MadNIS

Neural networks for multi-channel integration in MadGraph

MCnet Meeting - Graz 2022

Ramon Winterhalder — UC Louvain
Why talk about machine learning?

because

- rich toolbox of algorithms to develop expressive and flexible models for science
- fast development of new methods and algorithms in the past years
- promising applications in both theory and experiment
- large interest in HEP community: IML, ML4Jets, MCnet, workshops,..
Focus today

Event Generation
Theory predictions in HEP

Quantum numbers:
spin, colour charge etc.

Kinematics:
Momenta in Minkowski space, masses, etc.

Cross section:
more generally, differential observables*

PDFs:
convolution over all possible initial state configurations

Phase-space integral:
over final state kinematics

Squared amplitude:
summed over final states, averaged over initial states

\[ \mathcal{A}_{\lambda,c,...}(p_a, p_b|p_1, \ldots, p_n) : M \rightarrow \mathbb{C} \]

\[ \sigma = \frac{1}{\text{flux}} \sum_{a,b} \int dx_a dx_b f(x_a) f(x_b) \int d\Phi_n \langle |\mathcal{A}(p_a, p_b|p_1, \ldots, p_n)|^2 \rangle \]
Theory predictions in HEP

\[ \mathcal{A}_{\lambda,c,...}(p_a, p_b|p_1, \ldots, p_n) : \mathbb{M} \rightarrow \mathbb{C} \]

Quantum numbers: spin, colour charge etc.

Kinematics: Momenta in Minkowski space, masses, etc.

Cross section: more generally, differential observables*

PDFs: convolution over all possible initial state configurations

Phase-space integral: over final state kinematics

Squared amplitude: summed over final states, averaged over initial states
Monte Carlo Integration

Standard Monte Carlo integration

$$I = \int_V d^d x \, f(x) \approx \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \langle f \rangle_x$$

$$\sigma_I \approx \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N - 1}}$$
Monte Carlo Integration

Standard Monte Carlo integration

\[ I = \int_V d^d x f(x) \approx \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \langle f \rangle_x \]

\[ \sigma_I \approx \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N - 1}} \]

Mapping

\[ y = G(x) \quad g(x) = \left| \frac{\partial G(x)}{\partial x} \right| \]

\[ G \quad G^{-1} \]

\[ \mathbb{R}^d \quad V \quad d^d x \quad \mathbb{R}^d \quad U \quad d^d y = \left| \frac{\partial G(x)}{\partial x} \right| d^d x \]
Monte Carlo Integration

**Standard Monte Carlo integration**

\[
I = \int_V d^d x f(x) \simeq \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \langle f \rangle_x \quad \sigma_I \simeq \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N - 1}}
\]

**Importance Sampling**

\[
I = \int_U d^d y \frac{f(x)}{g(x)} \bigg|_{x=G^{-1}(y)} \simeq \langle f/g \rangle_y \quad \sigma_I \simeq \sqrt{\frac{\langle (f/g)^2 \rangle_y - \langle f/g \rangle_y^2}{N - 1}}
\]

**Mapping**

\[
y = G(x) \quad g(x) = \left| \frac{\partial G(x)}{\partial x} \right|
\]
Monte Carlo Integration

Standard Monte Carlo integration

\[ I = \int_V d^d x f(x) \approx \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \langle f \rangle_x \quad \sigma_I \approx \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N - 1}} \]

Importance Sampling

\[ I = \int_U d^d y \left. \frac{f(x)}{g(x)} \right|_{x=G^{-1}(y)} \approx \langle f/g \rangle_y \quad \sigma_I \approx \sqrt{\frac{\langle (f/g)^2 \rangle_y - \langle f/g \rangle_y^2}{N - 1}} \]
Monte Carlo Integration

Standard Monte Carlo integration

\[ I = \int_V d^d x f(x) \simeq \frac{1}{N} \sum_{j=1}^{N} f(x_j) = \langle f \rangle_x \quad \sigma_I \simeq \sqrt{\frac{\langle f^2 \rangle_x - \langle f \rangle_x^2}{N - 1}} \]

Importance Sampling

\[ I = \int_U d^d y \left. \frac{f(x)}{g(x)} \right|_{x=G^{-1}(y)} \simeq \langle f/g \rangle_y \quad \sigma_I \simeq \sqrt{\frac{\langle (f/g)^2 \rangle_y - \langle f/g \rangle_y^2}{N - 1}} \]

Wanted

\[ \sigma_I \to 0 \]

Needs

\[ f/g \approx \text{const} \]
Importance sampling - VEGAS

Why not VEGAS for everything?

- High-dim and rich peaking
  → slow convergence
- If peaks are not aligned with grid axes → “phantom peaks”
Importance sampling - **VEGAS**

Why not **VEGAS** for everything?
- High-dim and rich peaking
  → slow convergence
- If peaks are not aligned with grid axes → “phantom peaks”

**Quality score?**
- Unweighting efficiency

$$
\epsilon = \frac{\langle w \rangle}{w_{\text{max}}}, \quad w = \frac{f}{g}
$$
Importance sampling - NN

Using a NN

- Unbinned and no grids
  → no “phantom peaks”
- Bijectivity not guaranteed
  → training unstable
- Numerical Jacobians
  → slow training and evaluation

[1707.00028, 1810.11509, 2009.07819]
Importance sampling - NN

Using a NN
- Unbinned and no grids
  → no “phantom peaks”
- Bijectivity not guaranteed
  → training unstable
- Numerical Jacobians
  → slow training and evaluation

Using a Flow instead
- Invertibility
  → bijective mapping
- Tractable Jacobians
  → fast training and evaluation

Normalizing Flow

$$\log p_y(y) = \log p_x(x) + \log \left| \frac{\partial G(x)}{\partial x} \right|$$
Multi-Channel Monte Carlo

Standard Multi-Channel Integration

\[ I = \int_V d^d x f(x) = \sum_i \int_V d^d x \alpha_i g_i(x) \frac{f(x)}{g(x)} = \sum_i \int_{U_i} d^d y_i \alpha_i \left. \frac{f(x)}{g(x)} \right|_{x \equiv x(y_i)} \]

\[ g(x) = \sum_i \alpha_i g_i(x) \text{, with } \sum_i \alpha_i = 1 \]

Channel Mappings

\[ y_i = G_i(x) \quad g_i(x) = \left| \frac{\partial G_i(x)}{\partial x} \right| \]
Multi-Channel Monte Carlo

**Standard Multi-Channel Integration**

\[
I = \int_V d^d x f(x) = \sum_i \int_V d^d x \alpha_i g_i(x) \frac{f(x)}{g(x)} = \sum_i \int_{U_i} d^d x \alpha_i \frac{f(x)}{g(x)} \bigg|_{x \equiv x(y_i)}
\]

\[
g(x) = \sum_i \alpha_i g_i(x) \text{, with } \sum_i \alpha_i = 1
\]

**Channel Mappings**

\[
y_i = G_i(x) \quad g_i(x) = \left| \frac{\partial G_i(x)}{\partial x} \right|
\]

**MadGraph Multi-Channel Integration**

only use \[
\sum_i \beta_i(x) = 1
\]
Standard Multi-Channel Integration

\[ I = \int_V d^d x f(x) = \sum_i \int_V d^d x \alpha_i g_i(x) \frac{f(x)}{g(x)} = \sum_i \int_{U_i} d^d y_i \alpha_i \frac{f(x)}{g(x)} \bigg|_{x \equiv x(y_i)} \]

\[ g(x) = \sum_i \alpha_i g_i(x) \text{, with } \sum_i \alpha_i = 1 \]

Channel Mappings

\[ y_i = G_i(x) \quad g_i(x) = \left| \frac{\partial G_i(x)}{\partial x} \right| \]

MadGraph Multi-Channel Integration

\[ I = \int_V d^d x f(x) = \sum_i \int_V d^d x \beta_i(x) f(x) = \sum_i \int_{U_i} d^d y_i \beta_i(x) \frac{f(x)}{g_i(x)} \bigg|_{x \equiv x(y_i)} \]

only use \[ \sum_i \beta_i(x) = 1 \]
Multi-Channel Monte Carlo

Standard Multi-Channel Integration

\[ I = \int_V d^d x f(x) = \sum_i \int_V d^d x \alpha_i g_i(x) \frac{f(x)}{g(x)} = \sum_i \int_{U_i} d^d y_i \alpha_i \frac{f(x)}{g(x)} \bigg|_{x \equiv x(y_i)} \]

\[ g(x) = \sum_i \alpha_i g_i(x), \text{ with } \sum_i \alpha_i = 1 \]

MadGraph Multi-Channel Integration

\[ I = \int_V d^d x f(x) = \sum_i \int_V d^d x \beta_i(x) f(x) = \sum_i \int_{U_i} d^d y_i \beta_i(x) \frac{f(x)}{g_i(x)} \bigg|_{x \equiv x(y_i)} \]

\[ \text{only use } \sum_i \beta_i(x) = 1 \]

Channel Mappings

\[ y_i = G_i(x) \quad g_i(x) = \frac{\partial G_i(x)}{\partial x} \]

Connection

\[ \beta_i(x) = \alpha_i \frac{g_i(x)}{g(x)} \]
Multi-Channel Importance Sampling

$$I = \sum_i \int_{\Omega} \alpha_i(x) \frac{f(x)}{g_i(x)} \, dG_i(x)$$
Neural Multi-Channel Monte Carlo

Multi-Channel Importance Sampling

\[ I = \sum_i \int_\Omega \alpha_i(x) \frac{f(x)}{g_i(x)} dx \]

Neural Channel Mappings

\[ G_i(x) \rightarrow G_i^\theta(x), \quad g_i^\theta(x) = \left| \frac{\partial G_i^\theta(x)}{\partial x} \right| \]

n-dimensional remapping

- tractable Jacobian with normalizing flow
Neural Multi-Channel Monte Carlo

Multi-Channel Importance Sampling

\[ I = \sum_i \int_{\Omega} \alpha_i(x) \frac{f(x)}{g_i(x)} dG_i(x) \]

Neural Channel Mappings

\[ G_i(x) \rightarrow G_i^\theta(x), \quad g_i^\theta(x) = \left| \frac{\partial G_i^\theta(x)}{\partial x} \right| \]

n-dimensional remapping

- tractable Jacobian with normalizing flow

Multi-Channel Importance Sampling

\[ I = \sum_i \int_\Omega \alpha_i(x) f(x) \frac{g_i(x)}{g_i(x)} d[G_i(x)] \]

Possible Improvements?

Neural Channel Mappings

\[ G_i(x) \rightarrow G_i^\theta(x), \quad g_i^\theta(x) = \left| \frac{\partial G_i^\theta(x)}{\partial x} \right| \]

n-dimensional remapping
• tractable Jacobian with normalizing flow

Neural Multi-Channel Monte Carlo

Multi-Channel Importance Sampling

\[ I = \sum_i \int_\Omega \frac{f(x)}{g_i(x)} dG_i(x) \]

Neural Channel Weights

\[ \alpha_i(x) \rightarrow \tilde{\alpha}_i^\theta(x), \quad \sum_i \tilde{\alpha}_i^\theta(x) = 1 \]

\( k \)-dimensional regression
- with boundary condition

Neural Channel Mappings

\[ G_i(x) \rightarrow G_i^\theta(x), \quad g_i^\theta(x) = \left| \frac{\partial G_i^\theta(x)}{\partial x} \right| \]

\( n \)-dimensional remapping
- tractable Jacobian with normalizing flow

Neural Multi-Channel Monte Carlo

Multi-Channel Importance Sampling

\[ I = \sum_i \int_\Omega \alpha_i(x) \frac{f(x)}{g_i(x | i)} \, dG_i(x | i) \]

Neural Channel Weights

\[ \alpha_i(x) \rightarrow \alpha_i^\phi(x), \quad \sum_i \alpha_i^\phi(x) = 1 \]

- \text{k-dimensional regression}
  - with boundary condition

Conditional Neural Channel Mappings

\[ G_i^\theta(x) \rightarrow G_i^\theta(x | i), \quad g_i^\theta(x) \rightarrow g_i^\theta(x | i) = \frac{\partial G_i^\theta(x | i)}{\partial x} \]

- \text{n-dimensional remapping}
  - tractable Jacobian with normalizing flow
  - conditioned on channel

→ \text{So far: improvement factors of } \sim 2-4 \text{ are achieved} [2001.05486, 2001.05478, 2001.10028, 2112.09145]
Neural Multi-Channel Monte Carlo

Multi-Channel Importance Sampling

Further Improvements?

- Two-stage training
- Variance-weighted training
- Overflow channels
- Bayesian neural networks

Conditional Neural Channel Mappings

$\alpha_i(x) \rightarrow \mathcal{G}_i^0(x)$, \quad $g_i^0(x) \rightarrow g^0(x)$

Further Improvements?

- Two-stage training
- Variance-weighted training
- Overflow channels
- Bayesian neural networks

k-dimensional regression

- with boundary condition

More expressive transformations: FFJORD, CubicSplines, LASeR, …

So far: improvement factors of ~2-4 are achieved
Neural Multi-Channel Monte Carlo

**Neural Channel Weights**

\[ l = \sum_{i} \int_{\Omega} a_i(x) \frac{f(x)}{y_i(x)} dG(x) \]

- k-dimensional regression
  - with boundary condition

**MadNIS**

\[ \alpha_i(x) \rightarrow \alpha_i^\theta(x) \]
\[ \sum_i \alpha_i^\theta(x) = 1 \]

- Neural Channel Splittings

**Neural Channel Splittings**

- n-dimensional remapping
  - tractable Jacobian
  - normalizing flow
  - conditioned on channel

**Two-Stage Training**

- With boundary condition

**So far:** Improvement factors of ~2-4 are achieved

MadNIS

Neural Channel Weights
Neural Channel Weights

Channel network

\[ \alpha_i(x) \rightarrow \alpha_i^\phi(x), \quad \sum_i \alpha_i^\phi(x) = 1 \]
Neural Channel Weights

Channel network

\[ \alpha_i(x) \rightarrow \alpha_i^\phi(x), \quad \sum_i \alpha_i^\phi(x) = 1 \]

Normalization

\[ \alpha_i^\phi(x) \rightarrow \hat{\alpha}_i^\phi(x) = \frac{\exp \alpha_i^\phi(x)}{\sum_i \exp \alpha_i^\phi(x)} \]
Neural Channel Weights

Channel network

\[ \alpha_i(x) \rightarrow \alpha_i^\phi(x), \quad \sum_i \alpha_i^\phi(x) = 1 \]

Normalization

\[ \alpha_i^\phi(x) \rightarrow \hat{\alpha}_i^\phi(x) = \frac{\exp \alpha_i^\phi(x)}{\sum_i \exp \alpha_i^\phi(x)} \]

MadGraph Channels

\[ \beta_i(x) = \frac{|M_i(x)|^2}{\sum_j |M_j(x)|^2} \]
Neural Channel Weights

Channel network

\[ \alpha_i(x) \rightarrow \alpha_i^\phi(x), \quad \sum_i \alpha_i^\phi(x) = 1 \]

Normalization

\[ \alpha_i^\phi(x) \rightarrow \hat{\alpha}_i^\phi(x) = \frac{\exp \alpha_i^\phi(x)}{\sum_i \exp \alpha_i^\phi(x)} \]

MadGraph Channels

\[ \beta_i(x) = \frac{|M_i(x)|^2}{\sum_j |M_j(x)|^2} \]

Residual channel network

\[ \alpha_i^\phi(x) = \log \beta_i(x) + \varphi_i \cdot \Delta_i^\phi(x) \]
Neural Channel Weights

Channel network

\[ \alpha_i(x) \rightarrow \alpha_i^{\phi}(x), \quad \sum_i \alpha_i^{\phi}(x) = 1 \]

\[ \alpha_i^{\phi}(x) \rightarrow \hat{\alpha}_i^{\phi}(x) = \frac{\exp \alpha_i^{\phi}(x)}{\sum_i \exp \alpha_i^{\phi}(x)} \]

Residual channel network

\[ \alpha_i^{\phi}(x) = \log \beta_i(x) + \varphi_i \cdot \Delta_i^{\phi}(x) \]

\[ \alpha_i^{\phi}(x) \rightarrow \hat{\alpha}_i^{\phi}(x) = \frac{\beta_i(x) \exp \varphi_i \cdot \Delta_i^{\phi}(x)}{\sum_j \beta_j(x) \exp \varphi_j \cdot \Delta_j^{\phi}(x)} \]

Normalization

\[ \beta_i(x) = \frac{|M_i(x)|^2}{\sum_j |M_j(x)|^2} \]
Neural Channel Weights

With prior weight

Pre-training

No prior weight

Pre-training

Probability density

Channel weights

Camel$(x)$ $g_1(x)$ $g_2(x)$ $\alpha_1(x)$ $\alpha_2(x)$ $\alpha_{opt,1}(x)$ $\alpha_{opt,2}(x)$ $\alpha_1(x)$ $\alpha_2(x)$ $\alpha_{opt,1}(x)$ $\alpha_{opt,2}(x)$
Neural Channel Weights

With prior weight  
After training

No prior weight  
After training
Neural Channel Weights

With prior weight
Pre-training

Camel(x)
$g_1(x)$
$g_2(x)$

With cut

Channel weights

With cut

No prior weight
Pre-training

Camel(x)
$g_1(x)$
$g_2(x)$

Channel weights
Neural Channel Weights

With prior weight
After training

With cut

Camel(x)
$g_1(x)$
$g_2(x)$

With cut

No prior weight
After training

Channel weights

Probability density
MadNIS

Two-Stage Training
Generative training
Generative training
Generative training

Sample $y$ \xrightarrow{G_\theta^{-1}(y)}$ PS points $x$
Integrand $f(x)$

Density $g_\theta(x)$

Loss $L(f(x), g_\theta(x))$

Sample $y$

PS points $x$

$G_\theta^{-1}(y)$

Generative training
Sample $y$ → $G_\theta^{-1}(y)$ → PS points $x$ → Integrand $f(x)$ → Backpropagation → Loss $L(f(x), g_\theta(x))$ → $G_\theta(x)$ → Density $g_\theta(x)$

Generative training

Integrand $f(x)$

Density $g_\theta(x)$
Generative training

Sample $y \xrightarrow{G_\theta^{-1}(y)}$ PS points $x \xrightarrow{G_\theta(x)}$ Integrand $f(x) \xrightarrow{g_\theta(x)}$ Loss $L(f(x), g_\theta(x))$

Backpropagation

Density $g_\theta(x)$

Saved samples $x, g_\theta(x), f(x)$
Integrand $f(x)$

Density $g_\theta(x)$

Loss $L(f(x), g_\theta(x))$

Backpropagation

Sample training

Saved samples $x, g_\theta(x), f(x)$

Generative training

Sample $y$

$G_\theta^{-1}(y)$
Integrand $f(x)$
Density $g_{\theta}(x)$
Loss $L(f(x), g_{\theta}(x))$

Backpropagation

Generative training
- Sample $y$
  $G_{\theta}^{-1}(y)$
- PS points $x$
- $G_{\theta}(x)$
- $G_{\theta}(x)$
- Density $g_{\theta}(x)$

Sample training
- Saved samples $x, g_{\theta}(x), f(x)$
- $G_{\theta}(x)$
- Density $\hat{g}_{\theta}(x)$
Sample training

\[ G_\theta^{-1}(y) \]

PS points

\[ x \]

Integrand

\[ f(x) \]

Backpropagation

Loss

\[ L(f(x), g_\theta(x)) \]

Density

\[ g_\theta(x) \]

Saved samples

\[ x, g_\theta(x), f(x) \]

Backpropagation

Weighted Loss

\[ L(f(x), \hat{g}_\theta(x); w(x)) \]

Density

\[ \hat{g}_\theta(x) \]
Integrand $f(x)$

Density $g_\theta(x)$

Loss $L(f(x), g_\theta(x))$

Backpropagation

Weighted Loss $L(f(x), \hat{g}_\theta(x); w(x))$

$w(x) = \frac{\hat{g}_\theta(x)}{g_\theta(x)}$
Two-Stage Training

![Graphs showing the relative increase and decrease in weight updates and training time with different reduction factors.](image)

- Relative increase in weight updates: $\frac{\Delta \theta_{\text{ref}}}{\theta_{\text{ref}}} = 40\%$
- Relative decrease in training time: $\frac{\Delta T}{T} = 10\%$

Parameters:
- $t_g = 40\mu s$
- $t_s = 30\mu s$
- $t_f = 1\mu s$
- $t_f = 10\mu s$
- $t_f = 100\mu s$
- $t_f = 1\text{ ms}$
Summary

• ML can reduce the error in Monte Carlo Integration

• New developments can improve performance and reduce computational overhead
Summary

- ML can reduce the error in Monte Carlo Integration
- New developments can improve performance and reduce computational overhead
- Overview of ML in HEP: https://iml-wg.github.io/HEPML-LivingReview/

HEPML-LivingReview

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

The purpose of this note is to collect references for modern machine learning as applied to particle physics. A minimal number of categories is chosen in order to be as useful as possible. Note that papers may be referenced in more than one category. The fact that a paper is listed in this document does not endorse or validate its content – that is for the community (and for peer-review) to decide. Furthermore, the classification here is a best attempt and may have flaws - please let us know if (a) we have missed a paper you think should be included, (b) a paper has been misclassified, or (c) a citation for a paper is not correct or if the journal information is now available. In order to be as useful as possible, this document will continue to evolve so please check back before you write your next paper. If you find this review helpful, please consider citing it using \cite{hepmlivingreview} in HEPML.bib.
Summary and Outlook

Summary

- ML can reduce the error in Monte Carlo Integration
- New developments can improve performance and reduce computational overhead
- Overview of ML in HEP: https://iml-wg.github.io/HEPML-LivingReview/

Outlook

- Test performance of flow on real LHC examples: (eg. multi-leg, NLO, complicated cuts, …)
- Possibly combine with other methods (eg. surrogate unweighting,…) 
- Make matrix elements run on the GPU and differentiable

HEPML-LivingReview

A Living Review of Machine Learning for Particle Physics

Modern machine learning techniques, including deep learning, is rapidly being applied, adapted, and developed for high energy physics. The goal of this document is to provide a nearly comprehensive list of citations for those developing and applying these approaches to experimental, phenomenological, or theoretical analyses. As a living document, it will be updated as often as possible to incorporate the latest developments. A list of proper (unchanging) reviews can be found within. Papers are grouped into a small set of topics to be as useful as possible. Suggestions are most welcome.

The purpose of this note is to collect references for modern machine learning as applied to particle physics. A minimal number of categories is chosen in order to be as useful as possible. Note that papers may be referenced in more than one category. The fact that a paper is listed in this document does not endorse or validate its content – that is for the community (and for peer-review) to decide. Furthermore, the classification here is a best attempt and may have flaws - please let us know if (a) we have missed a paper you think should be included, (b) a paper has been misclassified, or (c) a citation for a paper is not correct or if the journal information is now available. In order to be as useful as possible, this document will continue to evolve so please check back before you write your next paper. If you find this review helpful, please consider citing it using `cite(hepmlivingreview)` in HEPML.bib.