Domain-Adversarial Graph Neural Networks for Λ Hyperon Identification with CLAS12

M. McEneaney[a,*], A. Vossen[a,b]

[a]*Physics Department, Duke University, Durham, 27708, NC, USA*
[b]*Thomas Jefferson National Accelerator Facility, Newport News, 23606, VA, USA*

Abstract

Machine learning methods and in particular Graph Neural Networks (GNNs) have revolutionized many tasks within the high energy physics community. We report on the novel use of GNNs and a domain-adversarial training method to identify Λ hyperon events with the CLAS12 experiment at Jefferson Lab. The GNN method we have developed increases the purity of the Λ yield by a factor of 1.95 and by 1.82 using the domain-adversarial training. This work also provides a good benchmark for developing event tagging machine learning methods for the Λ and other channels at CLAS12 and other experiments, such as the planned Electron Ion Collider.

Keywords: Graph Neural Networks, Domain-Adversarial Neural Networks, Λ Hyperon, Event Tagging, CLAS12

1. Introduction

The internal dynamics of the nucleon are governed by the strong interaction, one of the four fundamental forces in nature, and present a significant and yet unconquered intellectual challenge. As the name suggests, the strong interaction creates a strongly coupled system within the nucleon making it extremely difficult to calculate. Fortunately, due to the asymptotic freedom of the strong interaction at high energies, one way study these interactions is through scattering events where subatomic particles such as electrons and protons are accelerated to high energies and collided together. The momentum and trajectories of particles produced in these interactions may be measured with a variety of detector systems. From this final state information, one may begin to infer the original dynamics of the nucleon before the collision.

The Λ hyperon is typically detected via the two-body decay $\Lambda \to p\pi^-$. It is of particular interest because the polarization of the Λ is preserved in the cross-section of the decay protons, allowing one to infer information about the spin structure of the Λ. One may identify the Λ signal by looking at the invariant mass spectrum of the proton pion ($p\pi^-$) pairs for a peak around the nominal Λ mass 1.1157 GeV. However, at CLAS12 this becomes complicated because of the large background from non-strange final states and combinatorics. One of the goals of this study has been to develop a method of increasing the Λ signal to background ratio while still preserving the signal shape in order to be able to extract the Λ yield from a fit to the signal.

Graph Neural Networks (GNNs) have been used to great efficacy for event classification, for example at the Large Hadron Collider (LHC) for tagging jet types using the four-momenta of the constituent particles [1]. GNNs build on the concept of a neural network by treating each node of the input graph as an input channel of a neural network layer. Node values in subsequent layers are obtained by computing a convolution over neighboring nodes within the current layer. GNNs are well-suited to event-level classification tasks because they are permutation invariant with respect to the order of the input particles and they can handle a variably sized input. GNNs also rely on the correlations between the nodes of the input graph which makes them a natural choice for identifying events based on the products of a decay.

1.1. Physics Observables

Semi-Inclusive Deep Inelastic Scattering (SIDIS) [2] events occur when a high-energy lepton ℓ collides with a target nucleon N and the scattered lepton and at least one hadron are detected in the final state. SIDIS provides one of the ideal laboratories in which to probe

---

*Corresponding author
Email addresses: `matthew.mceneaney@duke.edu` (M. McEneaney), `anselm.vossen@duke.edu` (A. Vossen)
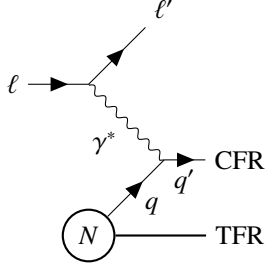
Figure 1: Semi-Inclusive Deep Inelastic Scattering

the internal dynamics of the nucleon. The hadrons may be produced from the struck quark q', i.e. in the Current Fragmentation Region (CFR), or from the remnant quarks, i.e. in the Target Fragmentation Region (TFR). Characteristics of final-state hadrons and correlations between them can reveal information about the spin and momentum structure of the nucleon. Some relevant kinematic variables for this study include the following:

$$Q^2 = -q^2 \qquad (1)$$

$$W^2 = (P + q)^2 \qquad (2)$$

$$\nu = E - E' \qquad (3)$$

$$x = \frac{Q^2}{2M\nu} \qquad (4)$$

$$y = \frac{P \cdot q}{P \cdot \ell} \qquad (5)$$

$$z = \frac{P \cdot P_h}{P \cdot q} \qquad (6)$$

$$x_F = \frac{2P_h \cdot q}{W|q|}. \qquad (7)$$

The virtual photon γ^* from the interaction of the incoming lepton ℓ with the struck quark imparts a momentum transfer $\ell - \ell'$ with virtuality Q^2. The incoming lepton has energy E and the outgoing lepton ℓ' has energy E' such that ν is the difference in energy between ℓ and ℓ' and y is the fraction of the incoming lepton energy transferred to the virtual photon. The total energy in the $\gamma^* N$ center of mass frame is W and the fraction of the nucleon momentum carried by the struck quark is x. A final-state hadron h carries a fraction z of the virtual photon energy and a fraction x_F, the so-called

x-Feynman variable, of the hadron's longitudinal momentum in the virtual photon direction relative to the maximum value $W/2$ allowed by momentum conservation in the struck-quark center of mass frame. x_F is used to distinguish between the CFR and TFR since it favors positive values for hadrons travelling in the direction of the incoming virtual photon and negative values for backward-going hadrons.

1.2. The CLAS12 Experiment

In order to realize these high energy scattering events the Continuous Electron Beam Accelerator Facility (CEBAF) at Jefferson Lab delivers a high luminosity polarized electron beam to four experimental halls for fixed target experiments [3]. The CLAS12 (CEBAF Large Acceptance Spectrometer 12 GeV) Experiment is located in the experimental Hall B and provides excellent momentum and angular coverage and good particle identification capabilities for both charged and neutral particles produced in high energy electron-proton or electron-deuteron scattering events [4]. The detector is centered around two large magnets, a solenoid in the central region of the detector and a torus magnet in the forward region. The torus magnet is operated in two different configurations, either bending negatively charged particles in toward the beamline (inbending configuration), or out away from the beamline (outbending configuration).

2. Data

The data used in this study was all taken during the fall 2018 run period in the outbending toroidal configuration with a 10.6 GeV polarized electron beam and an unpolarized liquid hydrogen target. Events were required to have an identified scattered electron (e^-) and a proton-pion ($p\pi^-$) pair in the reconstruction. Further restrictions on the scattered electron were that it be the trigger particle, be detected in the forward region of the detector, and have the highest momentum of all electrons in the event. The scattered electron was also required to have a particle identification (PID) assignment quality factor estimate $|\chi^2| < 3$. Note that the χ^2 PID quality factor is not an actual χ^2 statistic value although it has the same name. For electrons it is related to the difference of the energy deposition from an expected value estimated from the measured momentum, and for charged hadrons it is related to the difference from the expected vertex time. No estimate is given for neutral particles. Kinematic cuts were $Q^2 > 1$ GeV^2, $W > 2$ GeV, $y < 0.8$, $z_{p\pi^-} < 1$ and $x_F > 0$ for the

$p\pi^-$ pair to select Λs produced in the CFR. Additionally, the invariant mass of the $p\pi^-$ pair was restricted to $M_{p\pi^-} < 1.24$ GeV so that events were sufficiently close to the Λ mass peak.

2.1. Monte Carlo Data Set

The sample of Monte Carlo (MC) simulation events was produced with the same run configuration as the actual data from the fall 2018 run period. Events were generated with an MC algorithm based on Pepsi Lund program [5]. For the purpose of training our neural networks, Λ signal events were identified as those that contained a $\Lambda \to p\pi^-$ decay in the MC truth. A comparison of the full invariant mass spectrum to the spectrum of signal events is shown in Fig. 2.

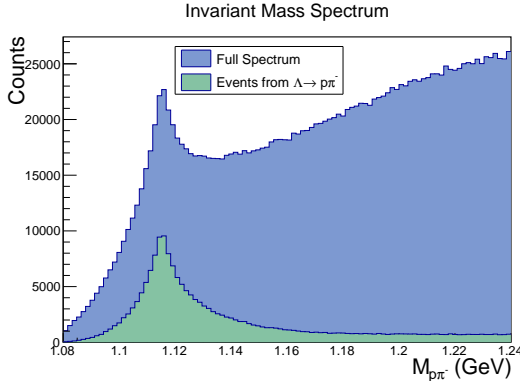Figure 2: The MC invariant mass spectrum for reconstructed $p\pi^-$ pairs is shown for both the full set of events and signal events with a Λ decay in the MC truth.

2.2. Signal Extraction

The invariant mass spectrum of all reconstructed $p\pi^-$ pairs passing kinematic cuts is shown in Fig. 3 for both the real data and MC simulation. A peak around the nominal Λ mass $M = 1.1157$ GeV is apparent, but the background contribution is very high especially in data. The main background contributions come from the $\Delta^0 \to p\pi^-$ decay which peaks at 1.232 GeV and from combinatorics of random $p\pi^-$ pairs. A Crystal Ball [6] signal over a quadratic background was used to model the invariant mass spectrum. The Crystal Ball function consists of a Gaussian peak with a power-law tail such that both the function and its first derivative are continuous

$$f(x; \alpha, n, \mu, \sigma) = N \begin{cases} \exp{-\frac{(x-\mu)^2}{2\sigma^2}}, & \frac{x-\mu}{\sigma} > \alpha \\ A(B - \frac{x-\mu}{\sigma})^{-n}, & \frac{x-\mu}{\sigma} \leq \alpha \end{cases} \quad (8)$$

where N, A, and B are constants defined by the function parameters and satisfying the continuity requirements and normalization. The signal is much more pronounced in MC but the fit parameters were similar up to scale factors between MC and data. The number of signal events N_{sig} is estimated using the histogram counts over the fitted background function within a $\pm 2\sigma$ region around the signal mean μ from the fit. The histogram counts are used instead of the fit function integral since the histogram has some noticeable excess above the fit function at the signal peak. The signal fit region limits have been optimized so as to maximize the signal purity and minimize the signal error.
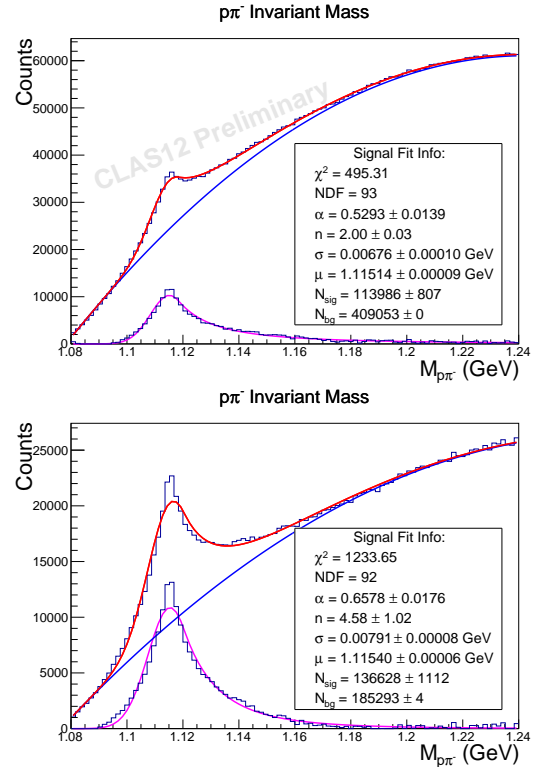

Figure 3: A Crystal Ball signal over a quadratic background fit function was used to extract the Λ signal statistics from data (top) and MC (bottom). The number of signal events N_{sig} is estimated using the histogram counts over the fitted background function (small outlined histogram).

3. Machine Learning Methods

The following sections assume a basic familiarity with supervised learning and neural networks; however, a comprehensive introduction may be found in Ref. [7].

3.1. Graph Neural Networks

GNNs are well suited to many physics applications since they are able to take inputs with irregular and unordered structure, e.g. unordered sets of varying numbers of final-state particles in a SIDIS event. A graph may be described as a set of vertices V and edges E. At the simplest level, GNNs operate similarly to Convolutional Neural Nets (CNNs) by taking a convolution of over the neighborhood of each node to compute its representation in the following layer as depicted in Fig. 4. CNNs take a rectangular input such as a two-dimensional pixel array and use an aggregation of the information from the neighboring pixels to compute the representation a pixel in the next layer. GNN convolution works the same way except that the neighbors are determined by the structure of the input graph, which may be highly irregular. In fact, CNNs are a subclass of GNNs since a rectangular array is in effect just a graph with a regular rectangular structure.

Specifically, to update the graph representation in the kth layer of a network, the representation $h_m^{(k)}$ of a given node or edge m is formed from some aggregation function A of the information contained in the neighboring nodes and edges in the previous layer.

$$h_m^{(k)} = A(\{h_n^{(k-1)} : n \in \mathcal{N}\}). \qquad (9)$$

The aggregation function must not depend on the order of the inputs and must be able to handle a variable number of arguments, e.g. a sum, maximum, or minimum function. The set of neighbors $\mathcal{N}$ of a node m is defined by the structure of the input graph and may include the node m itself. A more detailed discussion may be found in Ref. [8]. Using this concept of graph convolution, many different types of convolutional blocks may be formed, and these blocks may be combined to construct the different layers of a GNN.

3.1.1. Graph Isomorphism Networks

Graph Isomorphism Networks (GINs) are theoretically the most powerful form of GNNs and have been quite successful at graph classification tasks [9]. GINs essentially add two important modifications to the graph convolution algorithm. After the aggregation step over the neighboring nodes in the graph, a Multi-Layer Perceptron (MLP) is introduced so that on the kth iteration the updated node $h^{(k)}$ becomes

$$h_m^{(k)} = MLP((1 + \epsilon) \cdot h_m^{(k-1)} + A(\{h_n^{(k-1)} : n \in \mathcal{N}\})), \quad (10)$$

where ϵ is an optional learnable parameter and $\mathcal{N}$ is the set of all neighbors of node h_m [9].

The second modification is that in the final layer the hidden representations of the graph from each layer of the GIN are pooled together to obtain the final representation. The reasoning for this type of pooling is that localized information can be lost deeper in the network. Pooling over all previous representations gives a good compromise between the localized information contained within the initial layers and more generalized information reached in later ones.

3.2. Domain Adversarial Networks

Supervised learning relies on the training sample being representative of actual data. With SIDIS events, this means the simulation events should closely match the real data events intended for classification. However, simulation cannot perfectly match data for all physics channels. Thus, performance on real data may not live up to performance on MC test events.

One way to overcome this without trying to completely re-tune the simulation is to use a domain-adversarial training routine first introduced in [10]. If there is some shared phase space between the training sample and the real data another training objective may be introduced. In addition to minimizing the loss of the classifier, one also maximizes the loss of an additional head network on top of the base network (in this case the GNN) which is simultaneously trained to distinguish between training data and data from the real test domain. This forces the base network to converge towards a hidden representation that relies only on features common to both training and test domains.

In practice this means that there are three networks: a base network which produces a latent space representation of the input and then two adversarial head networks. The first head network, the discriminator, takes the latent space representation from the base network and makes a prediction whether the input came from the labelled training dataset or the unlabelled test dataset. Then the loss is computed between those predictions and the actual domains from which the inputs came. The discriminator loss L_D is back-propagated through the discriminator to update its weights. The second head network, the classifier, then takes the latent space representation from the base network and makes a prediction for our end classification task. The classifier loss L_C is back-propagated through the classifier, updating its weights. Finally, the loss of the discriminator and the classifier are back-propagated through the base network with a **reversed** gradient from the discriminator

$$\frac{\partial L_{total}}{\partial x} = -\alpha \frac{\partial L_D}{\partial x} + \frac{\partial L_C}{\partial x}. \qquad (11)$$
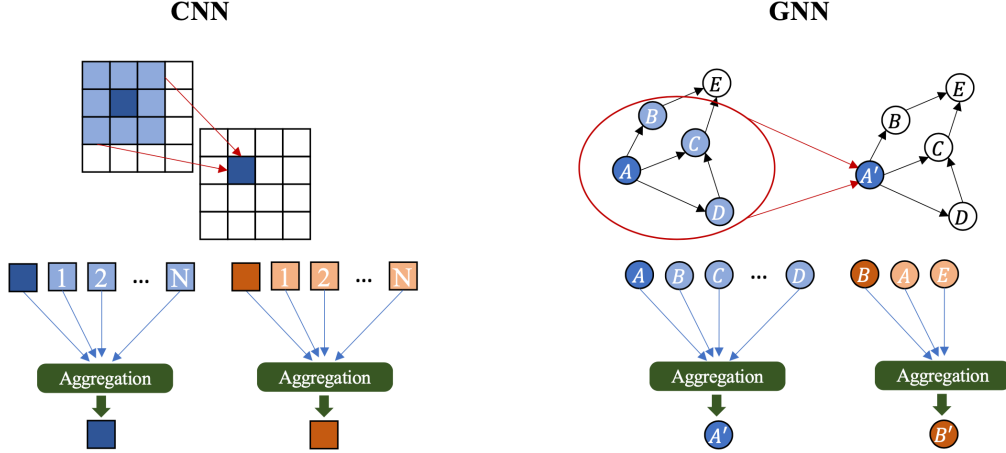
CNN

GNN

Figure 4: Convolution in a GNN is analogous to the convolution step in a CNN except that the neighbors of each node are defined by the structure of the graph.

The reversed gradient from the discriminator loss acts as a penalizing term for the base network so that it is biased toward representations which the discriminator is unable to distinguish. Here, α is another hyperparameter of the training, a coefficient that can depend on the training epoch or iteration. After training, the discriminator is discarded and the network consists solely of the base network and the classifier. The domain-adversarial process is depicted in Fig. 5.

3.3. Network Training and Evaluation

Our training sample consisted of 140k events from the MC outbending sample filtered so that they each contained an identified p, π^-, and scattered e^- in the reconstructed event. The sample consisted of a random sample of half MC truth Λ signal events and half MC truth background. Signal events were identified as those events with a $\Lambda \rightarrow p\pi^-$ decay in the MC truth, and all other events were marked as background. The real data sample for the domain-adversarial approach consisted of roughly 140k events from the fall 2018 outbending runs. The samples were split 80% for training, 10% for model validation, and 10% for model testing.

3.3.1. Data Preprocessing

The same preprocessing routine was applied to both MC and real data. Events were mapped into fully connected graphs containing all the particles from the reconstructed event. Each node in the graph represented a particle and its data consisted of the reconstructed transverse momentum p_T, polar and azimuthal angles θ and ϕ, velocity β, PID, χ^2 estimate for the PID assignment,

and detector status. The continuous quantities p_T, θ, ϕ, and β first had the event mean for that variable subtracted from them and were then normalized to the maximum difference from the event mean. The large default values for unassigned PID χ^2 estimates were set to +10 since these sometimes correspond to hadrons for which the χ^2 estimate is not required. Then, the PID χ^2 estimate was cut to be within the range $[-10, 10]$ and normalized. Discrete quantities, particle PID and detector status, were reassigned to arbitrary float values from -1 to 1. The PID reassignment values are listed in Table 1. For charged particles the replacement value had the same sign as the charge of the particle and particle-antiparticle pairs had identical magnitudes. Otherwise replacement value choice was arbitrary. Detector status values are confined within $(-5000, 5000)$ so the status was just normalized by 5000. The renormalization of inputs to values in $[-1, 1]$ is a standard step in the machine learning process to avoid large gradient values during optimization.

True PID	Replacement value
22	0.0
± 11	∓ 1.0
± 2212	± 0.8
2112	0.5
111	0.1
± 211	± 0.6
311	0.3
± 321	± 0.4

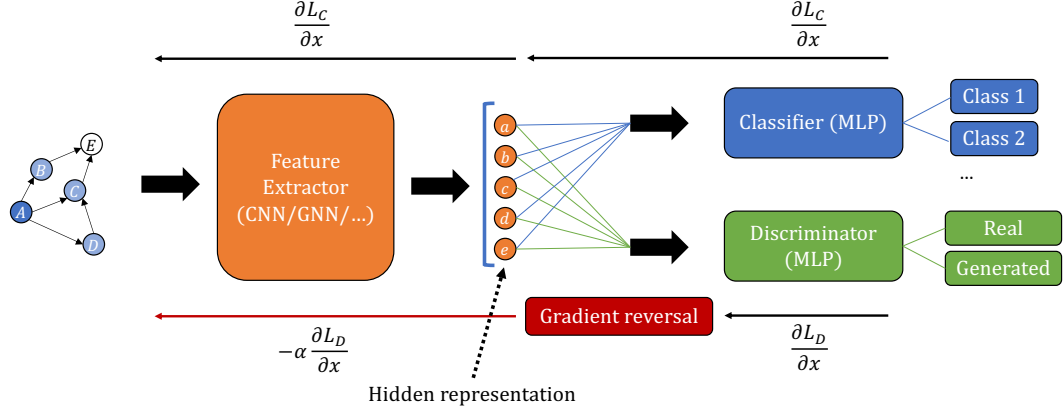Table 1: PID replacement values used to renormalize discrete PID values.

Figure 5: Domain-adversarial network optimization process

3.3.2. Model Architecture

The default architecture for the GIN consists of a 5 layer network with each MLP having 2 layers and a hidden dimension $hdim = 64$. The default final dropout rate is 0.5 and the default graph and node pooling functions are a simple mean. These architecture parameters (except pooling functions) were varied for the hyperparameter optimization search. Each MLP layer used a ReLU activation function was followed with a batch normalization layer to reduce overfitting. The final classification layer obviously only had $hdim = 2$ corresponding to our two classification categories signal and background and was followed with a sigmoid activation function. Final classification assignments were normalized to probabilities with the softmax function

$$\sigma(x) = \frac{e^{x_i}}{\sum_i e^{x_i}}. \tag{12}$$

For the Domain-Adversarial GIN (DAGIN) the GIN architecture was the same except the dimension of the final layer still had the same hidden dimension as the previous layers. The classifier and discriminator had identical architectures: each was an MLP with the same hidden dimension as the GIN in each layer except the final classification layer. Each layer used ReLU activation except the final layer which used sigmoid activation. Final classification predictions were also normalized with the softmax function from Eq. 12. The hyperparameters left for optimization for the classifier and discriminator architecture were just the number of layers in each network.

3.3.3. Training Parameters

The ADAM optimizer [11] was used to train each model but the choice of batch size and learning rate η was left for the hyperparameter optimization process. The loss function for each model was a cross-entropy loss, except the discriminator which used binary cross-entropy loss. For the DAGIN the learning rates of the three different networks base, classifier, and discriminator (η, η_C, η_D) were left as separate hyperparameters for optimization.

3.3.4. Hyperparameter Optimization

A Bayesian optimization search of the hyperparameter phase space was carried out using the Optuna Hyperparameter Optimization package [12]. Optuna's builtin *TPESampler* was used which is based on the Tree-structured Parzen Estimator (TPE) algorithm [13]. The ROC (Receiver Operating Characteristic) curve is a plot of the fraction of true positives versus the fraction of false positives as a function of the threshold value for a binary classification task. Hyperparameter choice was optimized based on the Area Under the (ROC) Curve (AUC) metric of trained model when evaluated on the reserved test data. The more area under this curve, i.e. the higher the AUC, the less dependence there is on the threshold value and the more robust the model is. This metric is preferred instead of a simple accuracy since it allows one to simultaneously maximize efficiency and purity for a binary classification task.

The hyperparameter phase space allowed for the optimization search is listed for the basic GIN model in Table 2 and for the DAGIN in Table 3. Roughly 1500 trials were performed for each hyperparameter search and each trial model was trained for up to 50 epochs. The

training and validation metrics as a function of training epoch for the GIN and DAGIN are shown in Figs. 6 and 7 respectively.

Hyperparameter	Range
nlayers	$[2, 8]$
hdim	$[2^5, 2^8]$
MLP nlayers	$[2, 4]$
dropout	$[0.5, 0.8]$
batch	$[2^6, 2^8]$
η	$[10^{-5}, 10^{-2}]$

Table 2: Hyperparameter ranges for basic GIN optimization search.

Hyperparameter	Range
nlayers	$[2, 8]$
Classifier/Discriminator nlayers	$[2, 8]$
hdim	$[2^5, 2^8]$
Classifier/Discriminator hdim	$[2^5, 2^8]$
MLP nlayers	$[2, 4]$
dropout	$[0.5, 0.8]$
batch	$[2^6, 2^8]$
η	$[10^{-5}, 10^{-2}]$
η_C	$[10^{-5}, 10^{-2}]$
η_D	$[10^{-5}, 10^{-2}]$
α	$[1, 100]$

Table 3: Hyperparameter ranges for DAGIN optimization search. α is the coefficient for the discriminator loss in the gradient-reversal layer.

As a sanity check, this same hyperparameter optimization routine was run with similar datasets filtered so that the MC background events came from varying fractions of the two main background contributions: combinatorial background and $\Delta \to p\pi^-$ decays. The performance achieved with these datasets was essentially the same as that achieved on the original dataset, demonstrating that there was not any wild dependence of the training process on the composition of the background sample.

4. Results

4.1. Evaluation on Monte Carlo

After training, the performance of each model was evaluated on the reserved 10% of the MC dataset. The invariant mass spectrum of the GNN-identified signal events was fit to the same Crystal Ball signal over quadratic background function to determine if the signal shape was changed at all by the application of the GNN. Note that two of the fit parameters for the background
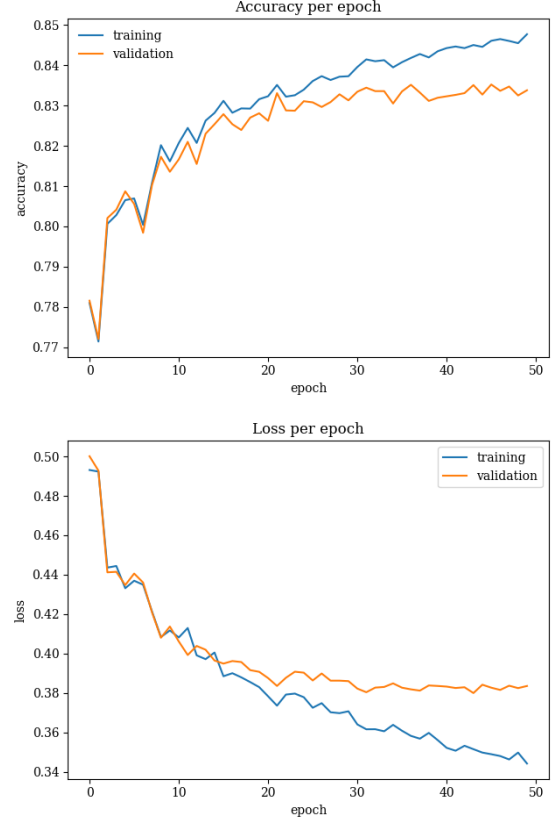

Figure 6: Training accuracy (top) and loss (bottom) vs. epoch for the hyperparameter-optimized GIN.

quadratic, those describing the peak height and location of the parabola, were held constant and only the curvature parameter of the background was varied for the fit. Both the GIN and DAGIN roughly preserve the signal shape as shown in Fig. 8 for the optimized models. There is some increase in the σ parameter of the Crystal Ball fit function, particularly for the DAGIN. However, this is mainly due to a loss in signal efficiency centered around the signal peak. This eliminates the slight excess above the fit function at the signal peak evident in Fig. 3 allowing the σ parameter to increase without actually significantly impacting the width of the signal. The efficiency of the network on background events is fairly stable as a function of the invariant mass as shown in Fig. 10. Employing the decorrelation method described in Ref. [14] as a sanity check to decorrelate the performance of the network from the invariant mass produced similar results. Efficiencies of the GIN and DAGIN networks trained with the decorrelation method are shown in Fig. 9.
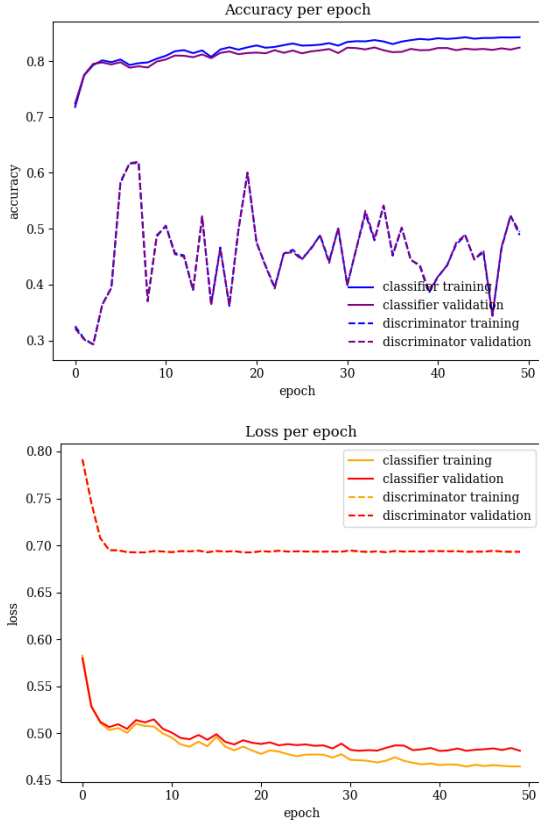
Figure 7: Training accuracy (top) and loss (bottom) vs. epoch for the hyperparameter-optimized DAGIN.
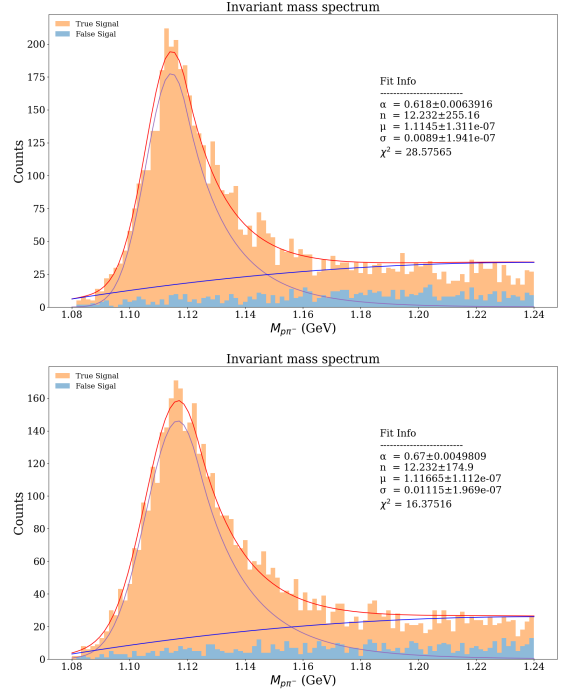

Figure 8: Signal mass spectra identified by the hyperparameter-optimized GIN (top) and DAGIN (bottom) with signal extraction fits applied. The stacked histograms are separated into true signal (orange) and false signal (blue).

4.2. Evaluation on Data

The figure of merit ($FOM = N_{sig}/\sqrt{N_{tot}}$) used to evaluate the performance on data is calculated within the $\pm 2\sigma$ region around the signal peak location μ from the fit to the mass spectrum of events the GNN identifies as signal. Maximizing this FOM should simultaneously maximize the purity N_{sig}/N_{tot} while minimizing the error, $1/\sqrt{N_{tot}}$ assuming Poissonian statistics. The FOM and purity were computed for various values of the cut on the GNN output in order to maximize the FOM and purity. The GNN output distributions on data are shown in comparison with the distributions on MC in Fig. 11, and the scans of the cut values are shown in Fig. 12 for both GIN and DAGIN. Note that the output probability from the DAGIN is taken just after the final classification layer without applying the softmax function. The maximal FOMs and corresponding purities obtained are listed in Table 4. The invariant mass spectra after the application of the optimized GIN and DAGIN respectively with maximized FOM are shown in Fig. 13. The signal shape is still relatively well preserved by the application

of either the GIN or DAGIN. The Kolmogorov-Smirnov distances between the NN output distributions and data and MC are also computed. These values are 0.262 and 0.217 for the GIN and DAGIN respectively.

Method	No GNN	GIN	DAGIN
$N_{sig}/\sqrt{N_{tot}}$	34.11	65.74	42.09
N_{sig}/N_{tot}	0.195	0.382	0.354
NN cut	N/A	0.13	0.22

Table 4: Figure of Merit from evaluation on data for all three methods.

5. Conclusions

We have developed a GNN architecture for tagging Λ hyperon events that reaches up to 83% test accuracy and 0.9 AUC values after training. The method also roughly preserves the Λ signal shape in both data and simulation. While there remains room for further improvement, already the GIN model increases the Λ signal purity by a factor of 1.95. Interestingly, the DAGIN model only increases the purity by a factor of 1.82. This could occur if the features that the domain-adversarial
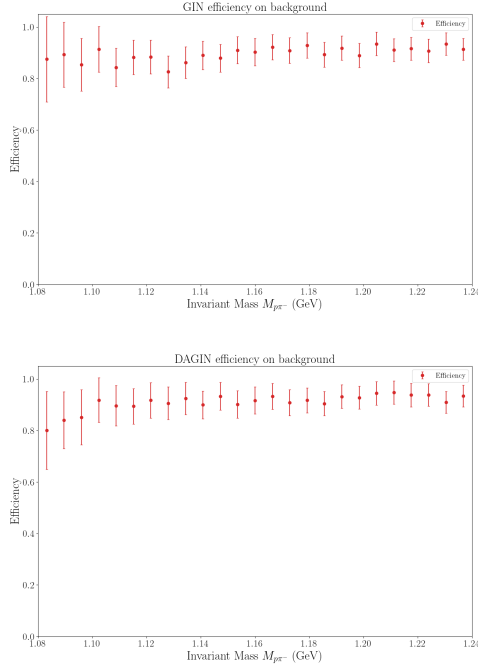
Figure 9: Efficiency of the hyperparameter-optimized GIN (top) and DAGIN (bottom) on background events as a function of the invariant mass. Errors are solely statistical.
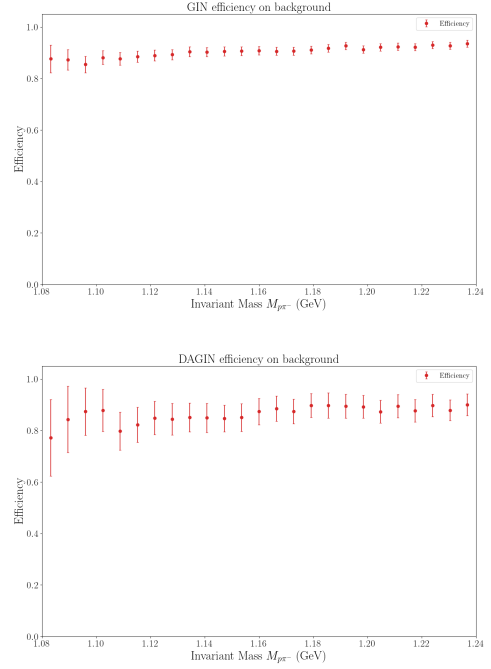

Figure 10: Efficiency of the GIN (top) and DAGIN (bottom) using the *MoDeLoss* decorrelation method on background events as a function of the invariant mass. Errors are solely statistical.

method forces the network to use are not as discriminating as those used by the GIN even if they are common to both datasets. However, the DAGIN model produces a more consistent output between data and MC with a Kolmogorov-Smirnov statistic of 0.217 compared to the GIN with a statistic of 0.262. The methods developed here may also be applied to other channels at CLAS12 and provide a baseline for future efforts using GNNs in this area. Another potential use case could be extending these networks to filter out feed-down contributions to the Λ signal from decays of heavier hyperons, e.g. $\Sigma^0 \to \Lambda\gamma$. This would be especially useful at the planned Electron Ion Collider where feed-down is expected to be more of an issue [15]. We plan to further check this method by evaluating how much it improves Λ longitudinal spin transfer measurements at CLAS12. Future improvements include exploring other network architectures, specifically Subgraph Networks, which are theoretically even more powerful than GINs [16] and expanding the input data to the graphs to include detector level information.

References

[1] P. T. Komiske, E. M. Metodiev, J. Thaler, Energy flow networks: deep sets for particle jets, Journal of High Energy Physics 2019 (1) (Jan 2019). doi:10.1007/jhep01(2019)121.

[2] C. A. Aidala, S. D. Bass, D. Hasch, G. K. Mallot, The spin structure of the nucleon, Reviews of Modern Physics 85 (2) (2013) 655–691. doi:10.1103/revmodphys.85.655.
URL https://doi.org/10.1103%2Frevmodphys.85.655

[3] C. W. Leemann, D. R. Douglas, G. A. Krafft, The continuous electron beam accelerator facility: CEBAF at the Jefferson Laboratory, Annual Review of Nuclear and Particle Science 51 (1) (2001) 413–450. arXiv:https://doi.org/10.1146/annurev.nucl.51.101701.132327, doi:10.1146/annurev.nucl.51.101701.132327.
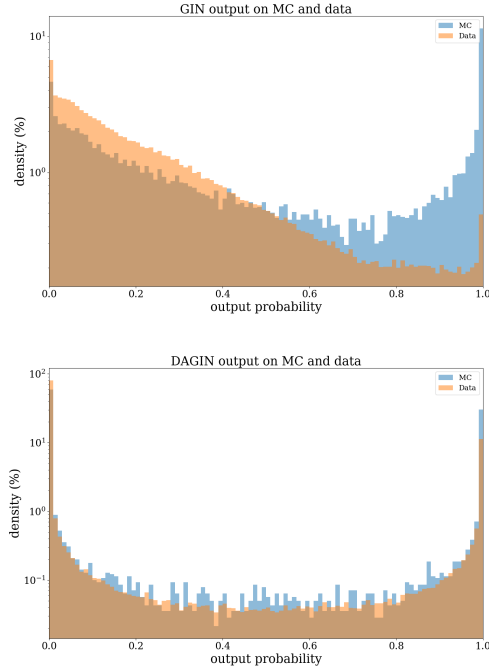URL https://doi.org/10.1146/annurev.nucl.51.101701.132327

Figure 11: Output distributions on real data and MC for the GIN (top) and DAGIN (bottom). The NN output probability spectrum is restricted within [0.0, 1.0] where 0.0 corresponds to a background event and 1.0 corresponds to a Λ signal event.
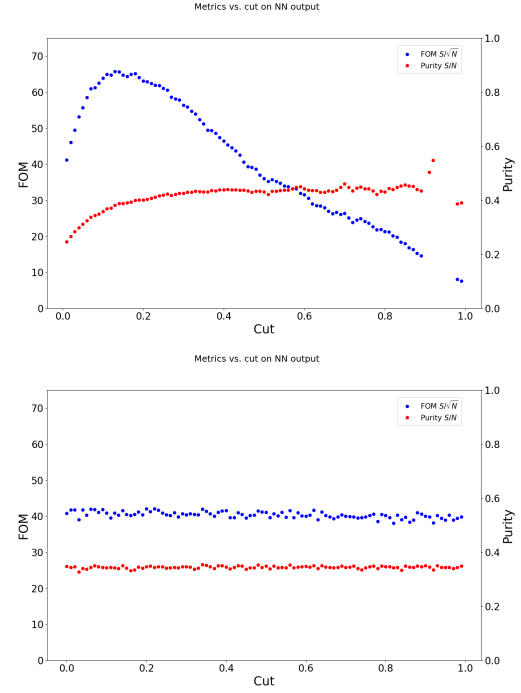
Figure 12: Scans of the FOM (blue) and purity (red) from the signal over background fits on data as a function of the cut on the GNN output for the GIN (top) and DAGIN (bottom). All events with GNN output greater than the cut value are identified as signal events.

[4] V. D. Burkert, et al., The CLAS12 spectrometer at Jefferson Laboratory, Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment 959 (2020) 163419. doi:https://doi.org/10.1016/j.nima.2020.163419. URL https://www.sciencedirect.com/science/article/pii/S0168900220300243

[5] L. Mankiewicz, A. Schäfer, M. Veltri, Pepsi — a monte carlo generator for polarized leptoproduction, Computer Physics Communications 71 (3) (1992) 305–318. doi:https://doi.org/10.1016/0010-4655(92)90016-R. URL https://www.sciencedirect.com/science/article/pii/001046559290016R

[6] J. E. Gaiser, E. Bloom, F. Bulos, G. Godfrey, C. Kiesling, W. Lockman, M. Oreglia, D. L. Scharre, C. Edwards, R. Partridge, C. Peck, F. C. Porter, D. Antreasyan, Y. F. Gu, J. Irion, W. Kollman, K. Strauch, K. Wacker, A. Weinstein, D. Aschman, T. Burnett, M. Cavalli-Sforza, D. Coyne, C. Newman, H. F. W. Sadrozinski, D. Gelphman, R. Hofstadter, R. Horisberger, I. Kirkbride, H. Kolanoski, K. Königsmann, R. Lee, A. Liberman, J. O'Reilly, A. Osterheld, B. Pollock, J. Tompkins, Charmonium spectroscopy from inclusive ψ' and j/ψ radiative decays, Phys. Rev. D 34 (1986) 711–721. doi:10.1103/PhysRevD.34.711. URL https://link.aps.org/doi/10.1103/PhysRevD.34.711

[7] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015, available at: http://neuralnetworksanddeeplearning.com.

[8] P. W. Battaglia, et al., Relational inductive biases, deep learning, and graph networks (2018). doi:10.48550/ARXIV.1806.01261.

[9] K. Xu, W. Hu, J. Leskovec, S. Jegelka, How powerful are graph neural networks? (2019). arXiv:1810.00826.

[10] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks (2015). doi:10.48550/ARXIV.1505.07818.

[11] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). arXiv:1412.6980.

[12] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2019.

[13] J. Bergstra, R. Bardenet, Y. Bengio, B. Kégl, Algorithms for hyper-parameter optimization, in: J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, K. Weinberger (Eds.), Advances in Neural Information Processing Systems, Vol. 24, Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

[14] O. Kitouni, B. Nachman, C. Weisser, M. Williams, Enhancing searches for resonances with machine learning and moment decomposition, JHEP 21 (2020) 070. arXiv:2010.09745, doi:10.1007/JHEP04(2021)070.

[15] Z.-B. Kang, J. Terry, A. Vossen, Q. Xu, J. Zhang, Transverse Lambda production at the future Electron-Ion Collider, Physical Review D 105 (9) (may 2022). doi:10.1103/physrevd.105.094033. URL https://doi.org/10.1103/physrevd.105.094033
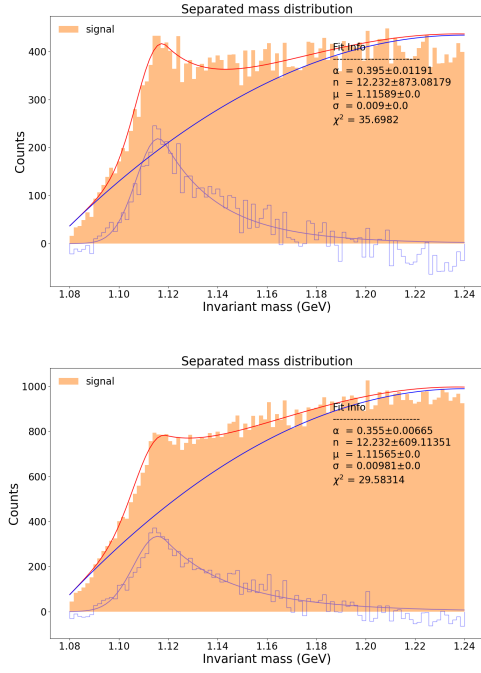
Figure 13: The same Crystal Ball signal over quadratic background fit was applied to the NN-identified signal mass spectrum (pale orange histogram) for the GIN (top) and DAGIN (bottom). N_{sig} is estimated using the histogram counts over the fitted background function (small outlined histogram).

[16] F. Frasca, B. Bevilacqua, M. M. Bronstein, H. Maron, Understanding and extending subgraph GNNs by rethinking their symmetries (2022). doi:10.48550/ARXIV.2206.11140. URL https://arxiv.org/abs/2206.11140