Anomaly Detection in CMS HGCAL Using Radial Distribution Function and Autoencoder

Qingxiang Guo
University of Chinese Academy of Sciences
Advisor: Bora Isildak

August 12, 2025

Abstract

The High Granularity Calorimeter (HGCAL) is a key component in the Phase-2 upgrade of the CMS detector at the LHC. Due to its unique hexagonal geometry, conventional Convolutional Neural Networks (CNNs) are not directly applicable. In this work, we explore the Radial Distribution Function (RDF) as an alternative feature representation to detect anomalies in HGCAL occupancy maps. We construct an Autoencoder model using both CNN and RDF features, compare their performance on Monte Carlo (MC) simulated datasets with injected anomalies, and demonstrate the superiority of RDF-based features in this context.

Contents

1 Introduction

1.1 HGCAL for CMS

The Compact Muon Solenoid (CMS) experiment [1] is one of the four major detectors at the Large Hadron Collider (LHC). It is a general-purpose particle detector designed to investigate a wide range of physics phenomena, including the search for the Higgs boson, extra dimensions, and particles that could make up dark matter. CMS uses a powerful superconducting solenoid magnet to bend the paths of charged particles, allowing precise measurement of their momentum.

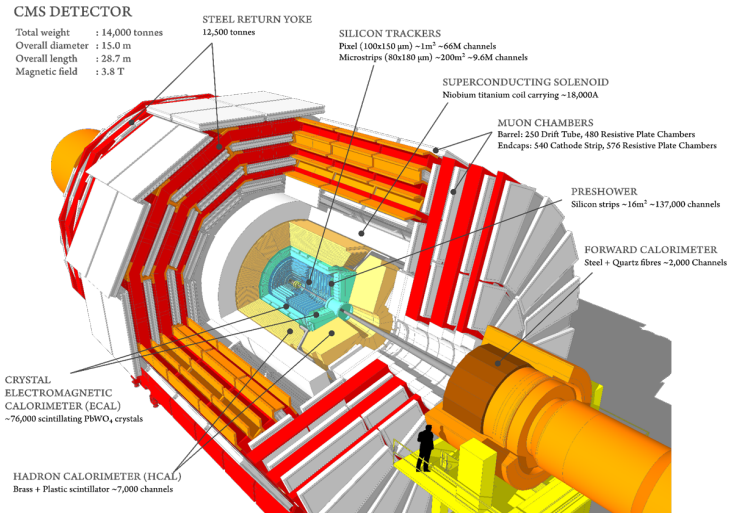

Figure 1: The CMS detector [1].

The High Granularity Calorimeter (HGCAL) [2] will replace the current endcap calorimeters in the CMS detector for the HL-LHC upgrade. It features unprecedented spatial resolution, with millions of hexagonal silicon sensor cells arranged in layers. This fine segmentation enables detailed reconstruction of particle showers, even in the extremely high radiation and pile-up environment expected at HL-LHC.

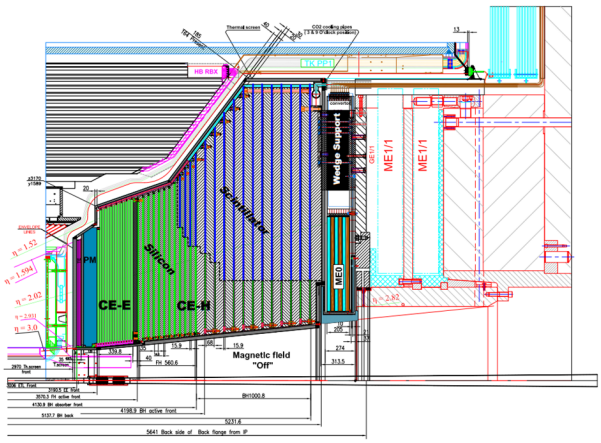

Figure 2: The HGCAL endcap calorimeter [2].It consists of an electromagnetic calorimeter and a hadronic calorimeter, where the green parts are silicon sensors and the blue parts are scintillator tiles.

In this research, we aim to develop an anomaly detection model for the HGCAL to enable real-time monitoring of its operational status. Given the detector's complex geometry and high channel density, unexpected behavior in individual sensors or regions can be challenging to

detect with conventional monitoring methods. Machine learning can potentially handle various complex situations—including those that are entirely unforeseen.

1.2 Wafer Types

For simplicity, this study focuses only on the electromagnetic calorimeter. The electromagnetic calorimeter is divided into multiple layers, each densely populated with hexagonal silicon sensor cells. To balance performance and cost, the HGCAL employs three different types of cells from the inside out. The innermost cells are smaller and thinner to cope with higher luminosity, while the outermost cells are relatively thicker and larger.

Table 1: Basic information of three cell types.

	Type 0	Type 1	Type 2
Position	innermost region	intermediate region	outmost region
Active thickness (μm)	120	200	300
Cell size (cm^2)	0.52	1.18	1.18
Cell capacitance (pF)	50	65	45
Bulk polarity	p	p	p / (n)
Expected range of influence ($\times 10^{15} n_{eq}$/cm^2)	2-7	0.5-2.5	0.1-0.5

A certain number of cells form a hexagonal wafer, with all cells within a wafer being of the same type. Wafers of different types have the same area.

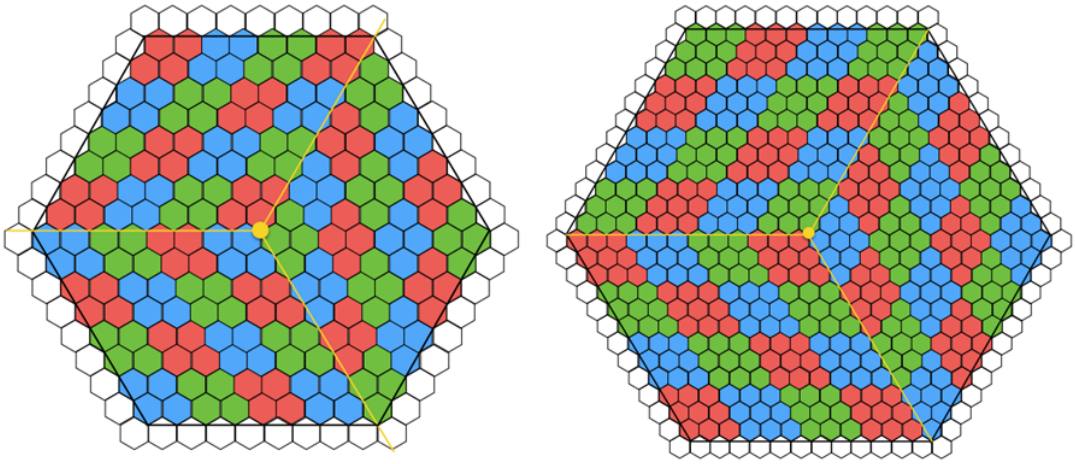

Figure 3: Two types of HGCAL wafers: large cell (left) and small cell (right).

In this study, potential operational faults are identified by monitoring the digi occupancy of each wafer. Here, digi occupancy refers to the number of hits within a wafer over a given time window, known as a lumi section, which lasts on the order of a few seconds depending on the conditions. The trained model should determine, based on the spatial distribution of digi occupancy within a given lumi section, whether an operational fault is present. Due to the substantial differences among the three wafer types, separate models are trained for each type.

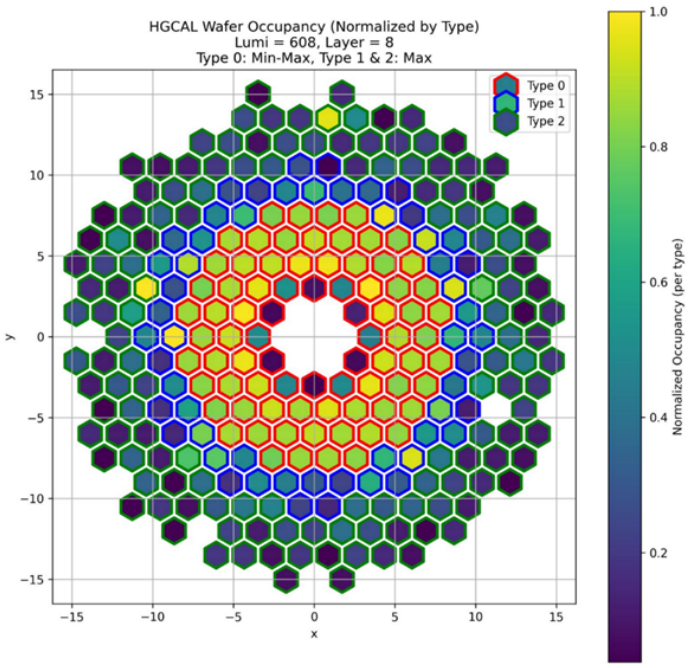

Figure 4: HGCAL wafer occupancy map with different borders for different types.

2 Model Choosing

2.1 Autoencoder Architecture

Autoencoders [3] are unsupervised neural networks with an encoder and decoder.

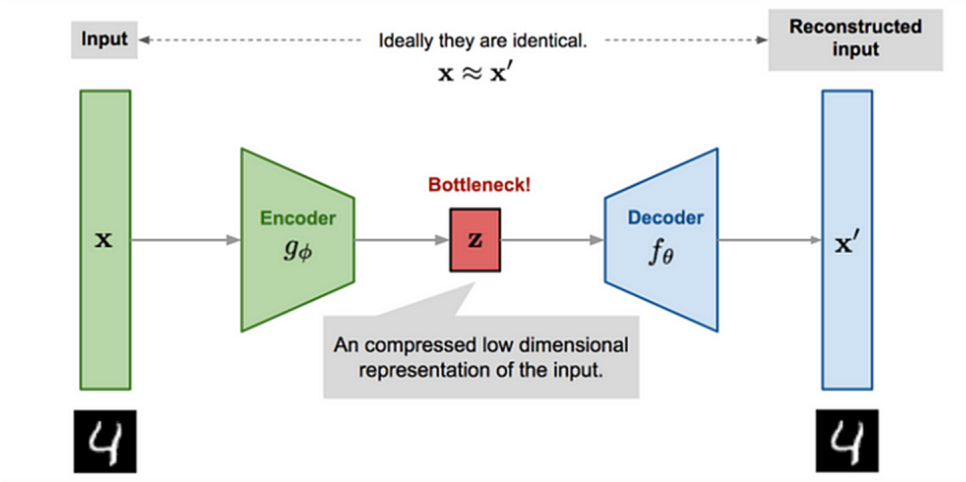

Figure 5: Autoencoder structure used in this work.

The encoder uses a neural network to reduce the dimensionality of the input data into a latent space, thereby extracting the key information from the input. The decoder then uses another neural network to reconstruct the input data from the information in the latent space. By training the autoencoder model with normal data as input, the decoder is optimized to produce outputs that are as close as possible to the inputs, resulting in minimal reconstruction

error. For anomaly data, which are absent from the training set, feeding them into the trained model will ideally yield significantly larger reconstruction errors compared to normal data. Thus, the magnitude of the reconstruction error can be used to determine whether a given data sample is normal or anomalous.

In this study, the use of an autoencoder offers two main advantages. First, during HGCAL operation, some anomalies may arise that were not anticipated in advance. An autoencoder can still identify such anomalies through reconstruction errors, even if they are absent from the training set. Second, the autoencoder architecture is sufficiently simple and computationally efficient, making it well suited for online monitoring tasks.

2.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) [4] is a class of deep learning models particularly well suited for processing data with grid-like structures, such as images. It consists of convolutional layers that apply learnable filters to extract local features, often followed by pooling layers to reduce spatial dimensions, and fully connected layers for final prediction or classification. CNNs are widely used in computer vision and related tasks due to their ability to automatically learn hierarchical feature representations.

However, in this study, the use of a CNN faces certain challenges. This is primarily because the wafers in the HGCAL are arranged not in the rectangular grid that CNNs are well suited to process, but in a hexagonal pattern. The article [5] describes a CNN algorithm specifically designed for hexagonally arranged data. However, due to its narrow range of applications, this approach has not been as well optimized as conventional CNNs, and current GPUs commonly used for machine learning are also not efficient at processing hexagonal structures. For these reasons, this method was discarded at the very beginning of the study.

In addition, there are two alternative approaches to applying CNNs. The first is to transform the original hexagonal arrangement into a rectangular one through a coordinate transformation. As shown in Fig. 6, such a transformation disrupts the original symmetry and neighborhood relationships, and also requires filling certain gaps with empty data. Nevertheless, the efficiency advantage of conventional CNNs is largely retained, and we included this method as one of the approaches tested in this study.

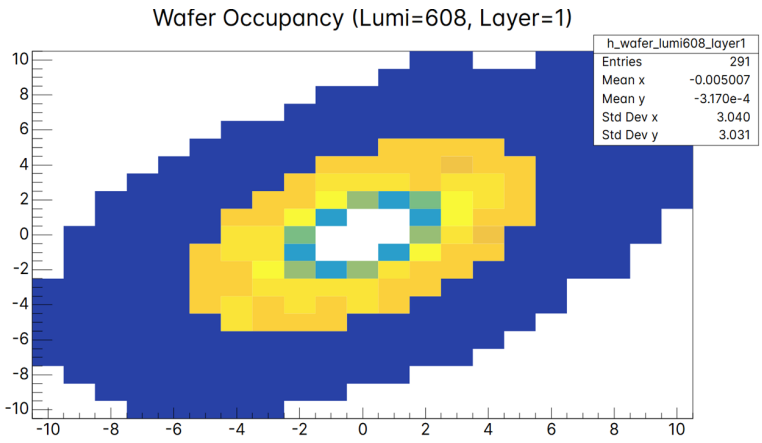

Figure 6: Transformation of the original hexagonal arrangement into a rectangular grid. This transformation breaks the original symmetry and neighborhood relationships, and requires filling some gaps with empty data.

The second approach is to treat the hexagonal distribution as a high-resolution image and use it directly as the CNN input. This, however, inevitably introduces substantial redundancy and requires a much larger training set for the model to learn the meaning of the image represen-

tation. Given the limitations of the available training data, this method was deemed impractical and was therefore discarded.

2.3 Radial Distribution Function

The Radial Distribution Function (RDF) [6] is a method for extracting key information from a distribution while reducing its dimensionality. For the discrete digi occupancy distribution on the HGCAL, the RDF approach can be used to transform it into a one-dimensional discrete function, which can then be used as the input to a conventional deep neural network (DNN) for building an autoencoder.

The RDF is calculated as

$$\mathrm{RDF}(d) = \sum_{i,j} w_i w_j \, \delta(d_{ij} - d), \tag{1}$$

where

- i, j denote indices over all sensor cells within a wafer,
- w_i and w_j are the digi occupancies of cells i and j, respectively,
- d_{ij} is the distance between cells i and j,
- d is the specific distance bin being evaluated,
- $\delta(\cdot)$ is the discrete delta function, equal to 1 if $d_{ij} = d$ and 0 otherwise.

This is equivalent to transforming the original distribution into a function of the distance d. If d does not correspond to the distance between any two points in the distribution, then $\mathrm{RDF}(d)$ is guaranteed to be zero. If there exists one or more pairs of points whose separation is d, then $\mathrm{RDF}(d)$ will be nonzero, with its value reflecting the weighted contribution of all such pairs.

The u, v coordinate system is a natural choice to describe the positions of cells within each hexagonal wafer. These coordinates correspond to axes aligned with the hexagonal lattice directions, enabling a discrete and integer-valued representation of cell locations that simplifies distance calculations.

In this problem, computing d using the u, v coordinates is straightforward, and d^2 is guaranteed to be an integer. Specifically, letting $\Delta u = u_i - u_j$ and $\Delta v = v_i - v_j$ denote the coordinate differences between cells i and j, the squared distance in the uv-plane is defined as

$$d^2 = \Delta u^2 + \Delta v^2 - \Delta u \, \Delta v. \tag{2}$$

Furthermore, it is easy to extend the calculation by adding a third spatial dimension, such as processing multiple layers simultaneously. Introducing a coordinate z, the distance can be generalized as

$$d^2 = \Delta u^2 + \Delta v^2 - \Delta u \, \Delta v + \gamma_z \, \Delta z^2, \tag{3}$$

where $\Delta z = z_i - z_j$ and γ_z is a scaling factor to balance units. Similarly, a temporal dimension can be incorporated by including a time difference term $\Delta t = t_i - t_j$, resulting in the full form

$$d^2 = \Delta u^2 + \Delta v^2 - \Delta u \, \Delta v + \gamma_z \, \Delta z^2 + \lambda_t \, \Delta t^2, \tag{4}$$

where λ_t weights the relative importance of the temporal separation. This formulation allows simultaneous treatment of spatial and temporal distributions over a period of time.

Based on these advantages, the RDF method is expected to perform anomaly detection more effectively. In the following, we compare the performance of the RDF-based approach with that of CNNs to determine which method is better suited for anomaly detection in this context.

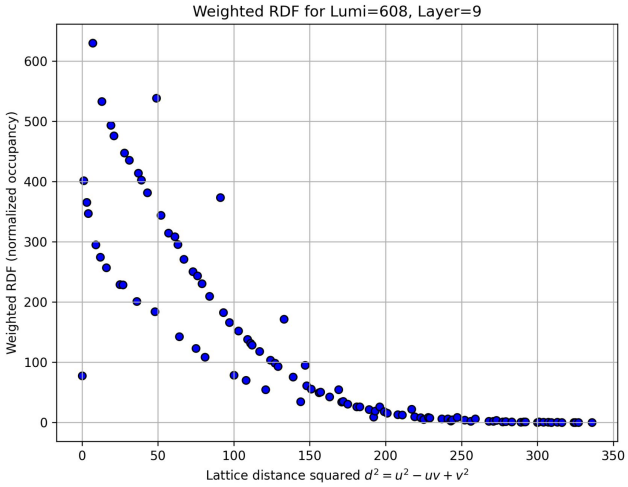

Figure 7: Example RDF distribution.

3 Dataset

3.1 MC Simulation

The HGCAL has not yet been installed on CMS, so only MC simulations are available to provide training and testing data. We use 1000 lumi sections of simulated data generated in other studies within the collaboration. In principle, more MC data could be generated, but this would require significant time and effort, and the simulation itself is not the focus of this study. Each lumi section is pileup combined with the four adjacent lumi sections to mitigate the effects of fluctuations.

The digi occupancy of the three wafer types is normalized separately. Types 0 and 1 have relatively high digi occupancy values and are normalized using min-max scaling, while type 2 uses max normalization. For the CNN training data, an additional median normalization step is applied. For the RDF training data, after calculating the RDF function, min-max normalization is performed again on the resulting values.

3.2 RDF Computation

For each wafer type, the possible values of distances between pairs of points are finite. Therefore, all possible values of d are first computed, and a $1 \times n$ vector (where n is the number of possible d values) is generated. The corresponding values of RDF(d) are then filled into this vector to form a single training data sample. Table 2 shows an overview of the RDF training data structure.

Table 2: Example structure of RDF training data for type 0 wafer.

d2_0	d2_1	...	d2_116
253.44856	1202.4387	...	16.361462
247.43417	1176.0359	...	15.851826

3.3 Anomaly Injection

The anomaly injection procedure is performed as follows. First, a wafer is randomly selected. Then, the original digi occupancy of this wafer is multiplied by a degradation level factor to

simulate different types of anomalies: 0 corresponds to a dead channel, 0.5 represents a degrading channel, and 2 indicates a hot channel.

Out of the entire dataset, 75% is used as training data. Due to the nature of the autoencoder, the training data consists entirely of normal samples. The remaining 25% serves as testing data, of which 12.5% are normal samples and the other 12.5% contain injected anomalies. Since there are three types of degradation, the same data is injected with each anomaly type separately, resulting in three distinct anomaly testing sets.

4 Training

4.1 Model Structure

The encoder structures used in this study are as follows. For the RDF-based autoencoder, the input dimensions differ according to the wafer type: Type 0: $33 \to 256 \to 126 \to 64 \to 32 \to 16$, Type 1: $29 \to 256 \to 126 \to 64 \to 32 \to 16$, Type 2: $101 \to 256 \to 126 \to 64 \to 32 \to 16$.

For the CNN-based autoencoder, the input and intermediate feature map sizes vary with wafer type: Type 0: $1 \times 9 \times 9 \to 16 \times 9 \times 9 \to 32 \times 9 \times 9 \to 32 \times 3 \times 3 \to 64 \times 3 \times 3$, Type 1: $1 \times 11 \times 11 \to 16 \times 11 \times 11 \to 32 \times 6 \times 6 \to 64 \times 3 \times 3$, Type 2: $1 \times 21 \times 21 \to 16 \times 21 \times 21 \to 32 \times 10 \times 10 \to 32 \times 5 \times 5$.

The decoder is essentially a mirror of the encoder, reconstructing the input from the latent representation.

4.2 Training Results

Table 3 and Table 4 presents all the training results. For models with an AUC less than or equal to 0.5, metrics such as Precision were not calculated.

Table 3: RDF results: Precision, Recall, F1 score and AUC for normal/anomaly cases.

Type	Degradation	Precision		Recall		F1 score	
		Normal	Anomaly	Normal	Anomaly	Normal	Anomaly
0	0.0	0.99	1.00	1.00	0.99	1.00	1.00
0	0.5	0.99	0.98	0.98	0.99	0.99	0.99
0	2.0	0.99	1.00	1.00	0.99	1.00	1.00
1	0.0	0.92	0.91	0.91	0.92	0.92	0.92
1	0.5	0.73	0.79	0.82	0.69	0.77	0.74
1	2.0	0.86	0.86	0.86	0.86	0.86	0.86
2	0.0	0.61	0.56	0.42	0.73	0.50	0.63
2	0.5	0.51	0.55	0.81	0.23	0.63	0.33
2	2.0	0.54	0.59	0.76	0.35	0.63	0.44

Type	Degradation	AUC
0	0.0	1.000
0	0.5	0.998
0	2.0	0.000
1	0.0	0.965
1	0.5	0.816
1	2.0	0.922
2	0.0	0.591
2	0.5	0.516
2	2.0	0.555

The results indicate that the RDF method indeed has advantages over CNN for this problem. However, detecting anomalies in type 2 wafers remains challenging. The CNN's AUC

Table 4: CNN results: Precision, Recall, F1 score and AUC for normal/anomaly cases.

Type	Degradation	Precision		Recall		F1 score	
		Normal	Anomaly	Normal	Anomaly	Normal	Anomaly
0	0.0	0.79	1.00	1.00	0.74	0.88	0.85
0	0.5	0.72	1.00	1.00	0.61	0.84	0.75
0	2.0	0.79	1.00	1.00	0.73	0.88	0.84
1	0.0	0.57	0.68	0.82	0.38	0.67	0.49
1	0.5	0.56	0.57	0.62	0.51	0.59	0.54
1	2.0						
2	0.0						
2	0.5						
2	2.0						

Type	Degradation	AUC
0	0.0	0.851
0	0.5	0.812
0	2.0	0.855
1	0.0	0.624
1	0.5	0.579
1	2.0	0.447
2	0.0	0.500
2	0.5	0.500
2	2.0	0.500

being below 0.5 for type 1 with degradation level 2 is due to the normalization strategy: after normalization, the total digi occupancy for degradation level 2 is much lower than that of the normal data, resulting in a lower reconstruction error than for normal samples. This reflects an inherent limitation of the autoencoder architecture. Performing more thorough normalization (e.g., normalizing by total digi occupancy) would cause loss of some critical information, while insufficient normalization causes the overall distribution magnitude to influence the reconstruction error. The RDF approach can partially mitigate this problem, which constitutes one of its main advantages over CNN.

5 Conclusion

In this work, we studied anomaly detection in the CMS HGCAL detector using autoencoders with two different feature types: Radial Distribution Function (RDF) and Convolutional Neural Networks (CNN). Because the HGCAL has a hexagonal layout, CNNs have some difficulties in directly handling the data. By converting the data into RDF, which summarizes the spatial relationships in a simple one-dimensional form, we were able to build a more effective and stable anomaly detection model.

Our tests on simulated data with different types of anomalies show that the RDF-based autoencoder generally performs better than the CNN-based one. The RDF method also helps reduce problems caused by how the data is normalized, which can affect how well the autoencoder detects anomalies. However, detecting anomalies in the type 2 wafers is still a challenge and needs further work.

Looking ahead, the RDF approach is still quite new in high energy physics, and there are many opportunities to explore its use in other areas. In particular, RDF could be useful for analyzing irregular spatial and temporal data in other scientific fields or industrial applications. We hope this study encourages more research into RDF and similar methods for handling complex data structures.

References

[1] CMS Collaboration, "The CMS experiment at the CERN LHC," *JINST* 3 S08004 (2008).

[2] CMS Collaboration, "The Phase-2 Upgrade of the CMS Endcap Calorimeter," *Technical Design Report*, CERN-LHCC-2017-023.

[3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] Yunxiang Zhao, Qiuhong Ke, Flip Korn, Jianzhong Qi, and Rui Zhang, "HexCNN: A Framework for Native Hexagonal Convolutional Neural Networks," *submitted to ICDM (International Conference on Data Mining)*, January 25, 2021.

[6] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, pp. 504–507, 2006.