Data Quality Monitoring through Transfer Learning on Anomaly Detection for the Hadron Calorimeters

Ⓘ **Mulugeta Weldezgina Asres**
Centre for Artificial Intelligence Research (CAIR)
University of Agder, Norway
mulugetawa@uia.no

Ⓘ **Christian Walter Omlin**
Centre for Artificial Intelligence Research (CAIR)
University of Agder, Norway
christian.omlin@uia.no

Ⓘ **Long Wang**
University of Maryland, USA
long.wang@cern.ch

Ⓘ **Pavel Parygin**
University of Rochester, USA
pavel.parygin@cern.ch

Ⓘ **David Yu**
Brown University, USA
david_yu@brown.edu

Ⓘ **Jay Dittmann**
Baylor University, USA
jay_dittmann@baylor.edu

The CMS-HCAL Collaboration
CERN, Switzerland

August 30, 2024

Abstract

The proliferation of sensors brings an immense volume of spatio-temporal (ST) data in many domains for various purposes, including monitoring, diagnostics, and prognostics applications. Data curation is a time-consuming process for a large volume of data, making it challenging and expensive to deploy data analytics platforms in new environments. Transfer learning (TL) mechanisms promise to mitigate data sparsity and model complexity by utilizing pre-trained models for a new task. Despite the triumph of TL in fields like computer vision and natural language processing, efforts on complex ST models for anomaly detection (AD) applications are limited. In this study, we present the potential of TL within the context of AD for the Hadron Calorimeter of the Compact Muon Solenoid experiment at CERN. We have transferred the ST AD models trained on data collected from one part of a calorimeter to another. We have investigated different configurations of TL on semi-supervised autoencoders of the ST AD models—transferring convolutional, graph, and recurrent neural networks of both the encoder and decoder networks. The experiment results demonstrate that TL effectively enhances the model learning accuracy on a target subdetector. The TL achieves promising data reconstruction and AD performance while substantially reducing the trainable parameters of the AD models. It also improves robustness against anomaly contamination in the training data sets of the semi-supervised AD models.

Keywords Transfer Learning · Anomaly Detection · Spatio-Temporal · Deep Learning · DQM · CMS

Acronyms

AD	Anomaly Detection	**FC**	Fully-Connected Neural Network
AUC	Area under the Curve	**FPR**	False Positive Rate
CMS	Compact Muon Solenoid	**GNN**	Graph Neural Network
CNN	Convolutional Neural Network	**GRAPHSTAD**	Graph Based ST AD model
DL	Deep Learning	**HCAL**	Hadron Calorimeter
DQM	Data Quality Monitoring	**HB**	HCAL Barrel subdetector
ECAL	Electromagnetic Calorimeter	**HE**	HCAL Endcap subdetector

HF	HCAL Forward subdetector	**RNN**	Recurrent Neural Network
HO	HCAL Outer subdetector	**SiPM**	Silicon Photo Multipliers
HPD	Hybrid Photodiode Transducers	**ST**	Spatio-Temporal
KL	Kullback-Leibler divergence	**TL**	Transfer Learning
LHC	Large Hadron Collider	**TS**	Time Series
LR	Learning Rate	**TW**	Time Window
LS	Luminosity Section	**VAE**	Variational Autoencoder
LSTM	Long Short-Term Memory	γ	Digi-occupancy Map
MAE	Mean Absolute Error	$i\eta$	*ieta* axis of the HCAL channels
MSE	Mean Squared Error	$i\phi$	*iphi* axis of the HCAL channels
QIE	Charge Integrating and Encoding	*depth*	*depth* axis of the HCAL channels
RBX	Readout Box		

1 Introduction

Spatio-temporal (ST) anomaly detection (AD) is one of the promising monitoring applications of deep learning (DL) in several fields [1–7]. A unique quality of ST data is the presence of dependencies among measurements induced by the spatial and temporal attributes, where data correlations are more complex to capture by conventional techniques [1]. A spatio-temporal anomaly can thus be defined as a data point or cluster of data points that violate the nominal ST correlation structure of the normal data points. DL models dominate the recent AD studies, as AD models capture complex structures, extract end-to-end automatic features, and scale for large-volume data sets [8–10]. The AD models can broadly be categorized as: 1) supervised methods require labeled anomaly observations, and 2) unsupervised approaches use unlabeled data and are more pragmatic in many real-world applications, as data labeling is tedious and expensive. Unsupervised AD models trained with only healthy observations are often called semi-supervised approaches. Semi-supervised AD models have accomplished promising performance in reliability, safety, and health monitoring applications in several domains [8, 9, 11].

The deployment of ST DL models in a new environment is often circumscribed by the limited amount of clean data [12]. Data curation for DL modeling remains cumbersome and particularly challenging for temporal data despite abundant availability. Transfer learning (TL) mechanisms have been proposed for DL models to mitigate the challenge of data insufficiency; it accelerates model training and enhances accuracy [12–17]. It aims to achieve in-domain and cross-domain learning by extracting useful information from the model or data of the source task and transferring it to the target tasks [18, 19]. TL is widely employed in computer vision (e.g., a large image classifier trained on over 1000 classes with IMAGENET1K [20] is fine-tuned to classify a few types of fruit categories) [18] and natural language processing (e.g., a BERT [21], initially trained on massive and diverse text corpus to learn general language features like syntax and semantics is fine-tuned with smaller task-specific data sets, to specialize for question answering tasks) [19]. It has also been proposed for time series (TS) sensor data related to machine monitoring [13], electricity loads [14], medical [15], dynamic systems [16], and ST data for crowd prediction [12, 17]. The TL on ST data for AD application remains limited [10, 19].

Our study discusses ST AD modeling for the *Compact Muon Solenoid* (CMS) experiment at the *Large Hadron Collider* (LHC) [22, 23]. The CMS experiment, one of the two high-luminosity general-purpose detectors at the LHC, consists of a tracker to reconstruct particle paths accurately, two calorimeters: the *electromagnetic* (ECAL) and the *hadronic* (HCAL) to detect electrons, photons and hadrons, and a *muon* system [23, 24]. The CMS experiment employs the *Data Quality Monitoring* (DQM) system to guarantee high-quality physics data through online monitoring that provides live feedback during data acquisition, and offline monitoring that certifies the data quality after offline processing [25]. The online DQM identifies emerging problems using reference distributions and predefined tests to detect known failure modes using summary histograms, such as a digi-occupancy map of the calorimeters [26, 27]. A digi-occupancy map contains the histogram record of particle hits of the data-taking channels of the calorimeters at the digitization level. The CMS calorimeters may encounter problems during data taking, such as issues with the frontend particle sensing scintillators, digitization and communication systems, backend hardware, and algorithms, which are usually reflected in the digi-occupancy map. The growing complexity of detectors and the variety of physics experiments make data-driven AD systems essential tools for CMS to automate the detection, identification, and localization of detector anomalies [2, 28]. Recent efforts in DQM at CMS have presented DL for AD applications [2, 25, 28–31]. The synergy in AD has thus far achieved promising results on spatial 2D histogram maps of the DQM for the ECAL [28, 29], the muon detectors [31] and ST 3D maps of the HCAL [2].

Further study on TL for ST AD models, often involving combinations of spatial and temporal learning networks, is essential considering the achievements of TL in other domains [19]. The recent ST DL models are hybrid and commonly made of combinations of variants of convolutional neural networks (CNNs), recurrent neural networks (RNNs), graph

neural networks (GNNs), and transformers for various data mining tasks [10, 12, 17, 32]. Our study investigates the potential leverages and limitations of TL on ST semi-supervised AD models. We discuss the GRAPHSTAD system [2]—an autoencoder model made of CNN, RNN, and GNN—to investigate TL for ST AD. The GRAPHSTAD has been proposed for online DQM to automate monitoring of the HCAL channels through DL [2]; it captures abnormal events using spatial appearance and temporal context on digi-occupancy maps of the DQM. The GRAPHSTAD employs CNNs to capture the behavior of adjacent channels exposed to regional collision particle hits, GNNs to learn local electrical and environmental characteristics due to a shared backend circuit of the channels, and RNNs to detect temporal degradation on faulty channels. We have transferred a pre-trained GRAPHSTAD model on the source *HCAL Endcap* (HE) subsystem into another target subsystem of the *HCAL Barrel* (HB) for the TL experiment. The HE and HB are subdetectors of the HCAL; they are designed to capture hadron particles at different positions of the calorimeter. The subdetectors share similarities but also have differences in design, technology, and configuration, such as detector segmentations [33].

We have provided insights on TL using various training modes with different network hierarchies of the autoencoder of the GRAPHSTAD system. The experiment has demonstrated the potential of TL when applied to the feature extraction encoder and the reconstruction decoder networks with different fine-tuning mechanisms on the target dataset. We have also examined the impact of within and across time-windows RNN state preservation on ST reconstruction when TL is employed during model inferencing. The TL has achieved promising ST reconstruction and AD while reducing the trainable parameters and providing better robustness against anomaly contamination in the training dataset. Our study demonstrates the efficacy of ST TL in overcoming training data sparsity and model training computation.

We discuss the related work in TL and the CMS DQM system in Section 2. We describe our data sets in Section 3 and the experiment AD and TL methodologies in Section 4. Section 5 presents the performance evaluation and discussion of the results. We provide the conclusion and review the impact of our results in Section 6.

2 Background

This section discusses TL in DL models and provides an overview of the DQM system of the CMS experiment.

2.1 Transfer Learning on Deep Learning

In the last decade, the effectiveness of DL in handling large data sets has caught the attention of both academia and industry. Its ability to learn nonlinear behavior, along with end-to-end automatic feature extraction, allows it to find complex patterns within high-dimensional large data sets. However, most DL models are complex and require extensive data sizes for modeling, which can be expensive and time-consuming to curate, especially in the case of temporal data. Transfer learning approaches, which incorporate pre-trained models into new tasks, are potential solutions for developing DL models when clean data is limited [13–16].

Transfer learning is a paradigm where knowledge from a source pre-trained model on different domains (e.g., different data sources or data sets) or tasks (e.g., different model applications) is utilized to improve the efficacy of a target model [18]. The TL techniques in the literature can broadly be categorized into various taxonomies [18]. One of the typical categorizations is based on the similarity of the task and domain between the source and target [19, 34, 35]: 1) inductive TL: the source and target tasks are different but their domains may remain the same, 2) transductive TL: the tasks remain the same, but the domains are different, and 3) unsupervised TL: similar to inductive transferring on different but related tasks with unlabeled data sets. TL can be carried out on 1) model parameters, where all or some parameters are transferred from a pre-trained source model, and 2) data, where all or part of the source domain data instances are utilized to train the target model. In this study, TL signifies the use of learned network parameters (weights and biases from a source model pre-trained on adequate data sets) on a target model for a related task on a different data set, with or without fine-tuning of the parameters [35]. The target data set may be smaller than the source data set.

Computer vision and natural language processing applications have hugely benefited from TL [18, 19]. The recent successes of generative models on image and sequential data have ameliorated the adoption of TL methods on several applications [19]. The notable contribution of TL is significant in transferring feature extraction networks (encoders) that are trained on immense data sets with very expensive computation grids. Robustly extracted features reduce the model complexity and training cost of the fine-tuned decision networks while enhancing accuracy. We refer to such TL mechanism as *freeze and fine-tune* approach. Although abundant studies are available for images and language text, TL is relatively less explored for temporal data, such as sensor measurement data sets [34, 35]; TS data sets are not readily available or accessible on the internet, unlike images and texts, and the data sets are often multidimensional and so diverse that require domain-specific knowledge for data curation and preparation.

TL on temporal data has been investigated in applications such as machine monitoring [13], electricity loads [14], medical [15], and dynamic systems [16]. Shao et al. [13] employ a freeze and fine-tune TL to improve the training speed and accuracy using pre-trained very deep convolutional networks model on machine sensor data converted into image format by a wavelet transformation to obtain time-frequency distributions. The initial network layers of the pre-trained model (the feature extraction networks) are frozen, whereas the lower layers (decision networks) are fine-tuned. Laptev et al. [14] have frozen the temporal long short-term memory (LSTM) networks while training only the forecasting fully-connected neural networks (FC) on an electricity power consumption data set. The authors have demonstrated the effectiveness of TL in learning different TS signal patterns and its robustness against noise on a small training data set. Gupta et al. [15] discuss TL on a deep-gated recurrent unit for multivariate classification for clinical data sets. The authors adopted pre-trained models on several classification tasks to provide generic features for simpler linear logistic regression models on new target tasks. They have shown that models trained with TL outperformed task-specific models, and are more robust to the size of labeled data. Boulle et al. [16] have investigated the potential of TL on fully convolutional networks and residual neural networks for chaotic TS classification in dynamic systems. They trained the models on a given chaotic signal pattern and tested them on different chaotic univariate signals. Wen et al. [35] present MU-NET to employ TL from univariate U-NET network to multivariate AD task using a freeze and fine-tune TL approach. They applied the pre-trained U-NET to each multivariate input variable for feature extraction when scaling it to the MU-NET.

There are also a few efforts in adopting TL for ST data [12, 17, 36]. Wang et al. [17] have applied TL for cross-city crowd-flow prediction where feature extraction CONVLSTM network of the forecasting model trained on one city is fine-tuned on another city data set. Wang et al. [12] present an end-to-end TL framework for cross-domain urban crowd-flow prediction using a deep adaptation mechanism on CONVLSTM networks. The deep adaptation network matches the embedding representations of the source and target domain distributions to learn the transferable features between two domains [12, 37]. Guo et al. [36] fine-tune an autoencoder for a store recommendation system from a model trained on a different city data set.

Recent DL models built on hybrids of CNNs [3, 38, 39], RNNs [40, 41], and GNNs [4, 41] have gained momentum for TS and ST data in AD and other data mining applications [10, 12, 17, 32]. Thus far, most TL studies focus on feature extraction encoding networks and predominantly on forecasting tasks [37]. We have studied the transferability of CNN, GNN, and RNN networks on both the encoder and decoder networks of an ST autoencoder and qualitatively evaluated the effectiveness of the TL on reconstruction and AD tasks.

2.2 The Hadron Calorimeter of the CMS Detector

Figure 1a shows the CMS experiment and the HCAL detector inside CMS [23, 24]. The calorimeters of the CMS detector are highly segmented to improve the accuracy of energy-deposition profile-measurement and particle identification [23, 24, 42]. The segmentation geometry of the detector is represented using η and ϕ spaces which correspond to *pseudo-rapidity* and *azimuth*, respectively (as shown in Figure 1b). The z-axis lies along the incident beam direction, ϕ is azimuthal angle between the x and y axis, and η is calculated from the polar angle θ_{cm} between z and $xy-$plane as:

$$\eta = -\ln(\tan(\theta_{cm}/2)) \tag{1}$$

where the x, y, and z are orthogonal axis of the the cylinder, the θ_{cm} is the center-of-mass scattering angle, and the $\ln$ is a natural log function. The $\eta - \phi$ space corresponds to a rectangular coordinate system representing an outgoing particle's direction from the center of the detector (where the collision occurs). Particles traveling in the same direction lie near each other in $\eta - \phi$ space.

Figure 2a illustrates the four major subdetectors of the HCAL covering different segments in the CMS detector: the HB, the HE, the *HCAL Outer* (HO), and the *HCAL Forward* (HF). Since this study's data sets are from the LHC Run-2 collision experiment, we will describe below the HCAL system configurations from 2018. The HB and HE are sampling calorimeters with a brass absorber and active plastic scintillators to measure the energy depositions [24]. The subdetectors surround the ECAL and are fully immersed within the strong magnetic field of the solenoid: the HB are joined hermetically with the barrel extending out to $|\eta| = 1.4$, and the HE covering the overlapping range $1.3 < |\eta| < 3.0$ (as shown in Figure 2b). The HF is located 11.2 meters from the interaction point and extends the pseudo-rapidity coverage (overlapping with the HE) from $|\eta| = 2.9$ to $|\eta| = 5$. The central shower containment in the region $|\eta| < 1.26$ is improved with the HO, an array of scintillators located outside the magnet.

The front-end electronics of the HCAL, responsible for sensing and digitizing optical signals of the collision particles, are divided into sectors of *readout boxes* (RBXes) that house the electronics and provide voltage, backplane communications, and cooling. The use cases of our study, the HE and HB, consist of 36 RBXes arranged on the plus (HE[HB]P) and minus (HE[HB]M) hemispheres of the CMS detector. The frontend acquisition systems transmit the photons produced in the plastic scintillators through the wavelength-shifting fibers to the *Silicon photomultipliers* (SiPMs) [HE] or the
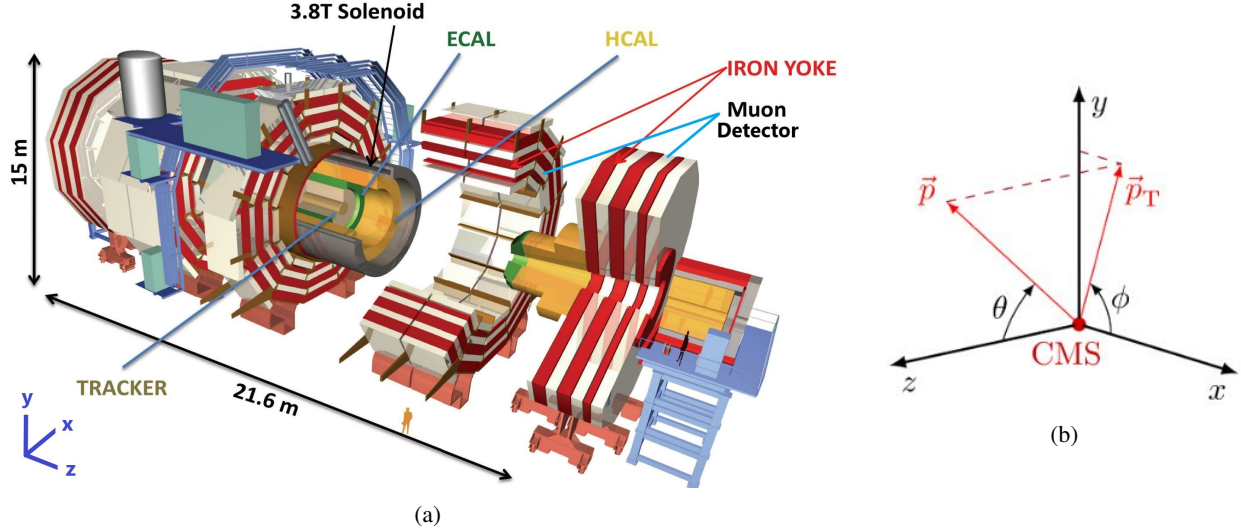
Figure 1: Schematic of the CMS detector: a) CMS with its major systems [42], and b) geometry axes and angles of the CMS with respect to collision intersection point [43].

hybrid photodiode transducers (HPD) [HB] [24]. Each RBX houses frontend electronics that include four digitization *readout modules* (RMs), the *next-generation clock and control module*), and the *calibration unit* [24]. Each RM is made of SiPMs [HE] or HPD [HB], a SiPM control card, and four readout *charge integrator and encoder* (QIE) cards, each with several QIE chips and *field programmable gate array* (FPGA) modules. A QIE chip integrates charge from one SiPM [HE] or HPD [HB] at 40 MHz, and the FPGA serializes and encodes the data from the QIE chips (channels).

2.3 The CMS Data Quality Monitoring

The collision data of the LHC is organized into *runs* where each run contains thousands of luminosity sections (also called lumisections). A *lumisection* (LS) corresponds to approximately 23 seconds of data taking and comprises hundreds or thousands of collision events containing particle hit records across the CMS detector. The DQM system in CMS provides feedback on detector performance and data reconstruction; it generates a list of certified runs for physics analyses and stores it in the "Golden JSON" [25]. The DQM employs online and offline monitoring mechanisms: 1) the *online monitoring* is real-time DQM during data acquisition, and 2) the *offline monitoring* provides the final fine-grained data quality analysis for data certification after 48 hours since the collisions were recorded. The online DQM populates a set of histogram-based maps on a selection of events and provides summary plots with alarms that DQM experts inspect to spot problems. The *digi-occupancy* map is one of the histogram maps generated by the online DQM, and it contains particle hit histogram records of the particle readout channel sensor of the calorimeters. A digi, also called a hit, is a reconstructed and calibrated collision physics signal of the calorimeter. Several errors can arise in the calorimeter affecting the frontend particle sensing scintillators, the digitization and communication systems, the backend hardware, or the algorithms. These errors appear in the digi-occupancy map as holes, under- or over-populated, or saturated bins. Previous efforts by Refs. [2, 25, 29–31] demonstrate the promising AD efficacy of using digi-occupancy maps for calorimeter channel monitoring using machine learning. Our GRAPHSTAD [2] has extended the efforts in AD for the HCAL with ST modeling of the 3D digi-occupancy maps of the DQM. The GRAPHSTAD incorporates both CNN and GNN [46, 47] to capture Euclidean and non-Euclidean spatial characteristics, respectively, and RNN for temporal learning for the HCAL channels.

3 Dataset Description

We utilized digi-occupancy data of the online DQM system of the CMS experiment to train and validate our models. The data contains healthy digi-occupancy maps with a 20 fC minimum threshold and were selected from certified good collision runs as referred to by the "Golden Json" of CMS. The digi-occupancy data sets were collected in 2018 during the LHC Run-2 collision experiment with the received luminosity per lumisection up to 0.4 pb^{-1}, and the number of events up to 2250. The source and target data sets contain three-dimensional digi-occupancy maps for the HE and HB subsystems of the HCAL, respectively (as shown in Figure 3). The similarities between the source and target data sets and tasks have been established to be essential factors that impact the performance of TL [35].
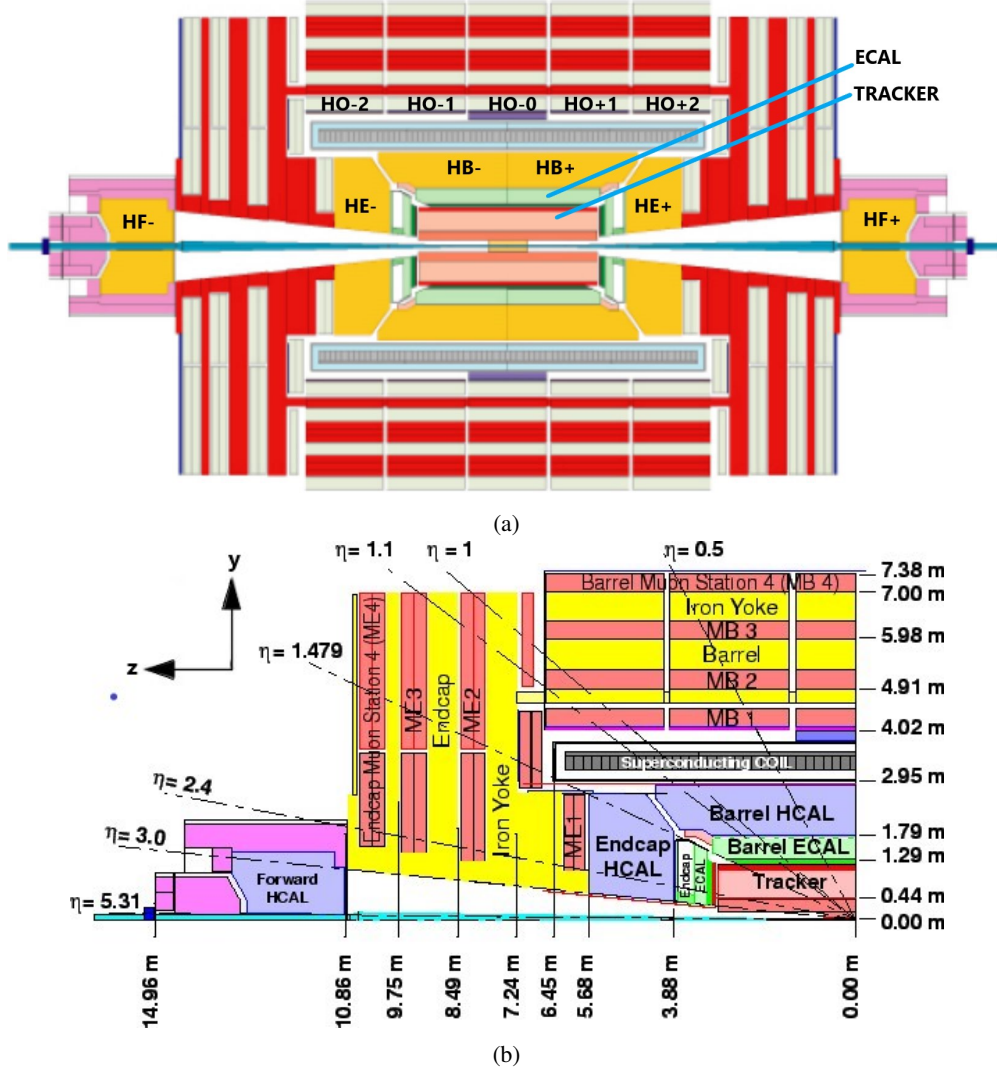
Figure 2: The subdetectors of the HCAL: a) longitudinal view of the HB, HE, HF, and HO subdetectors on CMS [44], and b) longitudinal view of one quadrant of CMS with segmentation angle specifications of the η, where the origin denotes the interaction point [24, 45].

The digi-occupancy map contains a particle hit count of the calorimeter readout channels for a given period of time. The HCAL covers a considerable volume of CMS and has a fine segmentation along three axes ($i\eta \in [-32, \ldots, 32]$, $i\phi \in [1, \ldots, 72]$ and $depth \in [1, \ldots, 7]$). The $i\eta$ and $i\phi$ denote integer notation of the towers covering ranges of η and ϕ of the CMS detector, respectively [24]. The digi-occupancy measurement corresponds to a hit record of the readout channels at the segmentation positions. The source system HE covers $|i\eta| \in [16, \ldots, 29]$, $i\phi \in [1, \ldots, 72]$ and $depth \in [1, \ldots, 7]$, whereas the target system HB takes $|i\eta| \in [1, \ldots, 16]$, $i\phi \in [1, \ldots, 72]$ and $depth \in [1, 2]$ (as shown in Figure 3). One of the key differences between the HE and HB in the 2018 LHC collision run was the frontend data acquisition optical-to-electrical technology, i.e., the HE was upgraded to SiPMs with QIE11 technology, and the HB utilized HPD with QIE8. We summarize the comparison of the digi-occupancy maps of the source and target data sets in Table 1.

4   Methodology

This section presents the GRAPHSTAD modeling and the experiment setups for the transfer learning study.

(a)

(b)   (c)

Figure 3: A sample digi-occupancy map (*year=2018, RunId=325170, LS=15*): a) digi-occupancy map for the HEHB, b) the source system HE channels are placed in $|i\eta| \in [16, \ldots, 29]$, $i\phi \in [1, \ldots, 72]$, and $depth \in [1, \ldots, 7]$, and c) the target system HB channels are placed in $|i\eta| \in [1, \ldots, 16]$, $i\phi \in [1, \ldots, 72]$, and $depth \in [1, 2]$. The missing sector at the top-left (HE) corresponds to the two failed RBX sectors during the 2018 collision runs.

Table 1: Description of source and target data sets.

Dataset	Technology	Channels/RBX	No. of RBX	Segmentation	Sample Size
Source (HE)	SiPM	192	36	$|i\eta| \in [16, \ldots, 29]$, $i\phi \in [1, \ldots, 72]$, $depth \in [1, \ldots, 7]$	20,000
Target (HB)	HPD	72	36	$|i\eta| \in [1, \ldots, 16]$, $i\phi \in [1, \ldots, 72]$, $depth \in [1, 2]$	7000

4.1 Data Preprocessing

This section describes the data preprocessing stages of the proposed approach, i.e., digi-occupancy renormalization and graph-adjacency matrix generation.

4.1.1 Digi-occupancy Map Renormalization

We apply digi-occupancy renormalization in the data preprocessing stages to normalize the values for the variation in the luminosity and the number of event configurations of the collision experiments [2]. The digi-occupancy (γ) map data of the HCAL varies with the received luminosity (β) and the number of events (ξ) (as shown in Figure 4). The per channel $\gamma_s(i)$ can range $\gamma_s(i) = [0, \xi_s]$, where s denotes the sth LS (3D map) on the data set and the i denotes the ith channel in the sth map. The ξ_s is usually adjusted with the β_s but not always. The β and γ are retrieved from different systems on the existing CMS system; directly accessing the β for the real-time γ AD monitoring requires further effort. We renormalize the maps ($\gamma_s \rightarrow \hat{\gamma}_s$) by the ξ_s to have a consistent interpretation of the γ maps across lumisections:

$$\hat{\gamma}_s = \frac{\gamma_s}{\xi_s} \tag{2}$$

The remaining impact of the β_s is left to be learned by the AD model. Moreover, the non-linearity at the LHC (when ξ remains high while γ drops following β) creates unpredictable spikes; renormalization of the γ with only the ξ does not entirely avoid the issue, and the spikes may affect the training performance of a temporal model. We employ additional reversible renormalization before and after invoking the AD model to mitigate the non-linearity of the digi-occupancy spatial data. The renormalization exploits the symmetric property of the $i\eta$ and *depth* (the γ values are less diverse along the $i\phi$ axis) and divides the input γ of the channels per each $i\eta$ and $depth$ by their median values and reverse the action on the model output.
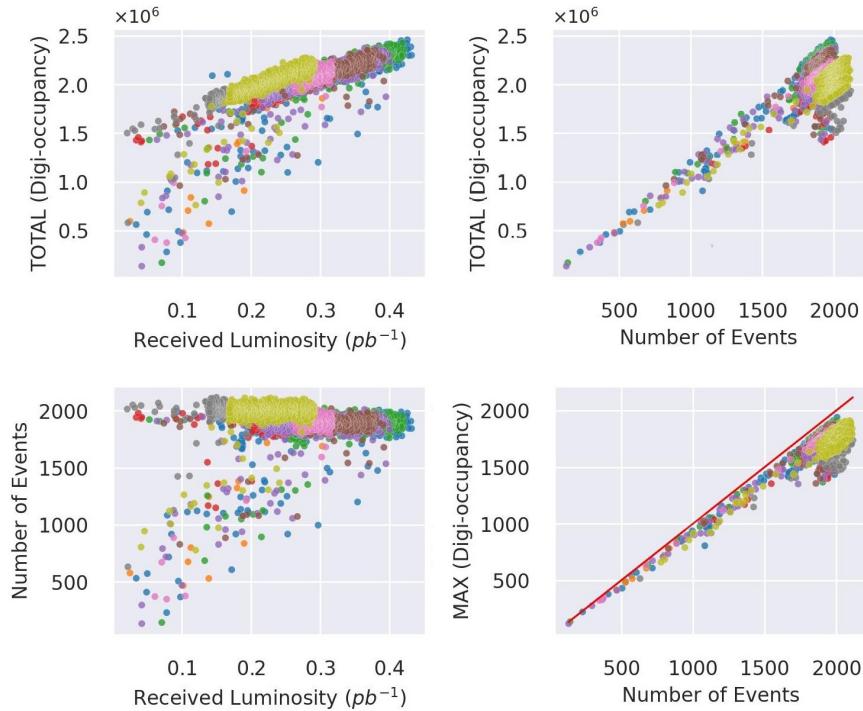

Figure 4: Visualization of the dependency between digi-occupancy of the HB and run setting: the received luminosity and the number of events per LS. The different colors correspond to different runs.

4.1.2 Adjacency Matrix Generation

We deploy an undirected graph network $\mathcal{G}(\mathcal{V}, \Theta)$ to represent the HCAL channels in a graph network based on their connection to a shared RBX system. The graph $\mathcal{G}$ contains nodes $\upsilon \in \mathcal{V}$, with edges $(\upsilon_i, \upsilon_j) \in \Theta$ in a binary adjacency matrix $\mathcal{A} \in \mathbb{R}^{M \times M}$, where M is the number of nodes (the channels). An edge indicates the channels sharing the same

RBX as:

$$A(v_i, v_j) = \begin{cases} 1, & \text{if } \Omega(v_i) = \Omega(v_j) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $\Omega(v)$ returns the RBX identification of the channel v. There are approximately 7000 for the HE and 2600 for the HB channels in a graph representation of the digi-occupancy map. We retrieved the channel to RBX mapping from the 2018 HCAL's calorimeter segmentation map.

4.2 Anomaly Detection Mechanism

We denote the autoencoder model of the AD system as $\mathcal{F}$. It takes ST data $\mathcal{X} \in \mathbb{R}^{T \times N_{i\eta} \times N_{i\phi} \times N_d \times N_f}$ as a sequence in a time window $t_x \in [t - T, t]$, where $N_{i\eta} \times N_{i\phi} \times N_d$ is the spatial dimension corresponding to the $i\eta$, $i\phi$, and $depth$ axes, respectively, and N_f is the number of input variables ($N_f = 1$ as we monitored only a digi-occupancy quantity in the spatial data). The $\mathcal{F}_{\theta,\omega} : \mathcal{X} \to \bar{\mathcal{X}}$, parametrized by θ and ω, attempts to reconstruct the input ST data $\mathcal{X}$ and outputs $\bar{\mathcal{X}}$. The encoder network of the model $\mathcal{E}_\theta : \mathcal{X} \to \mathcal{Z}$ provides low-dimension latent space, $\mathcal{Z} = \mathcal{E}_\theta(\mathcal{X})$, and the decoder $\mathcal{D}_\omega : \mathcal{Z} \to \bar{\mathcal{X}}$, reconstructs the ST data from $\mathcal{Z}$, $\bar{\mathcal{X}} = \mathcal{D}_\omega(\mathcal{Z})$ as:

$$\bar{\mathcal{X}} = \mathcal{F}(\mathcal{X}) = \mathcal{D}(\mathcal{E}(\mathcal{X})) \tag{4}$$

Anomalies can live for a short time on a single digi-occupancy map, or persist over time, affecting a sequence of maps. Aggregated spatial reconstruction error is calculated over a time window T using mean absolute error (MAE) to capture a time-persistent anomaly as:

$$e_{i,MAE} = \frac{1}{T} \sum_{t'=t-T}^{t} |x_i(t') - \bar{x}_i(t')| \tag{5}$$

where $x_i \in \mathcal{X}$ and $\bar{x}_i \in \bar{\mathcal{X}}$ are the input and the reconstructed digi-occupancy of the ith channel. We standardized $e_{i,MAE}$ to homogenize the reconstruction accuracy variations among the channels when generating the anomaly score a_i as:

$$a_i = \frac{e_{i,MAE}}{\sigma_i} \tag{6}$$

where σ_i is the standard deviation of the $e_{i,MAE}$ on the training data set. The standardized anomaly score allows us to use a single AD decision threshold α for all the channels in the spatial map. The anomaly flags are generated after applying α to the anomaly scores ($a_i > \alpha$). The α can be tuned to control the detection sensitivity.

The use-case GRAPHSTAD autoencoder model is made of CNN, GNN, and RNN networks; it employs CNN and GNN with a pooling mechanism to extract relevant features from spatial DQM data followed by RNN to capture temporal characteristics of the extracted features (see Figure 5). It integrates variational layer [48] at the end of the encoder for regularization of autoencoder overfitting by enforcing continuous and normally distributed latent representations [49–52]. We refer readers to [2] for the mathematical formulation and architecture discussion of the GRAPHSTAD autoencoder model.
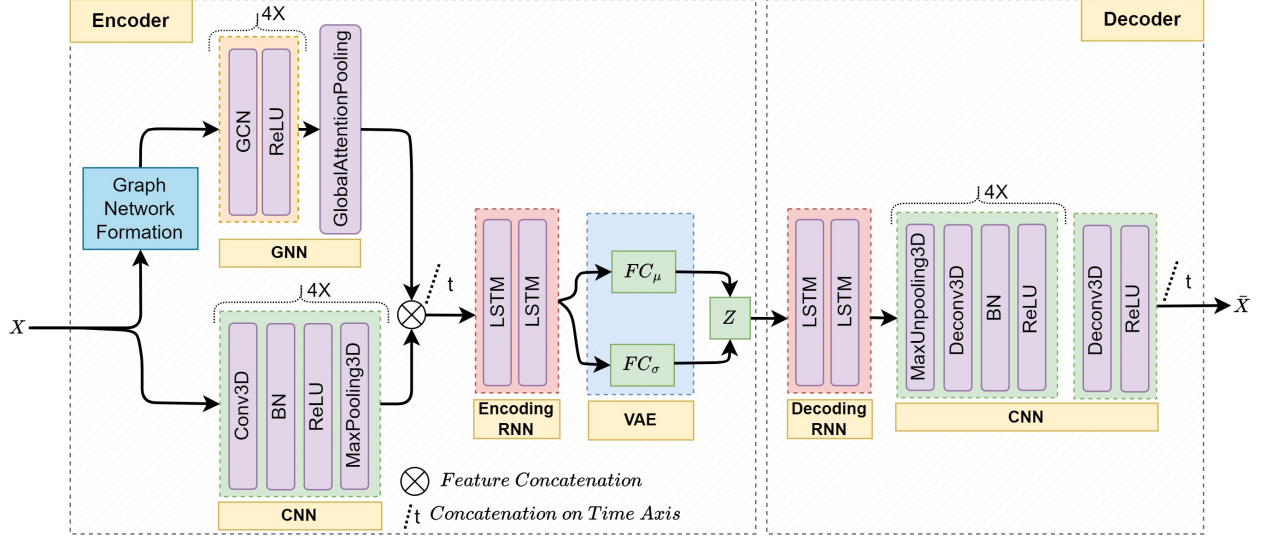
We trained the autoencoder on healthy digi-occupancy maps (without significant anomaly contamination, see Section 3) of the target HB system. We normalized the spatial data per channel into a $[0, 1]$ for effective model training across the variations in calorimeter channels. We utilized a mean squared error (MSE) loss function as:

$$\mathcal{L}_{MSE} = \frac{1}{M} \sum_i (x_i - \bar{x}_i)^2 \tag{7}$$

where x_i and $\bar{x}_i$ is the input and the reconstructed values of the normalized $\hat{\gamma}$ of the ith channel, respectively, and M is the total number of channels. The variational layer of the autoencoder (denoted as VAE in Figure. 5) regularizes the training MSE loss using the *Kullback-Leibler divergence* (KL) distance D_{KL} [48] to achieve the normally distributed latent space as:

$$\mathcal{L} = \underset{W \in \mathbb{R}}{\mathrm{argmin}} \left\{ \mathcal{L}_{MSE} - \lambda D_{KL} \left[\mathcal{N}(\mu_z, \sigma_z), \mathcal{N}(0, I) \right] + \rho \|W\|_2^2 \right\} \tag{8}$$

where $\mathcal{N}$ is a normal distribution with zero mean and unit variance, and $\|.\|_2^2$ is a squared *Frobenius norm* of L_2 *regularization* for the trainable model parameters W [53]. The $\lambda = 0.003$ and $\rho = 10^{-7}$ are tunable regularization hyperparameters. We finally employed the *Adam* optimizer [54] for training.

9

Conv3D: 3D convolutional neural network; GCN: graph convolutional neural networks; Deconv3D: 3D deconvolutional neural networks; BN: batch normalization; LSTM: long short-time memory recurrent networks; FC: fully-connected neural networks.

Figure 5: The architecture of the proposed autoencoder for the GRAPHSTAD system [2]. The GNN and CNN are spatial feature extraction on each time step, and the RNN network captures the temporal behavior of the extracted features. The feature extraction encoder incorporates the GNN for backend physical connectivity among the spatial channels, CNN for regional spatial proximity of the channels, and RNN for temporal behavior extraction. The reconstruction decoder contains RNN and deconvolutional neural networks to reconstruct the spatio-temporal input data from the low dimensional latent features.

4.3 Transfer Learning Approach

Model parameter TL generally consists of four basic steps: 1) selection of a source task with a related modeling problem and an abundance of data where we can exploit the mapping knowledge from the inputs to outputs, 2) development of the source model that performs well in the source task, 3) transfer source model to target model where whole or part of the source model is employed as part of the target model, and 4) fine-tuning the target model on the target dataset if necessary. We present knowledge transfer on GRAPHSTAD autoencoder models, i.e., trained AD model on digi-occupancy maps of the source HE subsystem is transferred to the target HB subsystem. Brute-forcing the knowledge from the source into the target irrespective of their divergence and thorough investigation of the several network building modules would cause certain performance degeneration, impacting the original data consistency in the target domain [18].

We have thus investigated several transferring cases when employing the TL on two principal model training phases: the initialization and training phases.

1. *Init mode*: the trainable network parameters (weights and bias) of the source model are transferred into the target model initialization. The target model is further trained on the target HB dataset, fine-tuning.

2. *Train mode*: The model parameters of the source model are directly reused as the final inference parameters of the target model; the parameters are frozen and excluded from fine-tuning on the target HB dataset.

Let $\mathcal{M}(\Psi, \Omega)$ be an AD model with parameters Ψ and Ω affected and not affected by TL, respectively, and $\mathcal{M}_e(\Psi_e, \Omega_e)$ and $\mathcal{M}_b(\Psi_b, \Omega_b)$ are the source and target models for the HE and HB, respectively. The TL modes $\mathcal{T}$ can be formulated mathematically as:

$$\underset{\text{init mode}}{T} : \mathcal{M}_b(\Psi_e, \Omega_b) \underset{\text{fine-tuning}}{\rightarrow} \mathcal{M}_b(\Psi'_e, \Omega'_b)$$

$$\underset{\text{train mode}}{T} : \mathcal{M}_b(\Psi_e, \Omega_b) \underset{\text{fine-tuning}}{\rightarrow} \mathcal{M}_b(\Psi_e, \Omega'_b)$$

(9)

where the superscript $'$ denotes the parameters that are updated after fine-tuning the $\mathcal{M}_b$ model on the target dataset.

The GRAPHSTAD autoencoder is made of CNNs and GNNs with a pooling mechanism to extract relevant features from high dimensional spatial data followed by RNNs to capture temporal characteristics of the extracted features (as

shown in Figure 5). Table 2 presents the TL mechanisms we apply to the different deep networks of the encoder and decoder to study the impacts on ST digi-occupancy map reconstruction and AD accuracy. We analyze the effect of RNN state preservation within and across time windows. We further investigate the TL to a variation in training iterations and learning rate scheduling methods. The discussion includes the impact of the TL on model accuracy, overfitting, and training stability.

Table 2: Transfer learning experiment configurations.

Config.	Init Mode			Train Mode (on Target Dataset)	
	Notation	Description		Notation	Description
1	RANDOM	Target model is initialized randomly		No-TL	Complete training (fine-tuning)
2	TL-4	Target model is initialized randomly, except the spatial learning networks (CNN and GNN) transferred from the source model		No-TL	
3				TL-1	The GNN of the encoder is frozen (not fine-tuned)
4				TL-2	The CNN of the encoder is frozen
5				TL-2$_d$	The CNN of the decoder is frozen
6				TL-3	The CNN and GNN of the encoder are frozen
7				TL-4	The CNN and GNN of the encoder, and the CNN of the decoder are frozen
8	TL-7	All the spatial and temporal learning networks (CNN, GNN, and RNN) of the target model are initialized by TL from the source model		TL-5	The CNN, GNN and the RNN of the encoder are frozen
9				TL-6	The CNN and GNN of the encoder, and the RNN of the decoder are frozen

TL - Transfer learning is applied.
TL-1: ENCODER[GNN], **TL-2**: ENCODER[CNN], **TL-2**$_d$: DECODER[CNN], **TL-3**: ENCODER[CNN, GNN], **TL-4**: EN-CODER[CNN, GNN], DECODER[CNN], **TL-5**: ENCODER[CNN, GNN, RNN], **TL-6**: ENCODER[CNN, GNN, RNN], DECODER[RNN], **TL-7**: ENCODER[CNN, GNN, RNN], DECODER[RNN], **TL-8**: ENCODER[CNN, GNN, RNN], DE-CODER[CNN, RNN].

The implementation of parameter transferring on DL networks can be accomplished in two ways: 1) start with the source model and then reset (remove and add) the networks that are not included in the TL, and 2) start with the target model with random initialization and update the parameter values of the networks included in the TL from the corresponding source networks. The first approach is widely utilized in DL TL literature and employed for feature extraction; however, it may not be suitable for flexibly choosing layers at different hierarchies, as the target models might have slight variations. Several configuration setups of the autoencoder are derived from the spatial configuration of the input 3D map, which differs for the source HE and target HB systems—e.g., variation in the depth spatial dimension between HE and HB. We have found the second approach more convenient for our study, as we intend to apply TL on different networks of the encoder and decoder of the autoencoder model.

5 Results and Discussion

This section will discuss the results of the TL on different network layers of the GRAPHSTAD autoencoder models. We will investigate the TL on reconstruction accuracy, trainable parameter reduction, and AD performance on the target HB digi-occupancy map dataset. We applied TL for model initialization (*init mode*) without and with fine-tuning (*train mode*) on the target HB dataset. We trained the models on NVIDIA Tesla V100 with 4 GPUs using 4000 digi-occupancy maps from LS 1 to 500 and evaluated the around 3000 maps from LS from 500 to 1500. We utilized 20% of the training dataset (the last time stamps) for validation loss calculation during training to determine the best states for the models. We set the learning rate at 0.001 and the batch size at 6 to train the models with five lumisections per time window.

5.1 Spatio-Temporal Reconstruction Performance

We will discuss below the $\mathcal{L}_{MSE}$ performance of TL applied on spatial (CNNs and GNNs) and temporal (RNNs) learning networks. We will also briefly present comparison results on the learning rate scheduling choice.

5.1.1 Transfer Learning on Spatial Learning Networks

We have assessed the transferability of the DL AD model at the initialization and inference phases for the spatial learning networks (CNNs and GNNs) on both the encoder and decoder networks on different numbers of training epochs (see Figure 6). The TL has reduced the reconstruction error $\mathcal{L}_{MSE}$ of healthy maps by 32.5% to 20.7% when the number of epochs is varied from 75 to 200 (as shown in Figure 6b). The minimum gain of 13% is achieved at epoch 150, just before the performance of the no-TL model starts to saturate. The complete fine-tuning, TL for initialization followed by fine-tuning the whole network, accomplished around 20% improvement. The $\mathcal{L}_{MSE}$ generally decreases, while the relative TL gain roughly decreases as the epoch increases to 150. The results are not entirely unexpected; the DL models may tend to improve performance as the training epoch increases, reducing the gap caused by the difference in initialization and training mechanism. When the epoch increased beyond 150, the randomly initialized model (no-TL) achieves only slight improvement, whereas the $\mathcal{L}_{MSE}$ continues to drop for the TL models—increasing the relative gain of the TL. TL for initialization of all the spatial learning networks of the autoencoder (*init mode* = TL-4) and fine-tuning only the decoder while freezing the encoder (*train mode* = TL-3) achieves the best improvement, from 26% to 32.5%. The TL gain of the GNNs is limited compared to CNNs; the CNNs are the primary networks that learn the input spatial data and have 15 times more parameters than GNNs in the use-case GRAPHSTAD autoencoder model. Transferring and freezing the CNNs of the encoder (TL-2 and TL-3) exhibit stable performance on repeated experiments.

(a)

(b)

Figure 6: Reconstruction performance evaluation ($\mathcal{L}_{MSE}$) of the TL on spatial networks across multiple number of epochs: a) Test MSE loss and the bars show the dispersion of five repeated experiments, and b) the average relative difference of the Test MSE loss with respect to the no-TL. The TL is applied with *init mode* on the encoder and decoder (TL-4: ENCODER[CNN, GNN], DECODER[CNN]) and *train mode* on the encoder using TL-1: ENCODER[GNN], TL-2: ENCODER[CNN], and TL-3: ENCODER[CNN, GNN]. The no-TL model starts to saturate at $epoch > 150$.

Table 3 provides the average and best model ST reconstruction performance, the $\mathcal{L}_{MSE}$. Inference TL on the decoder networks without fine-tuning *Training Mode* = TL-2_d fails to reconstruct the target data adequately. In an autoencoder architecture, the encoder maps the input into low dimensional latent space (information compression), while the decoder attempts to reconstruct (information expansion) the target data from the latent. The decoder networks thus require

fine-tuning on the target dataset to adjust its parameters to the target reconstruction effectively. Boulle et al. [16] have investigated TL on DL for a univariate chaotic TS classification model; they argued that BN without fine-tuning limits the transferability of CNNs. The scaling and shifting parameters for BN and bias parameters are estimated from the training dataset and strongly correlate to the data. We have further studied TL on the decoder when the BN layer and the bias parameters of the CNNs are fine-tuned on the target dataset. The $\mathcal{L}_{MSE}$ is substantially reduced, accuracy improved, by 50% as compared to the frozen decoder (see Table 4). The reconstruction error is still 20 times higher than the without TL model, indicating the CNNs of the decoder also require fine-tuning to achieve reasonable accuracy. The results demonstrate the promising leverage of TL for autoencoder model initialization on both feature extraction encoder and reconstruction decoder networks, whereas fine-tuning with the target data set is essential for the decoder networks.

Table 3: Average ST reconstruction $\mathcal{L}_{MSE}$ performance of TL on spatial networks at $epoch = 200$.

Init Mode	Train Mode	MSE Loss			ΔMSE w.r.t *Init*=RANDOM		
		Train	Validation	Test	Train	Validation	Test
Average Performance							
RANDOM	No-TL	2.650×10^{-04}	2.750×10^{-04}	3.361×10^{-04}	–	–	–
TL-4	No-TL	2.200×10^{-04}	2.250×10^{-04}	2.666×10^{-04}	-17.0%	-18.2%	-20.7%
TL-4	TL-3	1.775×10^{-04}	$\mathbf{1.775 \times 10^{-04}}$	2.489×10^{-04}	-33.0%	**-35.5%**	-25.9%
TL-4	TL-2	$\mathbf{1.725 \times 10^{-04}}$	1.800×10^{-04}	$\mathbf{2.463 \times 10^{-04}}$	**-34.9%**	-34.5%	**-26.7%**
TL-4	TL-1	2.075×10^{-04}	2.125×10^{-04}	2.604×10^{-04}	-21.7%	-22.7%	-22.5%
TL-4	TL-2_d	8.902×10^{-03}	7.380×10^{-03}	1.530×10^{-02}	3259.2%	2583.6%	4452.2%
Best Model (on Test Loss)							
RANDOM	No-TL	2.400×10^{-04}	2.600×10^{-04}	3.085×10^{-04}	–	–	–
TL-4	No-TL	2.100×10^{-04}	2.100×10^{-04}	2.569×10^{-04}	-12.5%	-19.2%	-16.7%
TL-4	TL-3	$\mathbf{1.700 \times 10^{-04}}$	$\mathbf{1.700 \times 10^{-04}}$	2.451×10^{-04}	**-29.2%**	**-34.6%**	-20.5%
TL-4	TL-2	$\mathbf{1.700 \times 10^{-04}}$	1.800×10^{-04}	$\mathbf{2.420 \times 10^{-04}}$	**-29.2%**	-30.8%	**-21.6%**
TL-4	TL-1	2.000×10^{-04}	2.100×10^{-04}	2.502×10^{-04}	-16.7%	-19.2%	-18.9%
TL-4	TL-2_d	8.88×10^{-03}	7.37×10^{-03}	1.5255×10^{-02}	3600.0%	2734.6%	4844.9%

TL-1: ENCODER[GNN], TL-2: ENCODER[CNN], TL-2_d: DECODER: [CNN], TL-3: ENCODER[CNN, GNN], TL-4: EN-CODER[CNN, GNN], DECODER[CNN], TL-5: ENCODER[CNN, GNN, RNN]), TL-6: ENCODER[CNN, GNN, RNN], DE-CODER[RNN], TL-7: ENCODER[CNN, GNN, RNN], DECODER[CNN, RNN].

Table 4: Average ST reconstruction $\mathcal{L}_{MSE}$ performance of TL on spatial networks, decoder with *init mode*=TL-4 at $epoch = 200$.

Train Mode	MSE Loss			ΔMSE w.r.t $TL-2_d$		
	Train	Validation	Test	Train	Validation	Test
TL-2_d	8.902×10^{-03}	7.380×10^{-03}	1.530×10^{-02}	–	–	–
TL-2_d / [BN]	$\mathbf{4.403 \times 10^{-03}}$	3.767×10^{-03}	$\mathbf{7.200 \times 10^{-03}}$	**-50.5%**	-48.9%	**-53.0%**
TL-2_d / [BN, BIAS]	4.405×10^{-03}	$\mathbf{3.760 \times 10^{-03}}$	7.354×10^{-03}	**-50.5%**	-49.1%	-51.9%

TL-2_d: DECODER[CNN], TL-4: ENCODER[CNN, GNN], DECODER[CNN], and / denotes without.

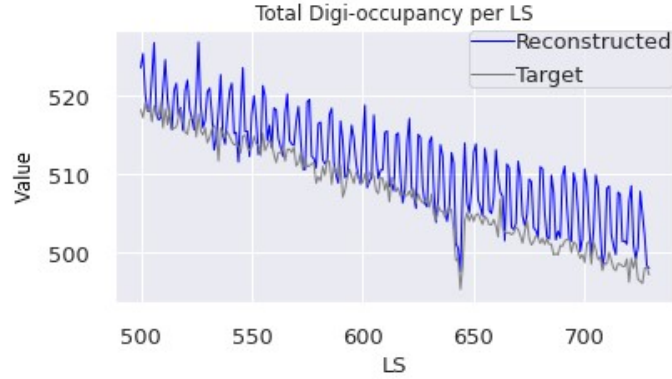## 5.1.2 Transfer Learning on Temporal Learning Networks

We have investigated TL on the temporal RNNs (LSTM layers) on both the encoder and decoder networks together with the spatial learning networks (CNNs and GNNs). We utilize the best performing TL from the above TL on spatial networks investigation, fine-tuning with *train mode* = TL-4 when conducting the TL experiment on temporal networks.

Table 5 presents the $\mathcal{L}_{MSE}$ when TL is applied to the ST networks at $epoch = 200$. We have evaluated models with RNN states preserved only within five maps in a time window, states are reset for each time window. Transferring RNNs has not accomplished performance gain. When the TL involves freezing the RNNs of the decoder for inference on the target data (*train mode*: TL-6) the performance suffered substantially, increasing the test $\mathcal{L}_{MSE}$ by more than 50% despite the reduction of the training $\mathcal{L}_{MSE}$ by 34%. A close investigation has revealed that the issue lies with RNN state resetting during inference. Figure 7 illustrates that the model is struggling to reconstruct the first time step in the time windows (TWs). This is caused by the model's reliance solely on the input map for the first time step reconstruction with reset states, while the states are adjusted and improved for the following time steps (see Figure 7a). This behavior is not entirely unexpected; the RNNs are trained on the source dataset, and employing them directly on the target data set (without taking advantage of the localized temporal information) for the first map in a TW would be challenging. We have further evaluated the models by preserving the RNN states across time windows that leverage the model reconstruction accuracy; it utilizes previous states even for the first maps in the TWs (see Figure 7b).
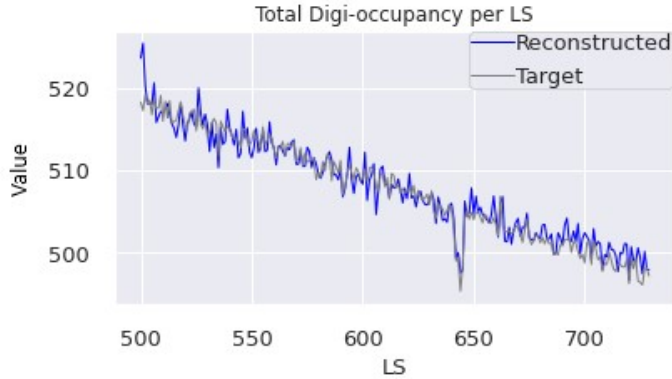
Table 5: ST reconstruction $\mathcal{L}_{MSE}$ performance of TL on ST networks, without RNN states preservation at $epoch = 200$.

Init Mode	Train Mode	MSE Loss			ΔMSE w.r.t $Init$=RANDOM		
		Train	Validation	Test	Train	Validation	Test
Average Performance							
RANDOM	No-TL	2.650×10^{-04}	2.750×10^{-04}	3.361×10^{-04}	–	–	–
TL-4	TL-3	1.775×10^{-04}	$\mathbf{1.775 \times 10^{-04}}$	$\mathbf{2.489 \times 10^{-04}}$	-33.0%	**-35.5%**	**-25.9%**
TL-7	TL-5	1.775×10^{-04}	1.800×10^{-04}	2.496×10^{-04}	-33.0%	-34.5%	-25.7%
TL-7	TL-6	$\mathbf{1.725 \times 10^{-04}}$	4.325×10^{-04}	5.054×10^{-04}	**-34.9%**	57.3%	50.4%
Best Model (on Test Loss)							
RANDOM	No-TL	2.400×10^{-04}	2.600×10^{-04}	3.085×10^{-04}	–	–	–
TL-4	TL-3	1.700×10^{-04}	$\mathbf{1.700 \times 10^{-04}}$	$\mathbf{2.451 \times 10^{-04}}$	-29.2%	**-34.6%**	**-20.5%**
TL-7	TL-5	1.700×10^{-04}	1.800×10^{-04}	2.457×10^{-04}	-29.2%	-30.8%	-20.4%
TL-7	TL-6	$\mathbf{1.500 \times 10^{-04}}$	4.600×10^{-04}	4.697×10^{-04}	**-37.5%**	76.9%	52.2%

TL-3: ENCODER[CNN, GNN], TL-4: ENCODER[CNN, GNN], DECODER[CNN], TL-5: ENCODER[CNN, GNN, RNN]), TL-6: ENCODER[CNN, GNN, RNN], DECODER[RNN], TL-7: ENCODER[CNN, GNN, RNN], DECODER[CNN, RNN].

(a)

(b)

Figure 7: Digi-occupancy map reconstruction on a sample ST data of the test-set. The model was trained using TL-6 (ENCODER[CNN, GNN, RNN], DECODER[RNN]), and the inference was executed: a) without, and b) with LSTM states preservation across time-windows. The autoencoder operates on ST γ maps, but the curves in these plots correspond to the aggregate renormalized γ per LS to illustrate the model's reconstruction performance in handling the fluctuation across lumisections.

Figure 8 illustrates the $\mathcal{L}_{MSE}$ on multiple epochs for the RNN networks with and without RNN states preservation across TWs. The plots illustrate a significant improvement by preserving the states on the frozen decoder RNNs (*train mode* = TL-6); the lower epochs have higher gaps among repeated experiments, and the stabilization gets better at higher epochs. State preservation across TW has a limited impact when the target dataset fine-tunes the decoder RNNs (the *train mode*: TL-5). We have summarized the TL performance in Table 6.

(a)

(b)

Figure 8: Reconstruction performance ($\mathcal{L}_{MSE}$) evaluation of the TL on the ST networks. The TL is applied to train the encoder and decoder with TL-5: (ENCODER: [CNN, GNN, RNN]), and TL-6: (ENCODER: [CNN, GNN, RNN], DECODER: [RNN]). Test MSE loss on a) non-preserved LSTM states that reset for each time window, and b) preserved LSTM states across consecutive time windows. The bars show the dispersion of five repeated experiments.

5.1.3 Applying Learning Rate Scheduling

The models reach saturation after $epoch > 150$ as illustrated in the previous plots in Figure 6a and Figure 8b for the models trained without TL and with TL, respectively. Learning rate (LR) scheduling mechanisms, e.g., lowering the LR when the loss gets flat or fast convergence methods, could improve the performance to mitigate training saturation. We have investigated the impact of scheduling on the TL by training the model with super-convergence *one-cyclic* LR scheduling [55] at $epoch = 200$. The LR scheduling sets the LR according to a one-cycle policy that anneals the LR from an initial LR to some maximum LR ($max\_lr = 0.001$) and then from that maximum LR ($min\_lr = 4 \times 10^{-7}$) to some minimum LR. We utilized a cosine annealing mechanism along with the other settings of the scheduler, provided in Table 7, for our experiment. We kept the default values of the remaining hyperparameters of the scheduler[1].

---

[1]https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.OneCycleLR.html

Table 6: ST reconstruction $\mathcal{L}_{MSE}$ performance of TL on ST networks with RNN state preservation.

Init Mode	Training Mode	MSE Loss		Trainable Parameters Reduction (%)
		Value	Δ(%) w.r.t *Init*=RANDOM	
At epoch = 75				
RANDOM	No-TL	3.826×10^{-04}	–	
TL-4	No-TL	3.180×10^{-04}	-16.9%	0.00%
TL-4	TL-2	2.686×10^{-04}	-29.8%	-2.23%
TL-4	TL-1	3.082×10^{-04}	-19.5%	-0.17%
TL-4	TL-3	2.705×10^{-04}	-29.3%	-2.39%
TL-7	TL-5	2.667×10^{-04}	-30.3%	-8.38%
TL-7	TL-6	$\mathbf{2.577 \times 10^{-04}}$	**-32.6%**	**-97.77%**
At epoch = 200				
RANDOM	No-TL	3.085×10^{-04}	–	–
TL-4	No-TL	2.569×10^{-04}	-16.7%	0.00%
TL-4	TL-2	2.420×10^{-04}	-21.6%	-2.23%
TL-4	TL-1	2.502×10^{-04}	-18.9%	-0.17%
TL-4	TL-3	2.451×10^{-04}	-20.5%	-2.39%
TL-7	TL-5	2.457×10^{-04}	-20.4%	-8.38%
TL-7	TL-6	$\mathbf{2.389 \times 10^{-04}}$	**-22.6%**	**-97.77%**

TL-1: ENCODER[GNN], TL-2: ENCODER[CNN], TL-3: ENCODER[CNN, GNN], TL-4: ENCODER[CNN, GNN], DE-CODER[CNN], TL-5: ENCODER[CNN, GNN, RNN], TL-6: ENCODER[CNN, GNN, RNN], DECODER[RNN], TL-7: EN-CODER[CNN, GNN, RNN], DECODER[CNN, RNN].

Table 7: Hyperparameter setting of one-cyclic LR scheduler.

Hyperparameter	Value	Description
max_lr	0.001	Upper learning rate boundaries in the cycle.
$steps\_per\_epoch$	$train\_data\_size/batch\_size$	The number of steps per epoch to train for.
$total\_steps$	$steps\_per\_epoch \times epochs$	The total number of steps in the cycle.
div_factor	25	Determines the initial learning rate via $initial\_lr = max\_lr/div\_factor$.
$final\_div\_factor$	10^2	Determines the minimum learning rate via $min\_lr = initial\_lr/final\_div\_factor$.

Table 8 shows that the LR scheduling has improved the $\mathcal{L}_{MSE}$ compared to the fixed LR (provided in Table 6) by 19% and 13% for *init mode* = RANDOM, and the *train mode* = TL-6 models, respectively. With respect to the model without TL, the relative improvement from TL is approximately 9% with the LR scheduling, which is lower than the 22.6% achieved with the fixed LR. The results are in accord with Figure 6b, following the projection of closing in the performance difference as the number of epochs increases past $Epoch > 150$ with resolved saturation on the *init mode* = RANDOM. The cyclic LR scheduling method requires more configuration tuning effort to improve the performance depending on the model and dataset compared to fixed LR or other simpler LR scheduling approaches.

Table 8: ST reconstruction $\mathcal{L}_{MSE}$ performance of TL with learning rate scheduling mechanism at $epoch = 200$.

Init Mode	Training Mode	MSE Loss	
		Value	Δ(%) w.r.t *Init*=RANDOM
RANDOM	No-TL	2.500×10^{-04}	–
TL-4	No-TL	2.400×10^{-04}	-4.0%
TL-4	TL-3	2.460×10^{-04}	-1.6%
TL-7	TL-5	$\mathbf{2.283 \times 10^{-04}}$	**-8.7%**
TL-7	TL-6	$\mathbf{2.286 \times 10^{-04}}$	**-8.6%**

TL-3: ENCODER[CNN, GNN], TL-4: ENCODER[CNN, GNN], DECODER[CNN], TL-5: ENCODER[CNN, GNN, RNN], TL-6: ENCODER[CNN, GNN, RNN], DECODER[RNN], TL-7: ENCODER[CNN, GNN, RNN], DECODER[CNN, RNN].

5.2 Anomaly Detection Performance

Machine learning studies performed thus far in the CMS DQM system primarily employed the simulated anomalies data to evaluate the effectiveness of the developed AD models [2, 29]; a small fraction of the DQM data is affected by

real anomalies and is limited to be used for model validation. We have validated the AD models on synthetic anomalies simulating real channel anomalies of the HCAL. We generated synthetic anomalies simulating *dead*, *hot*, and *degraded* channels, and injected them into healthy digi-occupancy maps of the test dataset. We formulate the simulated channel anomalies using:

$$\gamma_a = R_D\gamma_h, \ R_D \neq 1 \tag{10}$$

where γ_a and γ_h are the digi-occupancy of the generated anomaly channel and its corresponding expected healthy reading, respectively. The R_D is the degradation factor, and the simulated anomalies are defined as:

$$\begin{aligned}
&\text{Dead Channel}: \gamma_a = R_D\gamma_h = 0, \ R_D = 0 \\
&\text{Degraded Channel}: 0 < \gamma_a = R_D\gamma_h < \gamma_h, \ 0 < R_D < 1 \\
&\text{Noisy-Hot Channel}: \gamma_h < \gamma_a = R_D\gamma_h \leq \xi, \ R_D > 1 \\
&\text{Fully-Hot Channel}: \gamma_h < \gamma_a = \xi
\end{aligned} \tag{11}$$

The algorithm that generates the synthetic anomaly samples involves three steps: 1) selection of a random set of LSs from the test set, 2) random selection of spatial locations φ for each LS, where $\varphi \in [i\eta \times i\phi \times depth]$ on the HB axes (see Figure 3c), and 3) injection of the simulated anomalies into digi-occupancy maps of the LSs. The simulated anomalies include *dead*, *degraded*, *noisy-hot*, and *fully-hot* channels. We kept the same spatial locations of the generated different anomaly types for consistency. We evaluated the performance on several classification metrics using three anomaly thresholds set to capture 90%, 95%, and 99% of the injected anomalies.

We have evaluated the AD accuracy on 14,000 digi-occupancy maps (2000 maps for each anomaly type) for the dead ($R_D = 0\%$), decaying channel anomalies ($R_D = [80\%, 60\%, 40\%, 20\%]$), noisy hot ($R_D = 200\%$), and fully hot ($\gamma_a = \xi$) channels. We have investigated persisting channel anomalies that affect consecutive maps in a time window. We have processed 70,000 digi-occupancy maps, including five history maps in the time window for each of the 14,000 maps; we generated 1.17% abnormal channels in the 70,000 maps.

We compare the AD performance of models without TL and the best TL from Table 8, i.e., without TL (*init mode* = RANDOM) and with TL (*train mode* = TL-6). Table 9 presents the AD accuracy of the models on the dead, degrading, fully hot, and noisy hot channel abnormalities. Both models perform well in the *area under the receiver operating characteristic curve* (AUC) and *false positive rate* (FPR). The FPR exhibits slight variance between the two models; the TL model significantly improves dead and fully hot channel detection but performs slightly lower for noisy hot channels. Figure 9 illustrates the FPR score across all the anomaly types where the TL model outperforms the without the TL model in $R_D < 1$. The relatively lower precision at 70% for the $R_D = 80\%$ demonstrates that there are still a few anomalies challenging to catch, although the FPR is very low due to the accurate classification of numerous healthy channels (as shown in Figure 10). A channel operating at 80% is mostly an inlier or very close to the healthy operating ranges, and detecting such anomaly becomes even more difficult when the expected γ of the channel is very low. The FPR has significantly improved by 80% when the amount of the captured anomaly is reduced to 95%. Figures 11 and 12 demonstrate anomaly localization capability on a sample injected channel anomaly with the different anomaly types; Figures 11 and 12 demonstrate the ability of our algorithm to locate the anomalies on a sample that has been injected with different anomaly types in some channels ($4 < i\eta < 11$ and $11 < i\phi < 19$); the TL model accomplishes better localization on the fully hot channels with less dispersion in its anomaly score values.

Figure 13 portrays the distribution of the reconstruction error ($e_{i,MAE}$) and provides further illustration of the overlap region between the healthy and faulty channels at the different degradation rates. We have observed a slight increase in the reconstruction error of the healthy channels as the anomaly strength increases for the abnormal channels (the R_D is farther away from 100%); it is more pronounced when introducing hot channel anomalies with $R_D = 200\%$. Close investigation reveals that the healthy channels have higher anomaly scores (filtered out from the anomaly injection $\gamma_a = R_D\gamma_h > \xi$) due to their proximity to the abnormal channels (as depicted in Figure 14). Since channels belonging to the same RBX are positioned in proximity in the HB segmentation and share characteristics, the GRAPHSTAD autoencoder exploits the correlation for spatial data reconstruction. The exacerbating effect in the presence of the hot channels indicates the model's ability to focus on the channel with higher values to perform the map reconstruction.

The degrading and dead channels are another major difference between the without and with TL models. The tails of overlaps decrease on reconstruction error distribution of the normal and abnormal channels, the degrading channel as the R_D decreases, in both models (as shown in Figure 14). Table 9 shows the AD performance slightly deteriorates for the dead channels ($R_D = 0\%$) compared to the degraded channel at $R_D = 20\%$ defying expectation. Figure 14 illustrates the reconstruction error of most anomalies increases, enabling enhanced separation between the normal and anomaly channels except for a few dead channels that increase the FPR. The reconstruction error of the *without-TL* model drops to zero for dead channels although the channels have higher reconstruction errors at $R_D = 20\%$ (see Figure 13a); this is caused by the presence of real dead channels in the training dataset at the location of $[i\eta \in [-16, -15, -13], i\phi = 8, depth = 1]$ (see Figure 15). Figure 15 depicts that the *without-TL* model reconstructs the real dead channels as normal

Table 9: AD performance of TL on time-persistent abnormal channels.

Anomaly Type	FPR (90%)	FPR (95%)	FPR (99%)	AUC
Without-TL Model: (*Init*=**RANDOM**,*Training*=No-TL)				
Degraded Channel ($R_D = 80\%, \gamma_a = R_D\gamma_h$)	6.281×10^{-04}	1.519×10^{-03}	8.741×10^{-03}	0.993
Degraded Channel ($R_D = 60\%, \gamma_a = R_D\gamma_h$)	5.991×10^{-05}	1.438×10^{-04}	8.242×10^{-04}	**1.000**
Degraded Channel ($R_D = 40\%, \gamma_a = R_D\gamma_h$)	4.881×10^{-05}	5.628×10^{-05}	1.466×10^{-04}	**1.000**
Degraded Channel ($R_D = 20\%, \gamma_a = R_D\gamma_h$)	5.870×10^{-05}	6.273×10^{-05}	7.342×10^{-05}	**1.000**
Dead Channel ($R_D = 0\%, \gamma_a = 0.0$)	6.636×10^{-05}	7.080×10^{-05}	8.331×10^{-05}	**1.000**
Fully-Hot Channel ($\gamma_a = \xi, \gamma_a \neq \gamma_h$)	1.220×10^{-04}	1.317×10^{-04}	1.606×10^{-04}	**1.000**
Noisy-Hot Channel ($R_D = 200\%, \gamma_a = R_D\gamma_h$)	$\mathbf{3.300 \times 10^{-04}}$	$\mathbf{5.732 \times 10^{-04}}$	$\mathbf{1.765 \times 10^{-03}}$	**1.000**
With-TL Model: (*Init*=TL-6, *Training*=TL-6)				
Degraded Channel ($R_D = 80\%, \gamma_a = R_D\gamma_h$)	$\mathbf{5.019 \times 10^{-04}}$	$\mathbf{1.320 \times 10^{-03}}$	$\mathbf{6.527 \times 10^{-03}}$	0.996
Degraded Channel ($R_D = 60\%, \gamma_a = R_D\gamma_h$)	$\mathbf{2.118 \times 10^{-05}}$	$\mathbf{9.642 \times 10^{-05}}$	$\mathbf{8.141 \times 10^{-04}}$	**1.000**
Degraded Channel ($R_D = 40\%, \gamma_a = R_D\gamma_h$)	$\mathbf{1.614 \times 10^{-06}}$	$\mathbf{4.034 \times 10^{-06}}$	$\mathbf{7.161 \times 10^{-05}}$	**1.000**
Degraded Channel ($R_D = 20\%, \gamma_a = R_D\gamma_h$)	$\mathbf{1.614 \times 10^{-06}}$	$\mathbf{3.833 \times 10^{-06}}$	$\mathbf{8.472 \times 10^{-06}}$	**1.000**
Dead Channel ($R_D = 0\%, \gamma_a = 0.0$)	$\mathbf{1.815 \times 10^{-06}}$	$\mathbf{4.236 \times 10^{-06}}$	$\mathbf{8.472 \times 10^{-06}}$	**1.000**
Fully-Hot Channel ($\gamma_a = \xi, \gamma_a \neq \gamma_h$)	**0.000**	$\mathbf{6.051 \times 10^{-07}}$	$\mathbf{3.631 \times 10^{-05}}$	**1.000**
Noisy-Hot Channel ($R_D = 200\%, \gamma_a = R_D\gamma_h$)	1.380×10^{-03}	2.143×10^{-03}	4.099×10^{-03}	**1.000**

Figure 9: AD FPR on *dead*, *degraded* and *noisy-hot* channel. The anomaly flags are generated using thresholds that capture 90%, 95%, and 99% of the anomaly channels.

Figure 10: AD *Precision* on *dead*, *degraded* and *noisy-hot* channels. The anomaly flags are generated using thresholds that capture 90%, 95%, and 99% of the anomaly channels.

with low reconstruction error, whereas the *with-TL* provides a high error which signifies it is detecting the channels as anomalies. When the simulated dead anomalies are injected into the digi-occupancy maps near the real abnormal channels, the *without-TL* model reconstructs them as normal characteristics in which the model fails to detect the anomalies. The *with-TL* has thus achieved substantially better detection on the dead channels (see Figure 13b). The results demonstrate transfer learning robustness when a semi-supervised model's training dataset is contaminated with real anomalies.

6 Conclusion

We have presented transfer learning on a spatio-temporal semi-supervised anomaly detection model. We have discussed an anomaly detection model designed for Hadron Calorimeter channel monitoring using digi-occupancy maps of the Data Quality Monitoring system. We have successfully transferred the spatio-temporal anomaly detection model, employing convolutional, graph, and recurrent neural networks in an autoencoder architecture, from the source HCAL Endcap to the HCAL Barrel subdetector. The study has provided insights into several transfer learning scenarios at the initialization and training phases. We have successfully applied transfer learning to the encoder feature extraction networks and inner networks of the decoder. The transfer learning has achieved a promising spatio-temporal reconstruction and anomaly detection performance while proving a substantial reduction in trainable network parameters and enhancing robustness against contamination in the training data sets. The study indicates that transfer learning can facilitate machine learning development with limited cleaned training data and when expensive model training on large data sets is costly or time-consuming. The choice of model training settings, such as the number of iterations, learning rate schedule, and state preservation for recurrent neural networks during inference, can impact the transfer learning performance, in addition to the similarity between the source and target data sets.

Acknowledgments

Figure 11: Spatial reconstruction error ($e_{i,MAE}$) maps on a sample digi-occupancy map (at *depth* = 1) with degraded anomaly types: a) renormalized digi-occupancy map with simulated anomaly channels; the reconstruction error maps of the b) *without-TL* model and c) *with-TL* model. The anomaly region is localized well with proportional strength to the severity of the anomaly in both models.
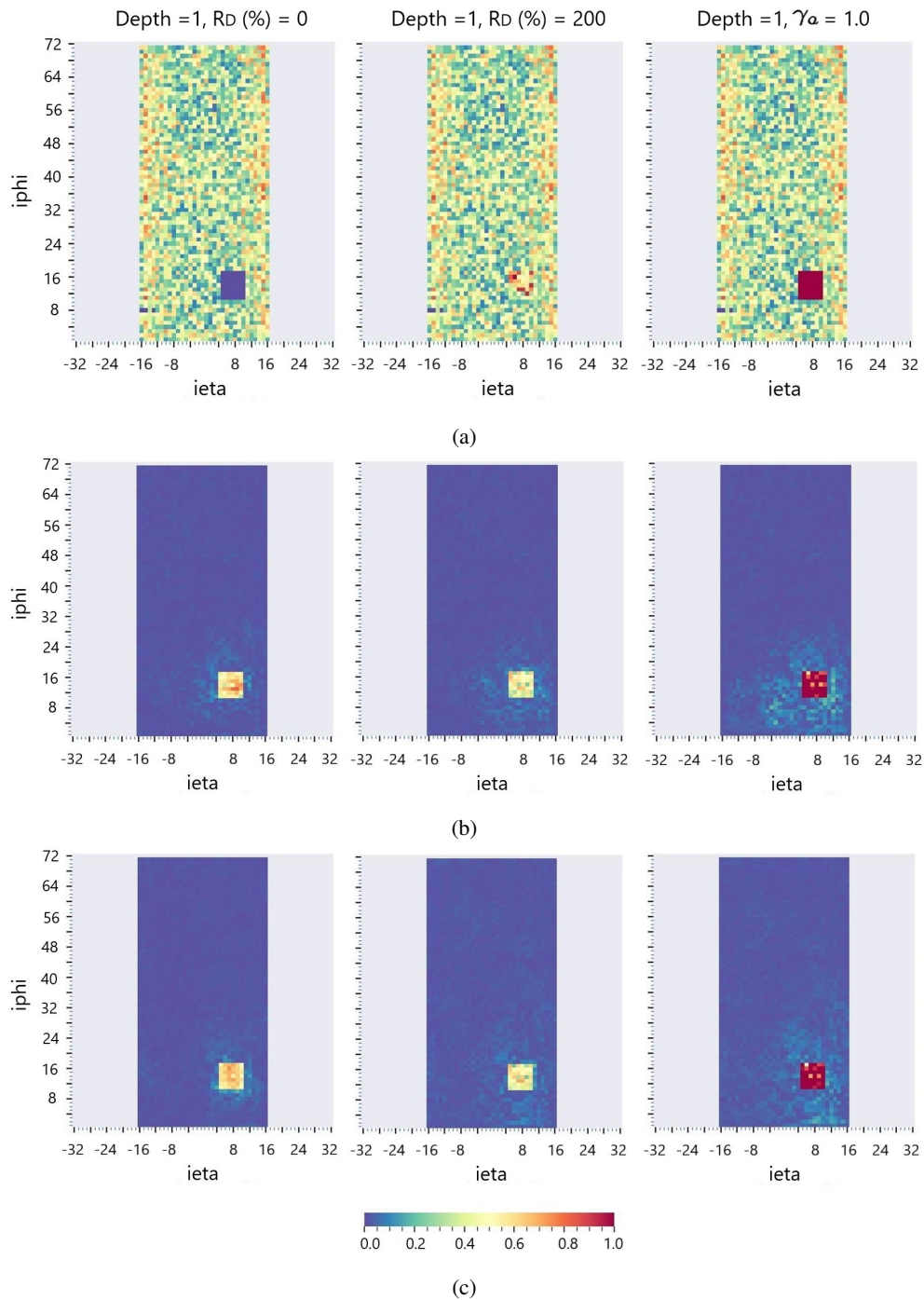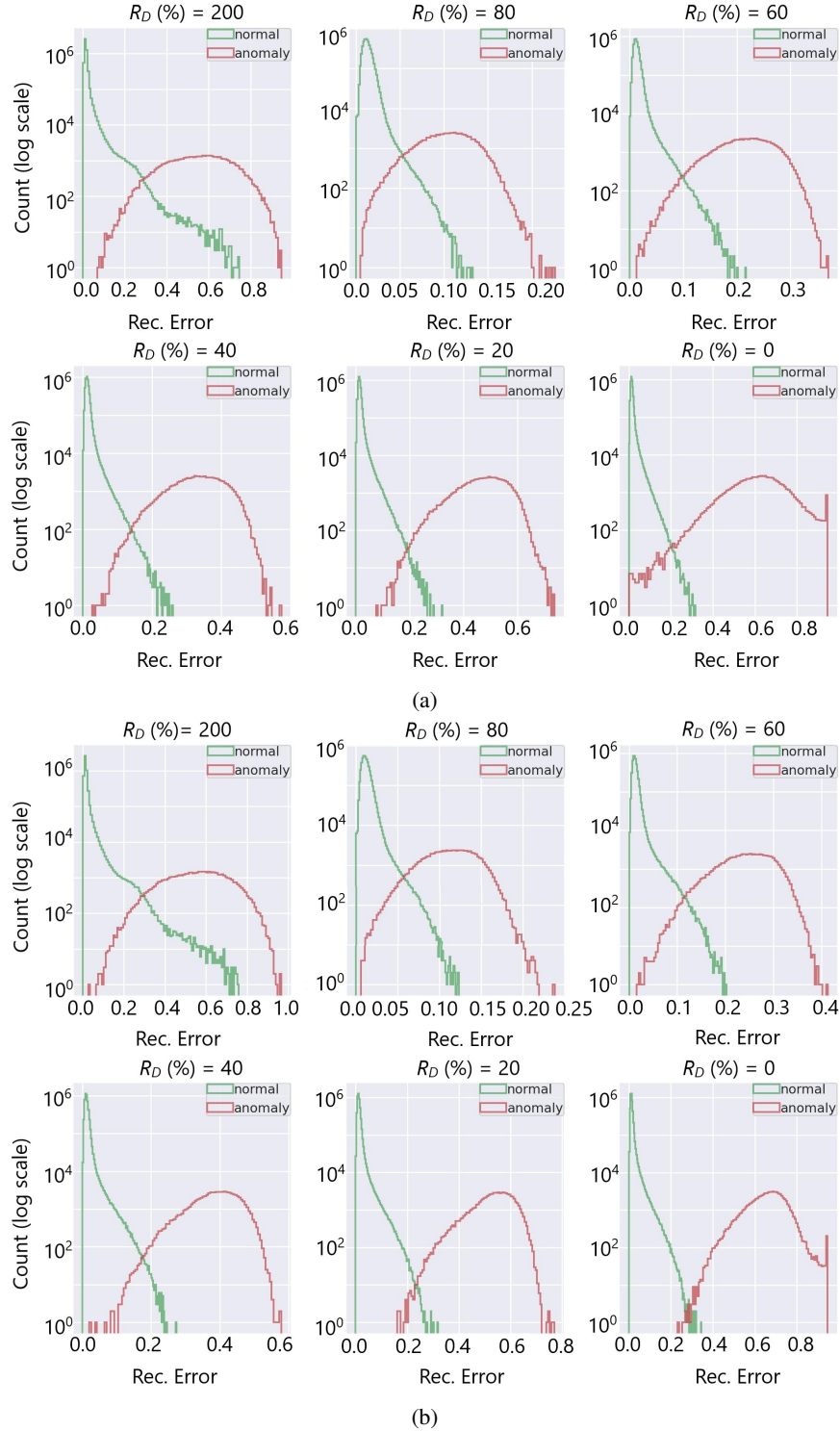
(a)

(b)

(c)

Figure 12: Spatial reconstruction error ($e_{i,MAE}$) maps on a sample digi-occupancy map (at *depth* $= 1$) with dead, noisy hot, and fully hot anomaly types: a) renormalized digi-occupancy map with simulated anomaly channels; the reconstruction error maps of the b) *without-TL* model and c) *with-TL* model.

(a)

(b)

Figure 13: Reconstruction error ($e_{i,MAE}$) distribution of healthy and anomalous channels at different channel degradation rates, excluding the real anomalies. The models are a) *without-TL* and b) *with-TL* models. The overlap region decreases substantially as the channel deterioration increases for $R_D < 100\%$.

Figure 14: Location embedding for channels with high reconstruction error, from the *with-TL* model, in the presence of noisy hot channel anomalies. We applied t-SNE embedding [56] on spatio-temporal locations (LS, $i\eta$, $i\phi$, and *depth*) to generate the two-dimensional representation. Normal channels with high reconstruction error occur in proximity to the anomalous channels.

Figure 15: Average reconstruction error ($e_{i,MAE}$) per channel at *depth* $= 1$ over the training dataset for the a) *without-TL* model, and b) *with-TL* model. The real dead HB channels (at $[i\eta = [-16, -15, -13], i\phi = 8, depth = 1]$) are highlighted in the red boxes. *without-TL* model reconstructs the real dead channels as normal with low rec error, whereas *with-TL* produces a high error which signifies detecting the channels as anomalies.

References

[1] G. Atluri, A. Karpatne, and V. Kumar, "Spatio-temporal data mining: a survey of problems and methods," *ACM Computing Surveys*, vol. 51, no. 4, pp. 1–41, 2018.

[2] M. W. Asres, C. W. Omlin, L. Wang, D. Yu, P. Parygin, J. Dittmann, G. Karapostoli, M. Seidel, R. Venditti, L. Lambrecht *et al.*, "Spatio-temporal anomaly detection with graph networks for data quality monitoring of the Hadron Calorimeter," *Sensors*, vol. 23, no. 24, p. 9679, 2023.

[3] Y. Chang, Z. Tu, W. Xie, B. Luo, S. Zhang, H. Sui, and J. Yuan, "Video anomaly detection with spatio-temporal dissociation," *Pattern Recognition*, vol. 122, p. 108213, 2022.

[4] L. Deng, D. Lian, Z. Huang, and E. Chen, "Graph convolutional adversarial networks for spatiotemporal anomaly detection," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2416–2428, 2022.

[5] L. Tišljarić, S. Fernandes, T. Carić, and J. Gama, "Spatiotemporal road traffic anomaly detection: a tensor-based approach," *Applied Sciences*, vol. 11, no. 24, p. 12017, 2021.

[6] D. Ahmedt-Aristizabal, M. A. Armin, S. Denman, C. Fookes, and L. Petersson, "Graph-based deep learning for medical diagnosis and analysis: past, present and future," *Sensors*, vol. 21, no. 14, p. 4758, 2021.

[7] G. Zhang, W. Zheng, W. Yin, and W. Lei, "Improving the the resolution and accuracy of groundwater level anomalies using the machine learning-based fusion model in the north China plain," *Sensors*, vol. 21, no. 1, p. 46, 2020.

[8] R. Chalapathy and S. Chawla, "Deep learning for anomaly detection: a survey," *arXiv:1901.03407*, 2019.

[9] Y. Zhao, L. Deng, X. Chen, C. Guo, B. Yang, T. Kieu, F. Huang, T. B. Pedersen, K. Zheng, and C. S. Jensen, "A comparative study on unsupervised anomaly detection for time series: experiments and analysis," *arXiv preprint arXiv:2209.04635*, 2022.

[10] R. Wang, K. Nie, T. Wang, Y. Yang, and B. Long, "Deep learning for anomaly detection," in *Proceedings of the 13th International Conference on Web Search and Data Mining*, 2020, pp. 894–896.

[11] A. A. Cook, G. Mısırlı, and Z. Fan, "Anomaly detection for IoT time-series data: a survey," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6481–6494, 2019.

[12] S. Wang, H. Miao, J. Li, and J. Cao, "Spatio-temporal knowledge transfer for urban crowd flow prediction via deep attentive adaptation networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 5, pp. 4695–4705, 2021.

[13] S. Shao, S. McAleer, R. Yan, and P. Baldi, "Highly accurate machine fault diagnosis using deep transfer learning," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2446–2455, 2018.

[14] N. Laptev, J. Yu, and R. Rajagopal, "Reconstruction and regression loss for time-series transfer learning," in *Proceedings of the Special Interest Group on SIGKDD Knowledge Discovery and Data Mining*, vol. 20, 2018.

[15] P. Gupta, P. Malhotra, L. Vig, and G. Shroff, "Transfer learning for clinical time series analysis using recurrent neural networks," *arXiv preprint arXiv:1807.01705*, 2018.

[16] N. Boullé, V. Dallas, Y. Nakatsukasa, and D. Samaddar, "Classification of chaotic time series with deep learning," *Physica D: Nonlinear Phenomena*, vol. 403, p. 132261, 2020.

[17] L. Wang, X. Geng, X. Ma, F. Liu, and Q. Yang, "Cross-city transfer learning for deep spatio-temporal prediction," *arXiv preprint arXiv:1802.00386*, 2018.

[18] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2014.

[19] S. Niu, Y. Liu, J. Wang, and H. Song, "A decade survey of transfer learning (2010–2020)," *IEEE Transactions on Artificial Intelligence*, vol. 1, no. 2, pp. 151–166, 2020.

[20] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805*, 2018.

[22] L. Evans and P. Bryant, "LHC machine," *Journal of instrumentation*, vol. 3, no. 08, p. S08001, 2008.

[23] The CMS Collaboration, "Development of the CMS detector for the CERN LHC Run 3," *arXiv preprint arXiv:2309.05466*, 2023.

[24] The CMS Collaboration, S. Chatrchyan, G. Hmayakyan, V. Khachatryan, A. Sirunyan, W. Adam, T. Bauer, T. Bergauer, H. Bergauer, M. Dragicevic *et al.*, "The CMS experiment at the CERN LHC," *The CMS Experiment at the LHC CERN Document Server*, vol. 3, p. S08004, 2008.

[25] V. Azzolini, D. Bugelskis, T. Hreus, K. Maeshima, M. J. Fernandez, A. Norkus, P. J. Fraser, M. Rovere, M. A. Schneider *et al.*, "The data quality monitoring software for the CMS experiment at the LHC: past, present and future," in *European Physical Journal Web of Conferences*, vol. 214, 2019, p. 02003.

[26] L. Tuura, A. Meyer, I. Segoni, and G. Della Ricca, "CMS data quality monitoring: systems and experiences," in *Journal of Physics: Conference Series*, vol. 219, no. 7. IOP Publishing, 2010, p. 072020.

[27] F. De Guio and The CMS Collaboration, "The CMS data quality monitoring software: experience and future prospects," in *Journal of Physics: Conference Series*, vol. 513, no. 3. IOP Publishing, 2014, p. 032024.

[28] The CMS-ECAL Collaboration *et al.*, "Autoencoder-based anomaly detection system for online data quality monitoring of the CMS electromagnetic calorimeter," *arXiv preprint arXiv:2309.10157*, 2023.

[29] V. Azzolin, M. Andrews, G. Cerminara, N. Dev, C. Jessop, N. Marinelli, T. Mudholkar, M. Pierini, A. Pol, and J.-R. Vlimant, "Improving data quality monitoring via a partnership of technologies and resources between the CMS experiment at CERN and industry," in *European Physical Journal Web of Conferences*, vol. 214. EDP Sciences, 2019, p. 01007.

[30] A. A. Pol, V. Azzolini, G. Cerminara, F. De Guio, G. Franzoni, M. Pierini, F. Sirokỳ, and J.-R. Vlimant, "Anomaly detection using deep autoencoders for the assessment of the quality of the data acquired by the CMS experiment," in *European Physical Journal Web of Conferences*, vol. 214. EDP Sciences, 2019, p. 06008.

[31] A. A. Pol, G. Cerminara, C. Germain, M. Pierini, and A. Seth, "Detector monitoring with artificial neural networks at the CMS experiment at the CERN Large Hadron Collider," *Computing and Software for Big Science*, vol. 3, no. 1, p. 3, 2019.

[32] P. Xie, T. Li, J. Liu, S. Du, X. Yang, and J. Zhang, "Urban flow prediction from spatiotemporal data using machine learning: a survey," *Information Fusion*, vol. 59, pp. 1–12, 2020.

[33] N. Strobbe, "The upgrade of the CMS Hadron Calorimeter with Silicon photomultipliers," *Journal of Instrumentation*, vol. 12, no. 1, p. C01080, 2017.

[34] P. Xiong, Y. Zhu, Z. Sun, Z. Cao, M. Wang, Y. Zheng, J. Hou, T. Huang, and Z. Que, "Application of transfer learning in continuous time series for anomaly detection in commercial aircraft flight data," in *International Conference on Smart Cloud*. IEEE, 2018, pp. 13–18.

[35] T. Wen and R. Keyes, "Time series anomaly detection using convolutional neural networks and transfer learning," *arXiv preprint arXiv:1905.13628*, 2019.

[36] B. Guo, J. Li, V. W. Zheng, Z. Wang, and Z. Yu, "CityTransfer: transferring inter-and intra-city knowledge for chain store site recommendation based on multi-source urban data," *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 4, pp. 1–23, 2018.

[37] M. Wang and W. Deng, "Deep visual domain adaptation: a survey," *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[38] P. Wu, J. Liu, M. Li, Y. Sun, and F. Shen, "Fast sparse coding networks for anomaly detection in videos," *Pattern Recognition*, vol. 107, p. 107515, 2020.

[39] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proceedings of Computer Vision and Pattern Recognition*. IEEE, 2016, pp. 733–742.

[40] W. Luo, W. Liu, D. Lian, J. Tang, L. Duan, X. Peng, and S. Gao, "Video anomaly detection with sparse coding inspired deep neural networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 3, pp. 1070–1084, 2019.

[41] D. Hsu, "Anomaly detection on graph time series," *arXiv preprint arXiv:1708.02975*, 2017.

[42] E. Focardi, "Status of the CMS detector," *Physics Procedia*, vol. 37, pp. 119–127, 2012.

[43] I. Neutelings, "CMS coordinate system," 2023, accessed on 14/12/2023. [Online]. Available: https://tikz.net/axis3d_cms

[44] H. W. Cheung, The CMS collaboration *et al.*, "CMS: Present status, limitations, and upgrade plans," *Physics Procedia*, vol. 37, pp. 128–137, 2012.

[45] T. Virdee and The CMS Collaboration, "The CMS experiment at the CERN LHC," in *6th International Symposium on Particles, Strings and Cosmology*. World Scientific, 1999.

[46] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[47] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[48] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[49] M. W. Asres, G. Cummings, P. Parygin, A. Khukhunaishvili, M. Toms, A. Campbell, S. I. Cooper, D. Yu, J. Dittmann, and C. W. Omlin, "Unsupervised deep variational model for multivariate sensor anomaly detection," in *International Conference on Progress in Informatics and Computing*. IEEE, 2021, pp. 364–371.

[50] Y. Guo, W. Liao, Q. Wang, L. Yu, T. Ji, and P. Li, "Multidimensional time series anomaly detection: A GRU-based Gaussian mixture variational autoencoder approach," in *Asian Conference on Machine Learning*. Proceedings of Machine Learning Research, 2018, pp. 97–112.

[51] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture on IE*, vol. 2, no. 1, pp. 1–18, 2015.

[52] G. S. Chadha, A. Rabbani, and A. Schwung, "Comparison of semi-supervised deep neural networks for anomaly detection in industrial processes," in *17th International Conference on Industrial Informatics*, vol. 1. IEEE, 2019, pp. 214–219.

[53] T. Van Laarhoven, "L2 regularization versus batch and weight normalization," *arXiv preprint arXiv:1706.05350*, 2017.

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

[55] L. N. Smith and N. Topin, "Super-convergence: very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. SPIE, 2019, pp. 369–386.

[56] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research*, vol. 9, no. 11, 2008.

The CMS-HCAL Collaboration

A. Gevorgyan[1], A. Petrosyan[1], A. Tumasyan[1], G.A. Alves[2], C. Hensel[2], W.L. Aldá Júnior[3], W. Carvalho[3], J. Chinellato[3,f], C. De Oliveira Martins[3], D. Matos Figueiredo[3], C. Mora Herrera[3], H. Nogima[3], W.L. Prado Da Silva[3], E.J. Tonelli Manganote[3], A. Vilela Pereira[3], M. Finger[4], M. Finger Jr.[4], G. Adamov[5], Z. Tsamalaidze[5,g], K. Borras[6,y], A. Campbell[6], F. Engelke[6,y], D. Krücker[6], I. Martens[6], L. Wiens[6,y], M. Csanád[7], A. Feherkuti[7], S. Lökös[7,v], G. Pásztor[7], O. Surányi[7], G.I. Veres[7], B. Kansal[8], S. Sharma[8], S.B. Beri[9], B. Bhawandeep[9], R. Chawla[9], A. Kalsi[9], A. Kaur[9], M. Kaur[9], G. Walia[9], S. Bhattacharya[10], S. Ghosh[10], S. Nandan[10], A. Purohit[10], M. Sharan[10], S. Banerjee[11], S. Bhattacharya[11], S. Chatterjee[11], P. Das[11], M. Guchait[11], S. Jain[11], S. Kumar[11], M. Maity[11], G. Majumder[11], K. Mazumdar[11], M. Patil[11], T. Sarkar[11] S. Sekmen[12,y], A. Juodagalvis[13], D. Agyel[14], F. Boran[14], S. Damarseckin[14], Z.S. Demiroglu[14], F. Dölek[14], I. Dumanoglu[14,ee], E. Eskut[14], G. Gokbulut[14], Y. Guler[14,ff], E. Gurpinar Guler[14,ff], C. Işik[14], E.E. Kangal[14], O. Kara[14], A. Kayis Topaksu[14], U. Kiminsu[14], G. Onengut[14], K. Ozdemir[14,gg], E. Pinar[14], A. Polatoz[14], A.E. Simsek[14], B. Tali[14,hh], U.G. Tok[14], S. Turkcapar[14], E. Uslan[14], I.S. Zorbakir[14], B. Bilin[15,y], G. Karapinar[15,ii], A. Murat Guler[15], K. Ocalan[15,jj], M. Yalvac[15,kk], M. Zeyrek[15], B. Akgun[16], I.O. Atakisi[16,ll], E. Gülmez[16], M. Kaya[16,ll], O. Kaya[16,mm], S. Tekten[16,nn], E.A. Yetkin[16,dd], T. Yetkin[16,qq], A. Cakir[17], K. Cankocak[17,ee], S. Sen[17,oo], O. Aydilek[18], S. Cerci[18,hh], B. Hacisahinoglu[18], I. Hos[18,pp], B. Isildak[18,qq], B. Kaynak[18], S. Ozkorucuklu[18], O. Potok[18], H. Sert[18], C. Simsek[18], D. Sunar Cerci[18,hh], C. Zorbilmez[18], A. Boyarintsev[19], B. Grynyov[19], L. Levchuk[20], V. Popov[20], P. Sorokin[20], H. Flacher[21], S. Abdullin[22], B. Caraway[22], J. Dittmann[22], K. Hatakeyama[22], A.R. Kanuganti[22], B. McMaster[22], M. Saunders[22], J. Wilson[22], A. Buccilli[23,q], P. Bunin[23,z], S.I. Cooper[23], C. Henderson[23,l], C.U. Perez[23], P. Rumerio[23,t], C. Cosby[24], Z. Demiragli[24], D. Gastler[24], E. Hazen[24], J. Rohlf[24], M. Hadley[25], U. Heintz[25], T. Kwon[25], E. Laird[25], G. Landsberg[25], K.T. Lau[25], X. Yan[25], D. Yu[25,cc], Z. Mao[25], J.W. Gary[26], G. Karapostoli[26,bb], O.R. Long[26], R. Bhandari[27], R. Heller[27], D. Stuart[27], J. Yoo[27,j], Y. Chen[28,n], J. Duarte[28], J.M. Lawhorn[28], M. Spiropulu[28], A. Apresyan[29], A. Apyan[29,c], S. Banerjee[29,d], F. Chlebana[29], Y. Feng[29] J. Freeman[29], D. Green[29], K.H.M. Kwok[29], J. Hirschauer[29], U. Joshi[29], D. Lincoln[29], S. Los[29], C. Madrid[29], N. Pastika[29], K. Pedro[29], W.J. Spalding[29], S. Tkaczyk[29], S. Linn[30], P. Markowitz[30], V. Hagopian[31], T. Kolberg[31], G. Martinez[31], O. Viazlo[31], M. Hohlmann[32], R. Kumar Verma[32], D. Noonan[32], F. Yumiceva[32,e], M. Alhusseini[33], B. Bilki[33], D. Blend[33], K. Dilsiz[33,rr], L. Emediato[33], R.P. Gandrajula[33], M. Herrmann[33], O.K. Köseyan[33], J.-P. Merlo[33], A. Mestvirishvili[33,aa], M. Miller[33], H. Ogul[33,ss], Y. Onel[33], A. Penzo[33], D. Southwick[33], E. Tiras[33,tt], J. Wetzel[33], A. Al-bataineh[34,s], J. Bowen[34,o], C. Le Mahieu[34], J. Marquez[34], W. McBrayer[34], M. Murray[34], M. Nickel[34], S. Popescu[34,r], C. Smith[34], Q. Wang[34], K. Kaadze[35], D. Kim[35], Y. Maravin[35], A. Mohammadi[35,d], J. Natoli[35], D. Roy[35], L.K. Saini[35,f], E. Adams[36], A. Baden[36], O. Baron[36], A. Belloni[36], A. Bethani[36], Y-M Chen[36], S.C. Eno[36], C. Ferraioli[36,i], T. Grassi[36], N.J. Hadley[36], R.G. Kellogg[36], T. Koeth[36], Y. Lai[36], S. Lascio[36], A.C. Mignerey[36], S. Nabili[36], C. Palmer[36], C. Papageorgakis[36], M. Seidel[36,u], L. Wang[36], K. Wong[36], M. D'Alfonso[37], M. Hu[37], B. Crossman[38], J. Hiltbrand[38], M. Krohn[38], J. Mans[38], M. Revering[38], N. Strobbe[38], A. Heering[39], Y. Musienko[39,z], R. Ruchti[39], M. Wayne[39], W. Chung[40], G. Kopp[40], K. Mei[40], C. Tully[40], A. Bodek[41], P. de Barbaro[41], C. Fallon[41], M. Galanti[41], A. Garcia-Bellido[41], A. Khukhunaishvili[41], C-L Tan[41], R. Taus[41], D. Vishnevskiy[41], M. Zielinski[41], B. Chiarito[42], J.P. Chou[42], S.A. Thayil[42], H. Wang[42], N. Akchurin[43], J. Damgov[43], F. De Guio[43,w], S. Kunori[43], K. Lamichhane[43], S.W. Lee[43], T. Mengke[43], S. Muthumuni[43], S. Undleeb[43], I. Volobouev[43], Z. Wang[43], A. Whitbeck[43], G. Cummings[44], S. Goadhouse[44], J. Hakala[44], R. Hirosky[44], D. Winn[45], V. Alexakhin[46], V. Andreev[46], Y. Andreev[46], M. Azarkin[46], A. Belyaev[46], S. Bitioukov[46], E. Boos[46], O. Bychkova[46], M. Chadeeva[46], V. Chekhovsky[46], R. Chistov[46], M. Danilov[46], A. Demianov[46], A. Dermenev[46], M. Dubinin[46,k], L. Dudko[46], D. Elumakhov[46], V. Epshteyn[46], Y. Ershov[46], A. Ershov[46], V. Gavrilov[46], I. Golutvin[46,a†], A. Gribushin[46], A. Kalinin[46,m], A. Kaminskiy[46], A. Karneyeu[46], L. Khein[46], M. Kirakosyan[46], V. Klyukhin[46], O. Kodolova[46,b], V. Krychkine[46], A. Kurenkov[46], A. Litomin[46], N. Lychkovskaya[46], V. Makarenko[46], P. Mandrik[46], P. Moisenz[46,a†], S. Obraztsov[46], A. Oskin[46], P. Parygin[46,x], V. Petrov[46], S. Petrushanko[46], S. Polikarpov[46], E. Popova[46,x], V. Rusinov[46], R. Ryutin[46], V. Savrin[46], D. Selivanova[46], V. Smirnov[46], A. Snigirev[46], A. Sobol[46], A. Stepennov[46,p], E. Tarkovskii[46], A. Terkulov[46], D. Tlisov[46,a†], I. Tlisova[46], R. Tolochek[46], M. Toms[46,h], A. Toropin[46], S. Troshin[46], A. Volkov[46], B. Yuldashev[46], A. Zarubin[46], A. Zhokin[46]

[1]Yerevan Physics Institute, Yerevan, Armenia
[2]Centro Brasileiro de Pesquisas Fisicas, Rio de Janeiro, Brazil
[3]Universidade do Estado do Rio de Janeiro, Rio de Janeiro, Brazil
[4]Charles University, Prague, Czech Republic
[5]Georgian Technical University, Tbilisi, Georgia
[6]Deutsches Elektronen-Synchrotron, Hamburg, Germany
[7]MTA-ELTE Lendület CMS Particle and Nuclear Physics Group, Eötvös Loránd University, Budapest, Hungary
[8]Indian Institute of Science Education and Research (IISER), Pune, India
[9]Panjab University, Chandigarh, India
[10]Saha Institute of Nuclear Physics, HBNI, Kolkata, India
[11]Tata Institute of Fundamental Research-B, Mumbai, India
[12]Kyungpook National University, Daegu, Korea
[13]Vilnius University, Vilnius, Lithuania
[14]Çukurova University, Physics Department, Science and Art Faculty, Adana, Turkey
[15]Middle East Technical University, Physics Department, Ankara, Turkey
[16]Bogazici University, Istanbul, Turkey

[17]Istanbul Technical University, Istanbul, Turkey

[18]Istanbul University, Istanbul, Turkey

[19]Institute for Scintillation Materials of National Academy of Science of Ukraine, Kharkiv, Ukraine

[20]National Science Centre, Kharkiv Institute of Physics and Technology, Kharkiv, Ukraine

[21]University of Bristol, Bristol, United Kingdom

[22]Baylor University, Waco, Texas, USA

[23]The University of Alabama, Tuscaloosa, Alabama, USA

[24]Boston University, Boston, Massachusetts, USA

[25]Brown University, Providence, Rhode Island, USA

[26]University of California, Riverside, Riverside, California, USA

[27]University of California, Santa Barbara - Department of Physics, Santa Barbara, California, USA

[28]California Institute of Technology, Pasadena, California, USA

[29]Fermi National Accelerator Laboratory, Batavia, Illinois, USA

[30]Florida International University, Miami, USA

[31]Florida State University, Tallahassee, Florida, USA

[32]Florida Institute of Technology, Melbourne, Florida, USA

[33]The University of Iowa, Iowa City, Iowa, USA

[34]The University of Kansas, Lawrence, Kansas, USA

[35]Kansas State University, Manhattan, Kansas, USA

[36]University of Maryland, College Park, Maryland, USA

[37]Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

[38]University of Minnesota, Minneapolis, Minnesota, USA

[39]University of Notre Dame, Notre Dame, Indiana, USA

[40]Princeton University, Princeton, New Jersey, USA

[41]University of Rochester, Rochester, New York, USA

[42]Rutgers, The State University of New Jersey, Piscataway, New Jersey, USA

[43]Texas Tech University, Lubbock, Texas, USA

[44]University of Virginia, Charlottesville, Virginia, USA

[45]Fairfield University, Fairfield, USA

[46]Authors affiliated with an institute or an international laboratory covered by a cooperation agreement with CERN.

[a]†Deceased

[b]Also at Yerevan State University, Yerevan, Armenia

[c]Now at Brandeis University, Waltham, USA

[d]Now at University of Wisconsin-Madison, Madison, USA

[e]Now at Northrop Grumman, Linthicum Heights, USA

[f]Now at Gallagher Basset, Schaumburg, USA

[g]Also at Tbilisi State University, Tbilisi, Georgia

[h]Now at Karlsruhe Institute of Technology, Karlsruhe, Germany

[i]Now at Windfall Data, Novato, USA

[j]Now at Korea University, Seoul, Korea

[k]Also at California Institute of Technology, Pasadena, California, USA

[l]Now at University of Cincinnati, Cincinnati, USA

[m]Now at University of Maryland, College Park, Maryland, USA

[n]Now a Massachusetts Institute of Technology, Cambridge, USA

[o]Now at Baker University, Baldwin City, USA

[p]Now at University of Cyprus , Cyprus

[q]Now at Bond, San Francisco, USA

[r]Also at IFIN-HH, Bucharest, Romania

[s]Now at Yarmouk University, Irbid, Jordan

[t]Also at Università di Torino, Torino, Italy

[u]Now at Riga Technical University, Riga, Latvia

[v]Also at Karoly Robert Campus, MATE Institute of Technology, Gyongyos, Hungary

[w]Now at INFN Sezione di Milano-Bicocca, Milano, Italy

[x]Now at University of Rochester, Rochester, New York, USA

[y]Also at CERN, European Organization for Nuclear Research, Geneva, Switzerland

[z]Also at an institute or an international laboratory covered by a cooperation agreement with CERN

[aa]Also at Georgian Technical University, Tbilisi, Georgia

[bb]Now at National Technical University of Athens, Greece

[cc]Now at University of Nebraska, USA

[dd]Also at Istanbul Bilgi University, Istanbul, Turkey

[ee]Also at Near East University, Research Center of Experimental Health Science, Mersin, Turkey

[ff]Also at Konya Technical University, Konya, Turkey

[gg]Also at Izmir Bakircay University, Izmir, Turkey

[hh] Also at Adiyaman University, Adiyaman, Turkey
[ii] Also at Istanbul Gedik University, Istanbul, Turkey
[jj] Also at Necmettin Erbakan University, Konya, Turkey
[kk] Also at Bozok Universitetesi Rektörlügü, Yozgat, Turkey
[ll] Also at Marmara University, Istanbul, Turkey
[mm] Also at Milli Savunma University, Istanbul, Turkey
[nn] Also at Kafkas University, Kars, Turkey
[oo] Also at Hacettepe University, Ankara, Turkey
[pp] Also at Istanbul University - Cerrahpasa, Faculty of Engineering, Istanbul, Turkey
[qq] Also at Yildiz Technical University, Istanbul, Turkey
[rr] Also at Bingol University, Bingol, Turkey
[ss] Also at Sinop University, Sinop, Turkey
[tt] Also at Erciyes University, Kayseri, Turkey