Published for SISSA by Springer

Received: *June 29, 2021*
Accepted: *July 22, 2021*
Published: *August 16, 2021*

Optimising simulations for diphoton production at hadron colliders using amplitude neural networks

Joseph Aylett-Bullock,[a,b] **Simon Badger**[c] **and Ryan Moodie**[a]

[a] *Institute for Particle Physics Phenomenology, Department of Physics, Durham University, Durham, DH1 3LE, United Kingdom*
[b] *Institute for Data Science, Durham University, Durham, DH1 3LE, United Kingdom*
[c] *Dipartimento di Fisica and Arnold-Regge Center, Università di Torino, and INFN, Sezione di Torino, Via P. Giuria 1, I-10125 Torino, Italy*

E-mail: j.p.bullock@durham.ac.uk, simondavid.badger@unito.it, ryan.i.moodie@durham.ac.uk

Abstract: Machine learning technology has the potential to dramatically optimise event generation and simulations. We continue to investigate the use of neural networks to approximate matrix elements for high-multiplicity scattering processes. We focus on the case of loop-induced diphoton production through gluon fusion, and develop a realistic simulation method that can be applied to hadron collider observables. Neural networks are trained using the one-loop amplitudes implemented in the NJet C++ library, and interfaced to the Sherpa Monte Carlo event generator, where we perform a detailed study for $2 \to 3$ and $2 \to 4$ scattering problems. We also consider how the trained networks perform when varying the kinematic cuts effecting the phase space and the reliability of the neural network simulations.

Keywords: NLO Computations, QCD Phenomenology

ArXiv ePrint: 2106.09474

https://doi.org/10.1007/JHEP08(2021)066

Contents

1 Introduction

Phenomenological studies of high multiplicity final states at collider experiments present a substantial theoretical challenge and are increasingly important ingredients in experimental measurements. During the last 15 years, a dramatic improvement in computational algorithms for one-loop amplitudes has led to a number of highly automated codes capable of predictions at next-to-leading order (NLO) accuracy in the Standard Model (SM) [1–5].

These codes are based around numerical algorithms which bypass the growth in algebraic complexity that analytic approaches suffer from. The computational cost of these algorithms is however relatively high, resulting in huge commitments of CPU and personnel resources to obtain the necessary theoretical predictions for current experiments.

The interface of general one-loop amplitude codes into multi-purpose Monte Carlo (MC) event generators has resulted in a wide variety of simulation options which can offer the best possible theoretical accuracy. Methods that go beyond fixed-order perturbation theory — such as parton shower matching, merging, and jet multiplicities — improve accuracy across important regions of phase space. However, these simulations add additional strain on the underlying amplitudes.

State-of-the-art tools make use of advanced phase-space mapping algorithms to improve the convergence of the multi-dimensional integration. General purpose MC event generators such as `Sherpa` [6, 7], `Pythia` [8, 9], `Herwig` [10–12], and `MadGraph` [4] often make use of the diagram structure of the underlying tree-level process to ensure an optimal distribution of points. Reusing tree-level distributions when generating virtual events is particularly effective at reducing the computational cost of using complicated one-loop amplitudes.

In this paper, we will consider a class of scattering processes that contribute to diphoton signals at hadron colliders. The process $gg \to \gamma\gamma + n(g)$ is a good test case for machine learning (ML) technology, since it is loop induced and has relevant contribution from high multiplicity matrix elements. Using automated tools at NLO, full QCD corrections are known for $pp \to \gamma\gamma + \leq 3$ jets [13–15]. There has been a flurry of recent activity around next-to-next-to-leading-order (NNLO) corrections to $pp \to \gamma\gamma + j$ in which the complete leading colour corrections have been presented [16–20]. As such, $2 \to 3$ and $2 \to 4$ scattering for the gluon-initiated diphoton channel are now extremely relevant for future phenomenological studies.

ML is now extremely popular in high energy physics with a wealth of applications. The majority of problems concern classification and exploit the scalability of ML algorithms to large datasets. The fact that neural networks (NNs) offer a general function parameterisation that are also useful in regression problems is not new in particle physics either: for example, parton distribution functions (PDFs) produced by the NNPDF collaboration [21] have been used for many years by the LHC experiments.

Interpolation methods such as polynomial fits and interpolation grids have often been used to provide fast simulations and predictions of differential observables, particularly at NNLO where there is an intricate structure of infrared (IR) singularities [22–26]. These techniques are extremely powerful for problems with two or three hard scattering variables, but scale poorly at high multiplicity.

ML offers a possible solution to the poor scaling with larger number of variables. Various directions are currently being explored for different aspects of event generation and MC integration. Boosted decision trees (BDTs) and NNs have been applied to phase-space sampling and integration [27–29] and have shown to increase the speed at which functions can be integrated, as well as allow for the integration of functions for which traditional algorithms failed. More recently, sampling using normalising flows [30, 31] has been attempted [32–34]. These have been shown to avoid the computational cost of calculating the gradient of the network itself, when determining the Jacobian, which previous NN approaches required.

There has also been a large focus on using ML for other components of MC event generator simulations. Specifically, Generative Adversarial Networks (GANs) [35] are being applied to event generation [36–47], event unweighting [48, 49] and subtraction [50], with recent works incorporating Bayesian methods for uncertainty estimation into these generative methods [51]. NN-based approaches (some of which also use GAN technology) applied to parton showering [52–55] and event reweighting [56] have also been developed.

Several works have focused on developing NN techniques for explicitly learning the cross section of a given process [57, 58]; however, little research has been done on learning the matrix element itself for a given phase-space point and process. Ref. [59] took a parallelised BDT approach to learning the matrix element, and tested this on the loop-induced $gg \to ZZ$ process at leading order, demonstrating the potential for large speedups in matrix element calculations. An ensembled NN methodology was presented in ref. [60] which divides the phase space into divergent and non-divergent regions, and further splits the former into sub-regions corresponding to different IR singular structures. This approach was tested on $e^+e^- \to \leq 5$ jets at both tree and one-loop level, to ensure robustness at high multiplicity, and was found to provide a good approximation of the cross section and various differential distributions.

Readers may like to refer to the living review of ML in particle physics for up-to-date information about the state-of-the-art [61].

Our paper is organised as follows. We begin by giving a brief review of the structure of the underlying perturbative amplitudes that we study. We then describe the computational setup in which we use the `NJet` amplitude library [62] to train an ensemble of NNs that are then interfaced to the `Sherpa` MC event generator [6, 7]. We discuss a strategy for the generation of the training dataset, and a reweighting strategy to refine the distributions of events where the networks are used for matrix element generation. We then discuss results for $gg \to \gamma\gamma g$ and $gg \to \gamma\gamma gg$ at leading order (LO). In each case, we present a selection of differential observables and provide a detailed comparison with a conventional simulation. We end with a presentation of our conclusions and a discussion of possible future uses of these technologies.

Our code is publicly available at https://github.com/JosephPB/n3jet_diphoton.

2 Gluon-initiated diphoton amplitudes

We study amplitudes with two photons and many gluons which first appear at one-loop level in the SM. With conventional simulations relying on cheaper LO tree evaluations to optimise event generation for NLO one-loop contributions, these loop-induced processes present an interesting sector to test new approaches for phase-space integration. Compact analytic computations for $gg \to \gamma\gamma$ and $gg \to \gamma\gamma g$ have been available for some time and offer extremely fast and stable evaluation. As a result, it is feasible to optimise event generation with the one-loop evaluation. For $2 \to 4$ scattering problems, only numerical codes are available and simulations can be extremely slow. It also not clear that analytic formula would be sufficiently compact to alleviate this situation even if they were available.

For this reason, we consider an alternative setup where the whole simulation uses a NN approximation of the amplitude.

The loop-level amplitudes proceed through a fermion loop and have a colour decomposition in the trace basis as

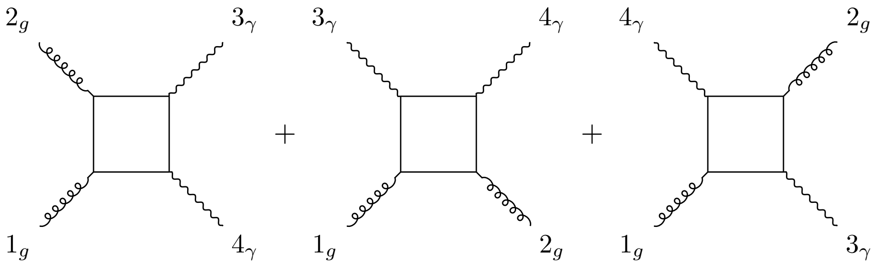
$$\mathcal{A}^{(1)}(1,\dots,n-2,(n-1)_\gamma,n_\gamma) = \\ {g_s}^{n-2} g_{\gamma\gamma} \sum_{\sigma\in S_{n-3}} \lambda(\sigma\,(a_1,\dots,a_{n-2}))A^{(1)}(\sigma(1,\dots,n-2),(n-1)_\gamma,n_\gamma), \quad (2.1)$$

where g_s is the strong coupling, $g_{\gamma\gamma} = e^2 \sum_q Q_q^2$ is the combined coupling of the diphoton system to the fermion loop, S_{n-3} is the set of even non-cyclic permutations of $\{1,\dots,n-2\}$, t^a are the fundamental SU(3) generators, q runs over active quark flavours with fractional quark charge Q_q, and the colour trace function λ is defined as

$$\lambda\,(a_1,\dots,a_{n-2}) = \mathrm{tr}\,(t^{a_1}t^{a_2}\dots t^{a_{n-2}}) + (-1)^n \mathrm{tr}\,(t^{a_1}t^{a_{n-2}}\dots t^{a_2})\,. \quad (2.2)$$

This yields $(n-3)!/2$ primitive amplitudes A for $n \geq 5$. For example, for $n=4$ there is a single primitive amplitude. It is given by the diagrams

$$A(1,2,3_\gamma,4_\gamma) =$$



(2.3)

where a plain line indicates a sum over quark loop arrow directions. At one-loop, these amplitudes are also related to the fermion loop corrections to pure gluon scattering through permutations [63]. The ingredients for differential cross sections are the squared amplitudes summed over helicities, h, and colour,

$$|\mathcal{A}^{(1)}|^2 = \left(\frac{\alpha_s}{4\pi}\right)^{n-2} g_{\gamma\gamma}^2 \sum_{h,i,j} A_i^{(1)*}(h)\,\mathcal{C}_{ij}\,A_j^{(1)}(h) + \mathcal{O}(\alpha_s^{n-1}) \quad (2.4)$$

where the matrix $\mathcal{C}$ is a function of the number of colours, N_c, obtained by squaring the colour basis elements and the index on the partial amplitudes, A, refers to the different permutations in the colour decomposition.

The amplitudes in this article are taken from the `NJet C++` library [62]. Here, there are different options: a general numerical setup using generalised unitarity and integrand reduction; and hard-coded analytic expressions for $n=4,5$. The $n=4$ analytic expressions were taken from ref. [64], while for $n=5$ they were obtained directly from a finite field reconstruction [65] and are in agreement with known analytic formula [63, 66]. By using a momentum twistor parameterisation of the external kinematics, cancellations in the rational coefficients of the special functions that lead to a manifestly finite representation are easily identified.

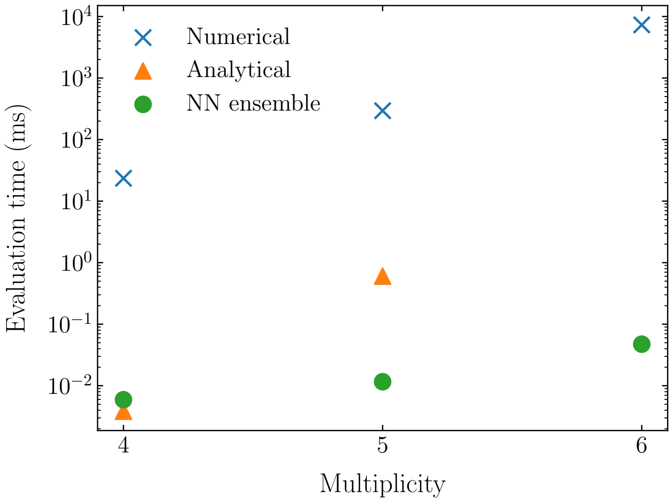


Figure 1. Matrix element typical CPU evaluation times for available methods — including `NJet` numerical evaluations, `NJet` analytical evaluations, and inference on a NN ensemble as described in section 3 — against the number of legs. These calls are single-threaded as parallelisation is applied at the level of events in simulations. An analytic expression for 2 → 4 is not available. The NN is comparable to the analytic call at 2 → 2, 50 times faster at 2 → 3, then 10^5 times faster than the 2 → 4 numeric call.

The numerical evaluation requires the sum of permutations of ordered primitive amplitudes. This is completely automated for arbitrary multiplicity, but evaluation times and numerical stability are increasingly difficult to control.

To study the growth of evaluation time with multiplicity, we evaluate the matrix element at 100 random phase-space points with each available technique and plot the mean times in figure 1. We generate the phase-space points isotropically with the algorithm from ref. [67]. While analytic methods are competitive at low multiplicity, we see they scale poorly and are unlikely to beat numerics at $n \geq 6$. Numeric scaling is better, but these algorithms come with a high cost. Our NN approach provides a performant alternative, with significantly better scaling than either numerics or analytics.

3 Computational setup

In this paper, we build on previous work which sought to demonstrate the viability of using NN-based approaches to approximate matrix element values for hard scattering processes [60]. In that work, a NN ensemble approach was presented in which a different NN is trained on each soft and collinear region of phase space, and was shown to be effective in handling IR divergent structures at both the Born and one-loop level at high multiplicity in e^+e^- collisions. We extend this to more complex 2 → 3 and 2 → 4 gluon-initiated diphoton amplitudes, while also showing the ability for these ML models to interface with existing event generators such as `Sherpa` [6, 7]. This is important to demonstrate since it is not immediately obvious that NN approximations trained in isolation will be robust to the added intricacies of event generators which are important for extracting physical results, such as PDF weighting and choices of integrators.

We begin this section with an overview of the setup for training our ML approach based on those presented in ref. [60] and finish with a discussion on interfacing with event generators.

3.1 Phase-space partitioning

IR divergences arise from soft and collinear real emissions and integrals over massless partons appearing in virtual corrections. Our ML approach requires the isolation of real emission IR singularities such that a different NN can be trained on each soft and collinear region [60]. To do this, we partition the phase space into divergent and non-divergent regions and then subsequently sub-divide the divergent region according to the FKS subtraction [68, 69]. The original implementation of this method was for $e^+e^- \to q\bar{q} + \text{jets}$ collisions in which the IR singularities only appear in the final state. In the processes discussed in this paper, IR singularities appear in initial-initial, initial-final and final-final state pair combinations. We therefore extend the partitioning to these states for hadronic processes as specified in ref. [69].

As in ref. [60], we parameterise our phase space according to the Lorentz invariant $y_{ij} = s_{ij}/s_{12}$, where $s_{ij} = (p_i + p_j)^2$. The PDF convolution creates a non-fixed partonic centre-of-mass energy $\sqrt{s_{12}}$.

We now define the divergent and non-divergent regions of phase space as

$$\mathcal{R}_{\text{div}} = \{p \,|\, \min(y_{ij}) \leq y_p,\, p = (p_1, p_2, \dots, p_n),\, i,j \in \{1,\dots,n\}\}\,, \tag{3.1}$$

$$\mathcal{R}_{\text{non-div}} = \{p \,|\, y_p \leq \min(y_{ij}),\, p = (p_1, p_2, \dots, p_n),\, i,j \in \{1,\dots,n\}\}\,, \tag{3.2}$$

where y_p is fixed for the process (see appendix B), p is a phase-space point consisting of the incoming momenta $\{p_1, p_2\}$, and the outgoing momenta $\{p_3, \dots, p_n\}$, where n is the number of particles in the process. The FKS pairs are then defined as

$$\begin{aligned}\mathcal{P}_{\text{FKS}} = \{(i,j) \mid 1 \leq i \leq n,\ 2 \leq j \leq n,\ i \neq j,\\ \mathcal{M}^{(n,0)} \text{ or } \mathcal{M}^{(n,1)} \to \infty \text{ if } p_i^0 \to 0 \text{ or } p_j^0 \to 0 \text{ or } \vec{p}_i \parallel \vec{p}_j\}.\end{aligned} \tag{3.3}$$

Finally, partition functions are used as in ref. [60]

$$\mathcal{S}_{i,j} = \frac{1}{D_1 s_{ij}}, \quad D_1 = \sum_{i,j \in \mathcal{P}_{\text{FKS}}} \frac{1}{s_{ij}}, \tag{3.4}$$

such that

$$\mathrm{d}\sigma = \sum_{i,j} \mathcal{S}_{i,j}\, \mathrm{d}\sigma, \tag{3.5}$$

where σ represents the cross section for the process. These partition functions are then used to weight the corresponding matrix elements of points lying in $\mathcal{R}_{\text{div}}$, and we train one network in this region for each $\mathcal{S}_{i,j}$ (see section 3.2 and ref. [60]).

The above definitions are appropriate for the processes we studied here as they account for the different singularity structure than that found in the case of e^+e^- (i.e. they explicitly include the initial-state singularities). To allow for easier generalisability to other processes, we include all pairs of initial- and final-state particles in our implementation, including the $\{\gamma\gamma\}$ pair which is redundant as it does not exhibit the relevant singularity structure. This redundancy does increase the computational time required; however, we find the performance of the NN ensemble is not adversely affected. The above implementation could therefore be simply generalised to the e^+e^- case, although again with some redundancy.

3.2 Neural network setup

Now that we have described how we partition the phase space for data generation, we will discuss the NN setup, and the data processing required to train and test our methodology, in more detail. While the focus of this paper is the use of the NN ensemble method originally presented in ref. [60], for completeness we present a comparison of this method against a naive (single network) approach in appendix C.

3.2.1 Data

The sampling of phase space is dependent on the integrator. Unless otherwise specified, the same integrator is used for training, validation and testing. We generate the datasets from two runs of the integrator: the first is divided into training and validation datasets according to a 80:20 split; the second uses a different random seed than the first and is used for the test dataset (note that this second stage is only performed when evaluating the performance of our ML approach). Phase-space points, and their corresponding matrix elements generated by NJet, are extracted during these stages after the cuts have been applied. For consistency with ref. [60], we train on 100k points and test on 3M. Input data consists of the initial-state 4-momenta, $x \in \mathbb{R}^{4n}$, and the target data is the corresponding (weighted) matrix element, $y \in \mathbb{R}$.

Once the training data has been generated, we split the data into divergent and non-divergent regions according to eq. (3.1) and eq. (3.2), and create copies of the points in $\mathcal{R}_{\mathrm{div}}$, each weighted by $\mathcal{S}_{i,j}$ for $(i,j) \in \mathcal{P}_{\mathrm{FKS}}$, which are used to train a different network in the NN ensemble for each soft and collinear divergent structure.

Before training the NN ensemble, we preprocess the data by standardising each variable input, i.e. we ensure that the training and validation input distributions for each variable have a mean of zero and a standard deviation of one. The output variable distribution is similarly standardised. During the testing phase, we insure the testing data is standardised to the training and validation dataset distribution parameters as is customary in ML literature. Once the model has been used for matrix element estimation, the output is destandardised to obtain the final value. Varying types of data processing are used for hyperparameter tuning (see appendix A for more details).

All our simulations use $\sqrt{s_{12}} = 1\,\mathrm{TeV}$; the methodology is agnostic to this choice. Although we test the performance of our models on different cuts, unless otherwise specified all models are trained, and analyses performed, using the following kinematic cuts adapted from those in ref. [14]

$$\begin{array}{lll} p_{T,j} > 20\,\text{GeV} & R_{\gamma,j} > 0.4 & |\eta_j| < 5 \\ p_{T,\gamma_1} > 40\,\text{GeV} & R_{\gamma,\gamma} > 0.4 & |\eta_\gamma| < 2.37 \\ p_{T,\gamma_2} > 30\,\text{GeV} & & \end{array}$$

where $p_T = \sqrt{{p_x}^2 + {p_y}^2}$ (beam along z-axis) is transverse momentum magnitude, $R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ is isolation cut cone radius, η is pseudorapidity, ϕ is azimuthal angle, γ denotes a photon, photons are ordered by p_T, and jets, j, are identified through the anti-k_T algorithm [70] implemented in `FastJet` [71] with $R = 0.4$. These cuts are typical for LHC analyses. Photons are selected by smooth cone isolation [72] such that all cones of radius $r_\gamma < R$ satisfy

$$E_{\text{hadronic}}(r_\gamma) \leq \epsilon\, p_{T,\gamma} \frac{1 - \cos r_\gamma}{1 - \cos R}$$

with $R = 0.4$ and $\epsilon = 0.05$.

Matrix elements are evaluated with renormalisation scale $\mu_R = m_Z$ with physical constant values $\alpha(Q^2 = 0)$, $\alpha_s(m_Z)$, and m_Z from the PDG [73]. Since the one-loop process is LO, the full amplitude is finite and has μ_R dependence in the couplings only.

3.2.2 Architecture

For optimal results, a different NN architecture construction would be fine-tuned to each choice of setup parameters, e.g. integrator, cuts, and process. However, the required computational and time resources required to perform this optimisation make this highly impractical. Instead, we further test the generalisability of the hyperparameter choices made in ref. [60] to this new set of processes. In addition, hyperparameter tuning was performed to assess how optimal this approach is relative to the ideal scenario where hyperparameters are process specific, and found the original setup to be among the most optimal (see appendix A for more details).

In summary, we use the same fully-connected NN architecture for every network in the ensemble. These are parameterised using Keras [74] and a TensorFlow [75] backend, with the number of input nodes equal to $n \times 4$.[1] The hidden layers comprise of 20-40-20 nodes and there is a single output node. All hidden layers use hyperbolic-tangent activation functions and the output node has a linear activation function. A mean squared error loss function is used, and the network is optimised using Adam optimisation [76]. Finally, the number of training epochs is determined through Early Stopping (see section 8.1.2 of ref. [77]), tracking the validation loss with no minimum change requirements. As in ref. [60], to minimise the limiting effects of using a validation set containing only 20% of the original training set, we train with a patience of 100 epochs.

[1] Testing was performed to assess the change in performance when removing redundant, non-independent, 4-momenta components; however, this had little effect.

3.3 Interfacing with event generators

Assessing performance after interfacing with existing event generator technology is important for demonstrating ‘real-world deployment’ of ML algorithms in particle physics simulations, as it exposes the model to a range of post-inference effects which may alter the final reliability of the model. For example, generators allow for the easy implementation of complex phase-space cuts, jet clustering algorithms, phase-space and PDF weights, as well as different integrators and integration optimisation routines.

3.3.1 The interface

Event generators are largely written in `C++` for computational efficiency. Therefore, after the model has been trained, the weights of each NN are extracted and written to file. A `C++` program reads these models files and performs the linear algebra operations required during the inference step using `Eigen` [78]. This means the `Python` libraries used for model training are circumvented, and the call time for model inference is reduced, while keeping everything in `C++` simplifies the interfacing of the model with standard event generators.

Given a set of 4-momenta, a custom `C++` interface provides the helicity- and colour-summed matrix element to `Sherpa`. This can be used to call `NJet` evaluations through a BLHA interface [79, 80] or to call the model inference result. `Rivet` [81, 82] is then used for analysis, using a script adapted from the reference analysis of ref. [83].

3.3.2 Phase-space integration

Phase-space integrators seek to achieve increasingly optimal rates of integration convergence though the careful sampling of points. While the choice of integrator can affect the overall rate of convergence, it also determines the placement of phase-space points which directly feeds into the distribution of points in the training dataset.

Since these processes are loop-induced, for simplicity we use the RAMBO integrator [84] throughout for event generation. However, we test different approaches to generating the integration grid. The first we term the ‘unit grid’ which is constructed by running the grid optimisation step while returning a unit value in place of the matrix elements. This effectively removes the dependence on the optimisation procedure and, since RAMBO is used, ensures a uniformly and isotropically sampled phase space. The second uses VEGAS [85, 86] optimisation when generating the integration grid, thereby putting a preference on sampling regions of particular importance to the cross section. We share the integration grid between training and testing phases, meaning this importance sampling is reflected in both. Given the expense of matrix element calculation for the $2 \to 4$ scattering process, we reserve the use of VEGAS optimisation only for the $2 \to 3$ case.

3.3.3 Weights

When training the models, we do not include explicitly any event generator effects. All additional weightings, i.e. phase-space weights and PDF weights, are introduced after the model has been used for inference as is done for other matrix element generators. The addition of these weightings has the potential to be problematic for model performance:

when the model is trained it is unlikely to learn all regions of phase space equally well and there is a chance that those regions in which the model has poor performance could be amplified by these additional weighing factors.

In order to test for this we include PDF weights using the LHAPDF library [87] and the NNPDF3.1 set `NNPDF31_nlo_as_0118` [88] as well as phase-space weights which depend on the integration grid optimisation method.

3.4 Reweighting

The approach used in this paper to train the NN ensemble provides good agreement between the network output and that of `NJet`. However, the ensemble approach will always be an approximation and is subject to perform poorly in certain regions of phase space, especially those in which it has not been trained on or in which training data is sparse. As a partial remedy to this, we propose the idea of reweighting the event weights with known matrix element values derived from `NJet`. When using weighted event generation this can either be performed after event generation or can be done 'on the fly' at the interface level. This former approach is possible since the original event weight can be recovered through

$$w_{E,i}^{(\texttt{NJet})} = w_{E,i}^{(\mathrm{NN})} \times \frac{\mathcal{M}_i^{(\texttt{NJet})}}{\mathcal{M}_i^{(\mathrm{NN})}}, \tag{3.6}$$

where $w_{E,i}^{\texttt{NJet}}$ and $w_{E,i}^{\mathrm{NN}}$ are the event weights using `NJet` and the NN ensemble respectively for a given phase-space point i, and $\mathcal{M}_i^{\texttt{NJet}}$ and $\mathcal{M}_i^{\mathrm{NN}}$ are the associated matrix elements.

As the ratio `NJet`/NN is not known a priori, we must construct criteria on which to reweight. Specifically, we explore the following:

1. A random sample of points (e.g. 10%) regardless of where they are in phase space;
2. A priori stating which regions of phase space in which to reweight and then doing so either randomly or over the entire region;
3. Using the NN uncertainties to inform reweighting, e.g. points with large uncertainties are reweighted.

There are several factors informing which approach is the most appropriate. The first of these is the added compute time required: all of these techniques necessitate the calculation of the matrix element by an analytic or numerical evaluator and therefore limit the desirable number of points to reweight. The second is the performance gain and confidence in the output in certain regions of phase space. If the analysis being performed is specific to an under-sampled region of phase space, such as distribution tails where the network may under-perform due to divergent structures in the matrix element, this could be an especially important region in which to reweight. However, if general process explorations are being performed, meaning all distributions and cross sections are of relative equal importance, then a less restrictive reweighting on regions of phase space may be optimal.

In this paper, we explore the application of reweighting to the processes described in section 4. While reweighting is not always found to be necessary given the performance of our methodology, we demonstrate how it can be applied and discuss which reweighting criteria show the greatest performance gain.

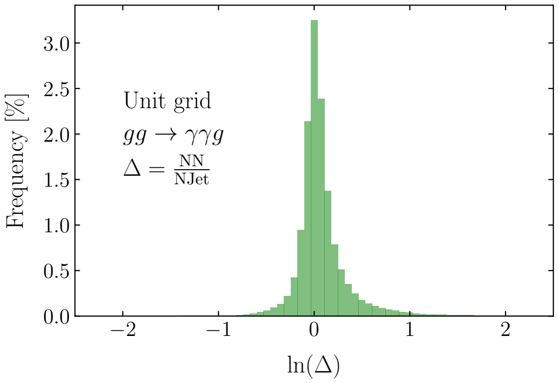


Figure 2. NN/`NJet` errors for the 2 → 3 scattering process using a unit integration grid.

4 Results

In this section, we present the results of our experiments for the $2 \to 3$ and $2 \to 4$ gluon-initiated diphoton amplitudes. As the former is significantly less computationally expensive, we use this for a deep analysis and exploration. The proposed pipeline for using our ML set up and interface with event generators is as follows:

1. Generate an integration grid;
2. Use this with a matrix element provider to generate training and validation datasets;
3. Train the model;
4. Use the model to estimate the values of the remaining phase-space points for event generation while using the same integration grid;
5. Reweight as necessary;
6. Obtain final results.

To assess performance, we also evaluated matrix elements with `NJet` in parallel with the models, with different random seeds.

4.1 $gg \to \gamma\gamma g$

First we investigate the performance of our methodology on the loop-induced $gg \to \gamma\gamma g$ process. Following the procedure outlined above, we use a unit integration grid, choose a random seed to generate the training and validation datasets with, and use another to infer on the trained model.

Figure 2 shows the performance of our trained NN ensemble at the matrix element level, here represented as the ratio of the model inferred values to the `NJet` evaluations. The errors form a narrow and approximately symmetric unit-centered distribution, thus demonstrating that the ensemble method has a reasonable per-point accuracy. The slightly

Cuts	NJet [pb]	NN ensemble [pb]
Baseline	$4.149 \times 10^{-6} \pm 6 \times 10^{-9}$	$4.19 \times 10^{-6} \pm 7 \times 10^{-8}$
Baseline + $p_{T,\gamma} > 50\,\text{GeV}$	$5.283 \times 10^{-7} \pm 8 \times 10^{-10}$	$5.4 \times 10^{-7} \pm 2 \times 10^{-8}$
Baseline + $m_{\gamma,\gamma} > 50\,\text{GeV}$	$3.300 \times 10^{-6} \pm 5 \times 10^{-9}$	$3.34 \times 10^{-6} \pm 5 \times 10^{-8}$

Table 1. Cross-sectional comparison between NJet and the NN ensemble approach using different cuts. Baseline cuts are those specified at the beginning of section 4. The NJet results are quoted with MC errors and the NN ensemble results with precision/optimality uncertainties calculated as described in ref. [60].

elongated right tale of the distribution is due to large matrix element values in highly divergent regions of phase space, yet these points are in the minority.

Once the ensemble is trained, it is converted to be called by the event generator interface which allows for the calculation of the cross section and differential distributions. table 1 shows the results of the cross section derived using NJet and the NN ensemble. We see that these two approaches are in excellent agreement, with the ensemble result overlapping within one standard deviation of that calculated by NJet. The errors on the NJet values are the MC errors, and the errors on the ensemble are precision/optimality uncertainties. The latter are calculated by training multiple ensembles with different random seeds in the weight initialisation, and in the shuffling of the training and validation datasets. MC errors are quoted to one standard deviation and the precision/optimality uncertainties to one standard error on the mean. A more in depth description of this uncertainty analysis can be found in section 2.3 of ref. [60].

The error plot and cross-section calculation provide good evidence for the performance of the NN ensemble method both in its ability to learn the distribution of phase-space points on average, as well as its robustness to being integrated into a wider event generation framework with additional phase-space and PDF weights. To further test the methodology in a more relevant way to how it would be used in practice, differential distributions can be used to assess robustness as they more explicitly expose performance on the divergent and tail events.

Figure 3 demonstrates the performance of the NN ensemble in comparison to NJet in six differential slices of phase space. These include p_T, angular, and diphoton system distributions which have been chosen to give a range of realistic constructions exploring different regions of phase space. In general, the NN ensemble is found to be in good agreement, particularly around the peaks, with the majority of the NN bin values being with the NJet MC error. The normalised NN uncertainties on the differential bins is negligible in comparison to the MC error. Strong performance is pronounced in the pseudorapidity distribution which shows variation at the percent level. The p_T and angular distributions show more fluctuations in the tail events, with the diphoton mass demonstrating the greatest deviations in these regions. However, despite these differences, fluctuations are clearly statistical rather than systematic meaning agreement will increase as the bins are aggregated. This is to be expected given the strong cross-section performance.

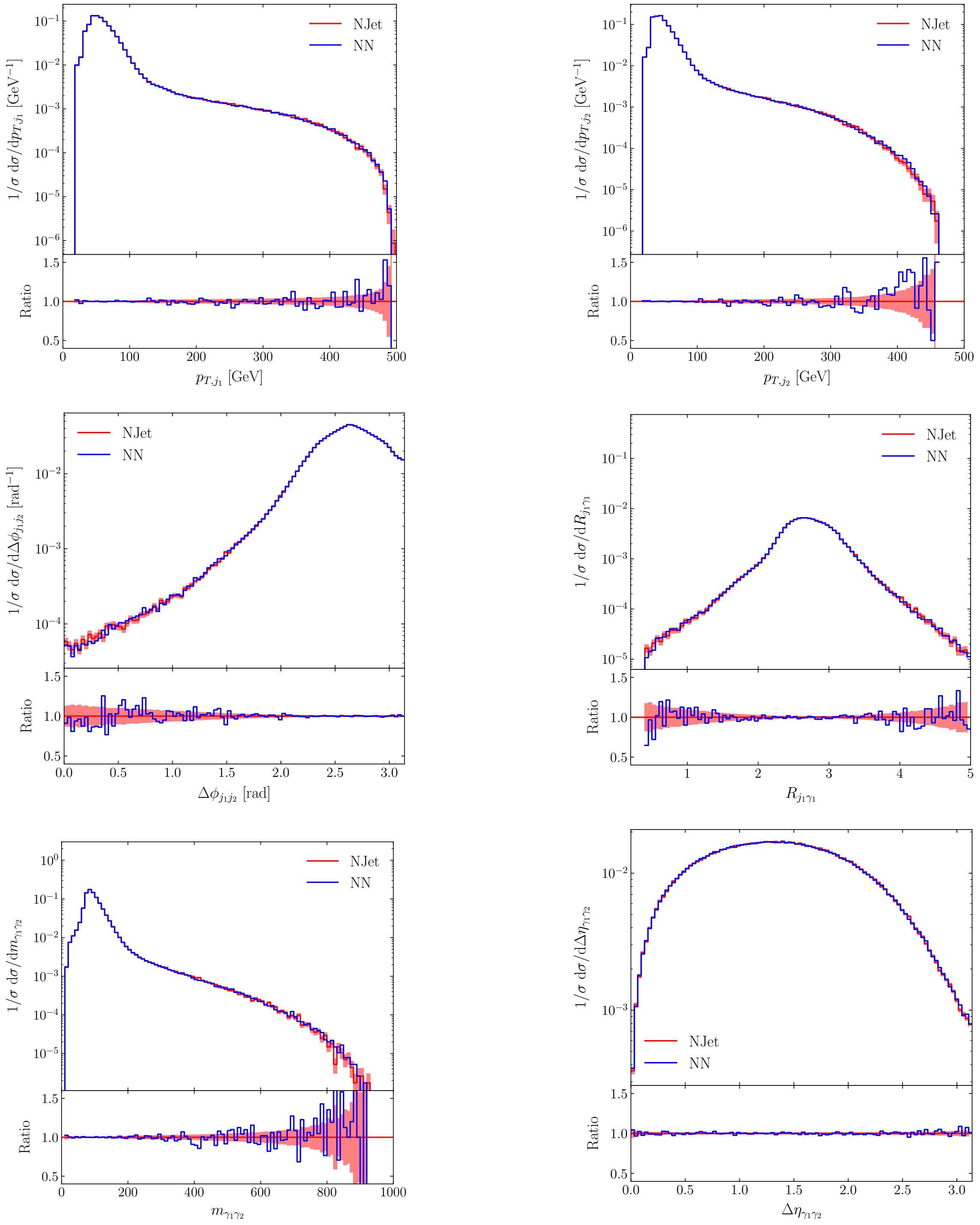


Figure 3. Differential distributions normalised to the cross section for the 2 → 3 process comparing `NJet` (red) with the NN ensemble (blue). The `NJet` results are quoted with MC errors, and the NN results with precision/optimality uncertainties calculated as described in ref. [60] but which are negligible in comparison. Pseudojets j_i and photons γ_i are ordered by energy, $\Delta\phi$ is azimuthal separation, R-separation is defined in section 3.2.1, and m_{γ_1,γ_2} and $\Delta\eta_{\gamma_1,\gamma_2}$ are the mass and pseudorapidity separation of the diphoton system.

The results presented so far have been derived from a NN ensemble trained and tested on the same integration grid and on the same cut parameters. However, in phenomenological explorations it is common to study a range of cut parameters, especially when measuring the effects of new phenomena. Since the NN ensemble performs well at the per-point level (as shown in figure 2), it should also be able to generalise to different cut parameter configurations. Specifically, the ensemble should still be applicable to harsher cuts than those used in training because the it expects the training and testing datasets to be drawn from the same statistical distributions. However, in the event that cuts are relaxed in comparison to those the model was trained on, reweighting could be employed for the relevant additional subset of points thereby guaranteeing the expected values in these 'unseen' regions of phase space.

Table 1 presents a comparison of cross-section values calculated using NJet and the NN ensemble with harsher cut values than the baseline. The agreement between the two approaches is comparable to the agreement found before the additional cuts were added, thereby suggesting good generalisability. Indeed, this is not surprising since the points with the largest errors between the NN and NJet were the most divergent points and therefore the ones more likely to be cut given the IR singularities present in these processes.

The generalisation to additional cut parameters both demonstrates the robustness of this training regime, as well as the practical gain in not having to retrain a network for each specified set of cuts. This allows us to generalise the training and testing procedure outlined at the beginning of this section to suggest that the NN ensemble be first trained on more relaxed cuts and then, as iterations of harsher cut parameters are explored during analysis, these can be applied without the ensemble significantly decreasing in performance. If cuts are to be relaxed then reweighting could be used to ensure good performance at the expense of compute time.

While the network performance has been shown to be strong overall, other reweighing methods can still be explored. Reweighting randomly across all phase space, even at the 20–40% level, was not found to significantly reduce the difference in the computed cross sections. Similarly, the NN ensemble uncertainties were not found to be correlated with the errors, and so were discarded as a good reweighting criteria. As mentioned above, the points in which targeted reweighting can be most beneficial are those which fall within the divergent regions of phase space. Figure 4 presents the results of reweighting points randomly in $\mathcal{R}_{\text{div}}$ (as defined in eq. (3.1)), and shows an improvement in the cross section — reweighting a greater number of points enables the reweighted cross section, $\sigma^{(\text{RW})}$, to converge to the value calculated by NJet, $\sigma^{(\texttt{NJet})}$. Indeed, to achieve almost equal values in the cross sections, the total proportion of phase space requiring reweighting is at the percent level. Therefore, we find reweighting in the $\mathcal{R}_{\text{div}}$ region of phase space and/or when relaxing cuts in relation to those used during training can improve model performance.

Finally, although the cross section and differential distributions provide a means to test the robustness of our approach against the additional weights introduced during event generation, we can more explicitly single out the effects of the PDF weights by calculating the NN ensemble error as a function of the momentum fractions, x_1 and x_2, of the initial state partons. Figure 5 shows the root mean squared error (RMSE) of the NN as a function

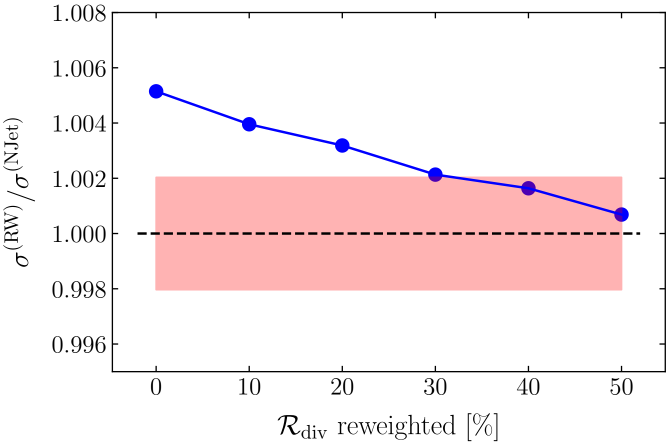


Figure 4. Effect of reweighting points in the divergent region of phase space, $\mathcal{R}_{\mathrm{div}}$, on the ratio between the reweighted cross section, $\sigma^{(\mathrm{RW})}$, and the cross section calculated using `NJet` $\sigma^{(\texttt{NJet})}$ for the 2 → 3 process. In this case, the divergent region comprises approximately 7–9% of the total phase space (see appendix B for details). The red band shows the MC error on the `NJet` result.

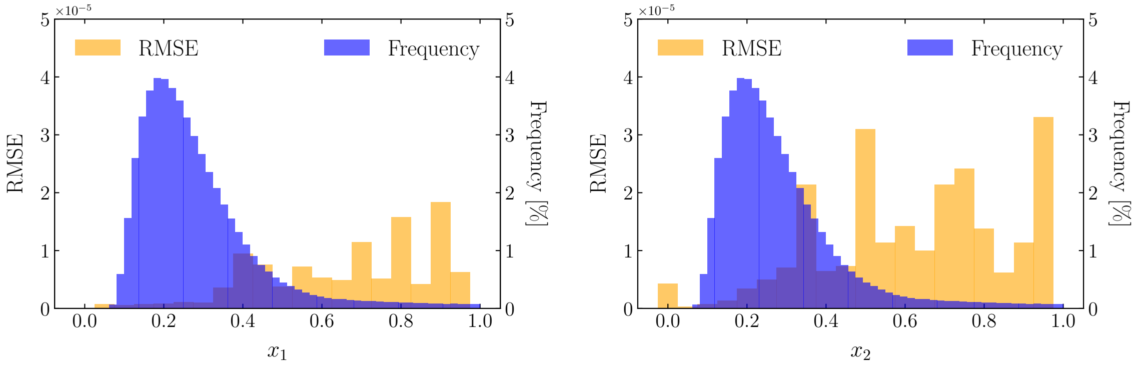


Figure 5. Root mean squared error (RMSE) of the NN ensemble approach in comparison to `NJet` as a function of x_1 and x_2, and the frequency of points with these values in the training dataset.

of these variables, along with the frequency of points as they appear in the training dataset. As expected, the ensemble performs better in locations with more points, and we only see the RMSE grow more significantly in the regions of low-statistics. Since the gluon PDF falls off as x approaches one, and peaks in the low x region, this provides another test of ensemble robustness during the external introduction of PDF weights.

4.1.1 Aside: VEGAS grid optimisation

The results presented so far have used a unit integration grid and RAMBO integrator in order to be process agnostic in the phase-space sampling. As mentioned in section 3.3.2, however, it is common to use importance sampling and other optimisation techniques to speed up integration conversion. To test the robustness of our approach to these alternative integrators, we use VEGAS during the optimisation grid generation stage. Figure 6 shows the error plots for the 2 → 3 scatting process using this optimisation setup while keeping all

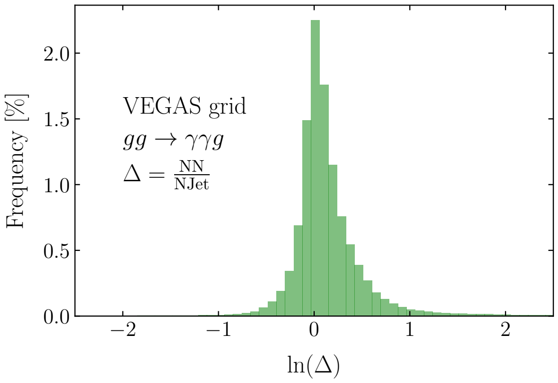


Figure 6. NN/`NJet` errors for the 2 → 3 scattering process using a VEGAS optimised integration grid.

other parameters setup parameters fixed. Here, we see that the shape exhibited in the error plots is similar to that of the unit grid shown in figure 2, although slightly broader around the peak. This is likely due to the larger number of points placed in the divergent regions by the VEGAS integrator. The cross section was also found to be in excellent agreement, with `NJet` giving $4.151 \times 10^{-6} \pm 1.1 \times 10^{-8}$ pb, and the ensemble giving $4.22 \times 10^{-6} \pm 8 \times 10^{-8}$ pb.

4.2 $gg \to \gamma\gamma gg$

We now turn to investigate the $gg \to \gamma\gamma gg$ process. Analytic expressions for this process are not available and the numerical implementation is significantly more computationally expensive than for the equivalent 2 → 3 process (see section 4.3). Integration grid optimisation is therefore highly inefficient, and so for the remainder of this section a unit grid will be used. To test generalisability, the NN setup is as in section 4.1, with the only change being in the chosen value of $y_p = 0.001$. At higher multiplicity, a greater proportion of points fall within the divergent region, $\mathcal{R}_{\text{div}}$, however, this can hinder model performance by unbalancing the training regime. It is therefore reasonable to aim to keep the proportion of points in this region approximately constant throughout our experiments which is achieved by lowering the value of y_p (see appendix B for more details).

Figure 7 shows the performance of our trained NN ensemble at the matrix element level. As expected, the performance has decreased relative to the 2 → 3 process shown in figure 2, yet the error distribution is still found to be approximately Gaussian, although with a shifted mean. Despite this, the cross section calculated using the NN ensemble — $4.5 \times 10^{-6} \pm 6 \times 10^{-7}$ pb — is found to be in excellent agreement with that derived from `NJet` — $4.9 \times 10^{-6} \pm 5 \times 10^{-7}$ pb. This suggests that although there are several points where the ensemble approach performs poorly, particularly in comparison to the 2 → 3 process, these are largely in the divergent region and found to not affect the cross-section calculation too greatly.

Figure 8 shows the performance of the ensemble approach in six differential slices of phase space. As in the previous example, the ensemble is found to perform well relative to `NJet`: while noise in the tails of the distributions is still observed, these appear to be

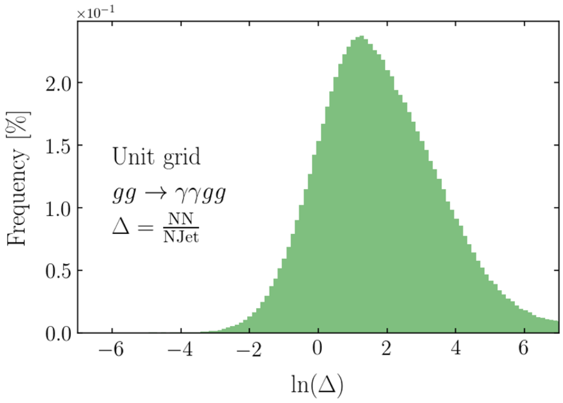


Figure 7. NN/`NJet` errors for the 2 → 4 scattering process using a unit integration grid.

reduced in comparison to the 2 → 3 process. This further supports the assertion that the points where the ensemble performs poorly are suppressed.

Given the difference in cross-section values calculated using `NJet` and the ensemble approach, we perform reweighting in the divergent region as discussed in section 3.4 and section 4.1. As shown in figure 9, reweighting in this region can bring the NN ensemble derived cross section closer to the value calculated using `NJet`. In the case of the 2 → 4 process, the MC error on the `NJet` result is significantly larger for the same number of points compared to the 2 → 3 process. Given these larger error, and that the ratio $\sigma^{(\text{RW})}/\sigma^{(\texttt{NJet})}$ resides within these errors, it is predictably noisy, yet still converges showing that this approach to reweighting can be generalised across multiple processes.

4.3 Timing

We repeat the performance evaluation of figure 1 with methods involving error estimation as these are likely to be employed in real-world usage. For conventional techniques, the dimension scaling test is a standard way to estimate error on the result and introduces a second matrix element call for each phase-space point evaluation. As discussed in section 4.1, we propose running 20 NN ensembles for each point to obtain a mean with standard error.

The results, shown in figure 10, demonstrate the per-point speedup in using amplitude NNs in practice. For the 2 → 4 process, where amplitude calls dominate conventional simulation time, a 10^4 times speedup in amplitude calls is observed which renders the inference stage as negligible in the total time of our NN-based simulation pipeline. Indeed, in comparison to the numerical calculation of the matrix element, the training time of the NN ensemble can also be considered negligible, meaning the total speed up in the overall simulation time is of the order $N_{\text{infer}}/N_{\text{train}}$ — the ratio of the number of inference point to the number of points in the training dataset.

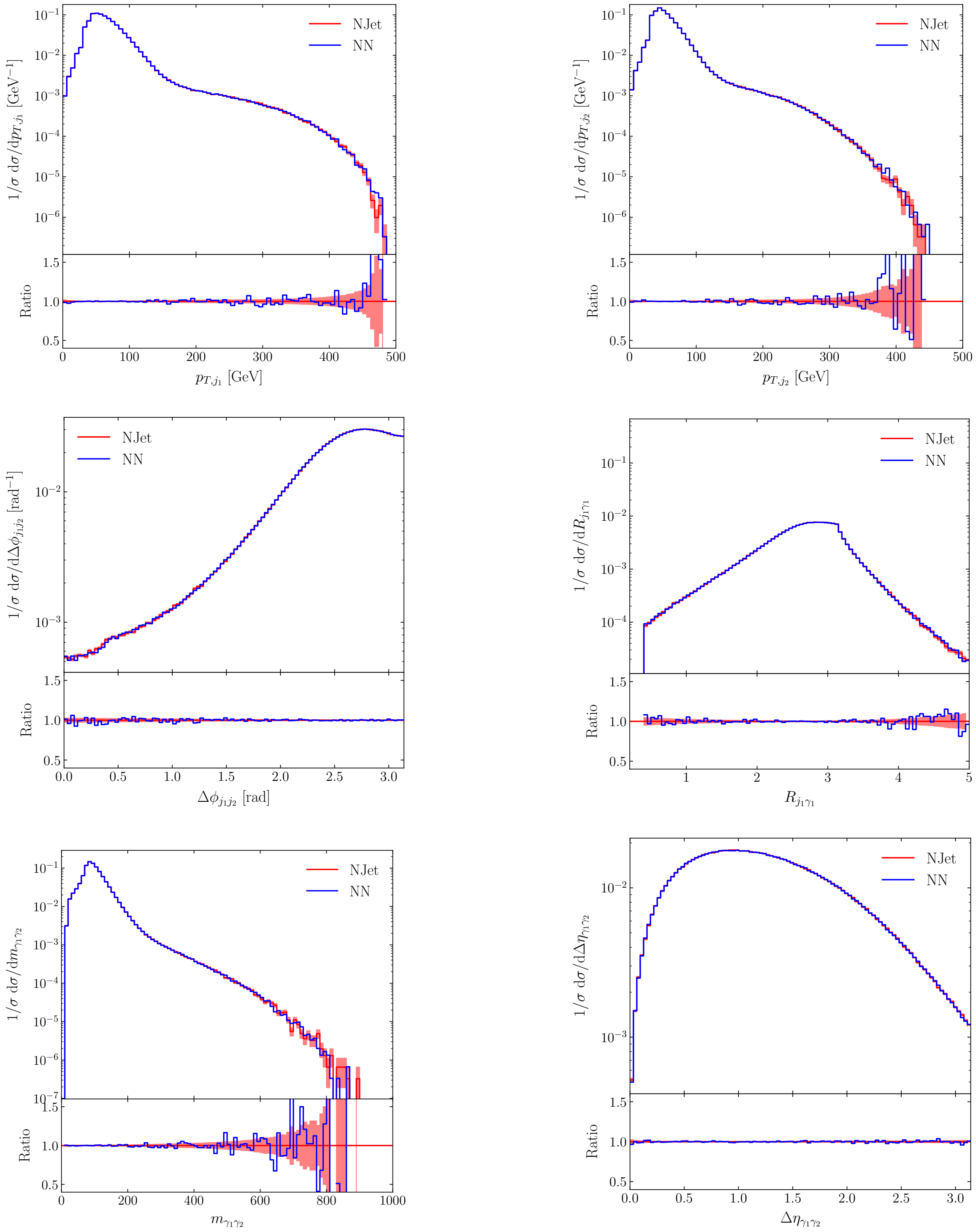


Figure 8. Differential distributions normalised to the cross section for the 2 → 4 process comparing `NJet` (red) with the NN ensemble (blue). The `NJet` results are quoted with MC errors, and the NN results with precision/optimality uncertainties calculated as described in ref. [60] but which are negligible in comparison.

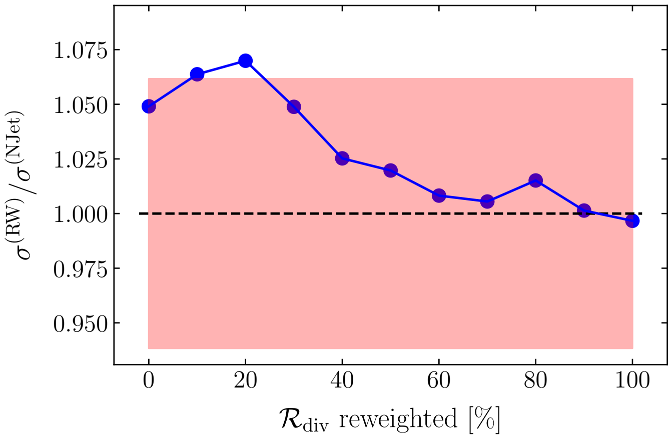


Figure 9. Effect of reweighting points in the divergent region of phase space, $\mathcal{R}_{\mathrm{div}}$, on the ratio between the reweighted cross section, $\sigma^{(\mathrm{RW})}$, and the cross section calculated using `NJet` $\sigma^{(\mathtt{NJet})}$ for the 2 → 4 process. In this case, the divergent region comprises approximately 2–3% of the total phase space (see appendix B for details). The red band shows the MC error on the `NJet` result.

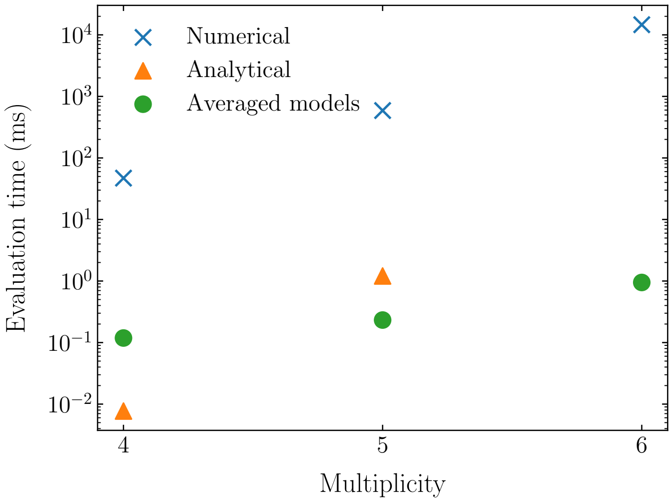


Figure 10. Typical per-point call times for the set of NN ensembles and scaling tests with numerical and analytical techniques against the number of legs. Compared to figure 1, this incurs a twofold cost on the conventional methods and multiplies the single NN ensemble time by 20. Analytical methods are fastest at 2 → 2 and NNs do not offer a dramatic improvement at 2 → 3 either, but their fast call time and weak dependence on the number of variables (which scales with multiplicity) win out at high multiplicity. At 2 → 4, where no analytical expression is available and extrapolation suggests it would be comparable in call time to numerics, our ML approach is four orders of magnitude faster than the numeric call.

5 Conclusions

In this article we provided further evidence that NNs provide a general framework for the optimisation of high multiplicity observables at hadron colliders. We extended preliminary studies [60] for electron-position scattering to hadron-hadron collisions and provided a general interface to the `Sherpa` MC event generator for NNs trained with the `NJet` amplitude library. We focused on the loop-induced processes $gg \to \gamma\gamma + n(g)$ which cause problems for conventional phase-space generation methods and require the computation of expensive scattering amplitudes.

We trained an ensemble of NNs which divide the scattering amplitudes into IR divergent sectors according to the FKS mapping and find excellent agreement between distributions generated with the networks and those generated with the conventional approach. Errors from the NN were included through variations of training parameters and show good agreement with the direct comparisons. We also showed that by reweighting the generated events according to their divergence structure, the accuracy of the simulation could be improved at a rather low computational cost. This step also provided additional confidence in the inference of the network.

We saw, especially for the $2 \to 4$ process, a good improvement in the total simulation time. Since the calls to the scattering amplitudes dominated the total time, the speedup was given by the ratio of the number of points used in training to the total number of calls used in the full simulation (during event generation). This came out around a factor of 30. While this was good to see, it is not the limit of the optimisation. If the trained networks can be used for many subsequent simulations with different kinematic cuts, the overall improvement would be much greater. We showed that our networks reproduce distributions with different cuts in the transverse momentum without the requirement for retraining, which was very encouraging. Additional improvements to the inference stage (event generation with the network) could be provided through GPUs which could be important if a large number of variations in the cut analysis were required. The distribution of the trained networks was also simpler than the large quantity of data generated with Root Ntuples [89, 90], another technique for optimising the information that can be extracted from expensive simulations.

There remain open questions of course. It would be very interesting to apply the technique to the more intricate problem of real radiation event generation since NLO and NNLO simulations are often dominated by these contributions. It may also be beneficial to make connections between the amplitude-level approach taken in this article and those focusing on phase space or complete simulation including parton showering and even detector simulation.

We hope that these studies will help to develop a general framework that can be used in future experimental analysis.

A Hyperparameter tuning

Hyperparameter tuning was performed on a dataset of 1M points (derived independently from the datasets used for validation and testing in section 4) to explore optimal data processing and model parameter choices. Given the computational expense of generating data, this was only done for the 2 → 3 process.

We tested different model architecture constructions (changing the number of hidden layers and/or the number of nodes in each hidden layer), data preprocessing methods, and model loss functions. All other training parameters are as described in section 3.2. For data preprocessing methods, we tested input variable standardisation, i.e. the training and validation data input variables are each standardised to have zero mean and unit variance, and normalisation, i.e. the training and validation data input variables are each normalised according to min/max normalisation

$$x^* = \frac{x - \min(X_{\text{train}})}{\max(X_{\text{train}}) - \min(X_{\text{train}})} \tag{A.1}$$

where $x \in X_{\text{train}}$, $X_{\text{train}} \subset \mathbb{R}$ is the set of training data for a given input variable, and x^* is the input variable normalised from x. This procedure means the dataset is normalised such that $x^* \in [0, 1]$ and therefore encourages a positive-definite output. When using the standardisation preprocessing step, we use hyperbolic-tangent activation functions in the hidden layers, but for normalisation we use rectified linear units (ReLU) [91]. This latter choice is to further encourage a positive-definite output and also aims to increase the rate of convergence.

It should be noted that a clear limitation of the positive-definite conditioning of the normalisation procedure is a reliance on the following conditions:

$$\min(X_{\text{train}}) = \min(X_{\text{train}} \cup X_{\text{test}}), \tag{A.2}$$

$$\max(X_{\text{train}}) = \max(X_{\text{train}} \cup X_{\text{test}}), \tag{A.3}$$

where $X_{\text{test}} \subset \mathbb{R}$ is the set variable inputs derived from the testing data, and therefore $X_{\text{train}} \cup X_{\text{test}}$ represents the combination of the training, validation and testing sets. Since the performance gain from using the ML approach is that the training and validation sets combined are much smaller than the testing set, the above conditions are likely to break down as $n(X_{\text{train}}) \gg n(X_{\text{test}})$.

Two model loss functions were tested during hyperparameter tuning. The first was the mean squared error (MSE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(f(x_i) - y_i)^2 \tag{A.4}$$

where n is the number of training points, $f : \mathbb{R}^d \to \mathbb{R}$ is the function describing the neural network, x_i is the i^{th} d-dimensional input data (here $d = 4n$), and y_i the corresponding target variable. The second is the mean squared logarithmic error (MSLE)

$$L = \frac{1}{n}\sum_{i=1}^{n}(\log(f(x_i) + 1) - \log(y_i + 1))^2. \tag{A.5}$$

	Processing		Layers				Loss		Error	
	Std	Norm	20-40-20	30-60-30	20-30-40-30-20	30-40-50-40-30	MSE	MSLE	RMSE	RMSLE
1	x		x				x		2.828×10^{-5}	2.790×10^{-5}
2	x		x					x	3.320×10^{-5}	3.288×10^{-5}
3	x			x			x		2.829×10^{-5}	2.790×10^{-5}
4	x			x				x	3.820×10^{-5}	3.791×10^{-5}
5	x				x		x		2.829×10^{-5}	2.791×10^{-5}
6	x				x			x	3.147×10^{-5}	3.113×10^{-5}
7	x					x	x		2.830×10^{-5}	2.792×10^{-5}
8	x					x		x	3.454×10^{-5}	3.422×10^{-5}
9		x	x				x		2.835×10^{-5}	2.797×10^{-5}
10		x	x					x	4.799×10^{-4}	4.802×10^{-4}
11		x		x			x		2.835×10^{-5}	2.797×10^{-5}
12		x		x				x	7.396×10^{-4}	7.405×10^{-4}
13		x			x		x		2.836×10^{-5}	2.797×10^{-5}
14		x			x			x	3.414×10^{-4}	3.416×10^{-4}
15		x				x	x		2.836×10^{-5}	2.797×10^{-5}
16		x				x		x	5.599×10^{-4}	5.604×10^{-4}

Table 2. Hyperparameter tuning results. Tuning was performed on a fixed training dataset size of 100k points sampled using the RAMBO integrator [84] on a unit integration grid. Performance was measured with respect to both the Root Mean Squared Error (RMSE) and Root Mean Squared Logarithmic Error (RMSLE) so as to avoid biasing the error measure to the optimisation criterion (loss function) chosen.

Given the problem of approximating matrix element values for complex scattering processes, the target variable can take on a wide range of values spanning several orders of magnitude. These large values can sometimes be especially important to the cross section and so the MSE's penalisation of large outlier values can be beneficial; however, this might also make the training unstable. We included the MSLE during training to test if reducing the sensitivity to large scale variations in the target value is beneficial.

The results of the hyperparameter tuning can be found in table 2. Here we see that using data standardisation with an MSE loss function generally produces better results, although there does not seem to be a clear dependence on the model architecture. Given these findings, we choose to train our models using data standardisation with hyperbolic-tangent activation functions, a MSE loss function, and an architecture of 20-40-20. This is consistent with that presented in ref. [60].

B y_p tuning

The choice of y_p defines the partition between the divergent region of phase space, $\mathcal{R}_{\text{div}}$, and the non-divergent region, $\mathcal{R}_{\text{non-div}}$ (see eqs. (3.1) and (3.2)). While it may be assumed that

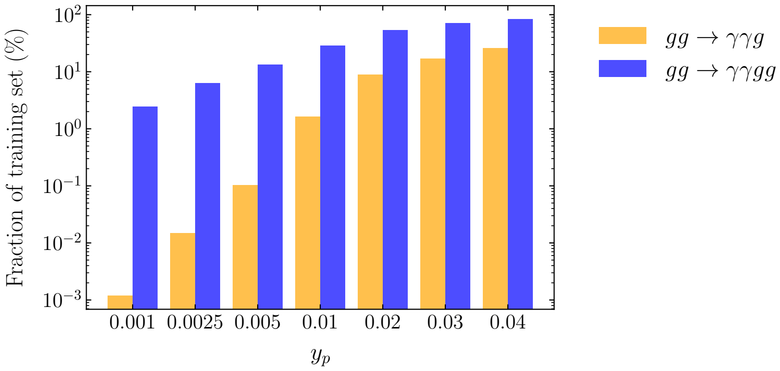


Figure 11. Proportion of the training dataset in the divergent region, $\mathcal{R}_{\text{div}}$, as a function of y_p for the $2 \to 3$ and $2 \to 4$ process.

having more points in each region is helpful since it provides more data for the networks trained in each region, this is not always the case. Including a mixture of points in the training dataset, with large imbalances in the distribution of different scales, can make the network optimisation procedure increasingly noisy. For this reason, we seek to choose a value of y_p which provides a balance between having enough divergent points to learn those regions well, whilst not providing too many points not in this limit and which share similar scales to points in the non-divergent regions of phase space.

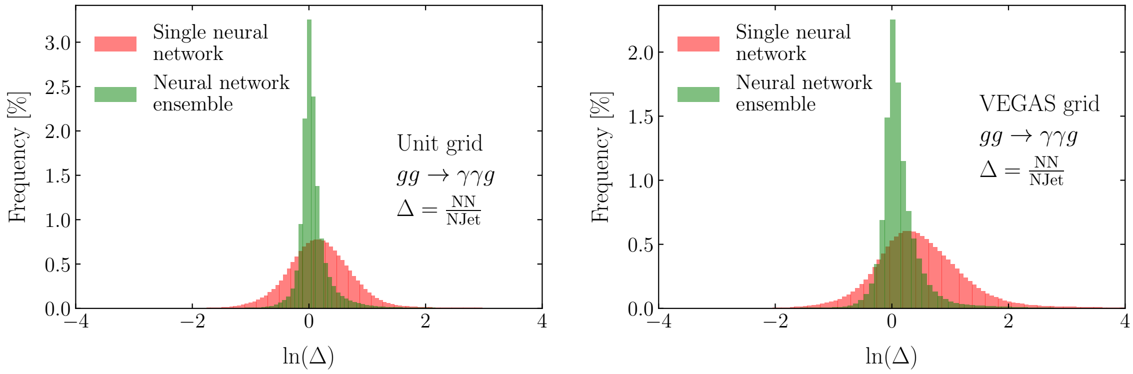
To be consistent with prior work [60], we initially chose $y_p = 0.02$ although the number of points falling into $\mathcal{R}_{\text{div}}$ depends on the multiplicity of the process. As presented in section 4.1, this value was shown to perform well,[2] yet the same value would place a significantly greater proportion of points into the divergent region when another external leg is added (see figure 11). Instead of choosing the same value of y_p for all processes, we aim to select a value which keeps the proportion of points in the divergent region at the level of 2–8% of the whole phase space sampled. We choose a value of $y_p = 0.02$ for the $2 \to 3$ process, and $y_p = 0.001$ for the $2 \to 4$ process.[3]

C Comparison with the naive setup

Throughout this paper, all results presented using a ML approach have used the NN ensemble methodology. This approach had been shown previously to outperform a naive single NN trained over the whole of phase space for e^+e^- collisions [60]. In particular, the motivation for this approach was enhanced performance in handling real emission, IR singular regions of phase space, which similarly occur in the processes studied in this work, especially at high multiplicity. For completeness, we perform a similar comparison on the $2 \to 3$ gluon-initiated diphoton processes; we do not compare on the $2 \to 4$ process as it is computationally expensive to do so and it is a natural higher multiplicity extension of the $2 \to 3$ process.

[2] The value of $y_p = 0.01$ was also tested and found to be in similarly good agreement.

[3] A value of $y_p = 0.0025$ for the $2 \to 4$ process would also allow for this; however, at high multiplicity the lower value of this cut provided more optimal performance.



(a) Unit integration grid.

(b) VEGAS integration grid.

Figure 12. Comparison of NN/`NJet` errors between the single NN and NN ensemble approaches for the 2 → 3 scattering process using different integration grids.

Figure 12 shows the matrix level error analysis of the 2 → 3 scattering process using both a unit and VEGAS optimisation grid. In both cases, the error distribution for the single NN approach has a significantly broader character than the ensemble method. This demonstrates that the findings of described in ref. [60] are consistent with those presented in this study.

Acknowledgments

We thank Frank Krauss for many insightful discussions, Alan Price and Marek Schoenherr for assistance with `Sherpa` and Joseph Walker for help with `Rivet`. This project received funding from the European Union's Horizon 2020 research and innovation programmes *High precision multi-jet dynamics at the LHC* (grant agreement No 772009). JAB is supported by the U.K. Research and Innovation Science and Technology Facilities Council (UKRI-STFC) grant number ST/P006744/1 and ST/P001246/1. RM is supported by UKRI-STFC ST/S505365/1 and ST/P001246/1. This paper made use of `Python` [92] and the following `Python` libraries: `Matplotlib` [93], `Numpy` [94] and `Pandas` [95, 96].

References

[1] C.F. Berger et al., *An Automated Implementation of On-Shell Methods for One-Loop Amplitudes*, *Phys. Rev. D* **78** (2008) 036003 [`arXiv:0803.4180`] [INSPIRE].

[2] G. Bevilacqua et al., *HELAC-NLO*, *Comput. Phys. Commun.* **184** (2013) 986 [`arXiv:1110.1499`] [INSPIRE].

[3] G. Cullen et al., *GOSAM-2.0: a tool for automated one-loop calculations within the Standard Model and beyond*, *Eur. Phys. J. C* **74** (2014) 3001 [`arXiv:1404.7096`] [INSPIRE].

[4] J. Alwall et al., *The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations*, *JHEP* **07** (2014) 079 [arXiv:1405.0301] [INSPIRE].

[5] A. Denner, J.-N. Lang and S. Uccirati, *Recola2: REcursive Computation of One-Loop Amplitudes 2*, *Comput. Phys. Commun.* **224** (2018) 346 [arXiv:1711.07388] [INSPIRE].

[6] T. Gleisberg et al., *Event generation with SHERPA 1.1*, *JHEP* **02** (2009) 007 [arXiv:0811.4622] [INSPIRE].

[7] SHERPA collaboration, *Event Generation with Sherpa 2.2*, *SciPost Phys.* **7** (2019) 034 [arXiv:1905.09127] [INSPIRE].

[8] T. Sjöstrand, S. Mrenna and P.Z. Skands, *PYTHIA 6.4 Physics and Manual*, *JHEP* **05** (2006) 026 [hep-ph/0603175] [INSPIRE].

[9] T. Sjöstrand et al., *An introduction to PYTHIA 8.2*, *Comput. Phys. Commun.* **191** (2015) 159 [arXiv:1410.3012] [INSPIRE].

[10] M. Bahr et al., *HERWIG++ Physics and Manual*, *Eur. Phys. J. C* **58** (2008) 639 [arXiv:0803.0883] [INSPIRE].

[11] G. Corcella et al., *HERWIG 6: An Event generator for hadron emission reactions with interfering gluons (including supersymmetric processes)*, *JHEP* **01** (2001) 010 [hep-ph/0011363] [INSPIRE].

[12] J. Bellm et al., *HERWIG 7.0/HERWIG++ 3.0 release note*, *Eur. Phys. J. C* **76** (2016) 196 [arXiv:1512.01178] [INSPIRE].

[13] T. Gehrmann, N. Greiner and G. Heinrich, *Precise QCD predictions for the production of a photon pair in association with two jets*, *Phys. Rev. Lett.* **111** (2013) 222002 [arXiv:1308.3660] [INSPIRE].

[14] S. Badger, A. Guffanti and V. Yundin, *Next-to-leading order QCD corrections to di-photon production in association with up to three jets at the Large Hadron Collider*, *JHEP* **03** (2014) 122 [arXiv:1312.5927] [INSPIRE].

[15] Z. Bern et al., *Next-to-leading order $\gamma\gamma + 2$-jet production at the LHC*, *Phys. Rev. D* **90** (2014) 054004 [arXiv:1402.4127] [INSPIRE].

[16] B. Agarwal, F. Buccioni, A. von Manteuffel and L. Tancredi, *Two-loop leading colour QCD corrections to $q\bar{q} \to \gamma\gamma g$ and $qg \to \gamma\gamma q$*, *JHEP* **04** (2021) 201 [arXiv:2102.01820] [INSPIRE].

[17] H.A. Chawdhry, M. Czakon, A. Mitov and R. Poncelet, *Two-loop leading-colour QCD helicity amplitudes for two-photon plus jet production at the LHC*, *JHEP* **07** (2021) 164 [arXiv:2103.04319] [INSPIRE].

[18] B. Agarwal, F. Buccioni, A. von Manteuffel and L. Tancredi, *Two-loop helicity amplitudes for diphoton plus jet production in full color*, arXiv:2105.04585 [INSPIRE].

[19] H.A. Chawdhry, M. Czakon, A. Mitov and R. Poncelet, *NNLO QCD corrections to diphoton production with an additional jet at the LHC*, arXiv:2105.06940 [INSPIRE].

[20] S. Badger et al., *Virtual QCD corrections to gluon-initiated diphoton plus jet production at hadron colliders*, arXiv:2106.08664 [INSPIRE].

[21] NNPDF collaboration, *Parton distributions for the LHC Run II*, *JHEP* **04** (2015) 040 [arXiv:1410.8849] [INSPIRE].

[22] M. Czakon, *Tops from Light Quarks: Full Mass Dependence at Two-Loops in QCD*, *Phys. Lett. B* **664** (2008) 307 [arXiv:0803.1400] [INSPIRE].

[23] S. Borowka et al., *Higgs Boson Pair Production in Gluon Fusion at Next-to-Leading Order with Full Top-Quark Mass Dependence*, *Phys. Rev. Lett.* **117** (2016) 012001 [*Erratum ibid.* **117** (2016) 079901] [arXiv:1604.06447] [INSPIRE].

[24] G. Heinrich, S.P. Jones, M. Kerner, G. Luisoni and E. Vryonidou, *NLO predictions for Higgs boson pair production with full top quark mass dependence matched to parton showers*, *JHEP* **08** (2017) 088 [arXiv:1703.09252] [INSPIRE].

[25] S.P. Jones, M. Kerner and G. Luisoni, *Next-to-Leading-Order QCD Corrections to Higgs Boson Plus Jet Production with Full Top-Quark Mass Dependence*, *Phys. Rev. Lett.* **120** (2018) 162001 [arXiv:1802.00349] [INSPIRE].

[26] G. Heinrich, S.P. Jones, M. Kerner, G. Luisoni and L. Scyboz, *Probing the trilinear Higgs boson coupling in di-Higgs production at NLO QCD including parton shower effects*, *JHEP* **06** (2019) 066 [arXiv:1903.08137] [INSPIRE].

[27] J. Bendavid, *Efficient Monte Carlo Integration Using Boosted Decision Trees and Generative Deep Neural Networks*, arXiv:1707.00028 [INSPIRE].

[28] M.D. Klimek and M. Perelstein, *Neural Network-Based Approach to Phase Space Integration*, *SciPost Phys.* **9** (2020) 053 [arXiv:1810.11509] [INSPIRE].

[29] I.-K. Chen, M.D. Klimek and M. Perelstein, *Improved Neural Network Monte Carlo Simulation*, *SciPost Phys.* **10** (2021) 023 [arXiv:2009.07819] [INSPIRE].

[30] D.J. Rezende and S. Mohamed, *Variational inference with normalizing flows*, in *32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15 (2015), pp. 1530-1538 arXiv:1505.05770 [https://www.jmlr.org/].

[31] L. Dinh, D. Krueger and Y. Bengio, *NICE: Non-linear independent components estimation*, in *3rd International Conference on Learning Representations*, ICLR'15 (2015). arXiv:1410.8516.

[32] C. Gao, J. Isaacson and C. Krause, *i-flow: High-dimensional Integration and Sampling with Normalizing Flows*, *Mach. Learn. Sci. Tech.* **1** (2020) 045023 [arXiv:2001.05486] [INSPIRE].

[33] E. Bothmann, T. Janßen, M. Knobbe, T. Schmale and S. Schumann, *Exploring phase space with Neural Importance Sampling*, *SciPost Phys.* **8** (2020) 069 [arXiv:2001.05478] [INSPIRE].

[34] C. Gao, S. Höche, J. Isaacson, C. Krause and H. Schulz, *Event Generation with Normalizing Flows*, *Phys. Rev. D* **101** (2020) 076002 [arXiv:2001.10028] [INSPIRE].

[35] I.J. Goodfellow et al., *Generative Adversarial Networks*, in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence and K.Q. Weinberger eds., Curran Associates Inc. (2014) arXiv:1406.2661 [INSPIRE].

[36] S. Otten et al., *Event Generation and Statistical Sampling for Physics with Deep Generative Models and a Density Information Buffer*, *Nature Commun.* **12** (2021) 2985 [arXiv:1901.00875] [INSPIRE].

[37] B. Hashemi, N. Amin, K. Datta, D. Olivito and M. Pierini, *LHC analysis-specific datasets with Generative Adversarial Networks*, arXiv:1901.05282 [INSPIRE].

[38] R. Di Sipio, M. Faucci Giannelli, S. Ketabchi Haghighat and S. Palazzo, *DijetGAN: A Generative-Adversarial Network Approach for the Simulation of QCD Dijet Events at the LHC*, *JHEP* **08** (2019) 110 [arXiv:1903.02433] [INSPIRE].

[39] A. Butter, T. Plehn and R. Winterhalder, *How to GAN LHC Events*, *SciPost Phys.* **7** (2019) 075 [arXiv:1907.03764] [INSPIRE].

[40] S. Carrazza and F.A. Dreyer, *Lund jet images from generative and cycle-consistent adversarial networks*, *Eur. Phys. J. C* **79** (2019) 979 [arXiv:1909.01359] [INSPIRE].

[41] SHiP collaboration, *Fast simulation of muons produced at the SHiP experiment using Generative Adversarial Networks*, 2019 *JINST* **14** P11028 [arXiv:1909.04451] [INSPIRE].

[42] A. Butter and T. Plehn, *Generative Networks for LHC events*, arXiv:2008.08558 [INSPIRE].

[43] A. Butter, S. Diefenbacher, G. Kasieczka, B. Nachman and T. Plehn, *GANplifying Event Samples*, *SciPost Phys.* **10** (2021) 139 [arXiv:2008.06545] [INSPIRE].

[44] Y. Alanazi et al., *AI-based Monte Carlo event generator for electron-proton scattering*, arXiv:2008.03151 [INSPIRE].

[45] A. Butter, *Generative Models in Event Simulation*, *PoS* **LHCP2020** (2021) 055 [INSPIRE].

[46] Y. Alanazi et al., *Simulation of electron-proton scattering events by a Feature-Augmented and Transformed Generative Adversarial Network (FAT-GAN)*, arXiv:2001.11103 [INSPIRE].

[47] T. Lebese, B. Mellado and X. Ruan, *The use of Generative Adversarial Networks to characterise new physics in multi-lepton final states at the LHC*, arXiv:2105.14933 [INSPIRE].

[48] M. Backes, A. Butter, T. Plehn and R. Winterhalder, *How to GAN Event Unweighting*, *SciPost Phys.* **10** (2021) 089 [arXiv:2012.07873] [INSPIRE].

[49] B. Stienen and R. Verheyen, *Phase Space Sampling and Inference from Weighted Events with Autoregressive Flows*, *SciPost Phys.* **10** (2021) 038 [arXiv:2011.13445] [INSPIRE].

[50] A. Butter, T. Plehn and R. Winterhalder, *How to GAN Event Subtraction*, arXiv:1912.08824 [INSPIRE].

[51] M. Bellagente, M. Haußmann, M. Luchmann and T. Plehn, *Understanding Event-Generation Networks via Uncertainties*, arXiv:2104.04543 [INSPIRE].

[52] E. Bothmann and L. Debbio, *Reweighting a parton shower using a neural network: the final-state case*, *JHEP* **01** (2019) 033 [arXiv:1808.07802] [INSPIRE].

[53] L. de Oliveira, M. Paganini and B. Nachman, *Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis*, *Comput. Softw. Big Sci.* **1** (2017) 4 [arXiv:1701.05927] [INSPIRE].

[54] J.W. Monk, *Deep Learning as a Parton Shower*, *JHEP* **12** (2018) 021 [arXiv:1807.03685] [INSPIRE].

[55] K. Dohi, *Variational Autoencoders for Jet Simulation*, arXiv:2009.04842 [INSPIRE].

[56] B. Nachman and J. Thaler, *Neural resampler for Monte Carlo reweighting with preserved uncertainties*, *Phys. Rev. D* **102** (2020) 076004 [arXiv:2007.11586] [INSPIRE].

[57] S. Otten, K. Rolbiecki, S. Caron, J.-S. Kim, R. Ruiz De Austri and J. Tattersall, *DeepXS: Fast approximation of MSSM electroweak cross sections at NLO*, *Eur. Phys. J. C* **80** (2020) 12 [arXiv:1810.08312] [INSPIRE].

[58] A. Buckley et al., *Xsec: the cross-section evaluation code*, *Eur. Phys. J. C* **80** (2020) 1106 [arXiv:2006.16273] [INSPIRE].

[59] F. Bishara and M. Montull, *(Machine) Learning amplitudes for faster event generation*, arXiv:1912.11055 [INSPIRE].

[60] S. Badger and J. Bullock, *Using neural networks for efficient evaluation of high multiplicity scattering amplitudes*, *JHEP* **06** (2020) 114 [arXiv:2002.07516] [INSPIRE].

[61] M. Feickert and B. Nachman, *A Living Review of Machine Learning for Particle Physics*, arXiv:2102.02770 [INSPIRE].

[62] S. Badger, B. Biedermann, P. Uwer and V. Yundin, *Numerical evaluation of virtual corrections to multi-jet production in massless QCD*, *Comput. Phys. Commun.* **184** (2013) 1981 [arXiv:1209.0100] [INSPIRE].

[63] D. de Florian and Z. Kunszt, *Two photons plus jet at LHC: The NNLO contribution from the gg initiated process*, *Phys. Lett. B* **460** (1999) 184 [hep-ph/9905283] [INSPIRE].

[64] Z. Bern, A. De Freitas and L.J. Dixon, *Two loop amplitudes for gluon fusion into two photons*, *JHEP* **09** (2001) 037 [hep-ph/0109078] [INSPIRE].

[65] T. Peraro, *FiniteFlow: multivariate functional reconstruction using finite fields and dataflow graphs*, *JHEP* **07** (2019) 031 [arXiv:1905.08019] [INSPIRE].

[66] Z. Bern, L.J. Dixon and D.A. Kosower, *One loop corrections to five gluon amplitudes*, *Phys. Rev. Lett.* **70** (1993) 2677 [hep-ph/9302280] [INSPIRE].

[67] E. Byckling and K. Kajantie, *Particle Kinematics: (Chapters I-VI, X)*, University of Jyvaskyla, Jyvaskyla Finland (1971).

[68] S. Frixione, Z. Kunszt and A. Signer, *Three jet cross-sections to next-to-leading order*, *Nucl. Phys. B* **467** (1996) 399 [hep-ph/9512328] [INSPIRE].

[69] R. Frederix, S. Frixione, F. Maltoni and T. Stelzer, *Automation of next-to-leading order computations in QCD: The FKS subtraction*, *JHEP* **10** (2009) 003 [arXiv:0908.4272] [INSPIRE].

[70] M. Cacciari, G.P. Salam and G. Soyez, *The anti-k_t jet clustering algorithm*, *JHEP* **04** (2008) 063 [arXiv:0802.1189] [INSPIRE].

[71] M. Cacciari, G.P. Salam and G. Soyez, *FastJet User Manual*, *Eur. Phys. J. C* **72** (2012) 1896 [arXiv:1111.6097] [INSPIRE].

[72] S. Frixione, *Isolated photons in perturbative QCD*, *Phys. Lett. B* **429** (1998) 369 [hep-ph/9801442] [INSPIRE].

[73] Particle Data Group collaboration, *Review of Particle Physics*, *PTEP* **2020** (2020) [https://academic.oup.com/ptep/article-pdf/2020/8/083C01/34673740/rpp2020-vol2-2015-2092_18.pdf].

[74] F. Chollet et al., *Keras*, (2015) https://github.com/fchollet/keras.

[75] M. Abadi et al., *TensorFlow: Large-scale machine learning on heterogeneous systems*, (2015) https://www.tensorflow.org/.

[76] D.P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, arXiv:1412.6980 [INSPIRE].

[77] I. Goodfellow, Y. Bengio and A. Courville, *Deep Learning*, MIT Press (2016).

[78] G. Guennebaud et al., *Eigen v3*, (2010) http://eigen.tuxfamily.org.

[79] T. Binoth et al., *A Proposal for a Standard Interface between Monte Carlo Tools and One-Loop Programs*, *Comput. Phys. Commun.* **181** (2010) 1612 [arXiv:1001.1307] [INSPIRE].

[80] S. Alioli et al., *Update of the Binoth Les Houches Accord for a standard interface between Monte Carlo tools and one-loop programs*, Comput. Phys. Commun. **185** (2014) 560 [arXiv:1308.3462] [INSPIRE].

[81] A. Buckley et al., *Rivet user manual*, Comput. Phys. Commun. **184** (2013) 2803 [arXiv:1003.0694] [INSPIRE].

[82] C. Bierlich et al., *Robust Independent Validation of Experiment and Theory: Rivet version 3*, SciPost Phys. **8** (2020) 026 [arXiv:1912.05451] [INSPIRE].

[83] ATLAS collaboration, *Measurements of integrated and differential cross sections for isolated photon pair production in pp collisions at $\sqrt{s} = 8$ TeV with the ATLAS detector*, Phys. Rev. D **95** (2017) 112005 [arXiv:1704.03839] [INSPIRE].

[84] R. Kleiss, W.J. Stirling and S.D. Ellis, *A New Monte Carlo Treatment of Multiparticle Phase Space at High-energies*, Comput. Phys. Commun. **40** (1986) 359 [INSPIRE].

[85] G.P. Lepage, *VEGAS: An Adaptive Multidimensional Integration Program*, (1980) [INSPIRE].

[86] T. Ohl, *Vegas revisited: Adaptive Monte Carlo integration beyond factorization*, Comput. Phys. Commun. **120** (1999) 13 [hep-ph/9806432] [INSPIRE].

[87] A. Buckley et al., *LHAPDF6: parton density access in the LHC precision era*, Eur. Phys. J. C **75** (2015) 132 [arXiv:1412.7420] [INSPIRE].

[88] NNPDF collaboration, *Parton distributions from high-precision collider data*, Eur. Phys. J. C **77** (2017) 663 [arXiv:1706.00428] [INSPIRE].

[89] Z. Bern et al., *Ntuples for NLO Events at Hadron Colliders*, Comput. Phys. Commun. **185** (2014) 1443 [arXiv:1310.7439] [INSPIRE].

[90] D. Maître, G. Heinrich and M. Johnson, *N(N)LO event files: applications and prospects*, PoS **LL2016** (2016) 016 [arXiv:1607.06259] [INSPIRE].

[91] V. Nair and G.E. Hinton, *Rectified linear units improve restricted boltzmann machines*, in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, Omnipress, Madison, WI, U.S.A. (2010) DOI.

[92] f G. Van Rossum and F.L. Drake, *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA U.S.A. (2009).

[93] J.D. Hunter, *Matplotlib: A 2D Graphics Environment*, Comput. Sci. Eng. **9** (2007) 90 [INSPIRE].

[94] C.R. Harris et al., *Array programming with NumPy*, Nature **585** (2020) 357 [arXiv:2006.10256] [INSPIRE].

[95] The pandas development team, *pandas-dev/pandas: Pandas*, (2020) https://doi.org/10.5281/zenodo.3509134.

[96] W. McKinney, *Data Structures for Statistical Computing in Python*, in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman eds. (2010) DOI.