Imaging Using Machine Learning for the LDMX Electromagnetic Calorimeter

Author:
Leo Östman

Supervisor:
Ruth Pöttgen

Thesis submitted for the degree of Master of Science
Project duration: 9 months

Department of Physics
Division of Particle Physics
May, 2020

Abstract

LDMX is a fixed target experiment designed to search for light dark matter. The experiment will search for dark matter signatures using missing momentum and energy in events with electrons scattering in the target. Part of the experimental setup in this search is a high granularity electromagnetic calorimeter to accurately measure the energies of recoil electrons.

In this thesis artificial neural networks are trained on simulated events in the electromagnetic calorimeter to classify events based on their particle contents, and to separate two showers from each other in the calorimeter. For the first task two sets of events are generated, one with a single electron that either undergoes bremsstrahlung or not, and one with multiple primary electrons. Two different neural networks models are used to classify these events. A basic convolutional neural network model achieves a classification accuracy of 96% on the first set of data, with an AUC of 0.99, and an accuracy 99.8% on the second set. A graph neural network model achieves a classification accuracy of 83% on the first set, with an AUC of 0.91, and an accuracy 99.3% on the second set. Another graph neural network model is trained on a subset of the first data set to label the individual hits in the calorimeter by whether they come from and electron shower or a photon shower. This network achieves an average accuracy of 86.6% over all events and correctly labels 90.5% of the energy in these events. The neural network models are compared to each other and evaluated based on their performance, limitations, and computing requirements.

Contents

List Of Abbreviations

ANN - Artificial Neural Network
AUC - Area Under Curve
CMB - Cosmic Microwave Background
CNN - Convolutional Neural Network
DM - Dark Matter
ECal - Electormagentic Calorimeter
FNR - False negative Rate
FPR - False Positive Rate
GNN - Graph Neural Network
HCal - Hadronic Veto Calorimeter
LDM - Light Dark Matter
LDMX - Light Dark Matter eXperiment
ROC - Receiver Operating Characteristic
SM - Standard Model
TNR - True Negative Rate
TPR - True Positive Rate
WIMP - Weakly Interacting Massive Particle

Acknowledgements

I would like to express my appreciation to Ruth, my supervisor, for all of the help, guidance, and support she has given me during this project. I would also like to thank Geoffrey for all the times when he has helped with any number of things I might have needed help with. The whole Lund LDMX group has also been very welcoming and helpful, always bringing useful discussion, good questions, and suggestions whenever I have presented something at the weekly meetings. I would like to mention all the other students in our office, and in particular Daniel, who I worked very closely with, for all of the discussion and suggestions. Special thanks to my partner Patri for all of the support she has given me on a daily basis, and for the help she has given me with this work. She, and of course all of my other friends here in Lund, have made this whole time so much better. Finally, my family for all of their support.

All computing was performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at LUNARC partially funded by the Swedish Research Council through grant agreement no. 2016-07213.

1 Introduction

Dark matter is a mystery haunting the universe with its presence. Evidence of the existence of dark matter has been known since the 1930s, when the presence of unseen matter was discovered in astrophysical observations [1]. This unseen matter was found in both galaxies that contained more mass seen in their gravitational effect than could be accounted for with the visible matter, and a similar observation in clusters of galaxies. Following these observations, there have been many more observations that point to the existence of some unknown matter that seems to only be interacting gravitationally.

What dark matter actually is made up of is not known, but there are multiple theoretical models of dark matter as one or more undiscovered species of particles. A feature of many of these models is that they include an assumption of a very small, but non-zero, coupling between dark matter and standard matter, opening up for the possibility to constrain, or potentially confirm, these models through experiments. The predictions made by these models can be tested in different ways, one of which is to use particle accelerators where potential interactions between standard matter and dark matter can occur. By comparison to the known physics of the Standard Model (SM) the predictions of the model can be tested.

This thesis will focus on an experiment, the Light Dark Matter eXperiment (LDMX), designed to search for some models of light dark matter (LDM), while also being able to explore other dark matter scenarios [2]. In this search for dark matter, the experiment will require high precision reconstructions of electrons, hence why this thesis will be looking at methods to aid in this using machine learning.

Machine learning has recently been used for various imaging related tasks in both computer vision and in particle physics. In particular, Convolutional Neural Networks and Graph Neural Networks have seen successes in tasks such as analyzing images and classifying them based on the contents of them, or detecting objects in 3D environments. Due to the similarity between this and the detection of particles in calorimeters, these types of neural networks have also seen use in various calorimetry related contexts in particle physics.

In this project, we make use of the two different types of neural networks mentioned above to study simulated events in the LDMX electromagnetic calorimeter. The aim of this is to study the usefulness of these different models together with calorimeter data in classifying different types of events, and to understand some of the potential limitations of these methods.

2 Background

2.1 The Standard Model of Particle Physics

The Standard Model of particle physics is a quantum field theory that describes all of the known elementary particles and their interactions with each other, these are shown in Figure 1. The elementary particles can be classified into two categories based on their spin, half-integer spin particles are fermions and integer spin particles are bosons.

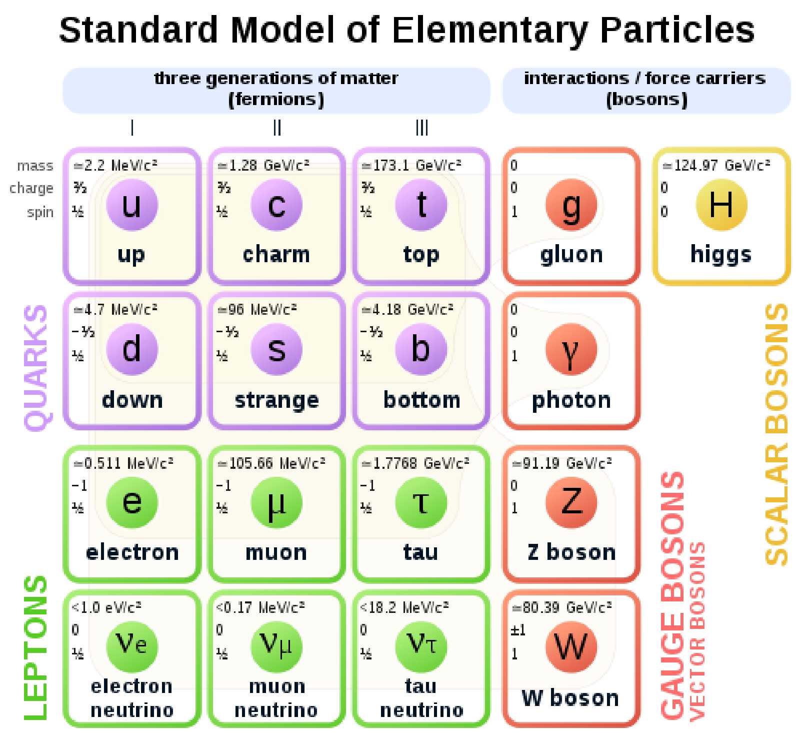


Figure 1: The particles of the Standard Model of particle physics, split into their respective categories [3].

The fermions can further be separated into two categories, quarks, which carry color charge, and leptons, which do not. The bosons too come in two different categories, the gauge bosons with spin 1 and the Higgs boson with spin 0. The gauge bosons are the force mediating particles, responsible for the strong, weak, and electromagnetic interactions between the particles that carry the relevant charge. Only electrically charged particles couple to the photon, and so only they experience the electromagnetic interaction, while only color charged particles, including the gluons themselves, couple to gluons and experience the strong force. All elementary fermions carry weak hypercharge and couple to the weak force, mediated by the two charged W bosons and the neutral Z boson. The Higgs does not mediate any forces, but it is responsible for the electroweak symmetry breaking that leads to masses for the massive bosons and for the Yukawa interaction that leads to fermion masses [4].

2.2 Dark Matter

The mass-energy content of the universe has been experimentally found to be about 68% dark energy and 32% matter. Baryonic matter described by the SM can only account for about one-sixth of the universal matter content, corresponding to about 5% of the total matter-energy of the universe [5]. What remains of the matter content must then consist of some type of matter that is not included in the SM, dark matter.

Observational evidence for the absence of visible matter in the universe was originally found in studies of the dynamics of galaxies and clusters of galaxies. Particularly the studies of rotation curves of galaxies showed the presence of unseen matter. Galaxies are seen to be denser closer to the galactic core, and so the greater gravitational potential closer to the center would lead to stars or gas clouds in that region to have a higher velocity than those farther out. What was observed in studies of multiple galaxies was that the velocity of a star or cloud as a function of distance from the galactic center did not drop, but rather remained constant as the distance increased. An example of this is shown in Figure 2. This behavior could be explained by the presence of some unseen matter [6, 7].

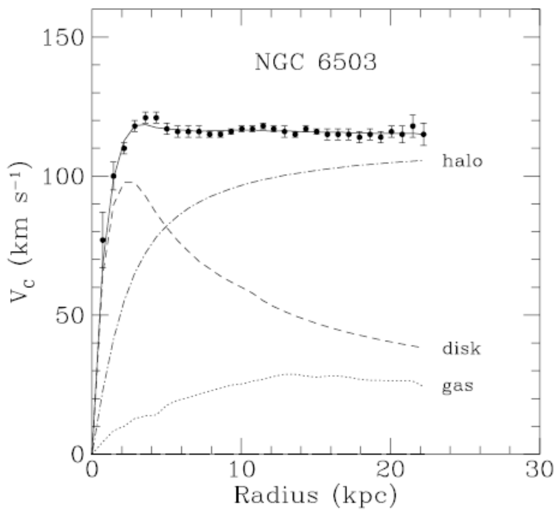


Figure 2: Plot of the rotation curve of the galaxy NGC 6503. Shown in the figure are the measured points of the galactic rotation curve and the contributions to the total rotational velocity from different components of the galaxy [8].

Evidence of the existence of dark matter in the early universe comes from measurements of anisotropies in the cosmic microwave background radiation (CMB). The CMB consists of the remaining photons from the recombination phase in the early universe when electrons and protons finally could combine to form neutral atoms, leaving these photons free through to propagate through what had previously been an electrically charged plasma. The photon-baryon interactions at this point interacted like a fluid that would compress due to local gravitational wells, until pressure in the fluid prevents this, and the fluid expands until the next compression. This type of oscillation lead to differences in the local matter density at recombination, and so when the photons became free to propagate they would leave the fluid at different energies depending on the local gravitational well. This leads to anisotropies in the CMB, which can provide evidence of the baryonic matter density in the early universe [7].

In addition to the above, the entire oscillation process requires the presence of some electrically neutral matter present in the fluid for the oscillations to occur before recombination. The density of this matter can similarly be measured from the anisotropies in the CMB. Results presented by the Planck collaboration, and earlier results by e.g. WMAP, show that the baryon density measured in the CMB is different from the total matter density, and that baryonic matter only makes up about one-sixth of the total matter in the early universe [5, 7, 9]. The remaining matter would need to be electrically neutral, as it otherwise would have coupled to photons and been included in the baryonic matter density.

Based on the available evidence pointing towards the existence of dark matter, there are certain properties that dark matter must have, in the case that it is one or more unknown species of particles. It does not belong to the SM, as the SM particles are well known and the SM leads to the observed abundance of baryonic matter in the universe based on the production mechanism following the Big Bang. Since it is invisible, it needs to be electrically neutral, so that it does not couple to the photon. If DM did couple to the photon it would have been seen in astrophysical observations. DM clearly needs to be massive for it to have gravitational interactions with SM particles, and there needs to exist at least one stable or sufficiently long-lived particle, as the DM observed in the early universe still exists. The neutrinos of the SM fulfill the above criteria, but their masses are too small for them to make up the dark matter in the universe [7, 10].

Since dark matter is not included in the Standard Model, new physics beyond the Standard Model is needed to explain presence and abundance of dark matter. New physics not included in the SM can be added to it given that this new physics is gauge invariant, and in this way it is possible to create models that are consistent with the SM, but also include new particles and interactions. Some models include new particles that could potentially couple to SM particles through these new interactions. Massive particles that through SM or non-SM interactions couple to SM particles could be dark matter candidates, given that the coupling is sufficiently weak in strength. The particles must be massive to produce the observed gravitational effects in the universe, and the coupling to SM particles must be weak enough to be able to produce the abundance of dark matter seen in the early universe, while having gone undetected until now. From this there are different possible masses that the particles could have, depending on their interaction with SM particles and their production mechanism [10].

Interactions between dark matter and standard matter in the early universe is one explanation for the abundance of dark matter in the current universe, and this puts restrictions on how strong the interactions can be and how massive the particles can be. A typical mechanism that leads to the ratio of matter to dark matter are non-gravitational interactions during the expansion of the early universe that eventually reaches equilibrium as the universe has become too large after expansion for the interaction to continue at an appreciable rate. This freeze out of dark matter and standard matter then leads to specific model-dependent velocities and cross sections on their interactions and self-interactions, and defines a mass range in which discoverable dark matter particles could possibly exist. Particle dark matter based on this scenario is confined to having masses from the order of MeVs to some 10s of TeVs.

From this range the particles can be broadly classified as either Light Dark Matter (LDM), roughly in the range below 1 GeV and, and Weakly Interacting Massive Particles (WIMPs) covering the range from 1 GeV and above [2, 10, 11].

Weakly Interacting Massive Particles are one type of dark matter candidate. These are massive particles that couple to the SM through some interaction on or below the strength of the SM weak interaction. WIMPs in the mass range of GeV-TeV that are created in the early universe serve as a dark matter candidate, as they rather easily lead to appropriate dark matter densities in the current universe. In a hot enough early universe these WIMPs are in equilibrium with SM particle, but, as the universe expands and the temperature decreases, they experience the freeze out described above. At this moment the WIMPs can no longer be created, but can still annihilate with each other, until the WIMP density is low enough that it remains constant, creating the relic abundance that is observed today [10, 8].

Light Dark Matter is another, less investigated, dark matter candidate, whose abundance is similarly the result of a freeze out mechanism. One LDM model is that of dark matter particles that couple to SM particle through a new interaction mediated by a force carrier called a dark photon. The dark photon is light compared to the massive gauge bosons of the SM, and obeys a $U(1)$ symmetry like the SM photon, hence the name. The dark photon couples to the SM through a kinetic mixing mechanism where the photon and dark photon mix, allowing the dark photon to couple to the electrically charged SM fermions with a strength governed by a mixing parameter ϵ [2].

Interactions between DM and SM particles imply that the interactions could be observed in experiments, given that the interactions are strong enough to meet experimental sensitivities. Typically, these searches aim to either produce dark matter using particle accelerators, detecting the direct interaction between DM and standard matter, or detecting the result of dark matter annihilation or decay. Detecting a beyond the SM particle does not imply that a dark matter candidate has been discovered. Dark matter candidates need to fulfill the above mentioned criteria on stability, electric charge, and mass, and would need to be present in galactic halos to conclusively be dark matter [10, 12, 13].

2.3 LDMX

LDMX is a proposed fixed target experiment intended to search for light dark matter. The experimental setup consists of an electron beam incident on a target followed by a recoil tracker, an electromagnetic calorimeter (ECal), and then a hadronic calorimeter (HCal). The processes that LDMX will look for are the creation of dark matter particles or mediator particles in the target. This would occur through a bremsstrahlung process where the electron through an interaction with a target nuclei will either radiate a dark matter mediator particle, which is likely to decay to two dark matter particles, as shown in 3. In the process of recoiling against the created particles the electron will lose a significant fraction of its energy and obtain transverse momentum relative to the beamline, while the dark matter particles would escape undetected. As the electron trajectory is displaced from the beam direction it will enter the electromagnetic calorimeter as shown in Figure 4, which also shows the experimental signal of the production process described above [2].

The missing momentum and displaced low energy electron make it possible to estimate some of the properties of the produced dark matter mediator particle, provided that the kinematics of the electron can be precisely measured and any potential background can be efficiently rejected. On top of a low energy electron displaced from the beamline, the signature for an event where dark matter could have been produced also requires that the electron produced the only shower in the ECal. Measuring the electron energy and momentum is done with a recoil tracker after the target that measures the momentum of the recoiling electron and an electromagnetic calorimeter designed to precisely determine the energy of the electron. A large number of background events can then be rejected based on the total energy deposition in the ECal, as an event where a SM photon is produced and showers in the ECal will have a total ECal energy close to the beam energy. Background events where the photon does not shower in the ECal, due to for example a photo-nuclear interaction that produces one or more neutral hadrons, cannot be rejected based on just the ECal energy, and so the HCal is used to detect and reject these events [2].

The experiment is intended to run in multiple phases, starting with single electron bunches with a high rate of bunches to obtain a large number of electrons on target. In later phases the experiment will use larger bunch sizes in order to reach full sensitivity [2].

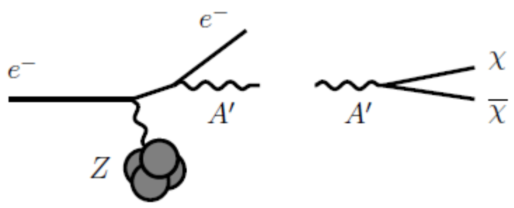


Figure 3: Diagram showing the production of a dark photon A' in a dark bremsstrahlung reaction. The dark photon decays invisibly to dark matter particles, one of the processes that LDMX is designed to look for [2].

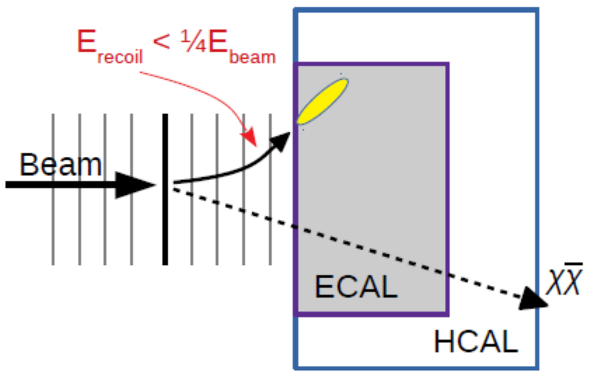


Figure 4: Simplified schematic of the LDMX target, detector, and a signal process. The incoming electron undergoes an interaction that creates two invisible particles. The electron creates an electromagnetic shower in the ECal [2].

2.3.1 The Ecal and its Own

The LDMX ECal design is a high granularity sampling calorimeter with silicon and tungsten layers. The ECal consists of 34 layers of 7 hexagonal sensors placed as in Figure 5, where the red dots correspond to the centers of the individual readout cells on the sensors. The cells have sizes of around 0.5 cm^2, and the dimensions of the ECal can be seen in Figure 5. The high granularity of the ECal means that it might be possible to more precisely resolve separate overlapping showers in the ECal that would arise in scenarios where more than one electron is used per beam bunch. Additionally, the granularity and structure of the ECal could be beneficial for machine learning based imaging techniques that make use the structure and relative positioning of objects in an image for classification. The higher granularity can give a clearer shower structure using 3D readout cell data from the ECal, and this makes it easier to use these machine learning techniques. How these machine learning algorithms work will be described in the following section.

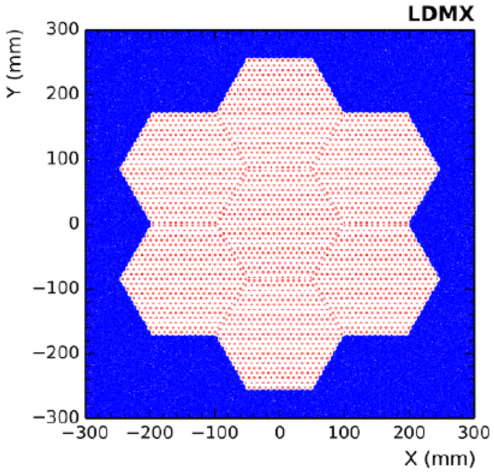


Figure 5: Cross section of the LDMX ECal showing the fiducial region of the calorimeter in red and white, where the red dots are the centers of the individual readout cells of the calorimeter [2].

2.4 Machine Learning

Artificial neural networks (ANN) are mathematical models used for information processing originally meant to emulate some function of a brain. For a given N-dimensional input $\{x_k\}$, the ANN will process the data and produce an output $\{y_j\}$ depending on the internal structure of the network. The structure, architecture, or model of an ANN typically consists of one or more layer of nodes with weighted connections to each other. It is these sets of weights $\{w_{ik}\}$ that determine how well an ANN performs at the task is designed for, as for any given model these weights can be adjusted by comparing the output to some target output $\{d_j\}$, and evaluating their difference using some loss function L. The process of adjusting the connecting weights based on performance is called training [14, 15].

The most basic unit of an ANN is the single neuron or node, shown in Figure 6 which takes the input set $\{x_k\}$, performs a weighted sum, and produces the output y_i using the formula

$$y_i = f\left(\sum_k^N x_k w_{ik} + b_i\right) \tag{1}$$

where f is called an activation function, the $\{w_{ik}\}$ are the set of weights relating input x_k to the specific output y_i, and b_i is a bias that can be added to shift the output of the activation function by a constant.

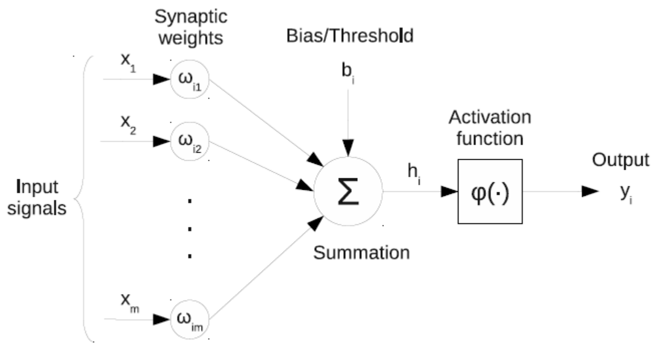


Figure 6: A single neuron taking m inputs x_j, each with their own weights w_ij. The neuron takes the weighted sum in Equation 1 together with a bias input as the argument of its activation function and produces the output y_i [15].

Using a single neuron as a building block, more complex ANNs can be constructed by arranging sets of neurons in one or more layers, to create a multi-layer networks, shown in Figure 7. An ANN with multiple layers functions in the same way as a single neuron, but neurons in the same layer are evaluated in parallel and neurons in different layers in series starting from the input layer. Each individual node takes the weighted sum of its input and produces an output using its activation function, which can then further be used as input for another node, or as the final output of the model [15, 16].

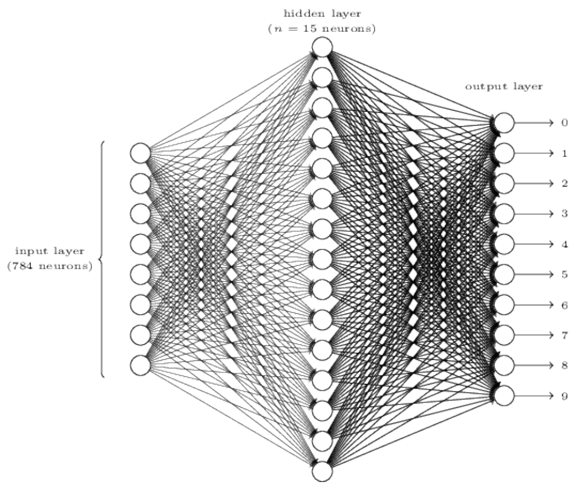


Figure 7: A multi-layer neural network, where the 15 hidden nodes in the middle are intermediate nodes between the input and the output nodes [16].

The result of the final layer of the model is the output from the model. The shape of the output, or the number of nodes in the final layer depends on what task the model is designed to do. A model aiming to classify the input into one of n classes will have n output nodes, where the numerical value resulting from each node would correspond to the probability that the input belongs to that class. If only a single value is desired as the output, then only a single node in the output layer is needed. If the model is aiming to recreate the input, the output layer should have the same size as the input.

2.4.1 Convolutions and its rivals

A convolutional neural network (CNN) is an artificial neural network where one or more of the layers are so called convolutional layers. In a convolutional layer, the D-dimensional input data is convolved with some D-dimensional matrix or tensor, called a kernel or filter, and the output of the convolution is fed through the activation function of that node. The kernel and input have the same dimensionality, but typically the size of the kernel is smaller than the size of the input. For a 2D input of size (x, y), the kernel might only be of size $(0.1x, 0.1y)$, so that multiple copies of the kernel can fit within the full input. In this way different parts of the input can be convolved with the kernel, to produce outputs that depend on the positioning of the kernel on the input. An example of this shown in Figure 8, where the 2×2 kernel is convolved with the input matrix 6 times at six different locations on the input. This generates another D-dimensional tensor, called a feature map, which is passed on to the next layer. In the 2-dimensional case, e.g. an image, the convolution operation is given by

$$I(x, y) = \sum_m \sum_n w(m, n) g(x - m, y - n) \tag{2}$$

where I is the convolved image or feature map, g is the input image, and w is the convolution kernel. The convolution operation is illustrated in Figure 8. In this way, for a given input and kernel, the convolutional layer creates a feature map. In a convolutional layer it is the kernel that contains the weights that are trained during the training process, as the elements of the kernel are what connects one layer to the next [14].

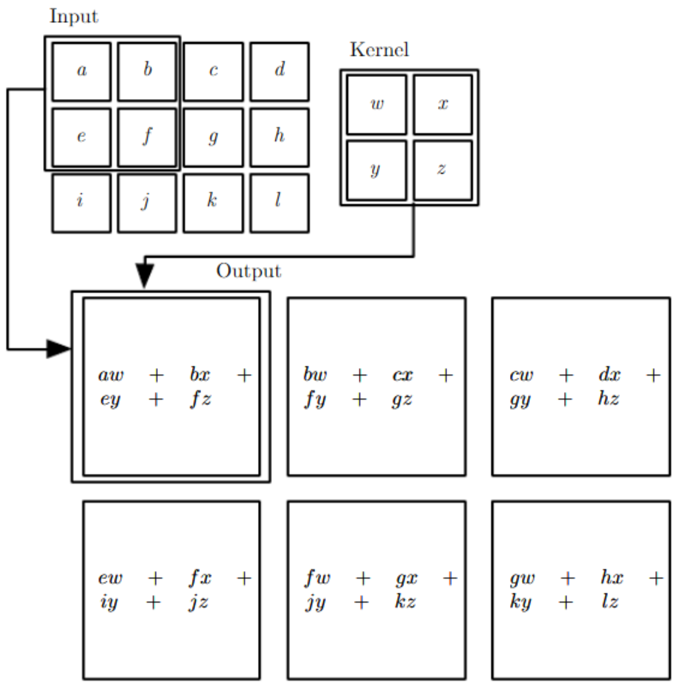


Figure 8: An example of a convolution operation using a 2×2 kernel on an input where the kernel can be convolved with the input six different times by sliding the kernel over the input [14].

If the kernels of a convolutional layer are smaller in size than the input tensor, it is moved through the tensor, as shown in Figure 8, and the same convolution is applied to every applicable patch of the input tensor. In this way, every output element from the convolutional layer does not have to interact with every single element of the input tensor and there is no need to have connecting weights between all of the input and output elements. As the same kernels are applied to every patch, it means that significantly fewer connecting weights are needed between layers, compared to a fully connected multi-layer network, and so fewer operations need to be performed, and less memory is required.

In a convolutional neural network, the analogue to the neuron or node described above can be found in the output from the convolution operation. A 2D convolution operation with a single kernel produces a 2D output where every single element corresponds to the output of a single neuron, as the value of one element is the dot product of the kernel with a region of the image. If every single neuron in the output feature map was uniquely connected to the input image the number of weights required would be equal to the product of the number of neurons in the feature map with the number of elements in the kernel. For large images and many kernels this can become very large, and so parameters are typically shared in convolutional layers so that every feature map has the same connection to the input, as described above. This reduces the number of weights per feature map to the number of elements in the kernel [17].

Another effect of the parameter sharing described above and the same kernel being applied in the same way over the entire tensor is that repeating or similar structures in the input image can be found. A given set of values in the input tensor will produce the same convolved representation on the feature map, regardless of where on the feature map they end up. In this way the kernel produces representations of repeating structure in the data on the feature map when it is created. Some examples of this are edges or similar curves in an image, which will have the same representation in the feature map regardless of their location in the image. This translational invariance of the convolution process makes convolutional neural networks ideal for detecting repeating features in the input tensor, and ideally the CNN would learn to distinguish inputs based on the features.

Convolutional layers in a CNN are typically accompanied by a layer that implements a pooling operation on the feature map, after the map has passed through an activation function. A pooling layer typically reduces the size of the feature map more than the convolution operation does, thus decreasing the size of the representation of the input and reducing the number of weights in the CNN. This is done by pooling together the elements in some region of the feature map using a function that produce a summary of that region of the feature map. Some function that are used for the pooling operation are max-pooling and average-pooling functions, which output the maximum value and the average value of the region respectively. The pooling and size reduction is shown in Figure 9, where a 2×2 max-pooling operation is performed in a 4×4 matrix in the centers of the 4 quadrants of the matrix. The decrease in size following the pooling operation depends on both the size of the pooling unit and how many steps are in between each operation, in Figure 9 two steps are used between each operation, if only one step had been used the output size would have been 3×3.

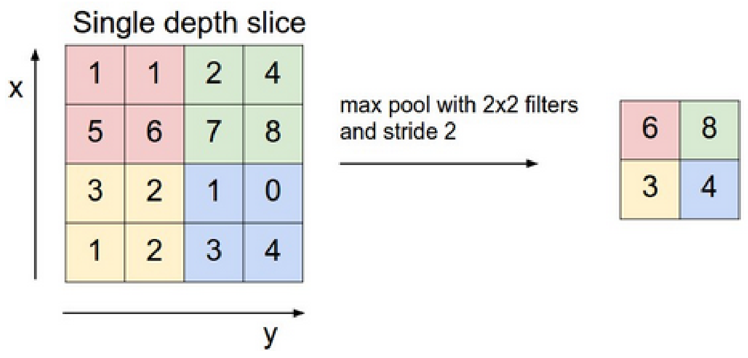


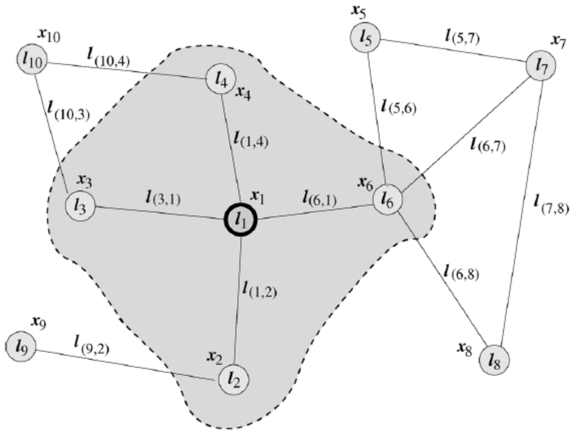
Figure 9: Illustration of the max pooling operation performed on a 4×4 matrix using a 2×2 pooling unit. The pooling unit is applied on the center of each quadrant of the input, with two steps between each pooling, thus producing an output of size 2×2 [17].

Similar to the convolution operation, the max-pooling operation includes translational invariance. Since max-pooling produces a summary of a region of the input using the maximum activation of the feature map, a specific feature in the feature map will likely produce the same pooled output regardless of where on the feature map it is located. The max-pooling operation is thus useful in cases where the existence of some feature in the input image is more important than the precise location of said feature.

The combination of one or more convolutional layers with a max pooling layer in a CNN can thus efficiently detect specific features in images that are indicative of a specific type or object or image. A CNN can also contain one or more layers following the convolutional and pooling layers that combine the convolved or pooled feature maps a vector of features that is used as input for what is called a dense layer. This dense layer is like a hidden layer in the multi-layer network described above, consisting of a large number of neurons, where each neuron in a layer is connected to every neuron in the preceding and following layers. One or more dense layers are used to process the final feature maps and generate the final output of the CNN depending on what the model is designed for. In a CNN designed for classification, the final layer of the model has a number of neurons equal to the number of classes that the model is classifying [14, 17].

2.4.2 Graph Neural Networks

Graph Neural Networks (GNN) are neural networks that make use of data that is structured in the format of a graph [18]. A graph G is made up of a pair of vertices and edges (V, E). The set of edges connect the vertices to each other and contain relations between vertices, such as distance between vertices. Each vertex in a graph has an associated vector F of features that describes the vertex. An example of a graph is shown in Figure 10.



$$x_1 = f_w(\underbrace{l_1, l_{(1,2)}, l_{(3,1)}, l_{(1,4)}, l_{(6,1)}}_{l_{co[1]}}, \underbrace{x_2, x_3, x_4, x_6}_{x_{ne[1]}}, \underbrace{l_2, l_3, l_4, l_6}_{l_{ne[n]}})$$

Figure 10: Illustration of a graph and the computation of the state of vertex 1. The central vertex is connected to the four neighboring vertices through edges, each with their own features, states, and edge features. The state of the vertex is a function of its own feature l_1 and the features of the edges associated to it, the states of the neighboring vertices, and the neighboring features [18].

From each vertex and its neighboring nodes local information can be extracted by assigning each vertex i a state x_i that depends on itself and the connection it has to some number of neighboring vertices. This is given by

$$x_i = f(l_i, l_{co(i)}, x_{ne(i)}, l_{ne(i)}) \tag{3}$$

where l_i is a features in the feature vector of vertex i, $, l_{co(i)}$ are the features of the edges associated to vertex i, $x_{ne(i)}$ are the states of the neighbors connected to i through edges, and $l_{ne(i)}$ are the features of the neighboring vertices. This operation is shown in Figure 10. By performing this operation at each vertex and creating vectors of all vertex states x and their features F a global output $y = g(x, F)$ can be computed for the entire graph [18]. This output can be on node-level, i.e. each node has its own output, or on the graph-level, i.e. each graph has a single output, depending on the task that the GNN is used for.

Graph Neural Networks are distinguished from Convolutional Neural Networks by the fact that the input is in the form of a graph, instead of a regular grid-based structure such as an image that CNNs take as input. This makes GNNs useful for data that does not have this regular structure or that is more easily represented by a point cloud, where each point has some spatial coordinates that are not necessarily restricted to a grid.

2.4.3 The EdgeConv Operation

The Edge Convolution operation (EdgeConv) is an operation on graphs that is analogous to the convolution operation used in CNNs. The operation is shown in Figure 11. EdgeConv makes use of the edge features between a vertex x_i and the N nearest neighbors of said vertex. The edge features between vertices i and j are given by

$$e_{ij} = h_\Theta(F_i, F_j) \tag{4}$$

where F_i and F_j are the features vectors of the vertices, and h_Θ is a function that generates a new feature vector. Θ denotes the parameters of the function h that are learnable during the training process, so the parameters are the weights of the EdgeConv layer similar to how the kernels contain the weights of the convolutional layers in a CNN. These parameters are shared for the entire graph, like the convolution kernel is shared for the entire image in a CNN. The computed edge features at vertex i are then aggregated using some function to produce the new feature vector F_i', given by

$$F_i' = \square_{i,j} h_\Theta(F_i, F_j) \tag{5}$$

where the $\square$ operation is some operation applied to all of the neighboring vertices j to aggregate information about them, e.g. an average of the edge features or a maximum, similar to the pooling operation in a CNN [19].

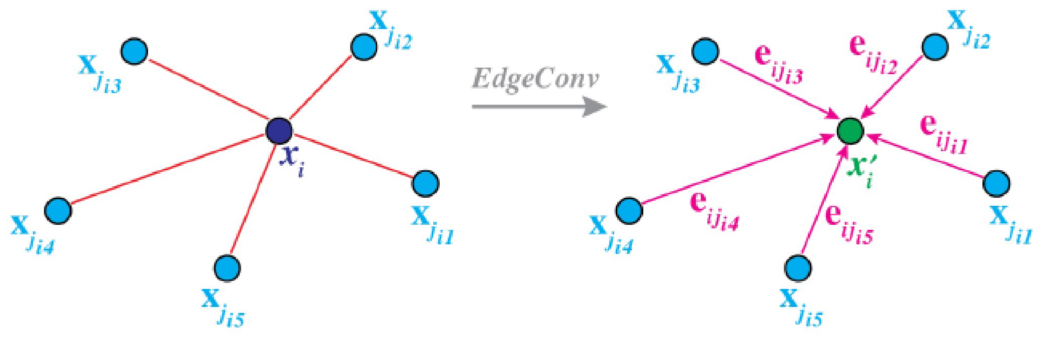


Figure 11: Illustration of the EdgeConv operation. EdgeConv aggregates the edge features associated to the central vertex x_i and its 5 nearest neighbors, each defined by some feature vector F [19].

Like in a CNN, the shared parameters of the edge function h_Θ means that the same operation is applied to every vertex in the graph, and so any features that are aggregated at a specific vertex will be the same if that vertex is located somewhere else in the graph, but with the same relation to its neighbors. This means that the operation can detect local features of the graph, such as clustered vertices, regardless of where in the graph they are located, as they will have similar aggregated edge features. Further, if the $\square$ operation is symmetric under permutations of the vertices in the graph, then the EdgeConv operation is symmetric to permutations that e.g. correspond to a rotation of the vertices making up the graph if the data is in the form of a point cloud, while a CNN is not necessarily rotation invariant.

2.4.4 GravNet Layers

The GravNet layer is a GNN layer designed for particle reconstruction. The graph that the layer operates on consists of a number of vertices V, each of which has some spatial coordinates, not necessarily Cartesian, and an associated vector of features F_i. Each feature f^i in the feature vector of each vertex j is modified according to

$$\tilde{f^i_{jk}} = f^i \cdot V(d_{jk}) \tag{6}$$

where V is a potential that is a decreasing function of the Euclidean distance between vertex j and the vertex k in the graph space. For each vertex, this process is repeated for the N nearest neighboring vertices in the graph space. These features are then combined to generate a number of new features for each vertex through an operation similar to the pooling operation in CNNs, by e.g. taking the average or maximum feature as the new vertex feature. By combining these new features with the original features at each vertex, a new feature vector is generated after each GravNet layer containing both the original features and the learnt features from the layer. This process is shown in Figure 12 [20].

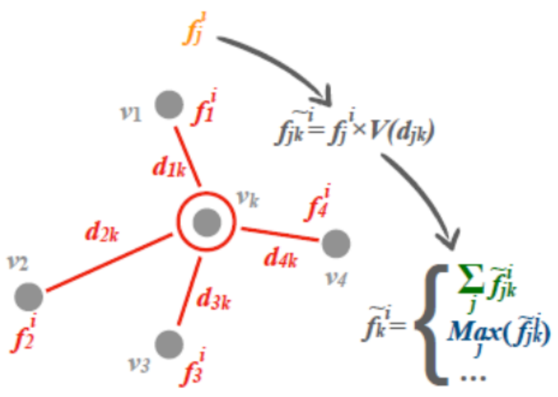

f_j^i
v_1
f_1^i
$\widetilde{f_{jk}^i} = f_j^i \times V(d_{jk})$
d_{1k}
v_k
f_4^i
d_{2k}
d_{4k}
v_4
v_2
d_{3k}
$\widetilde{f_k^i} = \begin{cases} \sum_j \widetilde{f_{jk}^i} \\ Max_j(\widetilde{f_{jk}^i}) \\ \ldots \end{cases}$
f_2^i
v_3
f_3^i

Figure 12: Schematic showing the operation of the GravNet layer. The circled vertex is connected to 4 other vertices, and new features for the vertex are computed based on the Euclidean distance between the central vertex and the others. This results in the new features following some pooling-like operation [20].

Similar to convolutional layers, the GravNet layer can make use of local information in the input data to learn features in the data. By considering only the N nearest neighbors at each vertex, the features learnt will be based on the surrounding vertices at each vertex. The potential as a function of distance puts further weight on nearby vertices, and the pooling function can collect both individual information at each vertex by taking the maximum and more collective information by taking the average. In this way, the learnt features at each vertex are likely highly correlated with the local neighborhood in cases of vertices being clustered in the graph space, and isolated vertices will have little influence on the rest of the sample. This makes the GravNet useful for tasks that have clustered data in the graph and highly local features.

2.5 Training a Neural Network

A neural network can be trained by assessing the performance of a model in terms of how close the model comes to the desired output and then updating the weights of the network in order to produce an output that is closer to the desired output. The updating of the weights is done using what is called an optimization algorithm, which is meant to update the weights such that the model performs better while avoiding over-correcting the model such that it performs worse than before. While there are many metrics that can be used to assess the performance of a model, such as the accuracy of a model for classification, the weights are updated based on what is called a loss function.

The loss function L is a function that takes as arguments the weights in a network and the input to the model and gives as an output a scalar value that is a measure of how closely the output of the model approximates the target. Typically, the closer L is to 0, the better the model is at approximating the target data, so the objective of optimizing the model is to minimize the loss function [16]. An example of a loss function is the binary cross entropy function, which can used as the loss function for binary classification tasks, where an object can belong to one of two classes,

$$L(w, x) = -\sum_{x} (d(x)ln(y(w, x)) + (1 - d(x))ln(1 - y(w, x))) \tag{7}$$

where x is the input to the model, w is the set of weights in the model, $d(x)$ is the class that input x belongs to, and $y(w, x)$ is the probability that x belongs to one class, thus $1-y(w, x)$ is the probability that x belongs to the other class [15].

The loss function is used as the basis for updating the weights in the method through a method called gradient descent. With gradient descent, the weights are changed based on the gradient of the loss function with respect to the weights. The update rule for a set of weights w_t at step t of the training process, using only a single object, e.g. an image, x as the input, is given by

$$w_t = w_{t-1} - \eta \nabla_w L(w_{t-1}, x) \tag{8}$$

where η is a parameter called the learning rate, that affects the speed at which the loss function converges to a minimum. A too high learning rate can lead to over-correcting for errors and shooting past the minimum, and too low learning rate can lead to small changes in weights and thus a slow convergence. The update process in Equation 8 can be repeated for some input x until the loss function converges [16, 21].

The gradient descent algorithm above can be modified to take a different number of inputs and evaluate the loss function for all of them before updating the weights. For a set of inputs x with N elements, the weight update can be performed either after all N objects have been evaluated using the loss function, or after subset of P objects have been evaluated, where $1 \leq P \leq N$. P is called the batch size, and the weight update is then given by

$$w_t = w_{t-1} - 1/P \sum_{i=1}^{P} \eta \nabla_w L(w_{t-1}, x_i). \tag{9}$$

If only a subset P of elements are used to update, the weights are updated once after the first batch of P objects, again after a second batch of P more objects, and so on until all N objects have been evaluated. This process of incrementally updating the weights on the entire input x is called an epoch [15].

Variations and extensions of the above gradient descent algorithm exist that include variable learning rates or that make use of not just the gradient of the previous step, but a running average of all previous gradients. One commonly used example of this is the Adam optimizer [22]. Adam updates each weight individually, using a running average of both the gradient and the square of the gradient of the weight. This gives Adam an individual learning rate for each weight, so that each weight is updated based on how incorrect it itself is.

When designing and training an ANN, there are many parameters of the network that do not have to take any specific values, but can be set during creation of the model. These are called hyperparameters. Some examples of hyperparameters are the learning rate of the Adam optimizer, or the number and size of the filters in a convolutional layer. For a given architecture and task there are typically some combinations of these hyperparameters that are better suited for optimal performance, though there is no a priori way of knowing which set of hyperparameters is preferable. Due to this it is often done by trial and error to observe which parameters give the best performance on some subset of the training data.

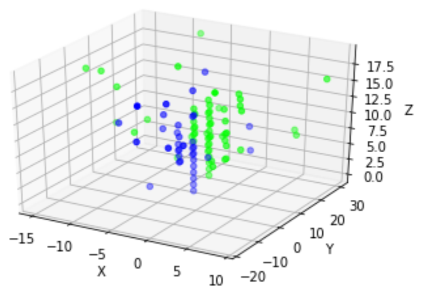
3 Physics Processes and Simulation Samples

This thesis aims to study how machine learning can be combined with data from the LDMX electromagnetic calorimeter for purposes of classifying and reconstructing events or showers. Here we have chosen to focus on two different scenarios to compare and evaluate the performance of using neural networks for these tasks for LDMX. All data used in this thesis have been simulated using Geant4 [23], with a custom build featuring the setup of the LDMX experiment described in Section 2.3 [2].

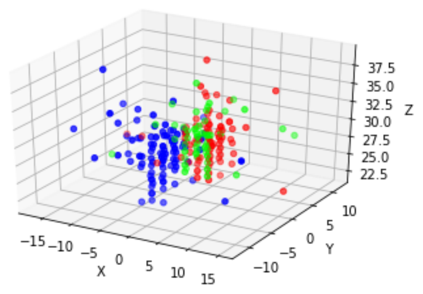
The first scenario that we consider consists of events where a single electron is incident on the target, with an initial energy of 4 GeV. The electrons are created 10 mm upstream of the target with initial momentum only in the beam direction, such that they impact the target perpendicularly. The initial position of the electron is uniformly distributed within a beamspot of size 20×80 mm in the plane perpendicular to the beam direction. From there we consider two possibilities, either the electron passes through the target without interaction and proceeds to shower in the ECal, or the electron undergoes a bremsstrahlung interaction in the target and both the electron and photon shower in the ECal. In the first case no cuts are made on the events, as long as the electron does not interact in the target the event is accepted. In the second case the event is only accepted if the recoil electron has an energy of less than 1500 MeV following the bremsstrahlung in the target, and thus the photon has an energy of at least 2500 MeV. This filtering is done during event generation, so only events that pass the filter on the energy of the recoil electron are saved. An example of what an event where an electron underwent bremsstrahlung looks like in the ECal is shown in Figure 13a. The blue markers are hits where the electron itself, or a particle other than the bremsstrahlung photon that has the primary electron as an ancestor, has contributed more than half of the energy of the hit. The green markers are hits where these particles contributed none or less than half of the energy.

The second scenario that we consider consists of events in which 1,2, or 3 electrons are incident on the target, all with an initial energy of 4 GeV and no momentum in the x- and y-directions. As in the first scenario, the electrons are created in a beamspot of size 20×80 mm at a distance of 10 mm upstream from the target. The events were generated with either 1,2, or 3 incoming electrons, and an event was only accepted if every initial electron underwent bremsstrahlung in the target, with the same criteria on the energies of the individual recoil electrons as above. This creates 3 different classes of events to classify and study, based on the number of primary electrons generated. An example of an event with 3 primary electrons is shown in Figure 13b. As in Figure 13a, the marker colors indicate which of the 3 primary electrons or particle with the primary electron as an ancestor contributed the most energy in the hit. No distinction is made between a recoil electron and its associated bremsstrahlung photon.

These two different scenarios create two different signatures in the ECal, and for that reason we have chosen to compare them to each other to gain an understanding of how machine learning can be used to study and reconstruct events in the ECal. These scenarios do not have any direct relation to the primary signals and backgrounds that concern LDMX, as the benchmark scenario for LDM that LDMX is looking for would only contain a single shower in the ECal coming from the recoil electron, in Phase one. What we are attempting to study more generally is the usefulness and limitation of the combination of the ECal with



(a) Plot of an event with a recoil electron and bremsstrahlung photon. The electron has an energy of 1.1 GeV. Blue markers correspond to electron hits, green to photon.



(b) Plot of an event with three primary electrons. Each color corresponds to a different primary electron and its associated bremsstrahlung photon.

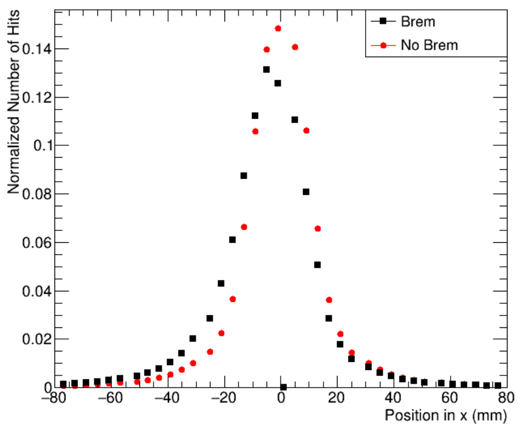
Figure 13: The markers represent hits in the ECal readout cells. The hit in the readout cell has been assigned to the particle that contributes the largest fraction of the energy of a hit in case of overlap between showers. The y- and y-axes are not to scale, while the z-axis is in ascending order of ECal layers starting from the layer closest to the target. The opacity of the markers indicate how far the points are from the point of view of the viewer, more transparent markers are further back in the image.

machine learning. While LDMX is intended to initially run with only one primary electron, it is likely that the experiment will be run with more than one electron per bunch in future phases, and so multi-electron events warrant similar study.

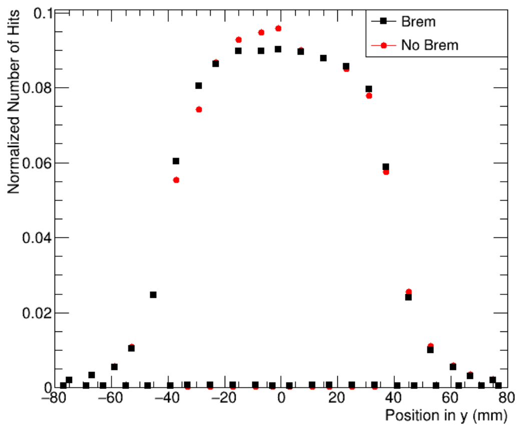
Events with only a single primary electron will typically have one or two objects in the calorimeter in our case, these being either a single 4 GeV shower coming from the electron, or two separate showers whose total energy should add up to 4 GeV. The discriminating variables between these two types of events could depend on the type of neural network used, as different networks learn based on different features of the input, e.g. a CNN is likely to pick up on specific spatial features that distinguish the types of events.

In comparison, events with one or more primary electrons will in our case have two or more objects in the calorimeter, and as many six separate showers in the events where three primary electrons undergo bremsstrahlung. While there are likely other discriminating variables that exist in this scenario, one big difference compared to the single electron is that the three different classes of events will have different total energies in the calorimeter. All primary electrons were generated with the same initial energy, and so the total energy in the calorimeter should be close to a multiple of 4 GeV.

In the following two subsections we will show some plots of the variables that characterize the samples used in this project and that we can use to study the performance of the machine learning models.



(a) Position of hits in x



(b) Position of hits in y

Figure 14: Histograms showing the x- and y-distribution of all of the ECal hits in all events. Both histograms show the distribution of hits for both Brem events and No Brem events.

3.1 Single Electron Studies

3.1.1 Spatial Features of Readout Hits

The input to the neural networks used in this project is based on the simulated readout hits in the ECal. How this is done for different models is explained in Section 4. Some variables that are relevant to the readout hits in an event are potentially useful discriminators that the neural networks might use, since the readout hits make up the foundation of the data used for training.

The plots in Figure 14 show the distributions in the x- and y-coordinates and 15 shows the distribution of hits in the layers of the ECal of all of the hits in the ECal in all of the events in one data set. Each plot shows two curves, one is the distribution of hits in events where a bremsstrahlung interaction took place, labeled "Brem", and the other is the distribution of hits in events with no bremsstrahlung, labeled "No Brem". Unless otherwise noted, the error bars in the following figures are the statistical errors given by the square root of the bin content. The total number of events in the data set is 96228, with 48220 Brem Events.

In the x-direction there is a clear difference between the two distributions, as the Brem events have a peak in negative x, while the No Brem events have a peak closer to $x = 0$. This is due to the presence of the magnetic field after the target, which deflects electrons towards the negative x-direction and this deflection is larger for recoil electrons that have lost energy due to the bremsstrahlung. The mean position in x for the Brem events is $x = -4.23$ mm, and for the No Brem events is $x = 0.01$ mm.

In the y-direction more of the hits in the No Brem events are near $y = 0$ than in the Brem events. Note that two out of three bins in the y-direction have a significantly lower bin content than other bins. This is because the readout cells in the left and right side hexagons in Figure 5 have a different set of y-coordinates from those in the central hexagon. Most events will be located in the central hexagon due to the beamspot size relative to the size of the ECal, and so these cells will have the most hits, and these are the cells with most of the

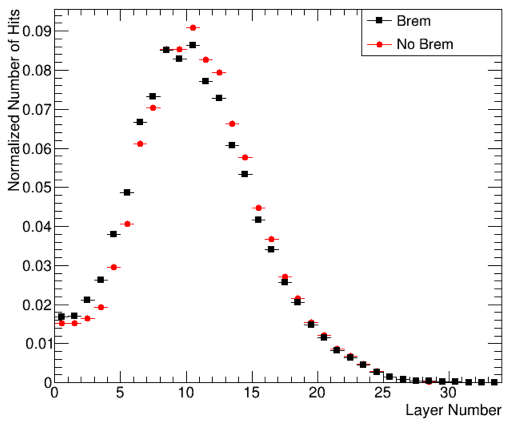


Figure 15: Histogram showing the distribution of the number of hits in each layer of the ECal for both Brem and No Brem events.

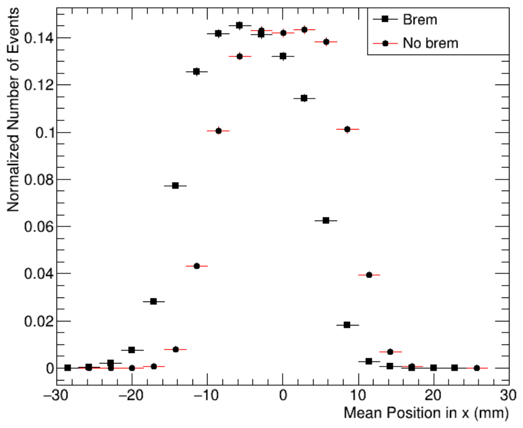
hits in the y-direction. The mean position in y for the Brem events is $y = -0.19$ mm, and for the No Brem events is $y = 0.09$ mm.

In the z-direction there are more hits early in the calorimeter from the Brem events than from the No Brem events. Since the No Brem events are generated with a single 4 GeV electron, the calorimeter will most of the time contain only a single shower close to 4 GeV in energy. This single shower will go deeper into the calorimeter because of the higher energy, in comparison to the Brem events with two lower energy showers. The mean layer number of all of the hits is 10 for both types of events.

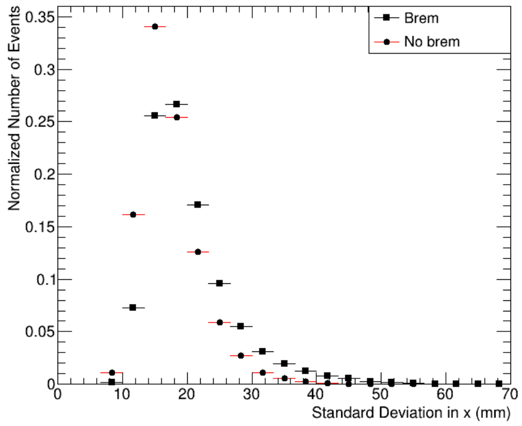
The distribution of all hits in the entire data set is not a variable that discriminates the two types of events that we consider in this scenario, but some variables that characterize individual events might do so. Figures 16 and 17 show the mean position of the hits per event in the x- and y-coordinates while Figure 18 shows the mean layer number hit and standard deviation in layer number per event.

In the x-direction the mean position of the hits is at more negative x-value in the Brem case than in the No Brem case, similar to the distribution of the hits. The standard deviation on the other is higher in the Brem case than the No Brem Case, indicating that the hits are more spread out. This is due to the change in trajectory and deflection that the recoil electron experiences following the bremsstrahlung interaction, which will make the electron shower in a location that is often separated from where the photon showers, and so even if there are not two distinct showers there will be a larger spread among the hits. The flatter peak of the mean position in the No Brem case occurs due to the fact that there is little deflection or change in trajectory of the primary electron since it does no interact in the target and experiences less deflection in the magnetic field. This contains a larger proportion of the events within the beamspot, which in the x-direction is located between $-10 \leq x \leq 10$ mm, assuming that the shower development happens roughly centered around the incoming position of the electron.

In the y-direction the mean position of the hits is more uniformly distributed within the beamspot, which is located between $-40 \leq y \leq 40$ mm, in the Brem case than in the No Brem case. The standard deviation is smaller in the Brem case than the No Brem case. There

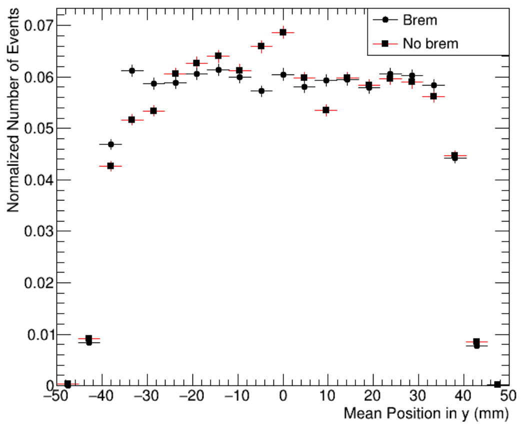


(a) Mean position of hits in x per event

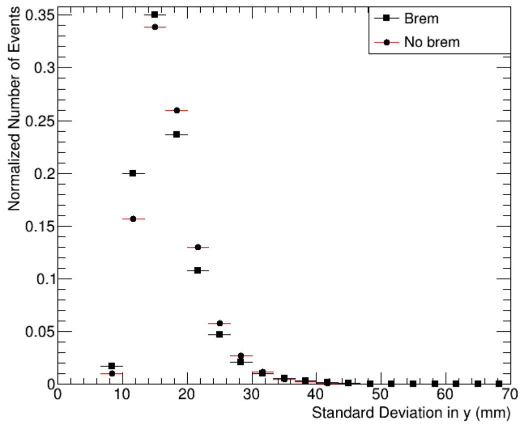


(b) Standard deviation in position of hits in x per event

Figure 16: Histograms showing the mean and standard deviations of the positions of the hits in the ECal per event in the x-direction for both Brem and No Brem events.

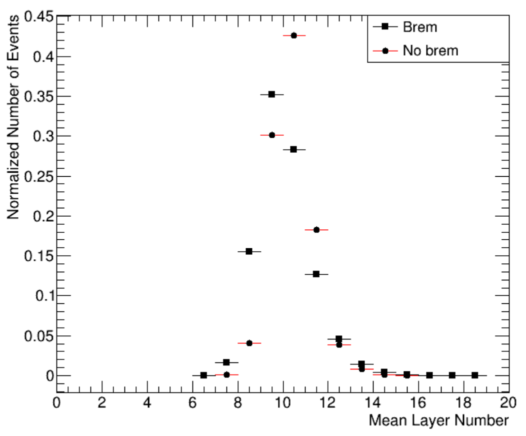


(a) Mean position of hits in y per event

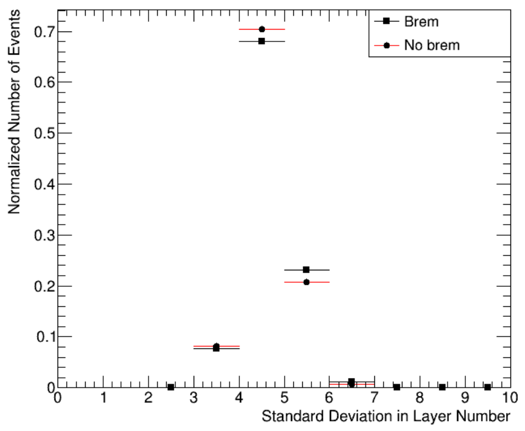


(b) Standard deviation in position of hits in y per event

Figure 17: Histograms showing the mean and standard deviations of the positions of the hits in the ECal per event in the x-direction for both Brem and No Brem events.



(a) Mean layer number of hits per event



(b) Standard deviation in layer number of hits per event

Figure 18: Histograms showing the mean layer number and standard deviation in layer number of the hits in the Ecal per event for both Brem and No Brem events.

is no deflection from the magnetic field in the y-direction, and so the separation between the recoil electron and the photon depends only on the initial trajectory change due to the bremsstrahlung.

In the z-direction the mean layer hit occurs slightly earlier in the Brem case than in the No Brem case, and the standard deviation in the number of layers hit is slightly larger in the Brem case.

In addition to the spatial coordinates of the hits, the total number of readout hits in the Ecal might affect the ability for the neural networks to distinguish between the two types of events. The plot in Figure 19 shows the total number of hits for the two types. Brem events typically have a larger total number of hits than No Brem events, due to the fact that the single electron with its higher energy will deposit more energy in the ECal per hit, which results in fewer total hits.

Whether the shape and distribution of hits in the ECal are relevant to the ability for a neural network to classify between the two types of events depends on if the neural network can utilize the shape of the event in the learning process. Since both CNNs and GNNs are capable of detecting spatial features in an image, they should be able to utilize the potentially distinct spatial features of the two types of events that we consider here. The same is not necessarily true for the number of hits, as the total number of hits in the event is not necessarily propagated through the network by itself. In a CNN, while the total number of hits is relevant to the input, it does not necessarily keep track of that information through the process of propagating the input through the network, as the original number of hits in the input is likely to be distorted due to the convolution and pooling operations that extract information from a large region of the input. In a GNN, the number of input vertices is fixed regardless of the number of readout hits in an event, and so the number of hits might play a role in the classification process if the number of hits is lower than the fixed input size.

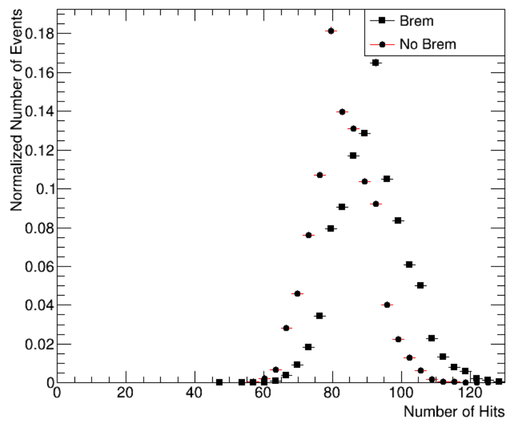


Figure 19: Histogram showing the total number of hits per event for both the Brem and No Brem case.

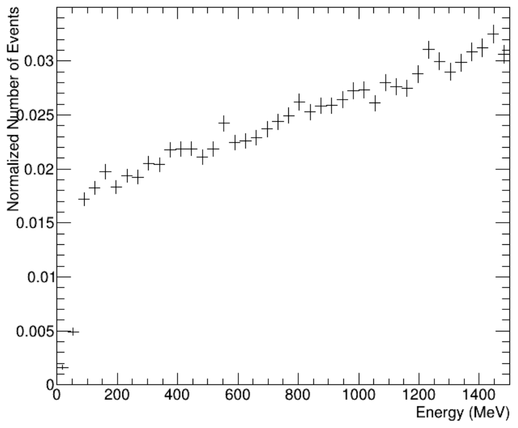
3.1.2 Features of Brem Events

There are some features of an event that do not directly relate to the input data used in training that would still cause differences between the two types of events. These features would mainly be properties associated with the recoil electron or bremsstrahlung photon in the Brem events, since there would be very little difference between the primary electrons in No Brem events other than initial position. A difference in initial position between two primary No Brem electrons will ideally not be an issue, as the shower development in the ECal would be similar, and so the main difference between the two events is the positioning of the shower. Since both CNNs and GNNs are designed to exploit translational invariance in the input they should be able to detect two similar showers in different positions as the same type of event. The other properties that might affect the accuracy of the models are then those that create distinctions between the two types of events and these are primarily present in Brem events.

The distribution of the energies of the recoil electron are shown in Figure 20a. The distribution increases slightly towards higher recoil energies, as it is less likely for higher energy photons to be produced in the bremsstrahlung interaction in the target.

The deflection angle of the recoil electron at the face of the ECal is shown in Figure 20b. Most recoil electrons undergo only a small deflection, the majority of events have a deflection angle smaller than 0.1 radians. This small deflection is due to the production of a massless photon in the bremsstrahlung reaction, if a massive particle were produced in the reaction instead, as would happen in the signal events LDMX is looking for, then deflection would be significantly larger, as seen in [2]. Further, even the deflection in the magnetic field following the target does not create a large deflection angle relative to the initial position of the recoil electron, as can be seen in Figure 14a, where the difference between the Brem and No Brem peaks is 4 mm.

While the deflection angle shows how much the recoil electron has been deflected from the position where it interacted in the target, it does not say anything about the position of the recoil electron relative to the photon. Figure 21 shows the transverse separation between the


(a) Histogram showing the distribution of the energy of the recoil electron in events where a bremsstrahlung interaction occurred.

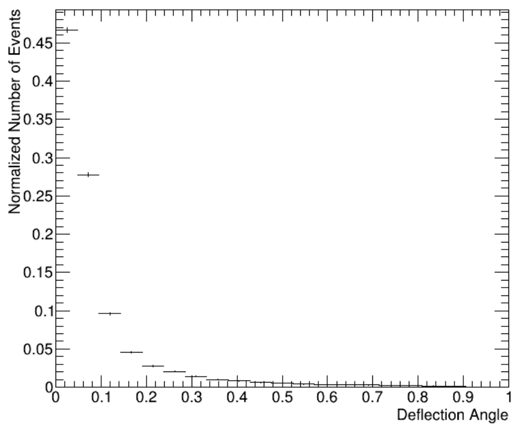

(b) Histogram showing the distribution of the deflection angle of the recoil electron relative to the position where it interacted in the target.

Figure 20

recoil electron and photon on the face of the ECal, so the distance between their respective entry positions.

Other than the properties of the electron itself, one feature that might be useful in distinguishing between events is the number of readout hits in the ECal that the electron causes. Figure 22 shows the number of readout hits in the ECal per event in black, where the recoil electron contributed the majority of the energy of the hit. This means that in case of overlap between the two showers coming from the electron and the photon, a hit will be designated as coming from the electron if the fraction of energy coming from the electron is greater than 0.5. For comparison the total number of hits in No Brem events the total number of hits in Brem events are included in Figure 22. The recoil electron thus typically contributes less than a quarter of all of the hits in Brem events. While the neural network models do not have knowledge of which hits come from which particle, it is potentially something that could be a discriminating variable between the two types either because of differences in structure of the events or because the model might learn that these are two different types of events.

The energy of the recoil electron can be anywhere in the range $0 \leq E \leq 1500$ MeV, though how the energy of the recoil electron affects accuracy might not be trivial, as there are also relations between how energetic the electron is and how many hits it contributes, or how much it separates from the photon. Figure 23 shows two 2D distributions of the energy together with the deflection angle and the number of electron hits respectively. This shows that the events with a large deflection angle also have low energy recoil electrons and further that low energy recoil electrons tend to cause fewer hits.

Similar to the spatial features discussed above, whether or not these variables are relevant to the training of the neural networks depends on how much they affect the input. None of the variables discussed in this section are directly given as input to the models, but since they affect the spatial distribution of the hits they might still have indirect effects on the

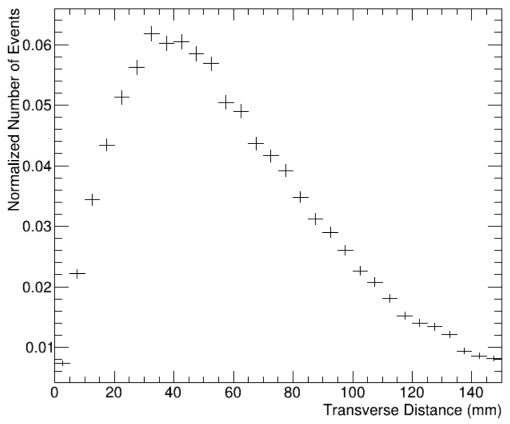


Figure 21: Histogram showing the transverse separation between the recoil electron and the photon at the face of the ECal in Brem events.

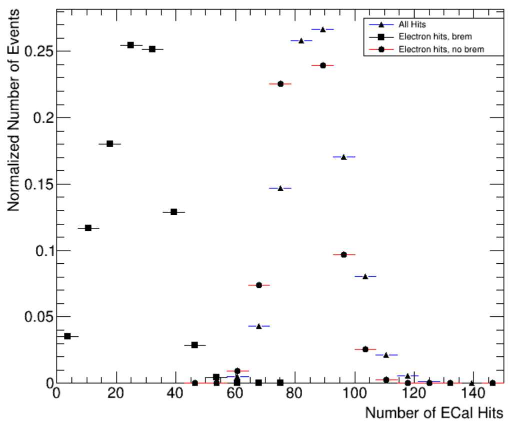
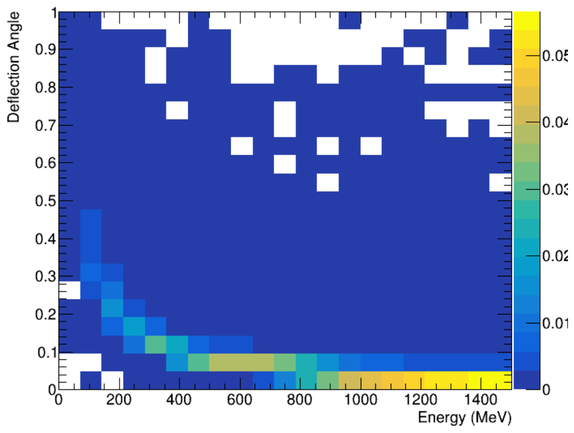
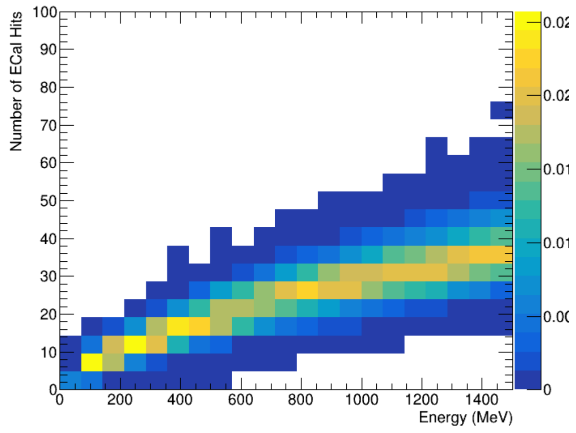


Figure 22: Histogram showing the number of hits caused by the recoil electron in Brem events together with the total number of hits in those events and the number of hits in No Brem events. Hits are designated as being caused by the electron if the recoil electron itself or a particle coming from it, other than the bremsstrahlung photon, contributed with more than half of the total energy of the hit.



(a) 2D distribution of the energy and deflection angle



(b) 2D distribution of the energy and number of hits

Figure 23: 2D distributions showing the energy of the recoil electron together with the deflection angle of the electron at the face of the ECal relative to where it interacted in the target and the number of hits where the electron contributed more than half of the total energy of the hit.

accuracy at which the models classify events. In particular, a larger separation between the two particles in the ECal will likely correspond to a larger separation between the showers that are created in the ECal. This could lead to a higher accuracy for these events, as a larger separation between the showers would lead to two different objects with potentially different structure in the input data, in comparison to a single large shower or two showers with more overlap where there might only be a single object.

3.2 Multi Electron Studies

Unlike in the single electron case, the main variable that differs between events with multiple primary electrons is the number of ECal hits in the event. As the number of primary electrons increases, the number of hits will also increase, and this distribution of hits is shown in Figure 24. There are 3 distinct peaks in the figure, each due to the number of primary electrons, with slight overlap in the number of hits between the 2 and 3 electron peaks. The primary electrons were generated with initial positions uniformly distributed within the 20×80 beamspot, and in the events with 3 electrons there is a higher chance that there will be overlap between some of the 6 showers in the ECal, leading to the overlap between the two peaks in Figure 24. The total number of events in the data set is 63661, with 20543, 22854, and 20264 1, 2, and 3 electron events respectively.

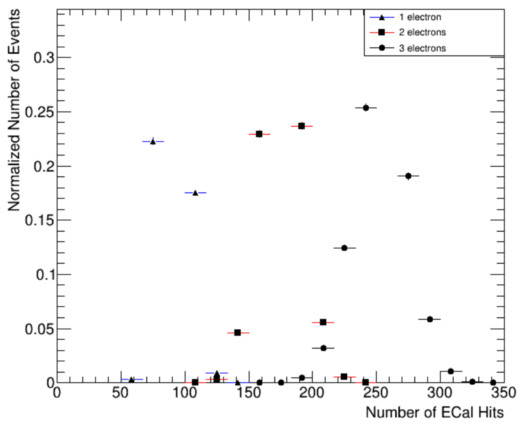


Figure 24: Histogram showing the number of hits in events with either 1, 2, or 3 primary electrons.

4 My Network and its Duties

4.1 Training and Testing Data

The data used for this project was generated using the Geant4 framework together with a custom built modification by the LDMX collaboration [2, 23]. For this project, two components of the simulated events were used in different stages of the work. The training process requires ECal data, and for this simulated hits in the ECal were used. To analyze and better understand the events, the underlying particles in the event, called "simparticles" were used.

The ECal hit data consists of simulated hits in the calorimeter that include information about the hit itself and the simparticles that contributed to it. The primary information in the hits themselves used here is the position of the cell that was hit and the energy deposit in the cell. The simulated energy deposited in the cell further undergoes a simulated digitization process that converts the deposit to a digital hit.

The simparticle data consists of simulated particles with kinematic data, particle id, particle ancestry, and information on the interaction of a particle with simulated parts of the the detector. Simparticle kinematics were used to determine the kinematic plots in Section 3. The ancestry of a simparticle was used to find out whether the particle was created in the electron or photon shower in the single electron studies and from which primary electron the particle was created in the multi electron studies. By combining the simparticle interaction and ancestry information with ECal hit information on particle contributions the showers shown in Figures 13a and 13b were obtained. Using this information the fraction of energy contributed to each hit from the respective primary particles in the event was also found for the segmentation studies.

The two types of neural networks used in this project, Convolutional Neural Networks and Graph Neural Networks, were both trained to perform the same tasks. In both cases the data used for training and evaluation was based on the simulated hits in the LDMX ECal and their energy deposits, but due to the different structure of the two neural networks the data was treated differently.

A single input to a CNN in 3D takes the form of a fixed size 3D tensor, in this case the 3 spatial dimensions (x, y, z). Each point in the tensor has one or more values, but is not encoded with the actual position in the ECal. The CNN does not make use of the Cartesian coordinates or absolute position of each hit in the ECal, only the relative positions of each hit compared to the others in the convolution operation. To create the 3D tensor used for the training, the Cartesian coordinates of each hit in an event were transformed into indices used to enumerate the tensor. As an example, a hit with coordinates $(x, y, z) \approx (4.3, 7.5, 223.8)$ mm would be transformed to $(1, 3, 1)$, where the first two indices roughly correspond to the number of the cell relative to the central cell located $(x, y) = (0, 0)$ in Figure 5 and the last index is the layer number. For the x- and y-coordinates this was done by dividing the value in Cartesian coordinates by 4.33 mm and 2.5 mm respectively and rounding down to the nearest integer if the value is positive, and up if it is negative. While this does not guarantee that there were no separate hits that were given the same index, no such case was found in the $8 \cdot 10^6$ ECal hits present in the single-electron data set. The allowed z-values in the ECal correspond to the 34 layers in the ECal, and so the transformation of the z-coordinates was achieved by simply mapping each allowed value to the corresponding layer number. These indices were then used to fill the fixed size 3D tensor such that the value of each element in the tensor is equal to the energy deposition in the cell it corresponds to if there was a hit in the cell, and 0 otherwise. This gives an input tensor for the CNN where the important elements are the hits in the ECal.

In comparison, a single input to the GNNs used here takes the form of a fixed number of vertices in the graph, where each vertex corresponds to a hit in a readout cell. Each vertex has a corresponding feature vector given by the Cartesian coordinates of the hit and the energy deposit of the hit. Since the number of vertices is fixed, an event with fewer hits than the predetermined number of vertices needs to be padded in order to have the correct number of vertices. This padding was done by creating enough vertices to reach the required number and giving them Cartesian coordinates corresponding to position far outside of the ECal and an energy deposit of 0, in order to minimize the chance that these padding hits would be among the nearest neighbors of any actual ECal hit. Similarly, an event with more hits than the number of vertices would need to cut some of the hits out to create the required graph. In each event the hits were sorted in descending order of energy deposit, and so the hits that were cut were those with lowest energy in each event.

For the CNN input size, a 3D tensor of size $(45, 70, 35)$ was chosen, which would roughly correspond to a slice with size 199×175 mm^2 in the xy-plane. As the size of the tensor was fixed, some hits in the ECal might be excluded from the tensor in case they were located far away from the center of the ECal in the x- and y-directions. This choice of tensor size lead to only 0.6% of total hits over all events to excluded. A smaller tensor could have been used, but for example a tensor of size $(40, 60, 35)$ excluded 1.5% of all hits while providing a small speed up and a leading to a decrease in performance.

Two different types of GNNs were trained, one for classification based on the EdgeConv operation, which was trained on both the single and multi electron data sets, and one for segmentation based on the GravNet layer, which was only trained on a subset of the single electron data set. The GNN for classification was trained to classify the events into the different types described in Section 3, while the GNN for segmentation was trained to classify each hit in an event as belonging to one of the showers present in the event.

For the GNN used for classification, the number of vertices was chosen to be 100 for the single-electron classification and 300 for the multi-electron classification. The excluded hits were the least energetic hits in each event with more than 100 or 300 hits respectively. Both a smaller and larger number of hits were tested, with no significant improvement in classification performance when more hits were used, and faster training but worse performance when fewer hits were used. For the GNN used for segmentation, the number of vertices was chosen to be 120 for the single-electron segmentation and 340 for the multi-electron segmentation. As the primary goal of this network was to classify each hit in an event, a larger number of hits were chosen.

Both types of neural networks use the energy of the readout hits as the only non-spatial feature of the hits. Another potential feature that could be used for both classification is the time at which the hit was registered. This was tested for both the CNN and GNN used for classification and was found to lead to worse performance using the same architecture and was thus discarded as a feature. Instead of using timing as a feature, it could potentially instead be used in a similar way to how the spatial information is currently used, which is to calculate proximity between hits in the GNN architectures.

For the training and evaluation processes two separate sets of data of roughly equal size were generated. For networks used for classification, the training data was created to have an equal proportion of each class available for training in order to not bias the model towards one class during training. The training data set was split into two parts, one for training and one for testing and evaluation. The fraction of the full training data set that was used for training was about 90%. The remaining 10% was used to evaluate the model both during the training and after the model had been fully trained in order to select the best model for further evaluation. In this way the model performance could be tested on a previously unseen sample of the data. This testing sample was randomly sampled from the data set generated for training. During the training process, each model was evaluated in the middle of the training process on this testing data at the end of each epoch. In cases where the model performance peaked before the full training process was completed, after e.g. 3/4 of all epochs were finished, then the best performing version of the model during the entire procedure was selected as the model to be evaluated. This process was repeated with varying architectures in order to find the overall best performing model for each task. These models will be detailed in the following subsections.

The overall best performing model was then used to make predictions on the second data set, in order to further evaluate the performance of the model on a larger data sample, roughly of the same size as the original training data set. For each task, the model was given unlabeled data and asked to estimate what the data is. For classification, this meant giving the model an event as input and as output the estimate of the probability of the specific event belonging to each of the different classes. The class with the highest probability was chosen as the predicted class, regardless of the magnitude of said probability. For segmentation, the model was given an event as input and asked to estimate the probability of each hit in the event belonging to one of the different classes. The hits were assigned to belong to the particle with the highest probability, regardless of magnitude.

The model performance during training was evaluated according to the classification accuracy of the model, so the model with the highest classification accuracy during training and testing was the one chosen for further evaluation. The classification accuracy is simply the fraction of samples that were correctly labeled, whether this was events or individual hits. To further understand the classification results of the models, the predictions were analyzed in terms of the data presented in section 3. The results of this analysis are presented in Section 5.

4.2 Machine Learning Models

All models presented here were created, trained, and evaluated using the Tensorflow and Keras libraries [24, 25].

The models that were used in this project were based on models that have enjoyed success for other tasks. Convolutional neural networks have been successful in a wide variety of imaging related tasks within the field of computer visions and for top tagging and particle identification within particle physics [26, 27, 28]. The CNN models used here were not directly based on models used in previous research, but were designed to be simple models built using the existing infrastructure for CNNs in Tensorflow and Keras. Graph Neural networks have been used for identification and clustering in particle physics. The GNN models used here were based on the ParticleNet used for jet tagging and the GravNet used for clustering and reconstruction [29, 20].

4.2.1 Convolutional Neural Network Models

The basic architecture of both CNNs used in this project is shown in Figure 25. This CNN consists of a single convolutional layer as the first layer following the input. The two CNN models used 32 filters of size $(8, 8, 8)$ in the convolutional layer. Following the convolution is a single max pooling layer of size $(5, 5, 5)$, thus decreasing the size of the input by a factor 5 in each dimension. After the max pooling the output is flattened, meaning that all of the feature maps are concatenated into a single vector, which is then fed through a dense layer. For the multi-electron samples the number of nodes in this dense layer was 100, and for the single-electron samples the number of nodes used was 128. In both models the dense layer was followed by a dropout operation, which for each node has some probability of setting the value of that node to 0 during training. The probabilities used were 50% and 16% for the multi-electron and single-electron samples respectively. The last layer of the model is the output layer, with a number of nodes equal to the number of classes that the model is considering, 3 classes corresponding to the number of primary electrons for the multi-electron samples and 2 classes corresponding to whether the event is of type Brem or No Brem for the single-electron samples. The convolutional and dense layer were both followed by the Rectified Linear Unit (ReLU) activation function [30], which is given by $f(x) = max(0, x)$. The output layer is followed by a softmax activation function in order to translate the output to probabilities between 0 and 1, where each output node i has an output given by $y(x)_i = e^{x_i}/\sum_j e^{x_j}$ and j runs over all output nodes, ensuring that the sum of all outputs is 1.

Both models were trained using the Adam optimizer, with a learning rate of 0.0001 and 0.0004 respectively, and a cross entropy loss function for 3 and 2 classes respectively. Batch sizes of 100 and 32, respectively, were used and both models were trained on the training data set for 20 epochs.

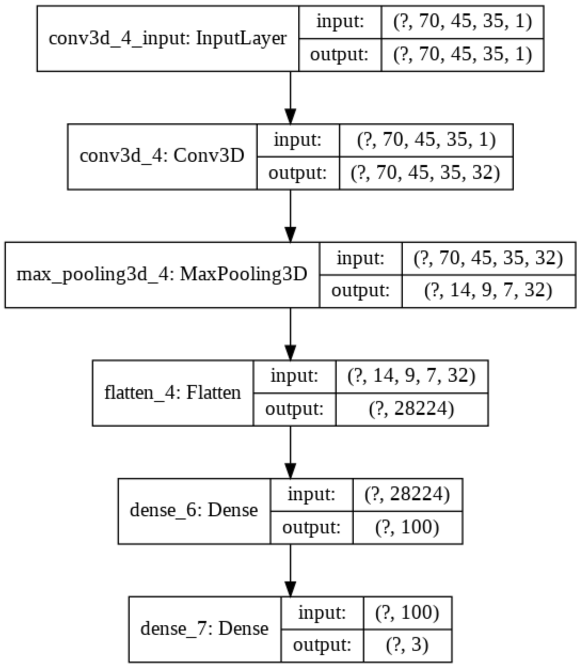

conv3d_4_input: InputLayer
input: (?, 70, 45, 35, 1)
output: (?, 70, 45, 35, 1)
conv3d_4: Conv3D
input: (?, 70, 45, 35, 1)
output: (?, 70, 45, 35, 32)
max_pooling3d_4: MaxPooling3D
input: (?, 70, 45, 35, 32)
output: (?, 14, 9, 7, 32)
flatten_4: Flatten
input: (?, 14, 9, 7, 32)
output: (?, 28224)
dense_6: Dense
input: (?, 28224)
output: (?, 100)
dense_7: Dense
input: (?, 100)
output: (?, 3)

Figure 25: Chart showing the basic architecture used for both 3D CNN models. The boxes on the right-hand side show the input and output sizes of each layer, where the question mark represents the batch size, which is set later. The convolutional layer takes the input of the given size and gives an output for each of the 32 filters. The pooling layer decreases the dimensionality of each of its input by a factor 5 in each spatial dimension. The flattening layer concatenates the output feature map into a single vector which is used as input to a dense layer, followed by the output layer.

The architectures given in Figure 25 were those that were evaluated using the separate evaluation data. In the case of the multi-electron samples, the model presented above was found to have such a good performance after selecting the model that only a small number of parameters were changed and tested, without giving a significant change in performance. In the case of single-electron samples, the model above was the one that was found to have the best performance after a tuning of some of the parameters of the model. Due to resource considerations, only a selection of parameters could be tested, and without considering a large number of different possibilities. The main parameter that was tuned was the learning rate of the Adam optimizer. Other than this, different number of filters, filter sizes, the number of nodes in the dense layer, and adding another convolutional layer was tested.

4.2.2 Graph Neural Network models

The GNN used for classification was based on the ParticleNet in [29], using the EdgeConv operation described in [19]. The EdgeConv was used in the GNN in the form of an EdgeConv block, shown in Figure 26. The block takes N input vertices with f features each in the feature vector F. Using a multi-layer neural network similar to what is shown Figure 7, the edge features are computed as $e_{ij} = h_{\Theta}(x_i, x_j - x_i)$ for the k nearest neighbors of the vertex, as in [19, 29]. The number of layers and nodes in this multi-layer network, and the number of nearest neighbors considered, were parameters that were varied during training to obtain the optimal performance. The final EdgeConv block was followed by a dense layer and finally an output layer like in the CNN models.

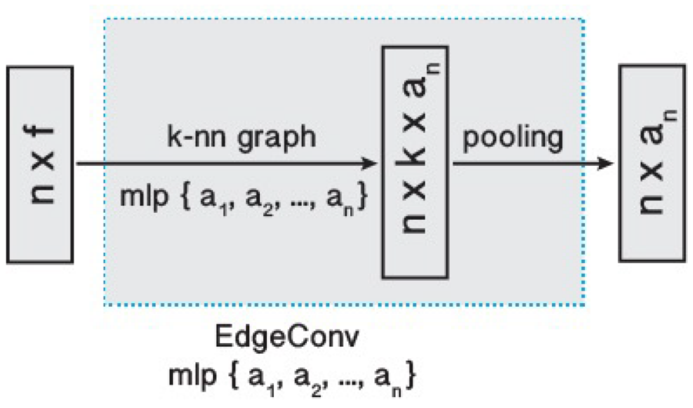


Figure 26: Schematic showing the EdgeConv Block used in the GNNs for classification. The block takes n input vertices with f features each, and passes it through a multi-layer neural network that computes the edge features at each vertex for k nearest neighbors. This is aggregated to an output of n vertices with a_n features each [19].

The best performing model for binary classification was one with 3 EdgeConv blocks stacked together, each with 3 layers in the multi-layer neural networks. These blocks all considered 16 nearest neighbors for the edge features. The number of nodes in the first block multi-layer neural network was $(64, 64, 64)$, $(128, 128, 128)$ in the second and $(256, 256, 256)$ in the final EdgeConv block. This was followed by a dense layer with 256 nodes, and a dropout probability of 0.14 applied on each node. All activation functions before the final layer were ReLU, and in the final layer a softmax function was applied to obtain probabilities for each class.

The GNN used for segmentation was based on the GravNet layers in [20]. The GravNet architecture uses 4 GravNet layers followed by a concatenation of all of the features extracted from the GravNet layers, and two dense layers with 128 nodes in them before the output layer. Each GravNet layer considered 40 neighbors at each vertex. The output layer in this model has 2 or 3 nodes per input hit, as each hit is assigned probabilities of which showers they belong to.

Both types of GNN and all models were trained using the Adam optimizer. The ParticleNet based models used an initial learning rate of 0.0004 for the binary classification and an initial learning rate of 0.00007 for the multi-electron classification. These models used the same kind of cross entropy loss as the CNNs. The GravNet models both used initial learning rates of 0.0008, but with a sparse cross entropy loss function, which applies the cross entropy loss not to the entire sample, but to individual vertices in the graph, as the models were tasked to label the hits in each event.

As with the CNNs, the models presented here were those that were found to have the best performance. In the multi-electron classification case the GNN performance was again found to be so high that very little further tuning was done. With the other models, the main tuned parameter was again the learning rate of the Adam optimizer, though some other parameters

Table 1: Training and evaluation times of the different models per event per epoch and the approximate size per event of the training and testing data.

Model and data set	Training time (ms)	Evaluation Time (ms)	Event Size (MB)
GNN 1E	32	6	0.003
GNN 3E	45	15	0.009
CNN 1E	77	35	0.8
CNN 3E	122	36	0.8
GravNet	12	2	0.003

that were tested were the number of nearest neighbors considered, and the number of nodes in the respective dense layers of the models.

4.3 Computing Considerations

For this project, all computing was done on the Aurora cluster at the center for scientific and technical computing at Lund University (LUNARC). The cluster has computing nodes consisting of 2 Intel Xeon E5-2650 v3 processors with 10 cores each.

The different models used in this project have different computing performances, which impact the usability of the models in real world use cases. Table 1 shows the training and evaluation times per event per epoch, and the size per event of the input data that was used for the different types of models and data sets that were discussed above. The difference in training and evaluation times is due to the fact that during training all the weights in the networks have to be computed and updated based on the loss function, while during evaluation only the output is required from the network.

All times displayed in Table 1 were obtained when training and evaluating on 20 cores. The Graph Neural Network models, and in particular the GravNet based models, are significantly faster than the CNN models, and require two orders of magnitude less memory per event. Due to the memory footprint of the events used to train the CNN models the total number of events used for training was limited. The storage of the events outside of training requires storage space and during training the process needs to load a large number of events into memory at any given time, while updating the weights of the model. This same limitation is not present for GNN models, and so they could have been trained on a larger number of events, or for a longer time, while consuming fewer resources than the CNN models. While this was not done, it is something that could benefit the performance of the GNN models.

One notable difference between the CNN and GNN models, other than the faster training time of GNN models, is that the CNN model evaluation time does not differ by a significant amount between the two data sets. This is because the CNN models take a fixed size input regardless of the type of event, and performs the same number of operations each time. The GNN models take different numbers of input hits depending on the data set, in order to include as many hits as possible, and will so have to perform more computations when evaluating events with more hits.

A CNN based model was also tested for segmentation, but due to the even larger resource requirements this was not further studied, and only the GravNet model was used for this study.

5 Results

5.1 Single Electron Studies

The results of the predictions from the best performing CNN and GNN models on the single-electron samples are presented in the following plots. The performance of these models is studied based on how well they predict events as belonging to the correct class. This is evaluated using the True Positive Rate (TPR) and True Negative Rate (TNR) of the models, defined as

$$TPR = \frac{TP}{P}$$
$$TNR = \frac{TN}{N}$$

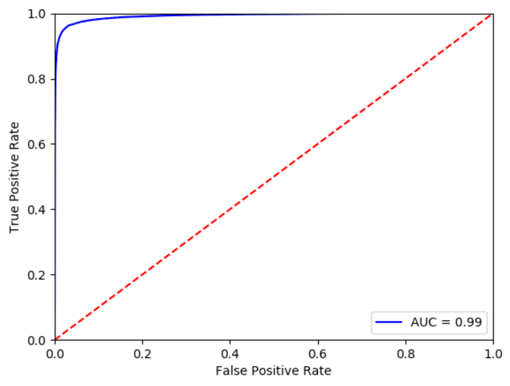
where TP is the number of correctly classified events that are positive, P is the number of events that are positive, TN is the number of correctly classified events that are negative, and N is the number of events that are negative. In this case, positive is defined as being Brem events, and negative is defined as being No Brem events. In addition to the TPR and TNR, there are also the False Positive rate and the False Negative Rate, given by $1 - TNR$ and $1 - TPR$ respectively. The False Positive Rate is the proportion of negative samples that are incorrectly classified as positive samples instead, and so a decrease in TNR means that more No Brem events will be classified as Brem, potentially leading to scenarios where the a model can have both a high TPR and FPR.

Figure 27 shows the ROC (Receiving Operating Characteristics) curves of the two different models. The curves show the TPR as a function of the False Positive Rate, which is given by $1 - TNR$, of the two models. The False Positive Rate is the proportion of negative samples that are incorrectly classified as positive samples instead. The area under the curve (AUC) is a measure of the accuracy of the models, the closer it is to 1, the higher the classification accuracy of the model is. The CNN model has AUC of 0.99, while the GNN model has an AUC of 0.91, so with the GNN model a higher FPR must be accepted to reach the same TPR as the CNN. The CNN has an accuracy of 96.4%, with $TPR = 0.964$ and $TNR = 0.981$, and the GNN has an accuracy of 82.9%, with $TPR = 0.822$ and $TNR = 0.836$, on the 96288 events in the evaluation data set. Both models classify more Brem events as No Brem than vice versa.

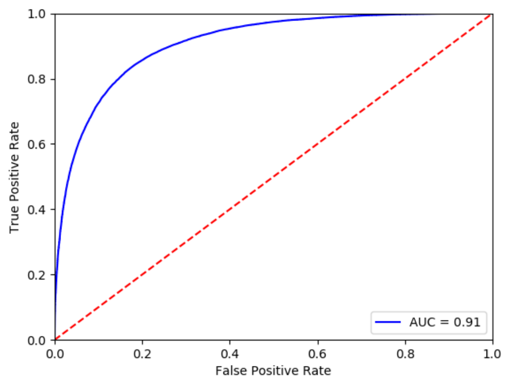
Unless otherwise noted, the error bars on the histograms are the statistical errors in the number of predicted events in each bin, given by the square root of the bin content. This means that the error bars on bins with only a small number of predicted events in that category will be large.

Figure 28 shows the TPR of the CNN and GNN models binned in terms of the angle of the recoil electron at the face of the ECal, the same distribution as in Figure 20b. The CNN correctly classifies the Brem events with a small recoil angle at a high rate, while the higher recoil angles have a lower TPR. The GNN incorrectly classifies a large fraction of events in the first bin, where most of the events are located, as can be seen in Figure 20b.

Figure 29 shows the TPR of the CNN and GNN models binned in terms of the energy of the recoil electron. Both models perform worse as the recoil energy approaches the cutoff of

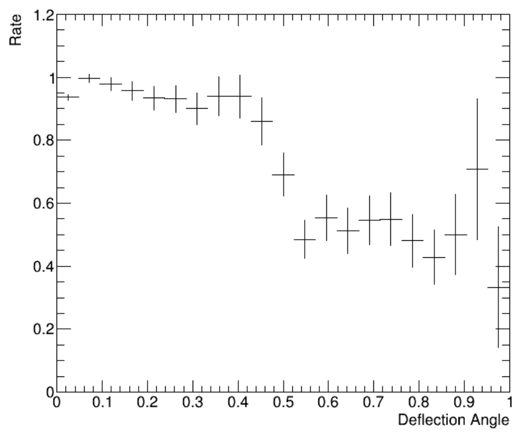


(a) ROC curve of the CNN model with AUC = 0.99.

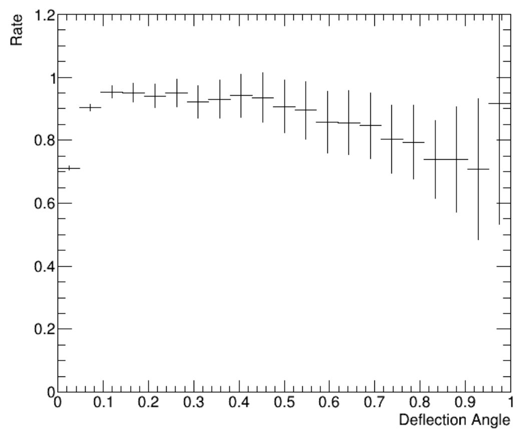


(b) ROC curve of the GNN model with AUC = 0.91.

Figure 27

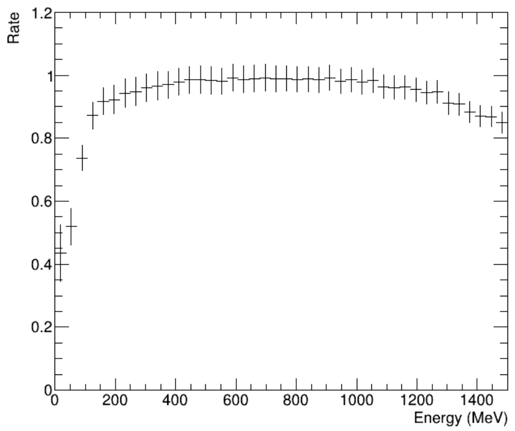


(a) True Positive rate for the CNN binned in terms of the deflection angle at the face of the ECal.

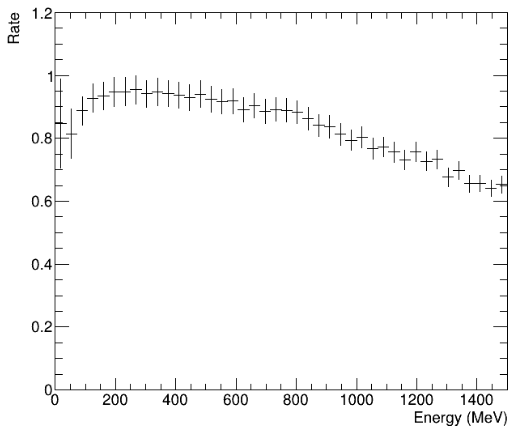


(b) True Positive rate for the GNN binned in terms of the deflection angle at the face of the ECal.

Figure 28



(a) True Positive rate for the CNN binned in terms of the energy of the recoil electron.



(b) True Positive rate for the GNN binned in terms of the energy of the recoil electron.

Figure 29

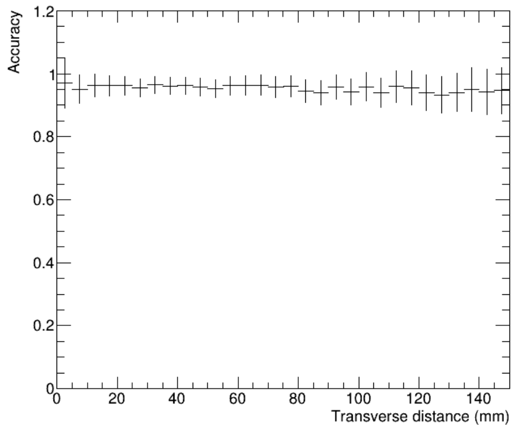
1500 MeV, but the GNN TPR has a peak around 300 MeV, while the CNN TPR remains roughly constant over a larger range of recoil energies. The CNN model TPR at the lowest energies is significantly lower at energies below 150 MeV, similar to the GNN performance at the highest energies.

Figure 30 shows the TPR of both models binned in terms of the transverse distance between the recoil electron and the photon at the face of the ECal. Here the CNN performs at roughly the same rate regardless of the separation between the two particles. The GNN shows a slight increase in the TPR as the separation between the two particles increases.

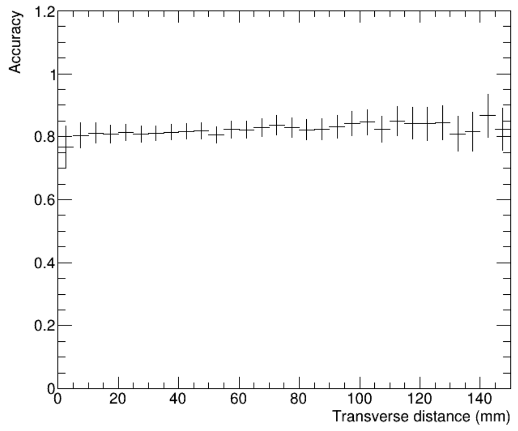
Figure 31 shows both the TPR and the TNR of the two models binned in terms of the number of hits attributed to the recoil electron. In the TNR case, since the only particle showering in the ECal is the 4 GeV primary electron, all hits are attributed to it. The CNN performs significantly worse when 10 or fewer hits are attributed to the electron, but otherwise maintains the average accuracy. The GNN performance fluctuates, and has the worst TPR for events with between 20 and 40 hits attributed to the recoil electron, while the TNR decreases significantly for events with a large number of hits from the 4 GeV primary electron.

Figure 32 shows the TPR of both models binned in terms of both the energy of the recoil electron and the deflection angle at the face of the ECal. Here the CNN has a higher TPR among the events that have high energy and a small deflection angle, which correspond to a large fraction of the events. The GNN performs worse in this section, while they have similar performance in the lower energy and slightly larger deflection angle region, where most of the low energy events are.

Figure 33 shows the TPR of the two models binned in terms of the energy of the recoil electron and the number of ECal hits attributed to the recoil electron. Here the CNN has a high TPR for most of the middle band in Figure 23b where the majority of events lie, while the GNN performs worse at higher energies, and in particular as the number of electron hits per event gets lower at the higher energies.

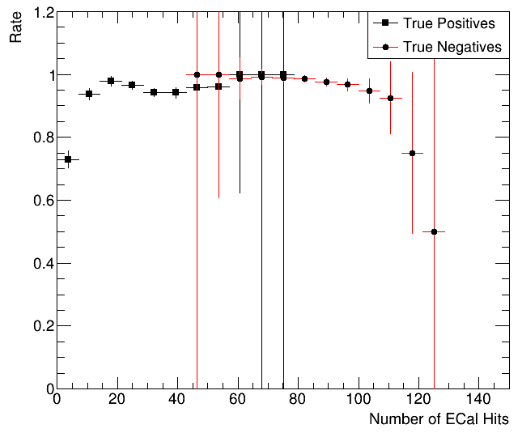


(a) True Positive rate for the CNN binned in terms of the transverse distance between the recoil electron and the photon at the face of the ECal.

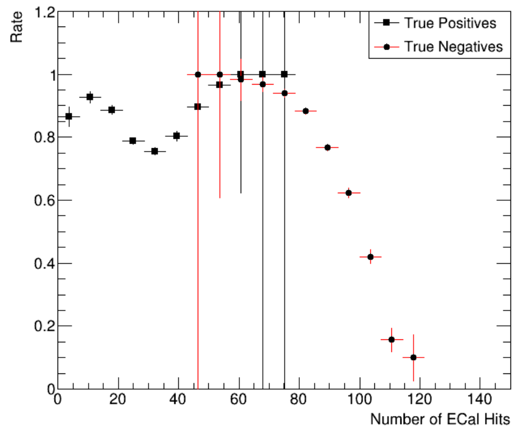


(b) True Positive rate for the GNN binned in terms of the transverse distance between the recoil electron and the photon at the face of the ECal.

Figure 30

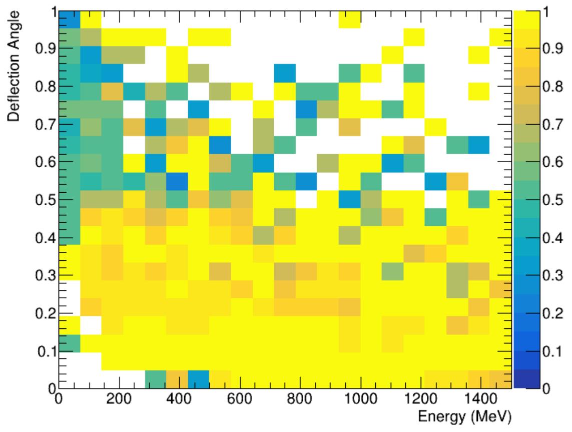


(a) True Positive and True Negative rates for the CNN binned in terms of the number of hits where the electron contributed the majority of the energy.

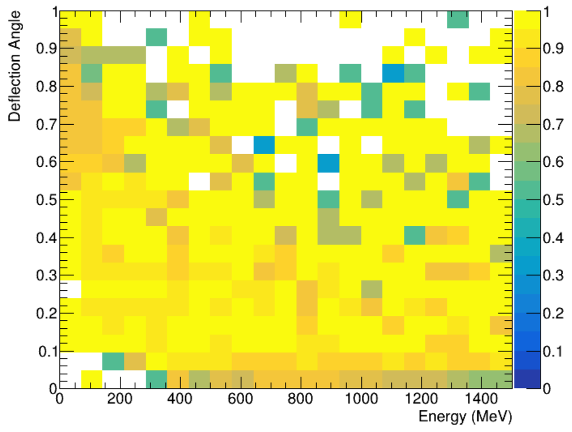


(b) True Positive and True Negative rates for the GNN binned in terms of the number of hits where the electron contributed the majority of the energy.

Figure 31

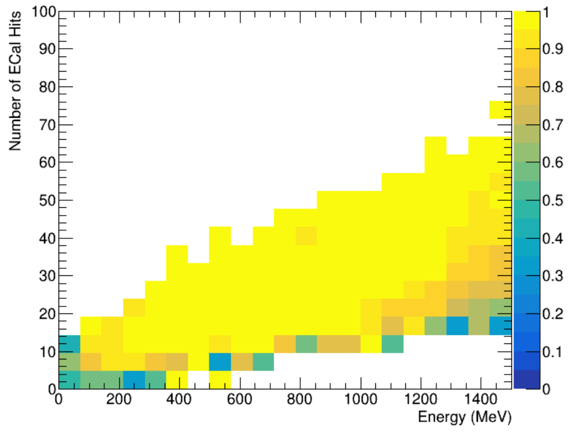


(a) True Positive Rate for the CNN binned in terms of both the energy of the recoil electron and the deflection angle at the face of the ECal.

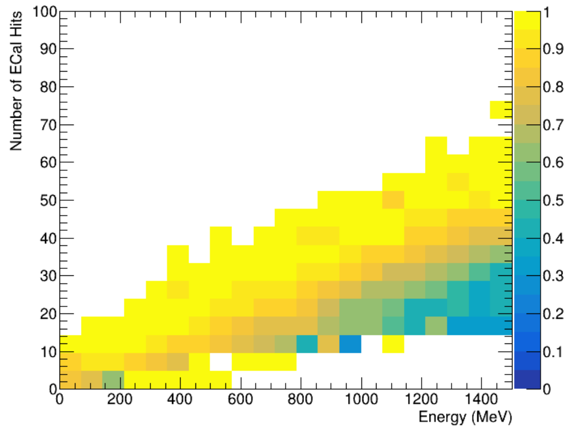


(b) True Positive Rate for the GNN binned in terms of both the energy of the recoil electron and the deflection angle at the face of the ECal.

Figure 32



(a) True Positive Rate for the CNN binned in terms of both the energy of the recoil electron and number of hits attributed to the recoil electron.



(b) True Positive Rate for the GNN binned in terms of both the energy of the recoil electron and number of hits attributed to the recoil electron.

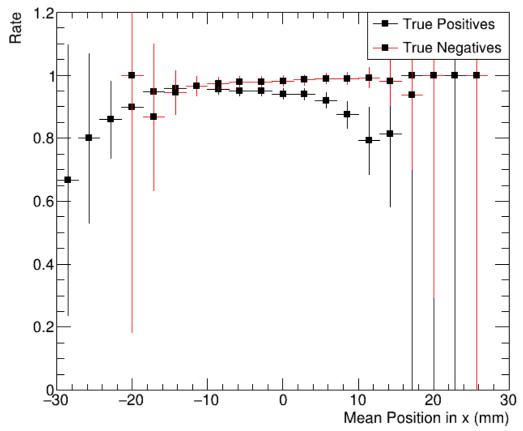
Figure 33

The main limitation of the CNN model seems to come from the number of hits in the ECal that can be attributed to the recoil electron. There is little change in the accuracy of the CNN model due to the separation of the two particles at the face of the ECal, as seen in Figure 30. A large separation between the two particles should give rise to two separated showers in the ECal, but if the energy of the recoil electron is low it will likely contribute only to a small number of hits. The reason for the problems that CNN has with a small number of hits could be due to a combination of there only being a few hits from the electron, and these hits making up only a small amount of the total energy in the ECal. In events with a small number of electron attributed hits, and with low energy in the hits, the feature map will likely have a larger response from the photon shower than the electron shower. This is due to the convolution and pooling operations combining a large number of clustered hits in the same area coming from the photon, which with the same filter would give a greater output than the small number of electron hits. Any effect from the electron shower in these events would potentially be overshadowed by the photon shower, or potentially just seen as noise. There is a further possibility that some events have lost a large fraction of electron hits in the creation of the input tensor to the CNN, which would lead to the only notable object in the calorimeter being the photon shower, and these would then be classified as No Brem events. The events where this happens would be those with a large separation between the recoil electron and the photon, but most events have a separation between the two particles of less than 150 mm, as seen in Figure 21. In any case, a distance this small between the two particles would fit within the slice of the calorimeter that the input is created from, as the events are centered to contain as many hits as possible and the slice is larger than this separation distance.

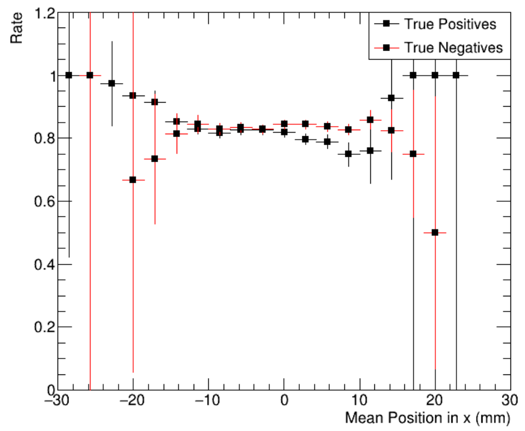
The above issues could potentially be overcome using some combination of smaller filter size, smaller pooling units, and transforming the energy deposit used as the elements of the tensors, though none of these that were individually tested provided any significant change or improvement in performance. Both smaller and larger filter sizes resulted in a similar performance, and using $\ln(E)$ instead of E as input resulted in worse performance on otherwise identical events.

The GNN on the other hand seems to be limited by the separation between the two particles and the deflection angle of the electron. This can be seen in how the TPR of the GNN is worse than the CNN in the regions of Figure 32 that have a high energy recoil electron and a low deflection angle. This weakness of the GNN is potentially due to how, as the separation between the two particles decreases, the showers increasingly overlap each other. The EdgeConv operation only considers a limited number of nearest neighbors when computing the edge features, and for overlapping showers this might not lead to any clear distinction between the two showers as many hits will have hits from both showers included in the EdgeConv operation. In events where the two particles are more separated it is more likely that the showers are clustered and that the EdgeConv then mostly includes hits from one shower.

Figures 34 and 35 show the TPR and TNR of the CNN and GNN models binned in terms of the mean position of the hits in the x-direction and the standard deviation of the hits in the x-direction. Both the CNN and the GNN perform close to their average accuracy in the region $-10 \leq x \leq 0$. The TPR of the CNN drops outside of this central region and the GNN has a worse performance at positive x, while a better at more negative x. Looking at the



(a) True Positive Rate for the CNN binned in terms of the mean position of the hits in the x-direction per event.



(b) True Positive Rate for the GNN binned in terms of the mean position of the hits in the x-direction per event.

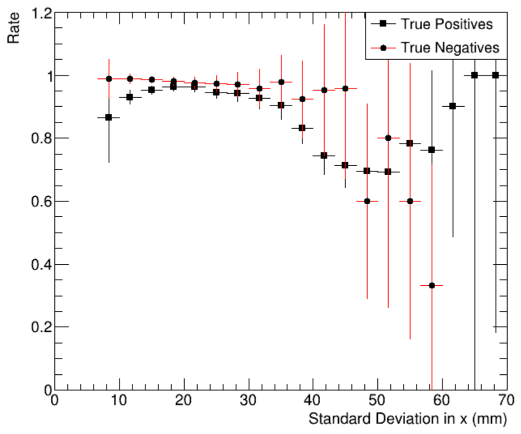
Figure 34

standard deviation, the CNN performs worse at events that have a larger standard deviation, while the TPR of the GNN becomes large as the standard deviation increases, but the TNR decrease, giving more false positives.

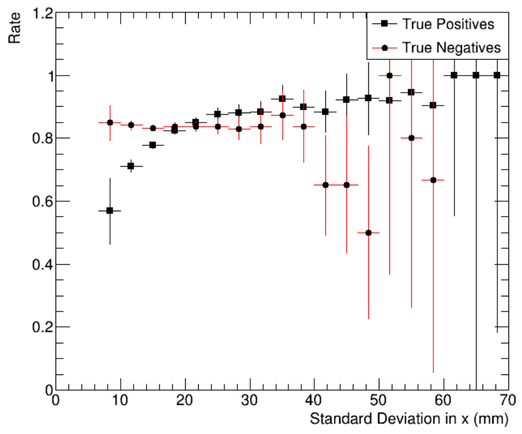
Figures 36 and 37 show the TPR and TNR of the CNN and GNN models binned in terms of the mean position of the hits in the y-direction and the standard deviation of the hits in the y-direction. Both models perform close to their average accuracy in terms of the mean in the y-direction, since the distribution of the means in the y-direction is uniform within the beam spot as there is no magnetic field that can affect the trajectories in that direction. Looking at the standard deviation, the CNN again performs worse at higher standard deviations, where the hits in the event are more spread out, and the GNN again has a slightly higher TPR at higher standard deviations, while still finding more false positives.

Figures 38 and 39 show the TPR and TNR of the CNN and GNN models binned in terms of mean layer hit and the standard deviation of the mean layer hit. The CNN performs around average in the region around the peak in Figure 18a, while the GNN has a worse than average TPR in this region. The GNN TPR increases with a larger mean layer hit, while the TNR decreases, giving more false positives.

The CNN overall has a lower TPR for events whose mean x-position is outside of the peak in Figure 16a and events with a relatively high standard deviation in both the x- and y-directions. A shift in the mean x-position away from the peak should not pose much of an issue for the CNN, if the shift is only due to a shift of the entire shower, as the CNN is designed to exploit translation invariance. Events with high standard deviations are ones that have more spread out hits in the ECal in at least one direction, the hits could be either less clustered together or separated into two distinct clusters as might happen in Brem events if the electron and photon enter the ECal sufficiently far away from each other. In addition to the decrease in TPR at high standard deviations, there is also a decrease in it at low standard deviations in the x-direction, but not in the y-direction. These low spread events would have

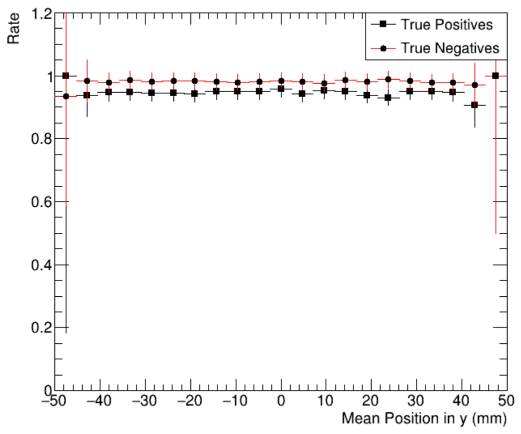


(a) True Positive Rate for the CNN binned in terms of the standard deviation of the position of the hits in the x-direction per event.

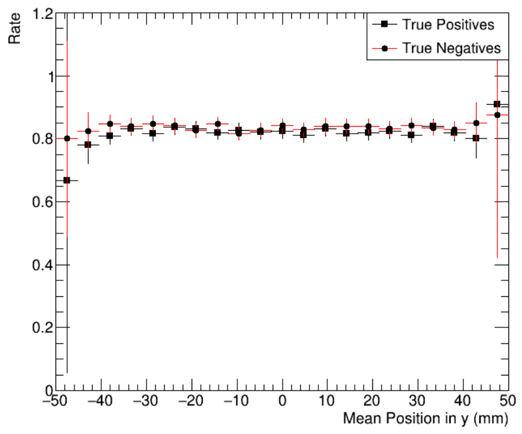


(b) True Positive Rate for the GNN binned in terms of the standard deviation of the position of the hits in the x-direction per event.

Figure 35

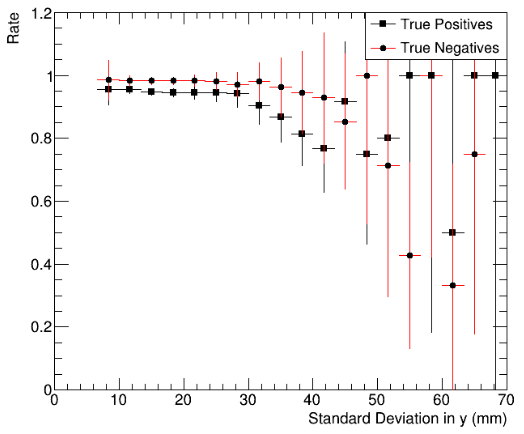


(a) True Positive Rate for the CNN binned in terms of the mean position of the hits in the y-direction per event.

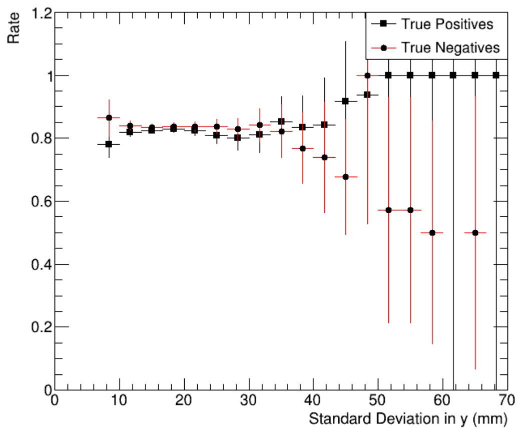


(b) True Positive Rate for the GNN binned in terms of the mean position of the hits in the y-direction per event.

Figure 36

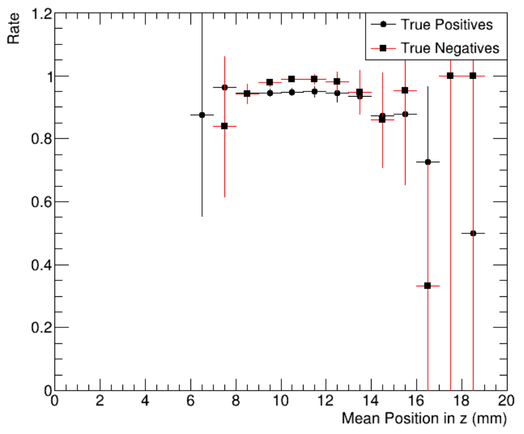


(a) True Positive Rate for the CNN binned in terms of the mean position of the hits in the y-direction per event.

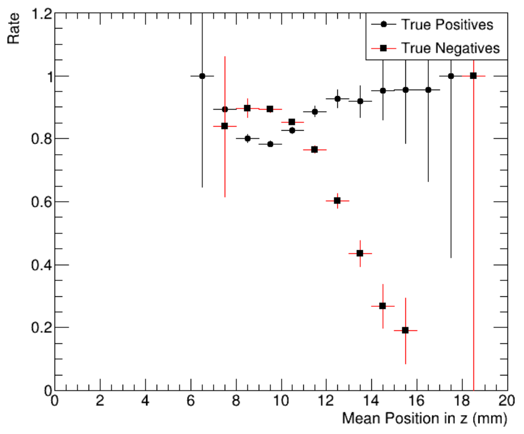


(b) True Positive Rate for the GNN binned in terms of the standard deviation of the position of the hits in the y-direction per event.

Figure 37

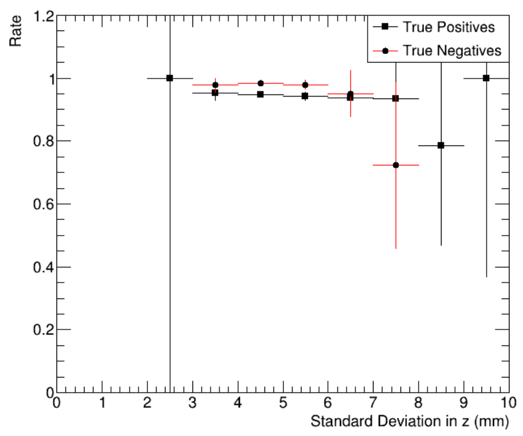


(a) True Positive Rate for the CNN binned in terms of the mean layer number hit per event.

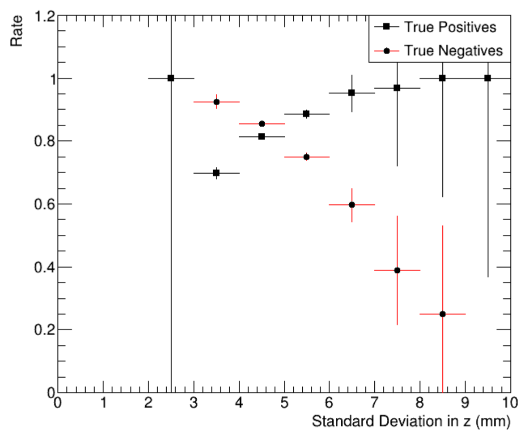


(b) True Positive Rate for the GNN binned in terms of the mean layer number hit per event.

Figure 38



(a) True Positive Rate for the CNN binned in terms of the standard deviation in the layer number of the hits per event.



(b) True Positive Rate for the GNN binned in terms of the standard deviation in the layer number of the hits per event.

Figure 39

more clustered showers, and that the TPR, but not the TNR, decreases here indicates that they are typically classified as No Brem events, which mostly have a single shower. That this only happens in the x-direction is possibly due to effects from the magnetic field. That the CNN has a lower TPR for the events with more spread indicates that there is something about these events that makes them more likely to be seen as No Brem events. Since it also performs poorly on events with few electron attributed hits, this performance on spread out events could be due to a small number of not very clustered electron hits making the event look like a single shower No Brem event. Worth noting is that these events that the CNN performs poorly on correspond to a small fraction of all events, and, if they are qualitatively distinct from the majority of Brem events, then it is possible that the CNN simply does not favor these events during training.

The GNN has a higher TPR on the more spread out events than the CNN does, while showing the same kind of decrease in TPR for events with a low standard deviation in the x-direction. The increase in TPR at high standard deviations might be due to the fact that the GNN considers a fixed number of nearest neighbors, regardless of the distance to the points, and so even in scenarios with spread out hits from the electron it could still recognize them as coming from the same particles if they are closer to each other than to the hits from the photon. The higher TPR is also accompanied by a decrease in TNR, and so the GNN will classify more spread out No Brem events as Brem events. In No Brem events that are more spread out the EdgeConv operation might again be considering nearest neighbors in such a way that hits that are not clustered together are considered as belonging to their own shower if they do not have any nearest neighbors from the main electron shower. The GNN also has a very low TNR on No Brem events that have a high mean layer hit and a high standard deviation in the mean layer hit, while having a low TPR on Brem Event with low mean layer number and standard deviation. The events with high mean layer hit and high standard deviation might appear as Brem events due to the spread in the z-direction that the GNN will see. The GNN does not use the geometry of the detector, only the Cartesian coordinates of the hits, and there is not necessarily any distinction between the three Cartesian coordinates when computing the nearest neighbors. The GNN might then be identifying spread in the z-direction as the same kind of distinguishing feature as spread in the x- and y-directions, and thus identifying the No Brem events as Brem events. This should of course not happen, as showers from the single primary electrons typically are deeper than showers from recoil electrons or photons, and this is qualitatively different from spread out hits in the x- and y-directions due to multiple showers. This could potentially be remedied by directly using the layer number of the hits instead of the z-coordinate as one of the vertex features in the graph. With this there could be less of an appearance of spread in the graph, which would not impact the classification in the same way.

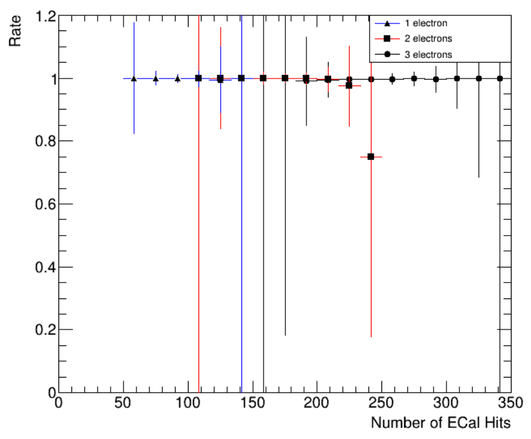
5.2 Multi Electron Studies

Both the CNN and GNN models trained on the multi-electron samples achieved very high accuracies, largely due to the different number of hits in the different events. The CNN achieved a classification accuracy of 0.998 on the 63661 events in the multi-electron data set, incorrectly classifying only 100 events, most of which were events with 3 primary electrons. The GNN achieved a classification accuracy of 0.992, incorrectly classifying 500 events, most of which again were events with 3 primary electrons. The TPR of the two models on the 3 different types of events are shown in Figure 40. Both models incorrectly classify events that are in the overlap between the peaks from the 2 primary electrons and 3 primary electrons in Figure 24, but the GNN significantly more so. In both cases, all of the incorrectly classified 3 electron events are instead classified as 2 electron events. This happens because two of the primaries shower close to each other, causing a greater than normal overlap between the showers and thus a smaller total number of hits than expected for a 3 electron event. The same applies to the small number of incorrectly classified 1 and 2 electron events.

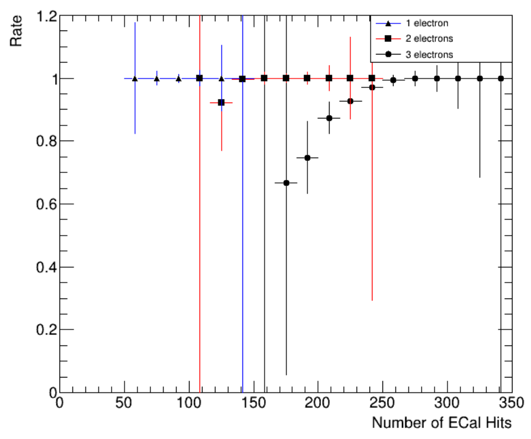
The difference in classification ability in the overlapping region is potentially due to the EdgeConv operation not being able to clearly distinguish between different showers when there is a large overlap between them. Events with large overlap between showers are then more difficult to distinguish both on the number of hits and the features of the showers that might otherwise distinguish them.

5.3 Shower Segmentation

The results of the GravNet based model used for shower segmentation are presented in the following plots. To evaluate the performance of the model, the accuracy per event has been computed. The GravNet model is created to predict which shower the hits in an event belong to when given an event. The accuracy of the model, on an event-by-event level, is then defined to be the number of hits in the event that were correctly labeled by the GravNet. The histograms show the average accuracy of all the events in a given bin, where the error on the bin content is the standard deviation of the accuracy of all of the events in that bin. In addition to the number of hits correctly classified, the model is also evaluated based on the fraction of the total energy in the event that belongs to hits that are correctly labeled. This is done to understand if the model accuracy comes from labeling a large number of low



(a) True Positive Rate for the CNN binned in terms of the total number of hits in the ECal.



(b) True Positive Rate for the GNN binned in terms of the total number of hits in the ECal.

Figure 40

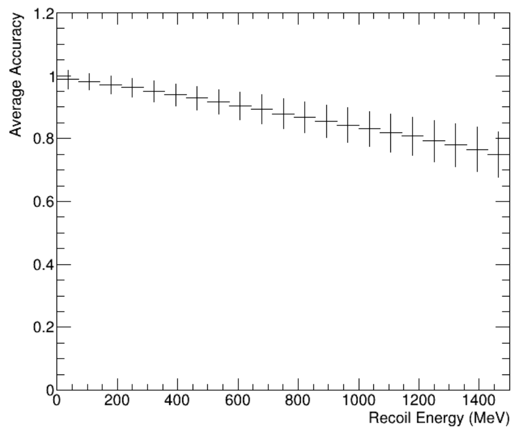
energy hits, or if the model correctly labels the high energy hits in an event. The model was evaluated on the same Brem events as the single-electron CNN and GNN models, not on the No Brem events, as these would only have a single object in the calorimeter most of the time. The average accuracy of the model over all events in the data set was 0.866 and the average fraction of the energy that was correctly labeled was 0.905.

Figure 41a shows the average accuracy and average fraction of the energy correctly labeled by the GravNet model binned in terms of the energy of the recoil electron. The average accuracy of the model decreases with increasing recoil energy, though the error increases, indicating a larger spread in the accuracy at these energies. The average fraction of the energy in each event predicted correctly is larger than the fraction of hits predicted correctly, indicating that the model does at least not solely predict low energy hits correctly, but on average predicts higher energy hits correctly.

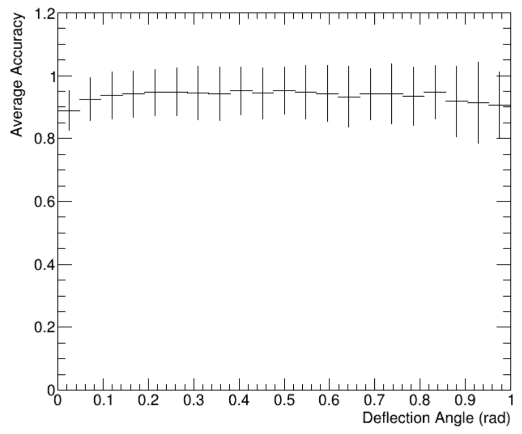
Figure 41b shows the average accuracy of the GravNet model binned in terms of the deflection angle of the recoil electron at the face of the ECal. The average accuracy of the GravNet model is the lowest in the first bin, where most of the events are located, while it increases for higher angles. All of the bins have a large error compared to in Figure 41a, indicating a large spread in the accuracy at all angles, even at the high deflection angles that only a few events have.

Figure 42a shows the average accuracy binned in terms of the transverse distance between the recoil electron and the photon at the face of the ECal. Similar to the plots in Figure 41b, the accuracy is consistent over the range of distances, with a possible slight increase at larger distances. The error of the bins are again larger compared to the low energy errors in Figure 41a.

The plots in Figure 42b shows the accuracy of the GravNet model binned in terms of the number of hits attributed to the recoil electron. The accuracy of the model decreases as the number of hits attributed to the electron increases. As in Figure 41a, the errors on the bins

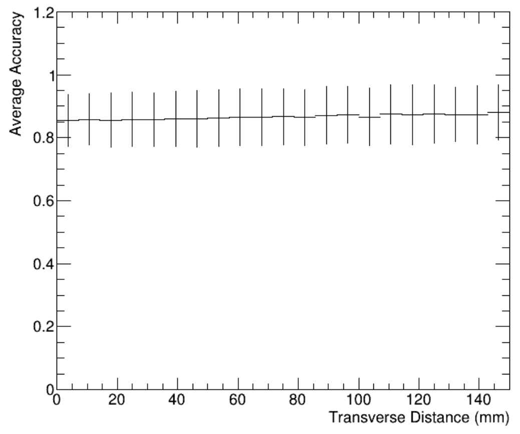


(a) Average accuracy of the GravNet model binned in terms of the energy of the recoil electron.

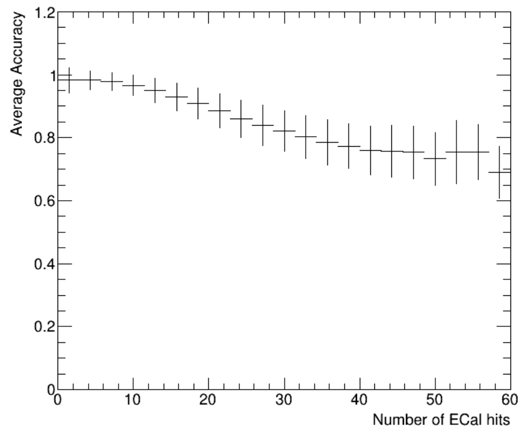


(b) Average accuracy of the GravNet model binned in terms of the deflection angle of the recoil electron at the face of the ECal.

Figure 41



(a) Average accuracy of the GravNet model binned in terms of the transverse distance between the recoil electron and the photon at the face of the ECal.



(b) Average correctly labeled fraction of the energy in each event by the GravNet model binned in terms of the number of ECal hits attributed to the recoil electron.

Figure 42

are lower for a small number of hits.

The GravNet model is designed to be able to separate different showers in a calorimeter. The model that was trained in this project showed a high average accuracy on the evaluation data, a total accuracy on all hits of 0.866. Comparing the 4 distributions of the model accuracy shows that the model performs worse in events with a high recoil energy and a relatively large number of recoil hits. The reason for this is potentially that the model has not learnt to distinguish an electron shower from a photon shower, but rather to separate a low energy shower from a larger energy shower. In this scenario the model can successfully cluster the hits attributed to the low energy shower correctly into one shower, given that the showers are sufficiently separate in energy. These events that the model performs better on are also the events that tend to have a smaller number of electron attributed hits, as seen in Figure 23b.

This could be tested by evaluating the GravNet model on a sample where the electron and photon have been switched, such that the photon is limited to an energy of less than 1500 MeV, to test whether it would attribute the electron hits in such a sample to the photon. Further, a similar GravNet model could also be trained on events where either the electron has no limit on its maximum energy, allowing both the electron and photon to take energies in the full range of the 4 GeV beam energy, or on events with two or more primary electrons, in order to separate two showers of the same energy.

A limitation of the GravNet model as used here is that it was trained on events where the hits were either attributed to the photon or to the electron, with no fractional attributions in case of overlap. The model could instead be modified to train on events with fractional attributions in the hits and then tested on the ability to predict fractions of shower energies in each cell. This could be done using the loss function presented in [20], which also takes into account the energy of each hit when computing the loss function, giving more importance to hits with a larger energy deposit, instead of the current cross entropy loss function which only takes into account the output of the GravNet when computing the loss.

6 Conclusion and Outlook

The LDMX experiment is designed to search for dark matter using missing momentum and energy. A crucial part of the experimental design is a high granularity electromagnetic calorimeter to accurately measure the energy deposited by electrons in the calorimeter. This thesis looks at some potential imaging techniques based on machine learning that could be used with calorimeter data in order to classify and potentially reconstruct events.

Classification of calorimeter data could be done in different ways, by e.g. identifying the particle responsible for a specific shower, classifying an entire event into some pre-determined class, or by finding the different showers in an event. This project takes two approaches to this, one which aims to study the usefulness and limitations of the calorimeter when distinguishing between different particles and event types, and one which aims to study how well two showers can be separated from each other.

For this project, two different types of neural networks were used, based on three different models. A basic Convolutional Neural Network model and a Graph Neural Network model based on [29] were trained to classify events based on their particle contents. This classification was done on two different data sets, one to distinguish between events where an electron has undergone bremsstrahlung or not, and one to classify events based on the number of primary electrons in the event. Another graph neural network model, based on [20] was trained to label calorimeter hits in the ECal based on which particle in an event contributed more to a given hit.

The best performing architectures of these models were evaluated on separate evaluation data sets. The CNN model achieves an accuracy of 96.4% on the single-electron data set, and an AUC of 0.99. The GNN model achieves an accuracy of 82.9% and an AUC of 0.91. These two models were both able to successfully distinguish between events based on the features of the events, though with different limitations.

The CNN model performs the worst on events with a low energy recoil electron and few hits where the electron has contributed the majority of the energy. This limitation is potentially due to the loss of information that occurs due to the large filters and large pooling unit used in the architecture leading to the contribution of these hits not being very important on the feature maps. The CNN architecture was relatively simple, with only a single convolutional layer and pooling layer, there is a lot of room for improvement by using different configurations of the convolution and pooling layers, say using two different convolutional layers, one with large filters and one with smaller filters, coupled with smaller pooling units in order to not lose information from low energy hits. Other potential 3D CNN architectures based on already existing and successful architectures in computer vision research could be used, e.g. the GoogLeNet architecture used in [27].

The GNN model performs the worst on events where the recoil electron has low deflection angles, small separation between the photon and recoil electron, and events where the hits are spread out in the calorimeter. The GNN could be modified to take different input variables, to substitute or complement, the Cartesian coordinates used here, such as the layer number of the hits or the unique ID of each readout cell, that capture the features of an event in a different way than the Cartesian coordinates do. While different architectures of the GNN were tested here and none was found to perform better than the one used, it is possible that the different configurations could lead to better performance, in particular considering a larger number of nearest neighbors. Due to the lower resource requirements of the GNN compared to the CNN it could also more easily be trained on more data than the CNN.

Both models achieve accuracies above 99% on the multi-electron data set. This is due to the different number of hits in the different types of events, as this is the biggest difference between the types. The only events that are misclassified are those with 3 primary electrons that have a large overlap between two or more of the showers causing a smaller number of hits. This is more of a limitation for the GNN model than the CNN model, due to the troubles it has with classifying events with overlapping showers.

The GravNet model achieves an average accuracy of 0.866 on the Brem events in the single-electron data set, and the correctly labeled hits correspond to 90.5% of the total energy in these events. The model performs better on events with lower recoil energies and fewer hits attributed to the recoil electron. These seems to be due to the model being able to distinguish lower energy showers or clusters of hits from higher energy showers. This could be better evaluated by training the model on data with equal energy showers, to study its ability to cluster showers not just based on their total energy. Further, the model could be modified to not label hits with integers corresponding to particles, but to label them with fraction corresponding to the energy coming from each particle, using the loss function provided in [20].

Other than strictly their performance, the models can be compared based on the computing resources they require to train and use. The 3D CNN models have significantly larger memory requirement than the GNN models, mainly due to the large size of the input data used in this project. The CNNs also require longer training times per event, though as the number of hits per event increases the advantage of the GNN models on training time decreases. CNN computing requirements could be decreased by using smaller event sizes, and while this can lead to worse performance it could be a useful thing if different 3D CNN architectures are used.

Improvements and changes based on the models used here depend on what their use case would be. The 3D CNN and ParticleNet models used for classification could be tested on different types of events to study their usefulness in different contexts, such as their ability to classify multi-electron events based on whether a photonuclear reaction occurred in the ECal, leading to the lack of a photon shower in the ECal following bremsstrahlung. They could also be further improved to not just classify event types, but also determine the energies of the different showers in the ECal, similar to what was done on single showers in [27]. This could be done by training them to detect the number of showers and the energies of each shower.

A fundamental limitation of this approach to classification is that the networks will be restricted to the output space defined during training. A more realistic later phase LDMX scenario, where the experiment used multiple primary electrons, might consist of events with one electron that undergoes bremsstrahlung and three that do not. Since there are many different permutations of what this could look like, with different numbers of particles undergoing bremsstrahlung and different interactions in the ECal, it would be difficult to enumerate all of these possibilities and train the networks on them. Other possibilities could be to broadly classify events, or to identify specific particles that are of interest, such as the single recoil electron in the above example event. The types of networks used for classification here are not immediately suitable for this second task, though could be modified to do it.

The ParticleNet based GNN can be modified for segmentation like tasks of identifying single particles as in [19]. While the 3D CNN for segmentation that was tested here was too resource intensive to further study, other architectures could be used, or even using 2D projections of the events together with 2D CNNs as in [31], which has high performance and would be less resource intensive than a 3D version.

The GravNet model [20] used here was designed for particle clustering, and so it is an attractive approach for these kinds of tasks. As it was used in this project to classify different hits it suffers from the same limitations as the other two models, in that the ways it can label hits depends on the output space set during training. If the model has been trained to expect at most four different showers in the ECal, then it would be useless in scenarios with more than that. One potential solution using the graph structure of the data is to use edge classification as in [32, 33, 34]. Instead of labeling the vertices on the graph, edge classification gives either a Boolean label or probability to each edge on the graph. Based on the edge labels the vertices are then classified according to whether they belong to the same object. Here, this could be used to say whether two hits belong to the same shower instead of saying which specific shower any given belongs to, and then use this information to determine where the showers are.

References

[1] F Zwicky. On the masses of nebulae and of clusters of nebulae. *The Astrophysical Journal*, 86:217, 1937.

[2] T. Åkesson et al. Light dark matter experiment (ldmx). *arXiv:1808.05219*, 2018.

[3] Wikimedia Commons. File:standard model of elementary particles.svg — wikimedia commons, the free media repository, 2020. [Online; accessed 2-April-2020].

[4] M. Thomson. *Modern particle physics*. Cambridge University Press, 2013.

[5] N. Aghanim et al. Planck 2018 results. vi. cosmological parameters. *arXiv:1807.06209*, 2018.

[6] V. C. Rubin, W. K. Ford Jr, and N. Thonnard. Rotational properties of 21 sc galaxies with a large range of luminosities and radii, from ngc 4605/r= 4kpc/to ugc 2885/r= 122 kpc. *The Astrophysical Journal*, 238:471–487, 1980.

[7] K. Garrett and G. Duda. Dark matter: A primer. *Advances in Astronomy*, 2011, 2011.

[8] M. Kamionkowski. Wimp and axion dark matter. *arXiv preprint hep-ph/9710467*, 1997.

[9] D. N. Spergel et al. First-year wilkinson microwave anisotropy probe (wmap)* observations: determination of cosmological parameters. *The Astrophysical Journal Supplement Series*, 148(1):175, 2003.

[10] G. B. Gelmini. Tasi 2014 lectures: the hunt for dark matter. *arXiv:1502.01320*, 2015.

[11] G. B. Gelmini. Light weakly interacting massive particles. *Reports on Progress in Physics*, 80(8):082201, 2017.

[12] G. Bertone and D. Hooper. History of dark matter. *Reviews of Modern Physics*, 90(4):045002, 2018.

[13] A. Boveia and C. Doglioni. Dark matter searches at colliders. *Annual Review of Nuclear and Particle Science*, 68:429–459, 2018.

[14] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. MIT press, 2016.

[15] M. Ohlsson. Lecture notes on introduction to artificial neural networks and deep learning.

[16] M. A. Nielsen. *Neural networks and deep learning*, volume 2018. Determination press San Francisco, CA, USA:, 2015.

[17] Lecture notes for cs231n: Convolutional neural networks for visual recognition. `https://cs231n.github.io/`.

[18] F. Scarselli et al. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[19] Y. Wang et al. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 38(5):1–12, 2019.

[20] S. R. Qasim et al. Learning representations of irregular particle-detector geometry with distance-weighted graph networks. *The European Physical Journal C*, 79(7):608, 2019.

[21] C. M. Bishop. *Pattern recognition and machine learning.* springer, 2006.

[22] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[24] M. Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[25] François Chollet et al. Keras. `https://keras.io`, 2015.

[26] G. Kasieczka et al. The machine learning landscape of top taggers. *arXiv preprint arXiv:1902.09914*, 2019.

[27] D. Belayneh et al. Calorimetry with deep learning: Particle simulation and reconstruction for collider physics. *arXiv preprint arXiv:1912.06794*, 2019.

[28] J. Alimena, Y. Iiyama, and J. Kieseler. Fast convolutional neural networks for identifying long-lived particles in a high-granularity calorimeter. *arXiv preprint arXiv:2004.10744*, 2020.

[29] H. Qu and L. Gouskos. Particlenet: Jet tagging via particle clouds. *arXiv preprint arXiv:1902.08570*, 2019.

[30] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[31] H. Kasaei. Orthographicnet: A deep learning approach for 3d object recognition in open-ended domains. *arXiv preprint arXiv:1902.03057*, 2019.

[32] C. Aggarwal, G. He, and P. Zhao. Edge classification in networks. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 1038–1049. IEEE, 2016.

[33] S Farrell et al. Novel deep learning methods for track reconstruction. *arXiv preprint arXiv:1810.06111*, 2018.

[34] Xiangyang Ju et al. Graph neural networks for particle reconstruction in high energy physics detectors. *arXiv preprint arXiv:2003.11603*, 2020.