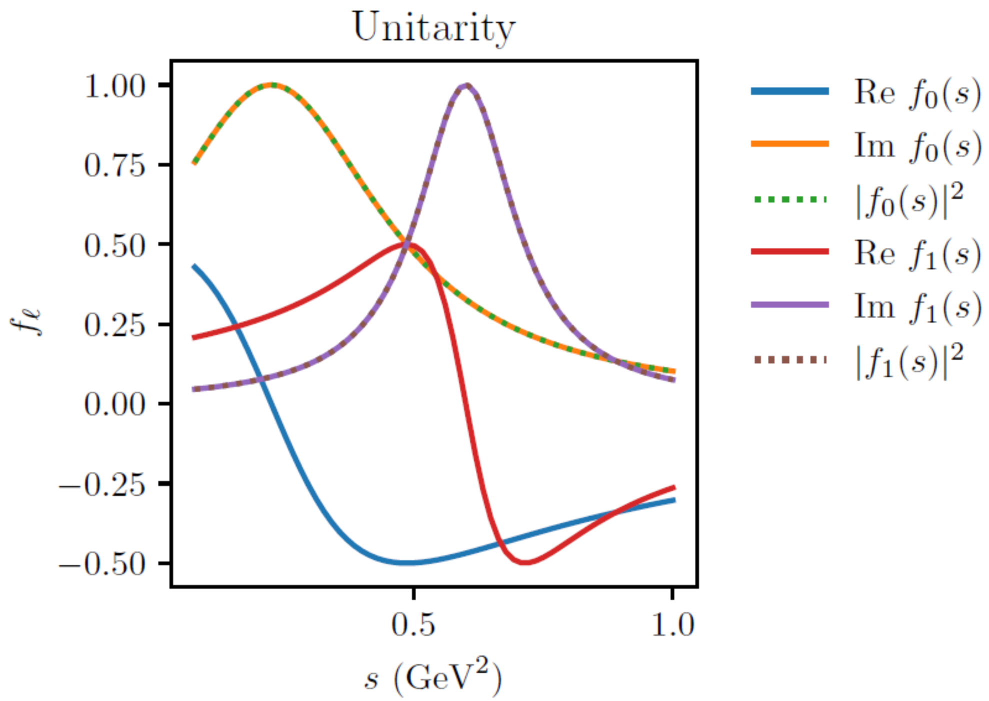
$$\frac{d\sigma}{d\Omega} = \frac{1}{64\pi^2} \frac{1}{s} |\mathcal{A}(s, \cos\theta)|^2$$



PHYSICS PROBLEM

- Unitarity of the partial waves

$$\text{Im } f_\ell(s) = |f_\ell(s)|^2$$

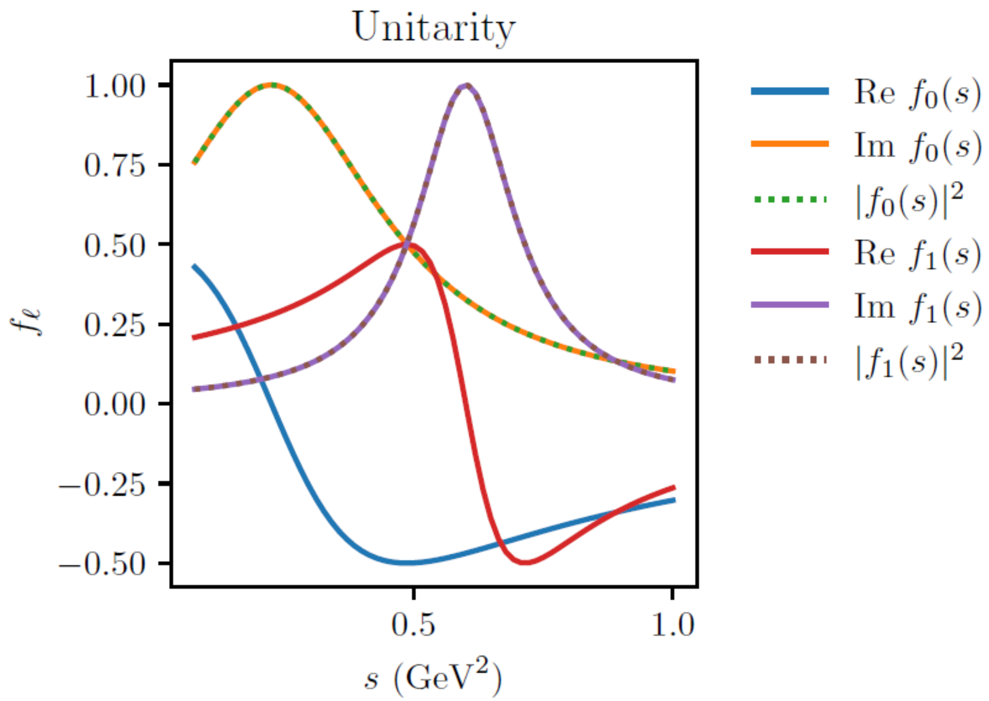


$$f_\ell(s) = \frac{1}{2}\int_{-1}^{+1} dz P_\ell(z)\mathcal{A}(s,z)$$

PHYSICS PROBLEM

- Unitarity of the partial waves

$$\text{Im } f_\ell(s) = |f_\ell(s)|^2$$



$$f_\ell(s) = \frac{1}{2}\int_{-1}^{+1} dz P_\ell(z)\mathcal{A}(s,z)$$

- Integral unitarity relation for the full amplitude

$$\text{Im } \mathcal{A}(s,z) = \frac{1}{4\pi}\int_0^{2\pi} d\phi \int_{-1}^{+1} dz' \mathcal{A}(s,z')\mathcal{A}^*(s,z'')$$

$$z'' = zz' + \sqrt{1-z^2}\sqrt{1-z'^{\,2}}\cos\phi$$

or, equivalently

$$\sin\Phi(s,z) = \int_0^{2\pi} d\phi \int_{-1}^{+1} dz' \frac{|\mathcal{A}(s,z')||\mathcal{A}(s,z'')|}{4\pi|\mathcal{A}(s,z)|} \times \cos\left[\Phi(s,z') - \Phi(s,z'')\right]$$

Phase ambiguity: $\mathcal{A}(s,z) \to -\mathcal{A}^*(s,z)$

$$\Phi(s,z) \to \pi - \Phi(s,z)$$

IMPLEMENTATION DETAILS

python
PyTorch

GAN architecture:

Generator

Layer Type	Input Dimensions	Output Dimensions	Activation/Other Details
Fully Connected	(batch_size, noise_dim)	(batch_size, $4 \times 4 \times 1024$)	BatchNorm, LeakyReLU (0.2)
Reshape	(batch_size, $4 \times 4 \times 1024$)	(batch_size, 4, 64, 64)	Reshapes tensor
ConvTranspose2d (1)	(batch_size, 4, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (2)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (3)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (4)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
Conv2d (Final)	(batch_size, 64, 64, 64)	(batch_size, 2, 64, 64)	Produces 2-channel image output

Lambda

Layer Type	Input Dimensions	Output Dimensions	Transformation Details
Lambda	(batch_size, 2, 64, 64)	(batch_size, 1, 64, 64)	Maps generator output to discriminator input

Discriminator

Layer Type	Input Dimensions	Output Dimensions	Activation/Other Details
Conv2d (1)	(batch_size, 1, 64, 64)	(batch_size, 64, 32, 32)	LeakyReLU (0.2), Dropout (0.3), Kernel=4
Conv2d (2)	(batch_size, 64, 32, 32)	(batch_size, 128, 16, 16)	BatchNorm, LeakyReLU (0.2), Dropout (0.3)
Flatten	(batch_size, 128, 16, 16)	(batch_size, $128 \times 16 \times 16$)	Flattens for FC layers
Fully Connected (1)	(batch_size, $128 \times 16 \times 16$)	(batch_size, 64)	LeakyReLU (0.2), Dropout (0.3)
Fully Connected (2)	(batch_size, 64)	(batch_size, 1)	Outputs real/fake score

IMPLEMENTATION DETAILS

python
PyTorch

GAN architecture:

Generator

Layer Type	Input Dimensions	Output Dimensions	Activation/Other Details
Fully Connected	(batch_size, noise_dim)	(batch_size, $4 \times 4 \times 1024$)	BatchNorm, LeakyReLU (0.2)
Reshape	(batch_size, $4 \times 4 \times 1024$)	(batch_size, 4, 64, 64)	Reshapes tensor
ConvTranspose2d (1)	(batch_size, 4, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (2)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (3)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
ConvTranspose2d (4)	(batch_size, 64, 64, 64)	(batch_size, 64, 64, 64)	BatchNorm, LeakyReLU (0.2), Kernel=17
Conv2d (Final)	(batch_size, 64, 64, 64)	(batch_size, 2, 64, 64)	Produces 2-channel image output

Lambda

Layer Type	Input Dimensions	Output Dimensions	Transformation Details
Lambda	(batch_size, 2, 64, 64)	(batch_size, 1, 64, 64)	Maps generator output to discriminator input

Too complex?
Too simple?

Discriminator

Layer Type	Input Dimensions	Output Dimensions	Activation/Other Details
Conv2d (1)	(batch_size, 1, 64, 64)	(batch_size, 64, 32, 32)	LeakyReLU (0.2), Dropout (0.3), Kernel=4
Conv2d (2)	(batch_size, 64, 32, 32)	(batch_size, 128, 16, 16)	BatchNorm, LeakyReLU (0.2), Dropout (0.3)
Flatten	(batch_size, 128, 16, 16)	(batch_size, $128 \times 16 \times 16$)	Flattens for FC layers
Fully Connected (1)	(batch_size, $128 \times 16 \times 16$)	(batch_size, 64)	LeakyReLU (0.2), Dropout (0.3)
Fully Connected (2)	(batch_size, 64)	(batch_size, 1)	Outputs real/fake score

IMPLEMENTATION DETAILS

- Loss Functions:

MSE Loss Measure the mean squared error between the target and output.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\mathrm{output}_i - \mathrm{target}_i)^2$$

Unitarity Loss Enforce unitarity by comparing the modulus squared of the integral of the scattering amplitudes over angular variables to the imaginary part.

$$\mathcal{L}_\mathrm{u} = \frac{1}{N \cdot N_s \cdot N_z}\sum_{i=1}^{N}\sum_{j=1}^{N_s}\sum_{k=1}^{N_z}\left(\left|\mathrm{Im}\,\mathcal{A}(s,z) - \mathrm{Re}\,\mathcal{I}(s,z)\right| + \left|\mathrm{Im}\,\mathcal{I}(s,z)\right|\right)$$

with $\mathcal{I}(s,z) = \frac{1}{4\pi}\int_{-1}^{1} dz' \int_{0}^{2\pi} d\phi \left(\mathcal{A}(s,z')\mathcal{A}(s,z''(z,z',\phi))\right)$

Integral approximator: Simpson's rule

Integral sampling points: $[\cos\theta \times \phi] = 64 \times 10$

IMPLEMENTATION DETAILS

Loss Functions:

MSE Loss Measure the mean squared error between the target and output.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\mathrm{output}_i - \mathrm{target}_i)^2$$

Unitarity Loss Enforce unitarity by comparing the modulus squared of the integral of the scattering amplitudes over angular variables to the imaginary part.

$$\mathcal{L}_\mathrm{u} = \frac{1}{N\cdot N_s \cdot N_z}\sum_{i=1}^{N}\sum_{j=1}^{N_s}\sum_{k=1}^{N_z}\Big(\big|\mathrm{Im}\,\mathcal{A}(s,z) - \mathrm{Re}\,\mathcal{I}(s,z)\big| + \big|\mathrm{Im}\,\mathcal{I}(s,z)\big|\Big)$$

with $\mathcal{I}(s,z) = \frac{1}{4\pi}\int_{-1}^{1} dz' \int_0^{2\pi} d\phi\,\big(\mathcal{A}(s,z')\mathcal{A}(s,z''(z,z',\phi))\big)$

Integral approximator: Simpson's rule

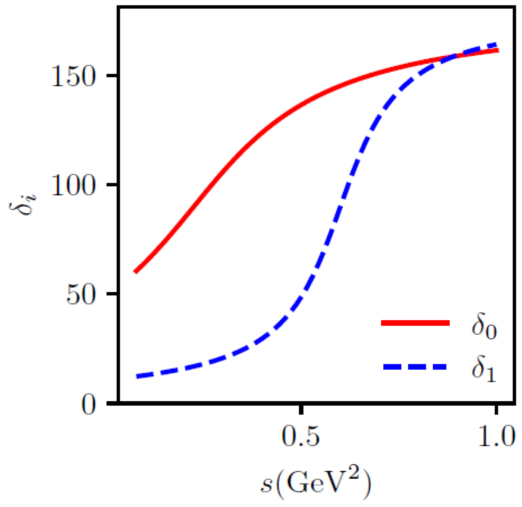
Integral sampling points: $[\cos\theta \times \phi] = 64 \times 10$

d0 Loss Ensure the positive derivative of the f_0 phase shift.

d1 Loss Ensure the positive derivative of the f_1 phase shift.

$$\delta_\ell = \mathrm{atan}\left(\frac{\mathrm{Im}\,f_\ell(s)}{\mathrm{Re}\,f_\ell(s)}\right)$$

$$\mathcal{L}_{D_\ell} = \frac{1}{N}\sum_{i=1}^{N}\log\big(\max\big(0, -\Delta\delta_\ell(s)\big)+1\big), \qquad f_\ell(s) = \frac{1}{2}\int_{-1}^{+1} dz P_\ell(z)\mathcal{A}(s,z)$$



IMPLEMENTATION DETAILS

Loss Functions:

MSE Loss Measure the mean squared error between the target and output.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\mathrm{output}_i - \mathrm{target}_i)^2$$

Unitarity Loss Enforce unitarity by comparing the modulus squared of the integral of the scattering amplitudes over angular variables to the imaginary part.

$$\mathcal{L}_{\mathrm{u}} = \frac{1}{N\cdot N_s\cdot N_z}\sum_{i=1}^{N}\sum_{j=1}^{N_s}\sum_{k=1}^{N_z}\left(\left|\mathrm{Im}\,\mathcal{A}(s,z) - \mathrm{Re}\,\mathcal{I}(s,z)\right| + \left|\mathrm{Im}\,\mathcal{I}(s,z)\right|\right)$$

with $\mathcal{I}(s,z) = \frac{1}{4\pi}\int_{-1}^{1}dz'\int_{0}^{2\pi}d\phi\left(\mathcal{A}(s,z')\mathcal{A}(s,z''(z,z',\phi)\right)\right)$

Integral approximator: Simpson's rule

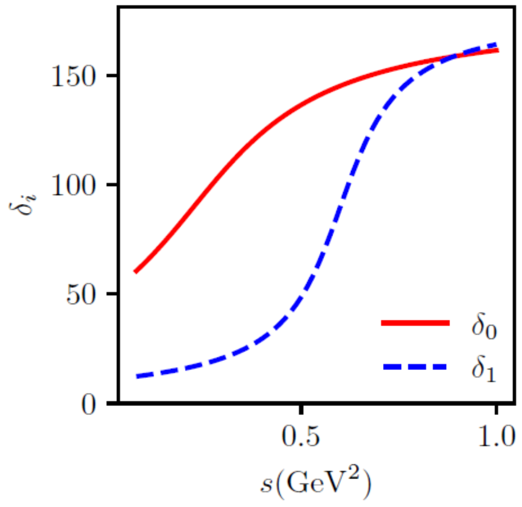
Integral sampling points: $[\cos\theta \times \phi] = 64 \times 10$

Better way to constrain the phase?

d0 Loss Ensure the positive derivative of the f_0 phase shift.

d1 Loss Ensure the positive derivative of the f_1 phase shift.

$$\delta_\ell = \mathrm{atan}\left(\frac{\mathrm{Im}\,f_\ell(s)}{\mathrm{Re}\,f_\ell(s)}\right)$$



$$\mathcal{L}_{D_\ell} = \frac{1}{N}\sum_{i=1}^{N}\log\left(\max\left(0, -\Delta\delta_\ell(s)\right) + 1\right), \qquad f_\ell(s) = \frac{1}{2}\int_{-1}^{+1}dz P_\ell(z)\mathcal{A}(s,z)$$

IMPLEMENTATION DETAILS

- Other hyperparameters:

Generator Optimizer	Adam Learning rate: 0.0001
Discriminator Optimizer	Adam Learning rate: 0.00001
Batch Size	256
Training Size	40×256
Input Noise Dimension	100
Epochs	Total: 200 (with stopping if convergence achieved)
Weights for Generator Losses	[MSE, unitarity, d0, d1] = [1,1,10,10]
Device	GPU

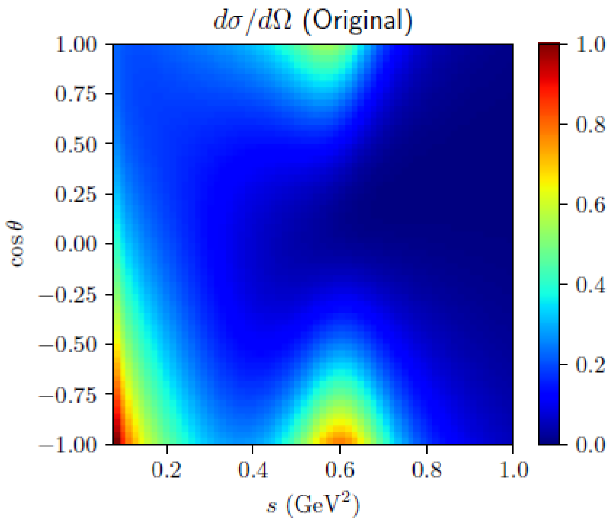
IMPLEMENTATION DETAILS

- Other hyperparameters:

Generator Optimizer	Adam Learning rate: 0.0001
Discriminator Optimizer	Adam Learning rate: 0.00001
Batch Size	256
Training Size	40×256
Input Noise Dimension	100
Epochs	Total: 200 (with stopping if convergence achieved)
Weights for Generator Losses	[MSE, unitarity, d0, d1] = [1,1,10,10]
Device	GPU

How to optimize?

TRAINING DATASET

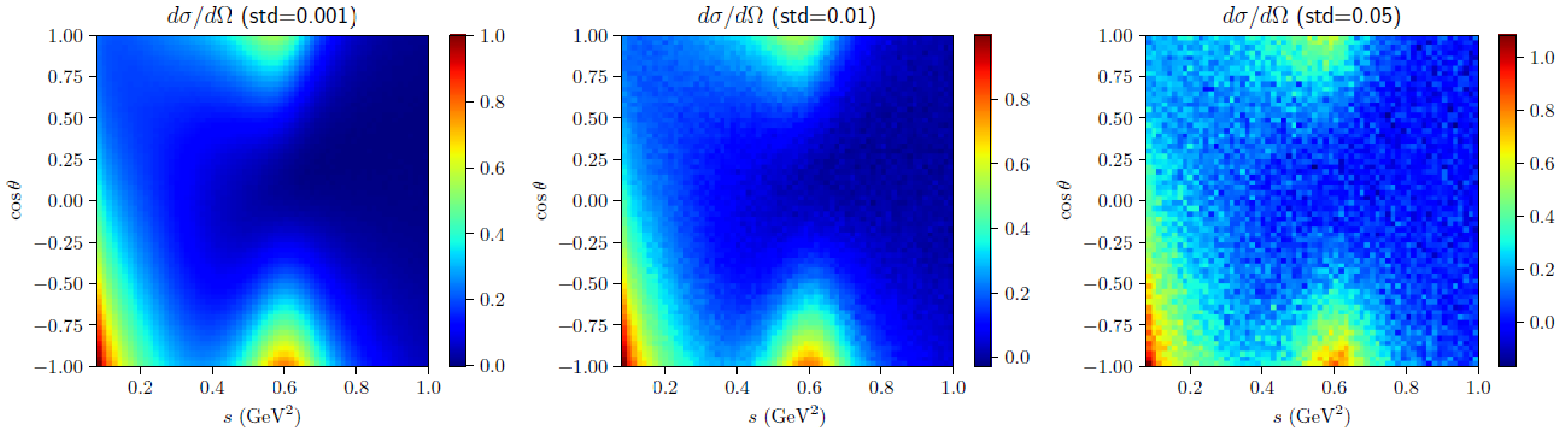


Normalized differential cross section discretized in grid:

$$64 \times 64,\ \ s \in [(2m_\pi)^2, 1\ \text{GeV}^2],\ \ \cos\theta \in [-1, 1]$$

Training samples with additional gaussian noise:

$$40 \times \text{BATCH\_SIZE} = 40 \times 256\ \ \text{samples}$$



PRELIMINARY RESULTS

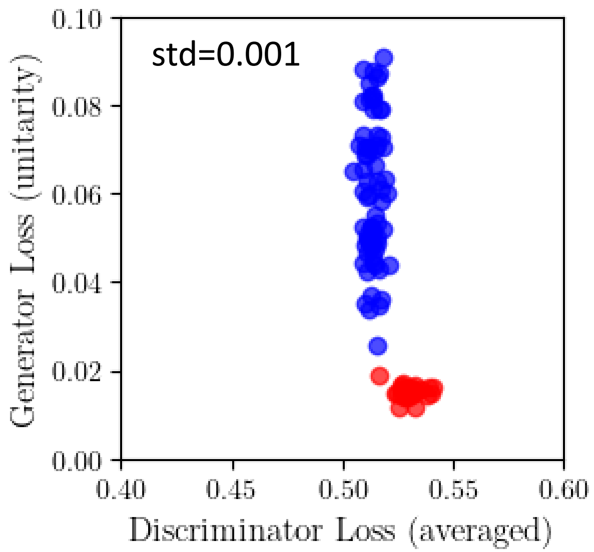
- Trained 100 GANs for 200 epochs:

Stop training if unitarity loss is smaller than 0.02 and changes less than 0.01 and for 10 consecutive epochs:

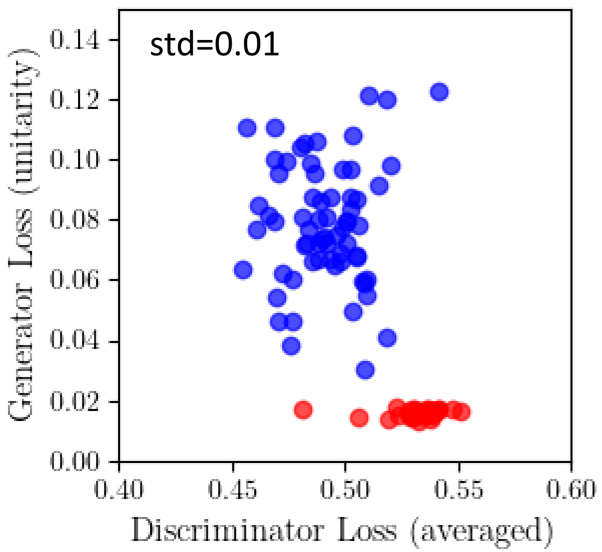
$$\mathcal{L}_{\mathrm{u}} < 0.02$$

$$\mathcal{L}_{\mathrm{u},n} - \mathcal{L}_{\mathrm{u},n-1} < 0.01$$

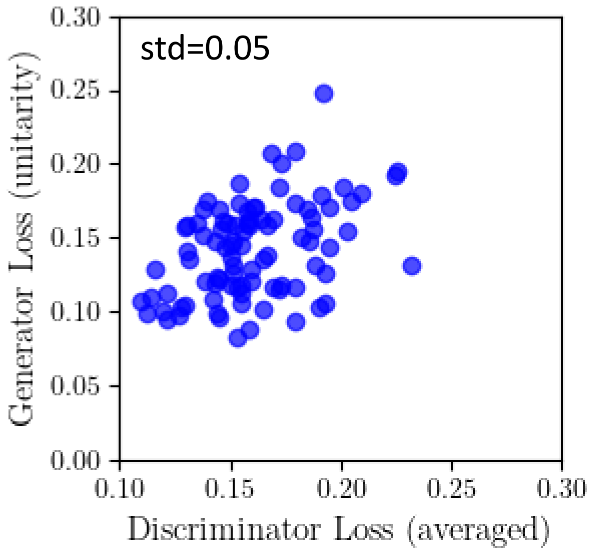
29/100 converged



33/100 converged

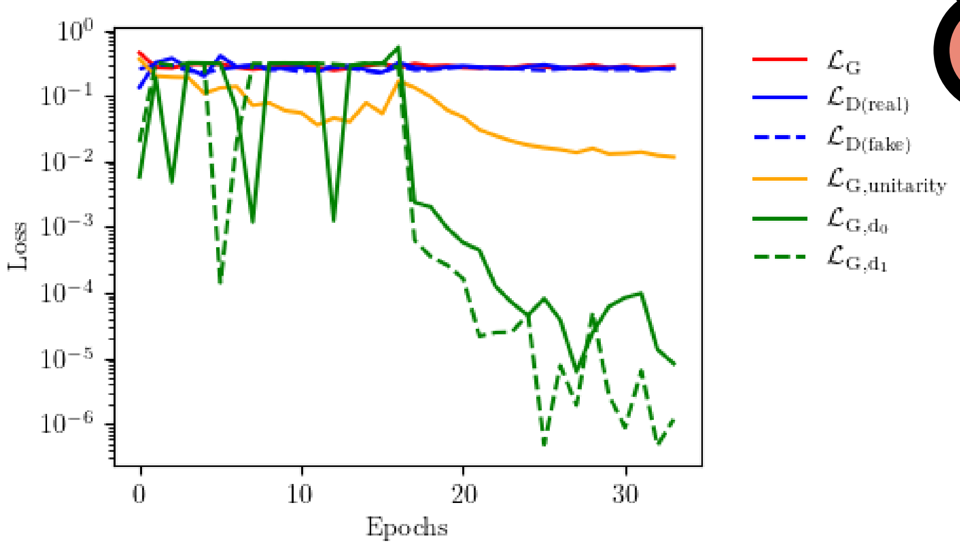


0/100 converged

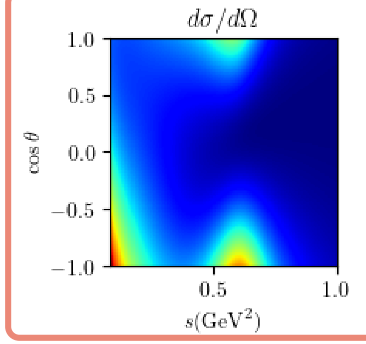


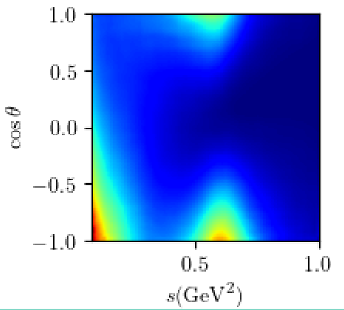
PRELIMINARY RESULTS

- Example of converged GAN with std=0.01:



model ("true" without noise)

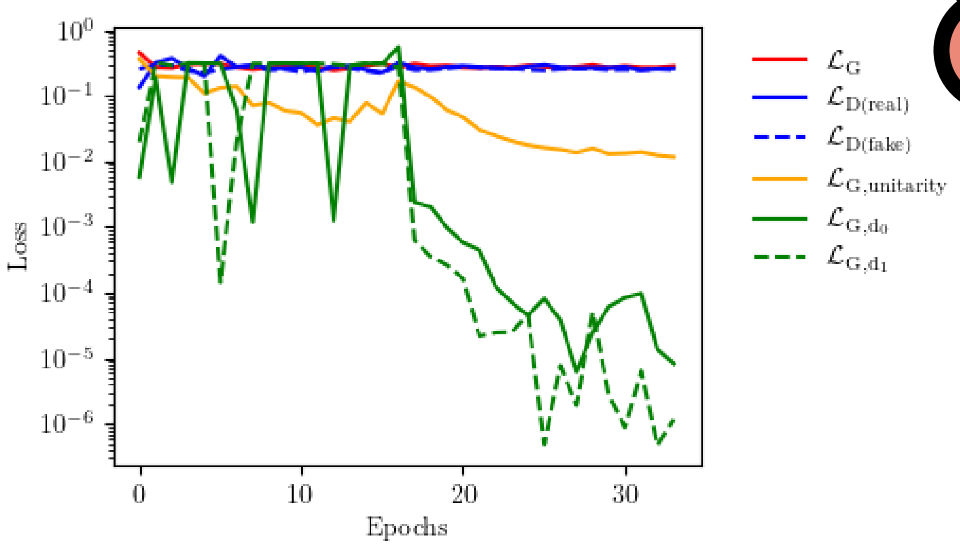




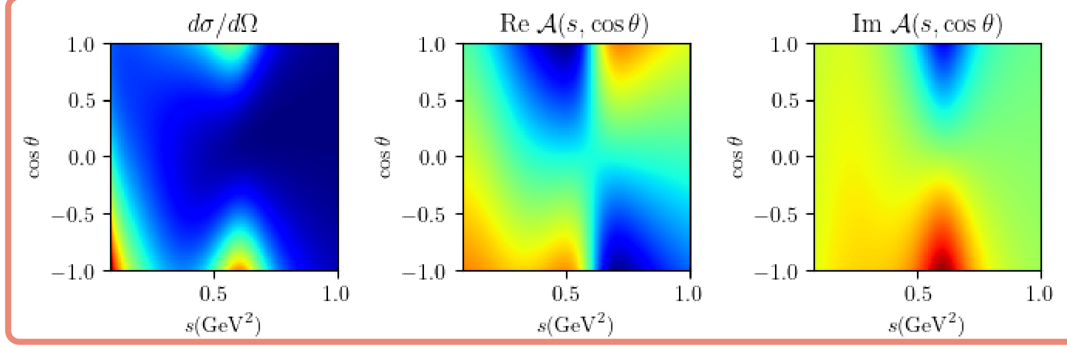
generated ("fake")

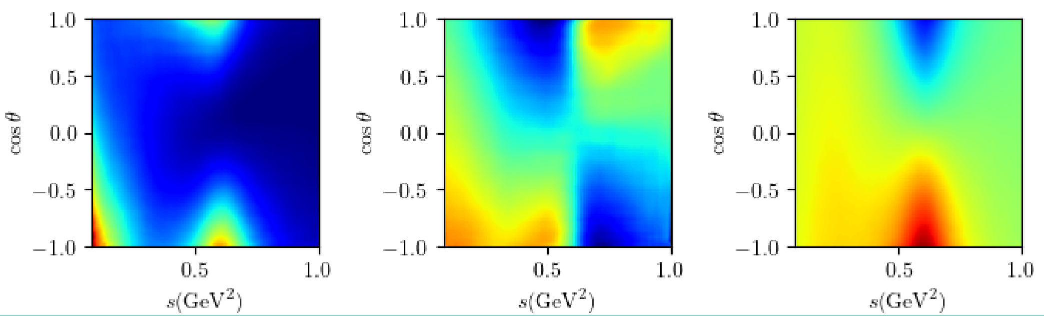
PRELIMINARY RESULTS

- Example of converged GAN with std=0.01:



model ("true" without noise)

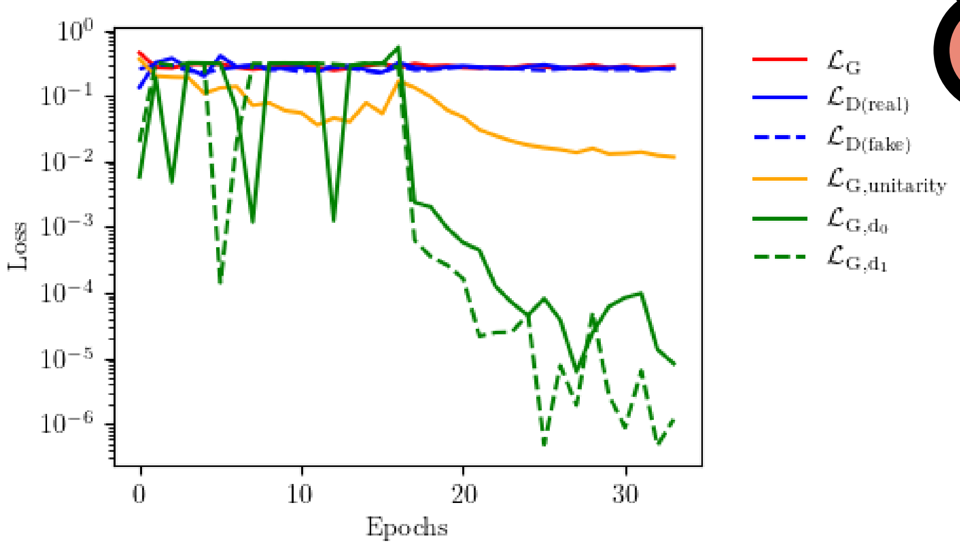




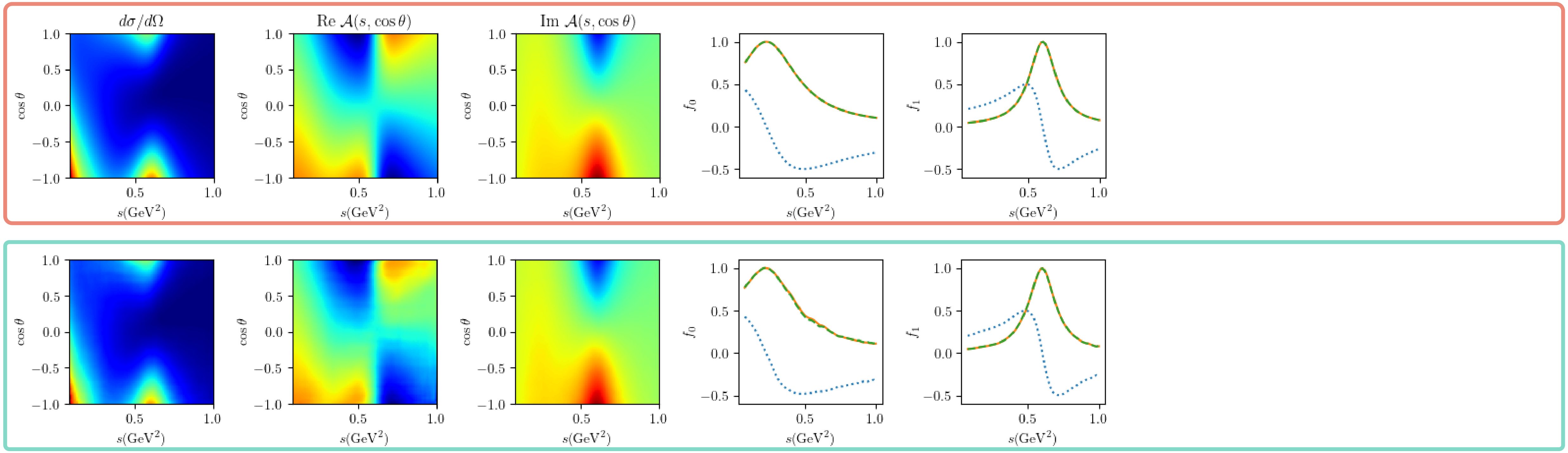
generated ("fake")

PRELIMINARY RESULTS

- Example of converged GAN with std=0.01:

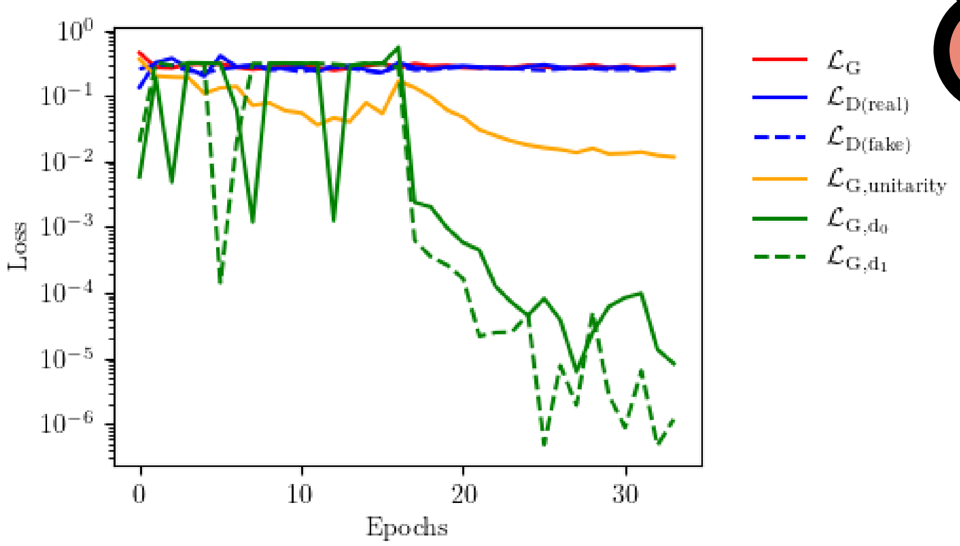


model ("true" without noise)



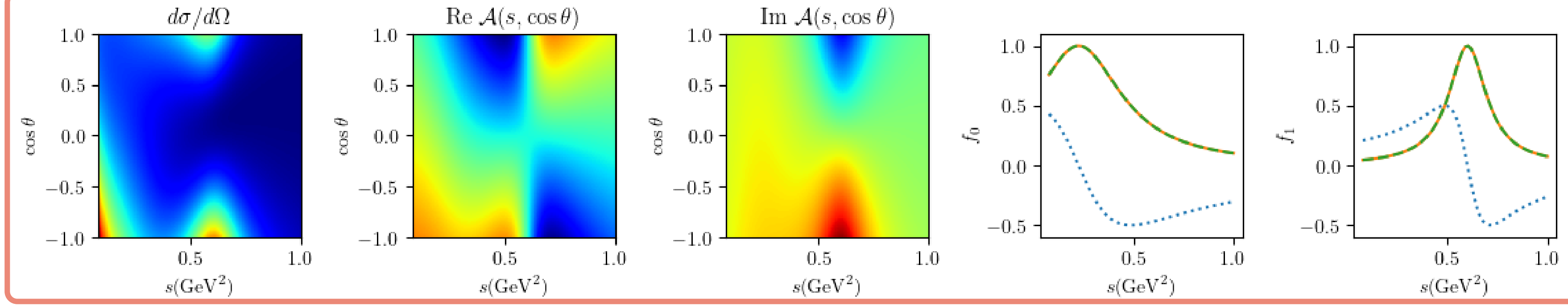
generated ("fake")

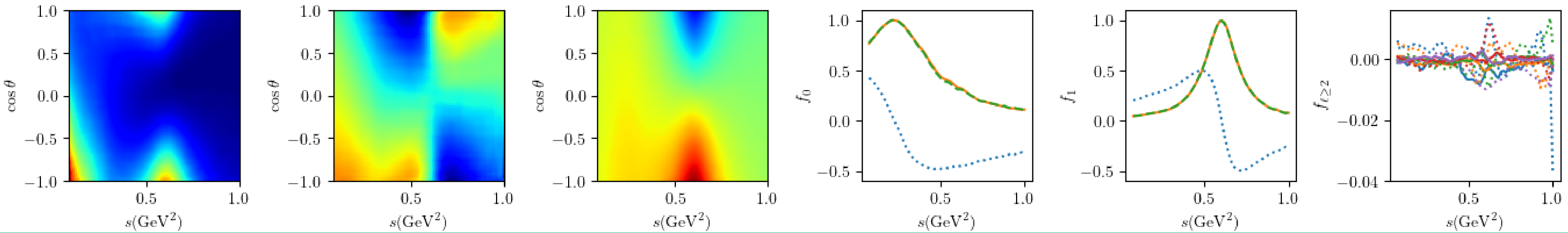
PRELIMINARY RESULTS



- Example of converged GAN with std=0.01:

model ("true" without noise)

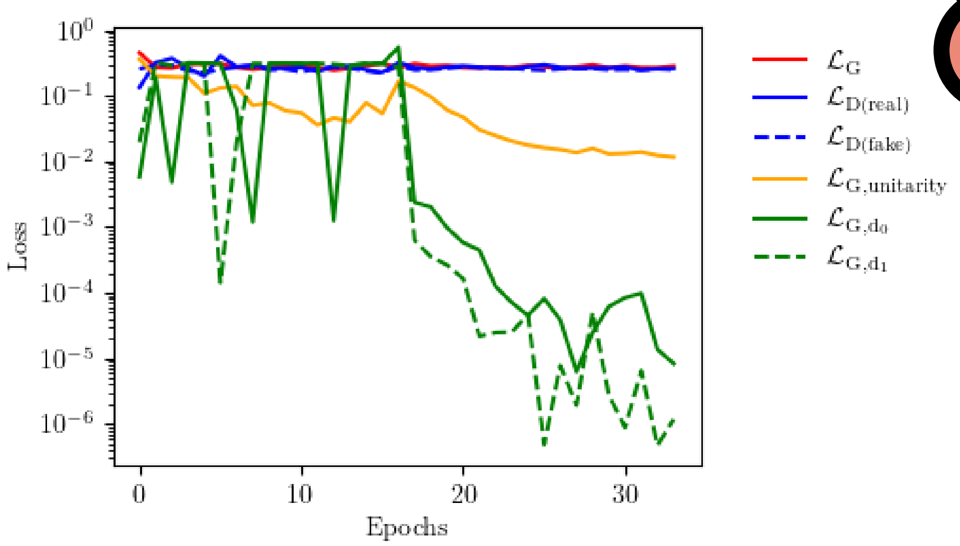




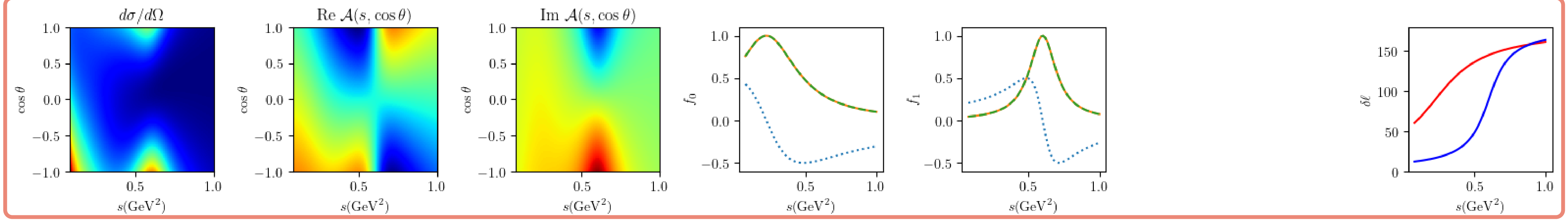
generated ("fake")

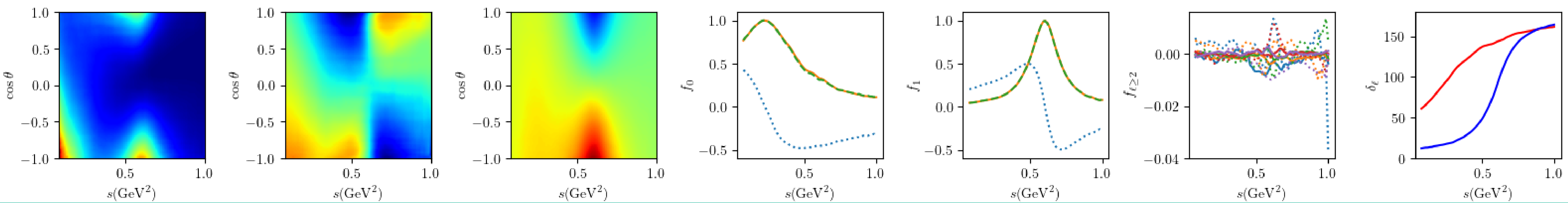
PRELIMINARY RESULTS

- Example of converged GAN with std=0.01:



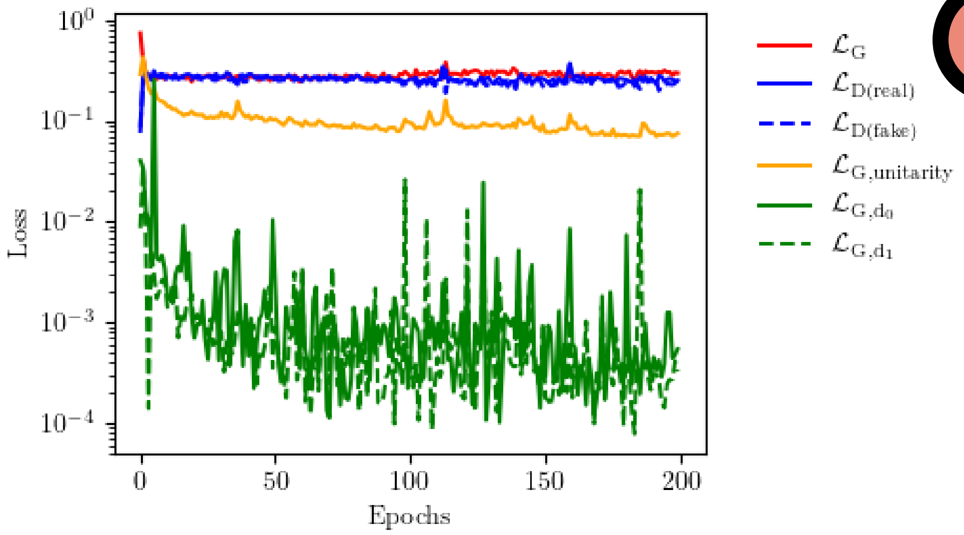
model ("true" without noise)





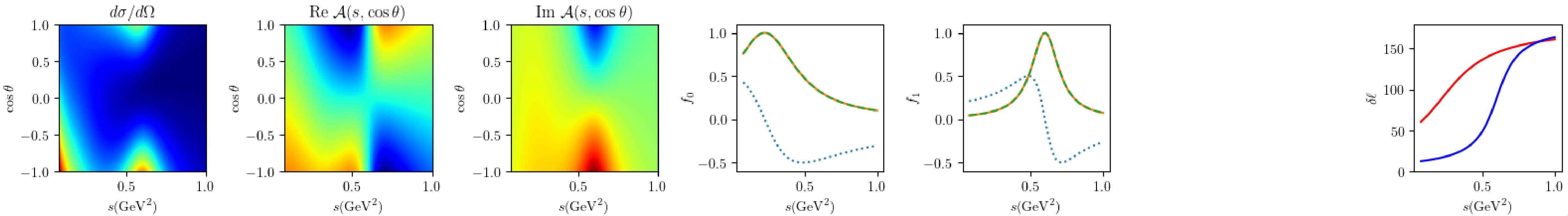
generated ("fake")

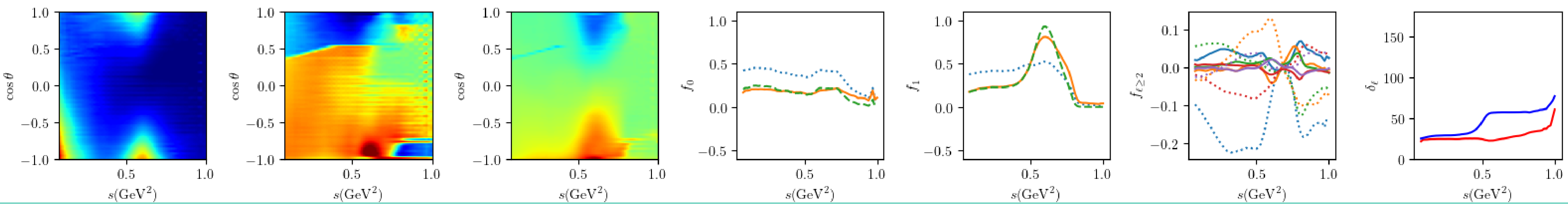
PRELIMINARY RESULTS



Example of non-converged GAN with std=0.01:

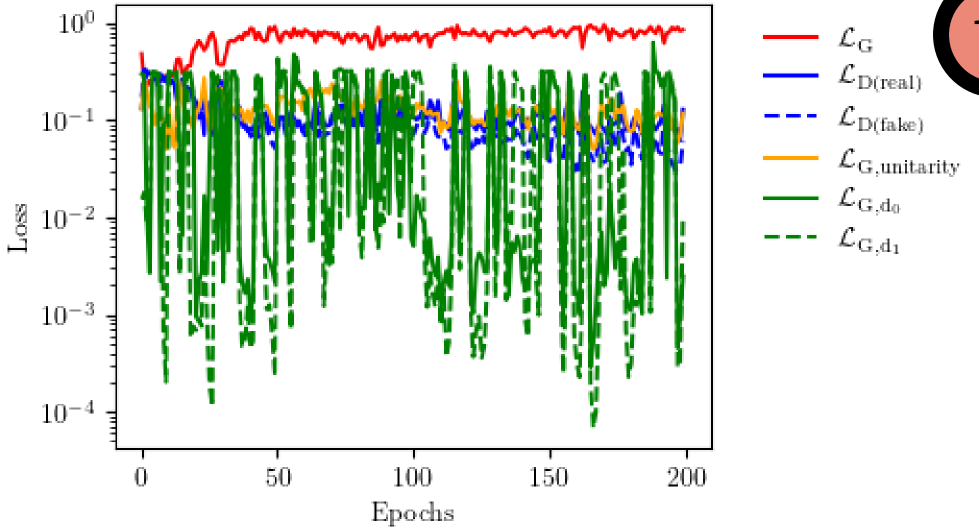
model ("true" without noise)





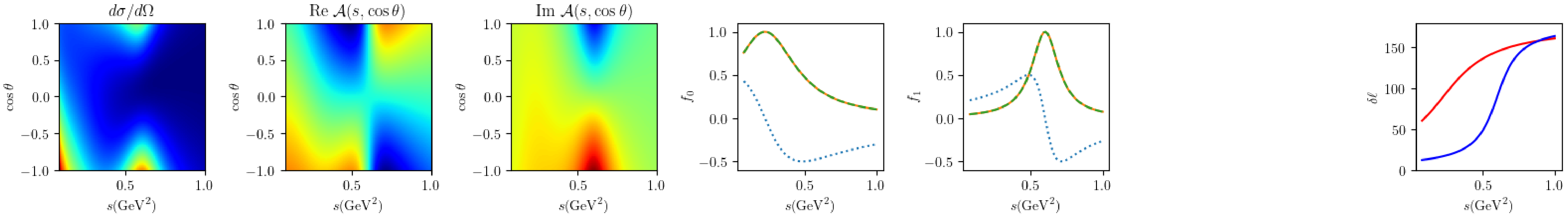
generated ("fake")

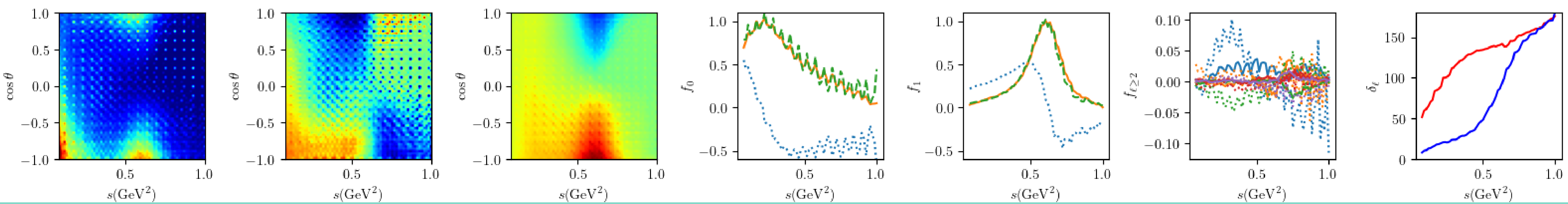
PRELIMINARY RESULTS



- Example of a “not too bad” non-converged GAN with std=0.05:

model (“true” without noise)





generated (“fake”)

CONCLUSIONS & OUTLOOK

Current achievements:

We developed a physics-constrained GAN to extract complex amplitudes from cross-section data.

The unitarity loss together with constraints on the phase allows us to recover the amplitude.

What's next?

Optimize the GAN architecture and the hyperparameters.

Explore additional/alternative physics-informed constraints to further stabilize the GAN training.

Perform a quantitative analysis and error estimation.

Future directions:

Extension to the event level using, e.g. normalizing flows.

Extension to more complicated processes and generalization of the physics constraints.

Preliminary status, but the results of using physics-constrained GANs to extract amplitudes from cross sections employing physics constraints are promising.