Interpreting Higgs Boson Interaction Network with Layerwise Relevance Propagation

Alex Y. Luo and Yue Xiao
University of California San Diego, La Jolla, California 92093, USA

(Dated: March 4, 2021)

ABSTRACT: While graph interaction networks achieve exceptional results in Higgs boson identification, GNN explainer methodology is still in its infancy. We apply layerwise relevance propagation to our existing Higgs boson interaction network to calculate relevance scores and reveal what features, nodes, and connections are most influential in prediction.

I. INTRODUCTION

Graph neural networks (GNN) are notoriously difficult to interpret [1 and 2], and those employed in the particle physics domain are no different. The graph interaction network has gained popularity with high energy physicists studying fundamental particles because this graph model achieves a highly competitive accuracy, while still working with relatively simple and unprocessed data [3]. However, it is often not fully understood how or why the Graph Interaction Networks make their classifications, or how these models' inner workings might relate to the physical properties of the universe.

The Higgs boson interaction network (HIN) is one such model that we seek to apply the latest research in graph explaining to. The purpose of the HIN is to determine whether or not a given jet, or spray of particles, decayed from a Higgs boson. The implementation examined in this paper is a simplified version of the HIN created in "Interaction networks for the identification of boosted $H \rightarrow \bar{b}b$ decays" [3]. Our HIN intakes one graph input and is trained solely on track/particle level features. The graph input represents a single jet instance, where each of the fully connected nodes is a particle, as in Figure 1. As an output, the HIN returns a classification: whether or not the origin of the jet decay was the elusive Higgs boson, or simply background noise. Specifically, the HIN is trained on a particular permutation of the Higgs boson decay known as $H \rightarrow \bar{b}b$, where the Higgs boson decays into b hadrons.

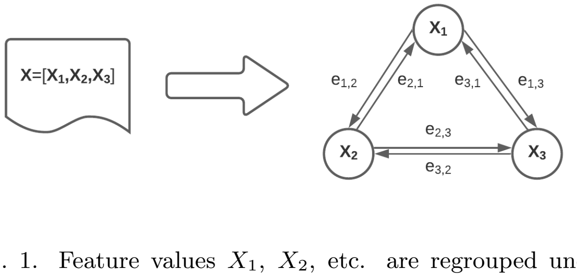

FIG. 1. Feature values X_1, X_2, etc. are regrouped under nodes such that one node is a particle, connected with directional edge weights.

We seek to use layerwise relevance propagation (LRP) to illuminate the inner workings of the HIN. LRP can explain highly complex deep learning networks with a strategic application of propagation rules based on deep Taylor expansion [4]. Historically, LRP has been advertised as a way to find and eliminate arbitrary or extraneous features in a complex neural network. However, with respect to our HIN, we are specifically interested in how the LRP can reveal the most important nodes and edges for predictions, which would essentially represent individual particle importance and particle-particle relationships that are particularly representative of Higgs boson decay, at least to the HIN model. That is to say, it is very possible that we would see known physics phenomena reflected in the decision making of the Higgs boson interaction network.

II. RELATED WORK

GNN interpretation is relatively new, largely kicked off by GNNExplainer in 2019, a methodology that approached the problem by taking a GNN and returning a salient graph substructure and the most influential node features [1]. We considered this a strong candidate for our model for a time, but were ultimately uncertain about how GNNExplainer's substructure strategy would react to a fully connected interaction network with essentially no preordained substructures. But among several emerging options for GNN interpretation, we ultimately decided to look into the direction of layerwise relevance propagation.

Layerwise relevance propagation has existed outside of the GNN context and has successfully been used to analyze a variety of model types, particularly convolutional neural networks [4]. More recently, LRP has been applied in the context of GNNs, such as in the case of GNN-LRP, which optimized LRP for GNNs by holistically analyzing graph pathways, or "relevant walks" [5]. LRP has seen usage in the chemistry domain, where it was implemented on a similar "InteractionNet" GNN to ours, except instead of particle relationships

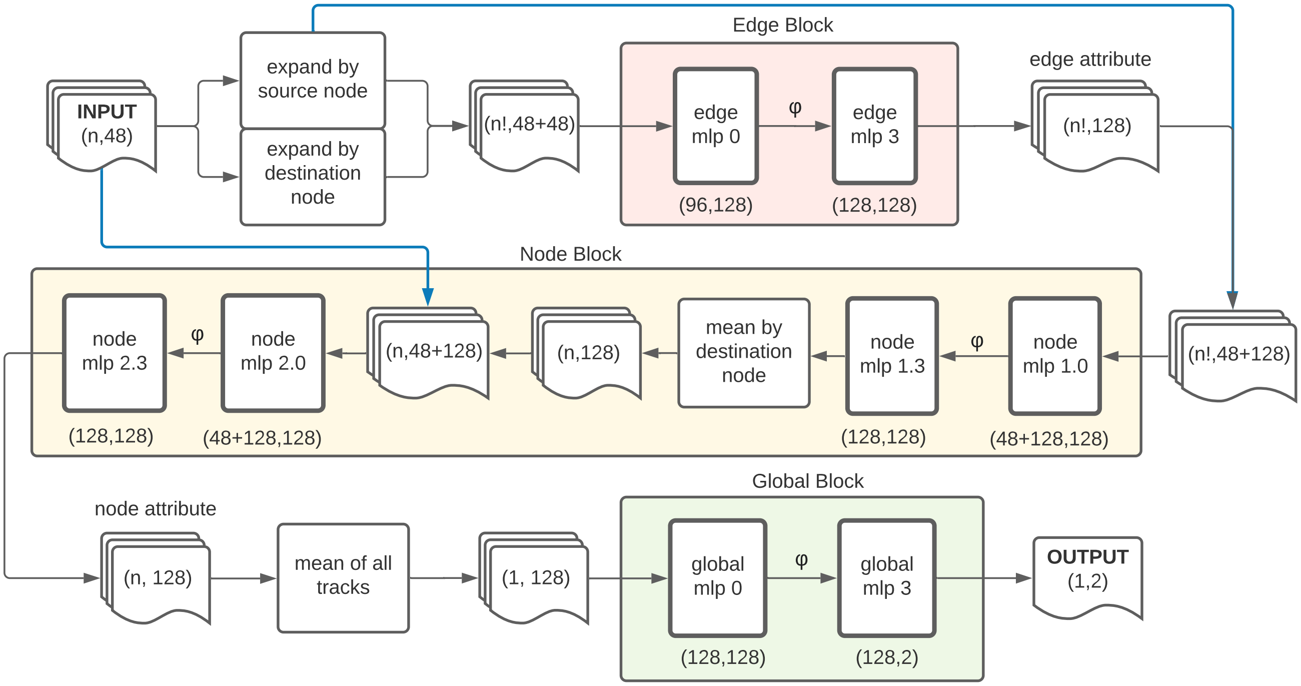

FIG. 2. HIN model architecture. Note the skip in layer indices, e.g. `edge mlp 0` and `edge mlp 3`, the skipped layers `edge mlp 1` and `edge mlp 2` are represented by the nonlinear mapping ρ, and merged with the closest linear layer preceeding them during the LRP computation. n represents the number of tracks in an arbitrary jet; ϕ encodes the nonlinear operation of a layer-wise normalization followed by ReLU. 128 is the dimension of the hidden layer abstract space. 48 is the number of particle features that can be found in a node. The blue pathways highlight the propagation relatively unique to GNNs, where an earlier input is reused multiple times through the layers.

it focuses on molecular structures where edges are bonds [6].

Broadly, jet tagging uses machine learning to help classify the particle collisions in an efficient and automated way. For a time, the success of these models depended on training with specially crafted features, where physics domain expertise plays a substantial role in aggregating and prioritizing useful information for the neural networks. The interaction network we explore here is notable for finding success training on more fundamental level features, particularly in the case of Higgs boson classification [3]. As far as jet tagging is concerned, layerwise relevance propagation has been used on CNNs and RNNs (convolutional and recurrent) in the particle physics domain [7], but not for GNNs. As such our goal is to make that step by applying LRP to the Higgs boson interaction network in this paper.

III. INTERACTION NETWORK

Our HIN model was programmed in PyTorch Geometric, which is a streamlined package specifically meant for GNN implementation [8]. PyTorch Geometric simplifies both the creation of the particle-particle interaction graphs and the training of the model itself. When the track features are adapted for the GNN, each jet level entry becomes a particle-particle interaction graph representing the relationships of every particle to all other particles in the graph bidirectionally. See Appendix A for the context of model training.

After processing the data into graphs, we use PyTorch Geometric to train the GNN by passing the data through three function blocks: and edge block, node block, and global block. As in Figure 2, the model's forward propagation adjusts corresponding edge weights, node weights, and global weights in each block in accordance with encoded transformation sequences: concatenations, linear transformations, batch normalizations, and ReLU activations. These transformation functions constitute the pathways that LRP will backpropagate across in order to acquire relevancy scores, which we will elaborate upon in the next section.

IV. LAYERWISE RELEVANCE PROPAGATION

LRP essentially redistributes relevance scores backward, starting from the model output, passing through the layers, all the way to the input. Each layer's relevance score is propagated from the previous layer (the layer closer to output) through the connections to the current layer, and as such the sum of the relevance score is kept approximately the same. Because LRP traverses the entire model, we can calculate a relevancy score for every edge, every node, every node feature, and more.

The flow of the relevance scores as it is backpropagated is analogous to the flow of water in a river: the total amount is conserved as it flows through the forks. This is described as a conservation property for LRP [4]. As such, the partial relevance scores ultimately attributed to the raw input from different paths should be directly summed up to approximate the actual prediction score of the output.

There is a distinction to acknowledge in applying LRP to GNNs, such as our Interaction Network, as opposed to other deep learning frameworks, like CNNs. Other models have a more straightforward layer by layer propagation, whereas the HIN re-propagates certain layers, particularly the input and the input source node transformation, reapplying those values in multiple instances across the block layers (see the blue highlighted pathways in Figure 2). This can be thought of as a weight sharing, and it affects the consideration of the conservation property. Namely, LRP backpropagation reaches the input at from multiple pathways, and each occurrence must be considered in the conservation calculations.

Figure 3 depicts an example of one backpropagation step in LRP. For an arbitrary node in Layer j, call it v_j, it receives the relevance scores from all the nodes that connects to it from the layer k that follows it. Hidden layer k, like other layers, draws from the output of an adjacent layer in activation from layer j, but unlike other layers, also pulls from the raw input. Thus, when propagating the relevance score through the model backwards, the relevance score R_k is split into two parts, R'_k and R_{src}, among which R'_k flows into layer j and the other layers beneath it, and R_{src} is attributed directly to the input.

LRP methodology provides several propagation rules that are outlined in "Layer-Wise Relevance Propagation: An Overview" [4]. In the case of the HIN, we apply a variant of the LRP-ϵ rule uniformly on the layers in our model, layers which can be roughly divided into two types: simple linear layers, and normalized rectified linear layers. For example, in our LRP propagation, `edge mlp 3` is one of the simple linear layers; on the other hand, `edge mlp 0` is joined with the non-linear operation ϕ into a normalized rectified linear layer.

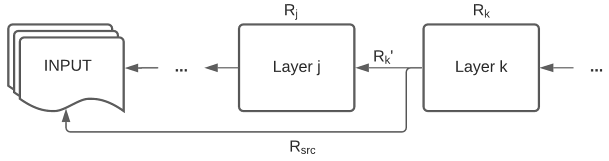

FIG. 3. Relevance propagation from Layer k backwards into Layer j and input. In the forward pass k sources from both Layer j and the input, so the relevance propagation would work accordingly. Intuitively, $\mathbf{R_k} = \mathbf{R'_k} + \mathbf{R_{src}}$

Denote the relevance score of node v_i in layer j as $(\mathbf{R_j})_i$, which can be computed as a proportion of the relevance score from the following layer, layer k. As shown, the amount of contribution can be conveniently computed by a forward pass. The LRP-ϵ rule for a given layer is as follows:

$$(\mathbf{R_j})_i = \sum_u \frac{(\mathbf{a_j}\mathbf{w_j^T})_{iu}}{\epsilon + \sum_u (\mathbf{a_j}\mathbf{w_j^T})_{iu} + \mathbf{b_j}} \mathbf{R_k}$$

Here $\mathbf{a_j}$ stands for the input at layer j, $\mathbf{w_j}$ and $\mathbf{b_j}$ respectively stand for the weight and bias at layer j. ϵ is introduced in the denominator to absorb some of the relevance score as well as to prevent division by zero, in accordance with the epsilon rule. Bold typeface indicates when the variables are tensors instead of scalars.

The proportion calculated in the LRP-ϵ rule follows a "gradient×input" convention to find out how each part of the input contributes to the layer activation, by computing by the product of the layer input and the partial derivative of the layer output with respect to the layer input. The partial derivative of the layer output can be viewed as "the rate of contribution to the activation" and thus the product can be viewed as the particular amount of contribution of the part of input towards the layer activation. For an arbitrary linear layer j, represented as a linear function ρ of layer input $\mathbf{a_j}$,

$$\begin{aligned} \rho(\mathbf{a_j}) &= \mathbf{a_j}\mathbf{w_j^T} + \mathbf{b_j} \\ \frac{\partial}{\partial \mathbf{a_j}} \rho(\mathbf{a_j}) &= \frac{\partial}{\partial \mathbf{a_j}} (\mathbf{a_j}\mathbf{w_j^T} + \mathbf{b_j}) \\ &= \mathbf{w_j^T} \\ \Rightarrow \mathbf{a_j} \cdot \frac{\partial}{\partial \mathbf{a_j}} \rho(\mathbf{a_j}) &= \mathbf{a_j} \cdot \mathbf{w_j^T} \end{aligned}$$

Following the paths illustrated in Figure 2, we can obtain several partial relevance scores from layers `node mlp`

2.0, node mlp 1.0, and edge mlp 0, which we denote $R'_{\text{input}}, R'_{\text{src}}, R_{\text{src,dest}}$ respectively. While the partial relevance score R'_{input} from node mlp 2.0 is directly attributed to the raw input, the scores attained from node mlp 1.0 and edge mlp 0 correspond to the relevance of the edges. Thus, the edge relevance score can be obtained by aggregating $R'_{\text{src}}, R_{\text{src,dest}}$ and R'_{input} expanded with respect to source node of the edges. When expanding w.r.t source node, each value in the matrix spreads out equally for every edge that originates from the same source node, resulting in a matrix with much larger dimension.

[some formula goes here]

V. RESULTS

TBA

VI. CONCLUSION

TBD

ACKNOWLEDGMENTS

We wish to acknowledge the support of our domain mentors Javier Duarte and Frank Würthwein, our domain TA Farouk Mokhtar, and our Capstone instructor Aaron Fraenkel. We also thank Justin Eldridge and Chen Cai for providing critical insight on graph theory.

Appendix A: Higgs Boson Interaction Network

1. Data

This Higgs boson interaction network was trained on monte carlo simulations of the LHC's collision data. [9] After many events are generated with the simulators, the most relevant jets are filtered through with a particle-flow algorithm that removes a certain amount of collision noise. Simulations are the preferred training data because the LHC produces approximately 10 quadrillion collisions per year, resulting in petabyte level amounts of particle data that are impractical to train on. Additionally, the actual data has an extremely imbalanced class ratio (approximately 99 to 1), as the Higgs boson is an extremely rare occurrence in real collision events.

Training models on the actual data would potentially result in a model with high accuracy, but low precision. Also, by using simulated data, we can concentrate solely on jets, because the actual LHC data often seeks to observe many other kinds of data. With all these considerations, we sourced the simulated data from the CERN Open Data Portal, and pulled out approximately 3 million jet entries. The particular Higgs boson event we draw from is the decay into b hadron pairs (Hbb), with background collision noise (QCD).

The Interaction Network ultimately trained on approximately 2 million jet entries, and evaluated on a random subset of 128000 jets due to time complexity. IN is trained with batch size of 128 jet particle-particle interaction graphs for 10000 minibatches per epoch (20% of the mini batches are put aside for validation purpose) with Adam optimizer and an initial learning rate of 10-4. We had planned for 150 epochs for the model with early stopping, but it actually converged quickly to a desirable result in less than 10 epochs.

2. Model

The edge, node, and global block structure is facilitated heavily by PyTorch Geometric abstraction, and further context for can be found in the documentation for PyTorch Geometric's Metalayer function [10], as well as the paper it was based on: "Relational inductive biases, deep learning, and graph networks" [11].

Appendix B: Project Proposal

Deep learning (DL) has commonly been regarded as a black box. By black box, we mean that we lack full understanding of how it works, despite knowing that it can produce outstanding performance. After implementing a successful GNN classifier for identifying Higgs bosons, we are left wondering how exactly our model makes the decisions. While there is an awareness in the physics domain that mass and momentum are large factors, it is difficult to understand concretely how the graphs weigh the features, especially since the more commonly understood features are usually decorrelated. We can look at the very model we just created, and with the same data, see if we can explain the model's understanding of Higgs boson jets. Is there potential for furthering the physics domain's understanding of the problem by explaining the model's composition?

This approach to deep learning is not exclusive to particle physics, it also applies to general applications of DL in different fields. Explainable DL models are becoming more and more popular recently, as people begin to notice its potential to understand various problems when the neural network's structure is more transparent.

Deep learning is still quite new, and deep learning interpretability is even newer, but quite a lot research has been done to interpret deep learning models. Many papers have already been written on understanding CNNs through the visualization of the network's activation, producing images that peel back the curtain on the neural net's mind. Unfortunately, we have far less understanding of how GNN works—the graph representation of the hidden layers are not as intuitively visualizable as feature maps of images in CNNs. And so we lack a comprehensive explanation of the Interaction Network we implemented for jet identification.

We know that a major advantage of IN is its ability to learn from low level features that are closer to the raw measurements from the collider, i.e. it does not depend as much on expert knowledge as the previously used classical ML solutions. We propose that by understanding a trained IN, we could potentially gain more insight into both the physics problem itself and possible direction for further improvement of the model. We hypothesize that we could render a "most signal looking input" or the graph that our model most strongly associates with the Higgs to b hadron decay.

Instead of studying a general question on how to explain the behavior of graph nets, we want to focus on the interpretation of the GNN based IN. A unique and good thing about studying IN is that, unlike many other GNNs, we have a rough expectation on its behavior based on existing particle physics theories. Probing into a trained IN, we expect to see that it actually learns the expert crafted variables from the raw input data, and it learns to distinguish the significance of different features; we also expect to discover something new or unexpected about the interaction of certain low level features. There is a potential to perceive actual physics phenomena by understanding how the GNN gets trained in a more coherent way.

Though explaining GNN is a relatively untapped subject, there are plenty of new and exciting reference points for us to reasonably build from. There is a long history of visualizing jets with image abstractions, so the hypothetical "most signal input" would have a straightforward extant visualization method. PyTorch Geometric does provide avenues to explain GNNs based on a paper that highlights and visualizes the most activated subsets of GNNs, which would synergize nicely with our existing PyTorch model. And just the other week a physics domain related paper was published about "Explainable AI for ML jet taggers using expert variables and layerwise relevance propagation" [7], which closely relates to the classical ML model we implemented with XGBoost in our replication.

[1] R. Ying, D. Bourgeois, J. You, M. Zitnik, and J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks (2019), arXiv:1903.03894 [cs.LG].

[2] D. Luo, W. Cheng, D. Xu, W. Yu, B. Zong, H. Chen, and X. Zhang, Parameterized explainer for graph neural network (2020), arXiv:2011.04573 [cs.LG].

[3] E. A. Moreno, T. Q. Nguyen, J.-R. Vlimant, O. Cerri, H. B. Newman, A. Periwal, M. Spiropulu, J. M. Duarte, and M. Pierini, Interaction networks for the identification of boosted h→bb¯ decays, Physical Review D **102**, 10.1103/physrevd.102.012010 (2020).

[4] W. Samek, M. Gregoire, A. Vedaldi, L. K. Hansen, and K.-R. Muller, *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning* (Springer International Publishing, 2019).

[5] T. Schnake, O. Eberle, J. Lederer, S. Nakajima, K. T. Schütt, K.-R. Müller, and G. Montavon, Higher-order explanations of graph neural networks via relevant walks (2020), arXiv:2006.03589 [cs.LG].

[6] H. Cho, E. K. Lee, and I. S. Choi, Interactionnet: Modeling and explaining of noncovalent protein-ligand interactions with noncovalent graph neural network and layerwise relevance propagation (2020), arXiv:2005.13438 [q-bio.BM].

[7] G. Agarwal, L. Hay, I. Iashvili, B. Mannix, C. McLean, M. Morris, S. Rappoccio, and U. Schubert, Explainable ai for ml jet taggers using expert variables and layerwise relevance propagation (2020), arXiv:2011.13466 [hep-ph].

[8] M. Fey and J. E. Lenssen, Fast graph representation learning with pytorch geometric (2019), arXiv:1903.02428 [cs.LG].

[9] .

[10] Source code for torch$_{g}eometric.nn.meta.P.\ W.\ Battaglia, J.\ B.\ Hamr$
$Gonzalez, V.\ Zambaldi, M.\ Malinowski, A.\ Tacchetti, D.\ Raposo, A.$
$1806.01261[cs.LG]$.