Pileup Synthesis and Transformer-based Anomaly Detection for Long-Lived Particles in the ATLAS Detector

Sam Young
Department of Physics
Stanford University
youngsam@stanford.edu

March 16, 2024

1. Introduction

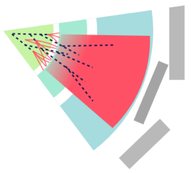

Figure 1. A "shower" of several LLPs traversing and decaying in a radial slice of the ATLAS detector at the LHC. The interaction point (i.e., where the protons collide to create new particles) is at the beginning of the slice. The LLPs are created at this point, but travel some distance (dotted line) through the detector before decaying into showers of particles that can be detected (in red). Diagram taken from Heather Russel's webpage: `https://hrussell.web.cern.ch/hrussell/graphics.html`

Though an extensive effort over many years as taken place, there currently is no evidence for undiscovered particles produced at CERN's Large Hadron Collider (LHC). Recently, there has been a growing interest in the community to develop machine learning techniques to enhance sensitivity to potential signals using all outputs of the detector.

The LHC, sat under Geneva, Switzerland, works by accelerating clumps of protons in a large ring at opposite speeds nearing that of light, colliding them, and detecting what particles are created after the collision. In some exotic extensions of the Standard Model (SM), a variety of so-called long-lived particles (LLPs) are hypothesized to be created in the proton-proton collisions. In this scenario, the particle after creation traverses some non-negligible distance through the detector before decaying, resulting in a detection of vertices[1] and tracks[2] that seemingly do not originate from the original collision (illustrated in Fig. 1).

Unfortunately, most of the background events in the detector also show up as displaced vertices (DVs). These signal-like background events are known as QCD events. There are also "true" background events, called pileup, which are nuisance particles created by protons in the clump lightly colliding with one another.

This project aims to be an initial survey of whether different methods in *supervised learning* for anomaly detection of these LLP-originated DVs in simulated data is feasible.

To do this, we train two different types of models we have touched upon in class – a multi-layer perceptron classifier (MLP) (Cybenko, 1989) and a transformer-based model (Vaswani et al., 2017) – to distinguish between events containing LLPs and signal-like background events. Our model's inputs will be single events parameterized as a variable collection of particles, and the output will be a single number corresponding to classification. A single particle ("track") is parameterized by five correlated features (See Section 3). Thus for each event, the classifier C maps $\mathbb{R}^{N_{track} \times 5}$ to $\mathbb{R}$.

2. Related Work

In recent years, a large community effort (Kasieczka et al., 2021) was undertaken to work on creating model-agnostic anomaly detection algorithms for use in the LHC. Particularly fruitful contributions to the field of anomaly detection for include optimal transport-based methods (Craig et al., 2024), variational recurrent neural networks (Kahn et al., 2021), generative adversarial network-based autoencoders (Vaslin et al., 2023), weak classifiers (Amram and Suarez, 2021), and CWoLA (Classification WithOut LAbels)-based methods (Metodiev et al., 2017).

There are also plenty of studies (e.g., (Qu et al., 2022)) using transformer for classification; however no studies that I have found have looked at using transformers for this task.

[1]A vertex is jargon for a point in space in which multiple particles are created.

[2]A track is the term for the detected path a particle takes throughout the detector.

3. Dataset and Features

I started the experiment by using MadGraph (Alwall et al., 2011), Pythia (Bierlich et al., 2022), and Delphes (de Favereau et al., 2014) to simulate three individual datasets:[3] 200,000 regular background (QCD) events containing p p $\rightarrow$ 2-5 particle showers (called “jets"), and two 100,000 event datasets containing a hypothetical LLP called the neutralino ($\tilde{\chi}_3^0$) with rest masses of 100 and 500 GeV[4], respectively. Added to each event within all datasets are an average of 60 pileup datasets (modelled as a Poissonian process), an average number consistent with what is seen at the LHC. Thus we have a balanced dataset of 200,000 signal + pileup events and 200,000 signal-like background + pileup events. As touched upon above, each particle is parameterized by 5 features:

- p_T (transverse momentum, units GeV): The component of the particle's momentum vector that is transverse to the beam axis (i.e., radially outwards).
- η (pseudorapidity, unitless): A commonly used metric that describes the angle of a particle relative to the beam axis, defined as $\eta = -\log\left[\tan\left(\frac{\theta}{2}\right)\right]$, where θ is the angle between the particle's momentum vector and the positive direction of the beam axis.[5]
- ϕ (azimuthal angle, units radians): The azimuthal angle between the particle's momentum vector perpendicular to the beam line.
- d_0 (transverse impact parameter, units mm): The shortest distance between a track and the beam line in the transverse plane (perpendicular to the beam line).[6]
- d_z (longitudinal impact parameter, units mm): The distance along the beam line between the primary interaction vertex and the track point.

When training the classifiers on this data, I perform several data augmentations: I remove all tracks with $p_T \leq 0.5$ GeV, remove all tracks with $|\eta| < 4.5$, remove all events with 0 tracks, scale the per-event p_T distributions to sum to 1, sort the values of each event by p_T, and only choose the 80 tracks with the highest p_T.

[3]Datasets are available upon request.

[4]We choose two rest masses to encourage the model to be able to be able to classify anomalous events for energies within a continuous mass range of $\mathcal{O}$(100 GeV), as models don't predict a specific rest mass.

[5]The intuition here is that a higher η corresponds to particles whose momentum are more along the beam line, and a lower eta corresponds to particles whose momentum is nearly perpendicular to the beam ($\vec{p} = p_T\hat{r}$).

[6]By their nature, LLPs will have larger values of d_0 because they take non-negligible time to decay.

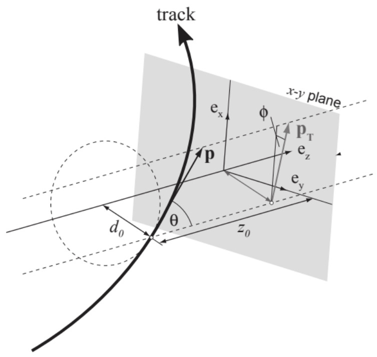

Figure 2. A track is parameterized by five parameters representing the shape of a helicoid. A reference point along the helix is used to calculate d_z and d_0. It is convention to use the helicoid's perigee to the primary vertex as this reference point. Figure taken from `https://atlassoftwaredocs.web.cern.ch/trackingTutorial/idoverview/`

After running this data augmentation, I use the maximum number of events possible to use while still maintaining a balanced dataset; this turns out to be 391,000 events in total. The dataset is randomly split into training and test datasets with a 80:20 split, corresponding to 312,800 events in the training dataset and 78,200 in the testing dataset.

4. Methods

4.1. Pileup synthesis

In an unfortunate circumstance, I only was able to get a hold of 1000 pileup events to be used in the sample. What is doubly unfortunate is that is nontrivial to simulate a large amount of pileup without devoting a considerable amount of computational resources to the task. Since we are dealing with approximately 400,000 events in total, we can't simply uniformly sample from the pileup distribution 400,000 times. Instead, we need to find a way to “simulate" 400,000 *independent* pileup events. To model the pileup we make heavy use of the Gaussian Mixture Model (GMM), which, given a number of components N, fits a weighted mixture of multivariate Gaussians to a group of observations, where the probability of observing values $\vec{x}$ is given by

$$p(\vec{x}; \{\mu_i, \Sigma_i, w_i\}_{i=1}^N) = \sum_{i=1}^{N} w_i \mathcal{N}(\vec{x}; \mu_i, \Sigma_i, w_i), \qquad (1)$$

where $\sum_{i=1}^N w_i = 1$. The latent variables $\{\mu_i, \Sigma_i, w_i\}_{i=1}^N$ are fit using the EM algorithm, coordinate ascent algorithm that iteratively finds the (local) maximum likelihood estimates of parameters in statistical models. This is done through two steps, the E-step and M-step. In the E-step, the responsibility of each cluster towards each individual input data-point given the fractional distribution of cluster assignments and the clusters parameters (μ and Σ for the case of the multivariate Gaussian). In the M-step, the latent variables are re-estimated using these new responsibilities by maximizing the log-likelihood of the complete distribution.

Since the number of components is a hyper-parameter, we elect to model distributions with the number of components that minimize the Bayesian Information Criterion (BIC):

$$BIC = k \log n - 2 \log \hat{L}. \tag{2}$$

Here, k is the number of parameters estimated by the model, n is the number of data-points in the distribution the GMM was fit to, and $\hat{L}$ is the maximized value of the sum of the logarithm of Eq. 1 over all data-points. The BIC is a model selection criterion and heuristic that presents a value that roughly aims to balance model complexity with overfitting. Generally, a lower value is better.

Because these are physical events, the event-level distributions are not merely uniform samplings of the global distribution. For example, you expect the daughter particles created at a vertex to be very close in distance to one another – i.e., on an event level, values of d_0 are essentially a combination of delta functions. Thus, to model these event-level distributions well, we must fit GMMs to each individual event, allowing the covariance to be arbitrarily small (i.e., to effectively model a delta function). We then fit *another* GMM to the distribution of GMMs, modelling the distribution of means, covariances, and weights. Then, for each event we'd like to synthesize, we sample an event-level probability distribution by sampling from this high-level GMM, and sample from that. To determine the number of tracks to sample from the synthetic event-level GMM, we simply fit a separate GMM to the track multiplicity distributions. This algorithm is illustrated in Fig. 3.

In the fitting pipeline, there are several augmentations we had to introduce for the distributions to model well. For each event, before fitting to a GMM, we:

1. Transform $p_T \to \log p_T$. This is because p_T varies over several orders of magnitude, is positive real-valued, and is log-normal-like.
2. **Remove** ϕ from the dataset. It unfortunately proved to be too difficult to model well under the time constraint; in the synthesis of pileup, we assume that it completely isotropic and uncorrelated from other parameters.[7]
3. **Remove** events with less than 3 tracks.[8]

After fitting the event, we take advantage of the fact that the covariance matrix is positive semi-definite and can be decomposed into a product of a lower-triangular matrix and its transpose, i.e., $\Sigma = LL^\top$. We then take the lower triangle indices of L as the representation of the covariance matrix, lowering the dimensionality of the GMM by $N_{param}^2 - \frac{N_{param}(N_{param}+1)}{2} = \frac{N_{param}(N_{param}-1)}{2}$. $N_{param} = 4$ for our pileup dataset, so the total number of parameters per component of a GMM is

$$4\,(\mu) \;+\; \frac{4(4+1)}{2}\,(L) \;+\; 1\,(w) = 15.$$

Saving the distribution of the number of components per event to be sampled from later, we concatenate the distribution of GMMs over all events in to a single tensor.

Before fitting this distribution of GMMs to the high-level GMM, we scale each of the 15 parameters to have a mean of zero and unit variance. We then fit a GMM with a maximum number of components of 100, using BIC to choose the optimal model.

4.2. Classifiers

To train the MLP, we utilize a simple fully-connected feed-forward neural network using PyTorch (Paszke et al., 2019) with 5 input features, 3 hidden layers with 32 hidden features, and one output feature ($\mathbb{R}^{N_{track}\times 5} \to \mathbb{R}$). Each input is of the shape $(N_{batch}, N_{track}, N_{features}) = (N_{batch}, 80, 5)$; if there are less than 80 tracks in a single event, the rest of the tensor is padded with zeros (though this does not happen when pileup is present). An additional mask of bools with size $(N_{batch}, 80, 1)$ is created and is multiplied after the activation of the initial layer to ensure the fact the padded values do not propagate through the network. We use the ReLU activation function (Fukushima, 1975) for the input and hidden linear layers apply a mean aggregation to end with a single value. LayerNorm (Ba et al., 2016) is applied after each Linear activation.

Our transformer-based architecture is an encoder MLP paired with a transformer encoder with multi-headed attention (MHA) and a MLP decoder. The encoder MLP is configured similarly to above, except that there is no output layer, and that the hidden and output activations are GELU the function (Hendrycks and Gimpel, 2016). The output of this encoder MLP (i.e., 32 encoded features) is input into a string of six transformer encoder blocks with GELU as the activation function, each with 8 heads[9] that learn different aspects of the encoded data, with the bool mask as the key

[7]It's true that ϕ is isotropic (uniformly distributed), but it is not true that ϕ is uncorrelated. Later in the project, ϕ should be incorporated into the GMM to ensure all correlations between parameters are modelled.

[8]This may seem arbitrary but about of the third of the events in the dataset contained 2 two tracks with very low p_T and high η. Considering the fact that in the classifier datasets we prune off low p_T and high η events, removing these low-track events will have negligible effects on the preprocessed classifier datasets.

[9]Following the original Transformer paper (Vaswani et al., 2017).

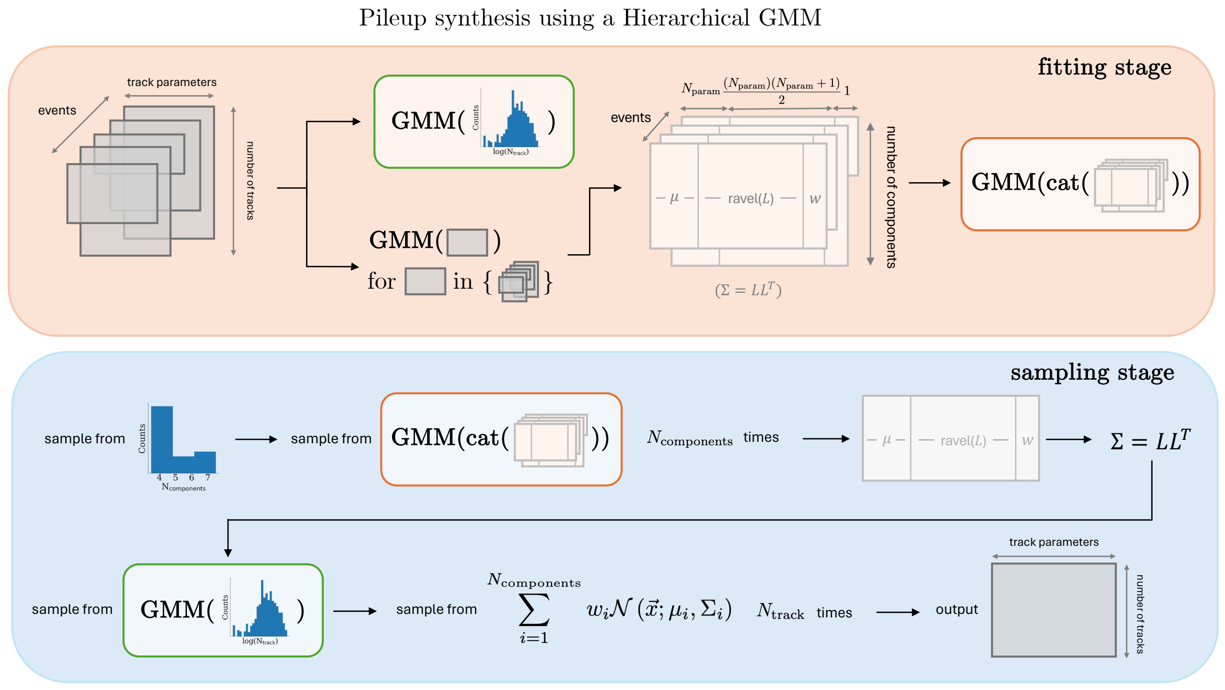

Figure 3. The algorithm for modeling pileup using a hierarchical GMM (HGMM).

padding mask. A dropout value of 10% is used, and the total output is aggregated via summation. The decoder MLP takes this output and decodes the output to a single value over 3 hidden layers with 32 hidden features, with GELU as the hidden activation function, no output activation function, and no layer normalization.

5. Experiments / Results / Discussion

5.1. Successful modeling of pileup

Our final model of pileup had 75 components. It appeared to model both the event-level and global track parameter distributions quite well; Fig. 4 shows the global distribution of parameters for 1,000 synthetic events compared to the original 1,000 events. Remarkably, even though we do not explicitly model this distribution, it is modeled well.[10] However, there are little "speckles" in the synthetic distribution which correspond to several events in which the majority of tracks were sampled from a multivariate Gaussian with a very small sampled covariance. This can be attributed to the fact that our 75-component Gaussian, although having the lowest BIC value, is still overfitting the underlying dataset. After all, we are fitting a 75-component Gaussian to a 15-dimensional dataset with $O(5000)$ entries (assuming 5 components on average per event)!

I attempted to get around this speckle issue by decomposing the synthetic covariance matrix into eigenvectors and eigenvalues (applying the spectral theorem), and clipping the eigenvalues to be at least, say 0.5, and reconstructing the covariance matrix from the modified eigenvalues/eigenvectors; this worked well for η, d_z and d_0, but because the p_T distribution is essentially $O(1)$, this clipping smeared distribution too much. This can, in principal, be fixed if all events (before being fit) were scaled to a zero mean and unit variance over all events. Nonetheless, due to time constraints, I used this speckle-ridden 75-component GMM.

5.2. Classifier Performance

I used the following hyper-parameters for training the vanilla MLP and transformer-based architecture: 500 epochs, a batch size of 2048, an initial learning rate of 0.001, a learning rate decay rate and factor of 100 epochs and 1/10, and the AdamW optimizer with (β_1, β_2)=(0.9,0.999).

500 maximum epochs was chosen to ensure no overfitting occurred, since we have a small dataset. A batch size of 2048 was chosen to ensure maximum GPU throughput, and the initial large learning rate followed by a step-wise decay every 100 epochs was employed because after many epochs of learning, it's possible that the optimizer is jumping across local minima with each gradient descent update. There

[10]Due to space constraints, I cannot include the 15-component corner plot of GMM components, although it models the distribution incredibly well. It is available upon request.

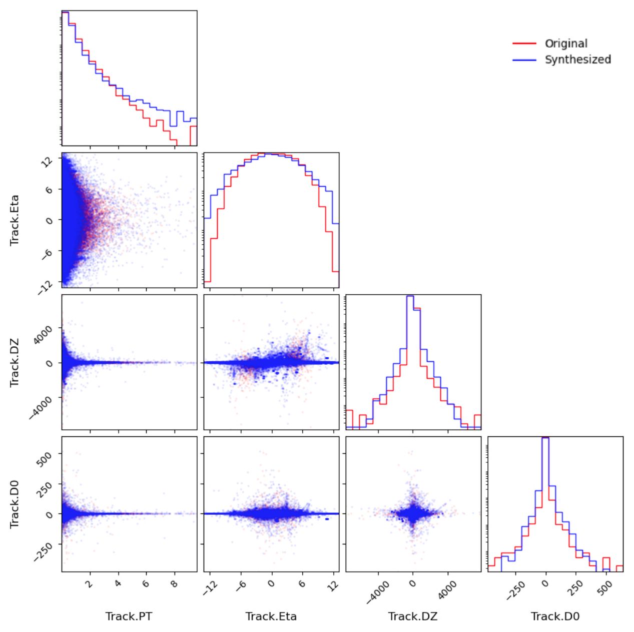

Figure 4. Corner plot of synthesized values of p_T, η, d_0, and d_z marginalized over 1000 real and synthetic pileup events. Each scatter plot compares the distribution of a pair of track parameters. The histograms correspond to single parameter distributions. The y-axis on the histograms are log-scaled.

Metric	Vanilla MLP	Transformer
AUC	0.762	**0.861**
Val. Acc	0.703	**0.817**
Recall	0.636	**0.678**
Precision	0.734	**0.938**

Table 1. Comparison of vanilla MLP and transformer model performance on their respective validation datasets.

were several primary metrics I looked at:

- **ROC Curve (Receiver Operating Characteristic Curve)**: A plot of the true positive rate against the false positive rate at each possible classification threshold setting.
- **AUC (Area Under the Curve)**: The area under the receiver operating ROC curve. The higher the AUC, the better the model is at predicting signal as signal and background as background. Ranges from 0 to 1.
- **Validation Accuracy**: The proportion of correct predictions (both true signal and true background) on the validation dataset.
- **Recall**: Measures the proportion of actual signal events that are correctly identified.
- **Precision**: Measures the proportion of signal classifications that were actually correct.

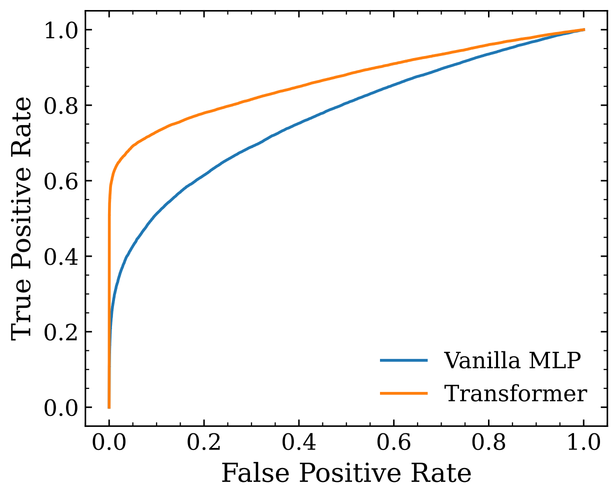

Figure 5. Comparison between the vanilla MLP and transformer-based models' ROC curves. The transformer model clearly outperforms the MLP at classification.

Table 1 shows the metrics shown of these models. Clearly, the transformer-based model performed much better than a 3-layer MLP, owing to the architecture's ability to understand the correlation between tracks with it's self-attention mechanism. Figure 5 shows the ROC curve between the two models.

6. Conclusion / Future Work

Employing hierarchical GMMs allows one to synthesize computationally expensive pileup events on both an event- and global- level. Issues were presented, but it is believed that a larger pileup dataset may help alleviate them. This opens a new avenue to simulating pileup for future studies without actually running physical simulations of ATLAS.

It is shown that it is indeed possible to delineate QCD and LLP events from one another in the presence of pileup, and using transformers for the task greatly increases model performance. Future studies should work with larger datasets, or more complex Transformer models like the Particle Transformer (Qu et al., 2022) for this task.

7. Acknowledgements

I would like to thank my research advisor Julia Gonski (SLAC) for providing me with assistance in generating the dataset and project direction, Daniel Murname (LBL) for giving me his PyTorch training workflow for a previous project to adapt from, and Nathan Suri (Yale) for valuable discussions. I also would like to thank the SLAC Shared Science Data Facility for allowing me to use their cluster to perform this study.

8. Contributions

SY undertook project ideation, execution, and report write-up.

References

Johan Alwall, Michel Herquet, Fabio Maltoni, Olivier Mattelaer, and Tim Stelzer. 2011. Madgraph 5: going beyond. *Journal of High Energy Physics*, 2011(6).

Oz Amram and Cristina Mantilla Suarez. 2021. Tag n' train: a technique to train improved classifiers on unlabeled data. *Journal of High Energy Physics*, 2021(1).

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization.

Christian Bierlich, Smita Chakraborty, Nishita Desai, Leif Gellersen, Ilkka Helenius, Philip Ilten, Leif Lönnblad, Stephen Mrenna, Stefan Prestel, Christian T. Preuss, Torbjörn Sjöstrand, Peter Skands, Marius Utheim, and Rob Verheyen. 2022. A comprehensive guide to the physics and usage of pythia 8.3.

Nathaniel Craig, Jessica N. Howard, and Hancheng Li. 2024. Exploring Optimal Transport for Event-Level Anomaly Detection at the Large Hadron Collider.

G Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Math. Control Signals Systems*, 2(4):303–314.

J. de Favereau, C. Delaere, P. Demin, A. Giammanco, V. Lemaître, A. Mertens, and M. Selvaggi. 2014. Delphes 3: a modular framework for fast simulation of a generic collider experiment. *Journal of High Energy Physics*, 2014(2).

Kunihiko Fukushima. 1975. Cognitron: A self-organizing multilayered neural network. *Biological Cybernetics*, 20(3–4):121–136.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus).

A. Kahn, J. Gonski, I. Ochoa, D. Williams, and G. Brooijmans. 2021. Anomalous jet identification via sequence modeling. *Journal of Instrumentation*, 16(08).

Gregor Kasieczka, Benjamin Nachman, David Shih, Oz Amram, Anders Andreassen, Kees Benkendorfer, Blaz Bortolato, Gustaaf Brooijmans, Florencia Canelli, Jack H Collins, Biwei Dai, Felipe F De Freitas, Barry M Dillon, Ioan-Mihail Dinu, Zhongtian Dong, Julien Donini, Javier Duarte, D A Faroughy, Julia Gonski, Philip Harris, Alan Kahn, Jernej F Kamenik, Charanjit K Khosa, Patrick Komiske, Luc Le Pottier, Pablo Martín-Ramiro, Andrej Matevc, Eric Metodiev, Vinicius Mikuni, Christopher W Murphy, Inês Ochoa, Sang Eon Park, Maurizio Pierini, Dylan Rankin, Veronica Sanz, Nilai Sarda, Urǒ Seljak, Aleks Smolkovic, George Stein, Cristina Mantilla Suarez, Manuel Szewc, Jesse Thaler, Steven Tsan, Silviu-Marian Udrescu, Louis Vaslin, Jean-Roch Vlimant, Daniel Williams, and Mikaeel Yunus. 2021. The lhc olympics 2020 a community challenge for anomaly detection in high energy physics. *Reports on Progress in Physics*, 84(12):124201.

Eric M. Metodiev, Benjamin Nachman, and Jesse Thaler. 2017. Classification without labels: learning from mixed samples in high energy physics. *Journal of High Energy Physics*, 2017(10).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library.

Huilin Qu, Congqiao Li, and Sitian Qian. 2022. Particle transformer for jet tagging.

Louis Vaslin, Vincent Barra, and Julien Donini. 2023. Gan-ae: an anomaly detection algorithm for new physics search in lhc data. *The European Physical Journal C*, 83(11).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.