Computing and Software for Big Science (2022) 6:2
https://doi.org/10.1007/s41781-021-00077-9

ORIGINAL ARTICLE

Deep Learning Strategies for ProtoDUNE Raw Data Denoising

Marco Rossi[1,2] · Sofia Vallecorsa[1]

Received: 3 July 2021 / Accepted: 14 December 2021 / Published online: 7 January 2022

Abstract
In this work, we investigate different machine learning-based strategies for denoising raw simulation data from the ProtoDUNE experiment. The ProtoDUNE detector is hosted by CERN and it aims to test and calibrate the technologies for DUNE, a forthcoming experiment in neutrino physics. The reconstruction workchain consists of converting digital detector signals into physical high-level quantities. We address the first step in reconstruction, namely raw data denoising, leveraging deep learning algorithms. We design two architectures based on graph neural networks, aiming to enhance the receptive field of basic convolutional neural networks. We benchmark this approach against traditional algorithms implemented by the DUNE collaboration. We test the capabilities of graph neural network hardware accelerator setups to speed up training and inference processes.

Keywords Deep learning · ProtoDUNE · Denoising · Convolutional neural networks · Graph networks

Introduction

Deep learning algorithms achieved outstanding results in many research fields in the last few years. The neutrino high energy physics community [1–4], and in general the particle physics one [5, 6], is making an effort to apply such technologies to create a new generation of automated tools. In general, capable of processing efficiently huge amounts of information, these algorithms work with increased performance to mine deeper in collected data and discover hidden patterns responsible for potential new physics scenarios. Event reconstruction algorithms, i.e., reconstruction, the process of extraction of useful quantities from detector or simulated data, are especially suited to the application of this approach.

DUNE [7] is a next-generation experiment in the neutrino oscillation research field. The DUNE Far Detector (FD) [8–10], based at the Sanford Underground Research Facility (SURF) in South Dakota, will be the largest monolithic Liquid Argon Time Projecting Chamber (LArTPC) detector ever built. To test and validate technologies for the construction of such detector, a prototype, ProtoDUNE Single Phase (SP) [11], has been built at the CERN Neutrino Platform.

Particle interactions with the liquid Argon produce ionization electrons that are drifted, thanks to an electric field, towards Anode Plane Assemblies (APAs) made of three readout planes that collect the deposited charge through time. Each readout plane is a bundle of wires, oriented in a specific direction and continuously monitored by the detector: two planes are called induction planes, while the last one is named the collection plane. In particular, the ProtoDUNE SP cage envelope is surrounded by 6 different APAs, each with two induction planes of 800 wires and a collection one, holding together 960 wires. Overall, a total of 15,360 wires on the sides of the detector provide a comprehensive and detailed picture of events from different points of view.

ProtoDUNE SP measures a digitized value of the induced current on each wire (ADC value). Since the detector maximum sampling rate is 2MHz and the common readout time windows at ProtoDUNE SP last 500ns, raw data (or raw digits) form a sequence of 6000 current measurements per wire. In this paper, we focus on event reconstruction of ProtoDUNE SP simulated data. We simulate interactions with the help of the LArSoft [12] framework and its `dunetpc` package. Figure 4 shows an example of the simulated data from a single collection plane: raw digits are cast into a 6000×960

✉ Marco Rossi
marco.rossi@cern.ch

[1] CERN openlab, 1211 Geneva 23, Switzerland

[2] TIF Lab, Dipartimento di Fisica, Università degli Studi di Milano and INFN Sezione di Milano, Via Celoria 16, 20133 Milan, Italy

resolution image plotting the ADC values heat map over time versus wire number axes.

Raw digits recorded by detector electronics intrinsically contain noise that must be filtered out. The traditional approach [8], developed by the MicroBooNE experiment [13, 14], is based on a 2-dimensional deconvolution and consists of two steps: first, a mask is produced to identify the Regions Of Interest (ROI) in the raw data containing signals; then, in those regions, Gaussian shape peaks are fitted to match the inputs, filtering them in Fourier space and deconvolving back the results.

The goal of this work is to tackle the denoising problem, implementing automatic tools at the readout plane view level. The nature of the inputs, above all sparsity and size, represents a challenging benchmark for Deep Learning models. Images indeed contain signals, mainly organized in long monodimensional tracks and clusters, divided by almost empty extended regions. Then, trying to train classic deep feed-forward models on sparse inputs might lead to sub-optimal results, since gradients would receive a large contribution from pixels in those noise-dominated empty regions. Moreover, the high image resolution might result in a pixel receptive field limited to just local neighborhoods, which are small compared to the extension of the meaningful spatial features included in the plane views. Finally, handling the size of the inputs requires careful RAM and GPU memory management while dealing with complex model architectures.

The paper is organized as follows. First, in section" Proposed Models", we describe the models and operations used to tackle the problem. Then, the section "Dataset and Training" illustrates the dataset we generated and the methods employed to train the networks. The section "Experiments Results" is devoted to the presentation of experimental results. Finally, in section "Conclusions", we present our conclusion and future development directions.

Proposed Models

Convolutional neural networks (CNN) are based on a stack of sliding kernels comprised of multiple filters, that are trained according to some optimization method to output a feature map. The convolutional kernel generates each feature map pixel as a function of a small neighboring portion of the input image. Hence, the receptive field of the pixels in each layer is constrained by the kernel size, which could be enlarged by increasing the depth of the network. The scope of the present section is to describe an alternative approach to the issue: by enriching the network with alternative operations we hope to increase the expressiveness of the internal representation by exploiting non-local correlations between pixel values, alongside the already discussed local neighborhood pixel intensities.

Graph Convolutional Neural Network

We implement a Graph Convolutional Neural Network (GraphCNN) inspired by [15, 16] and based on the Edge Conditioned Convolution (ECC) operation, first presented in [17]. We employ a simplified version of the ECC layer: it builds the output representation as a pixel-wise average of a common convolution with a 3×3 kernel and a Non-Local Aggregation (NLA) operation. We wrap such operations into a network layer called Graph Convolution (GCONV). Figure 1 sketches the GCONV layer mechanism.

The NLA connects each pixel to its k closest ones in feature space, according to the Euclidean distance, and mixes the information through a feed-forward layer. If at layer l, the i-th of an n-pixel input image is described by the vector $\mathbf{H}_i^l \in \mathbb{R}^{d_l}$, then the NLA output $\mathbf{H}_i^{l+1}$ has the following form:

$$\mathbf{H}_i^{l+1} = \sigma\left(\frac{1}{|\mathcal{N}_i^l|} \sum_{j \in \mathcal{N}_i^l} \Theta^l \left(\mathbf{H}_i^l - \mathbf{H}_j^l\right) + \mathrm{W}^l \mathbf{H}_i^l + \mathbf{b}^l \right) \in \mathbb{R}^{d_{l+1}}. \tag{1}$$

With $\mathcal{N}_i^l$ being the neighborhood of pixel i in the H_i^l representation at layer l; $\{\Theta^l, W^l\} \in \mathbb{R}^{d_{l+1} \times d_l}$ and $\mathbf{b}^l \in \mathbb{R}^{d_l+1}$ are trainable weights and biases shared throughout pixels; σ is the element-wise sigmoid function.

Note that the operation in Eq. (1) requires building a k-NN graph which requires an amount of memory proportional to the area of the input image times the number of neighbors per pixel k. Assuming we fix $k = 8$, with an input image of 960×6000 pixels and employing single precision floating point numbers, the graph construction operation burden is of order $\mathcal{O}(200\ \text{MB})$. If the architecture involves multiple graph building operations, the GPU memory is easily saturated. Therefore, we have to limit the model inputs to just crops of the actual image as explained in section "Network Training".

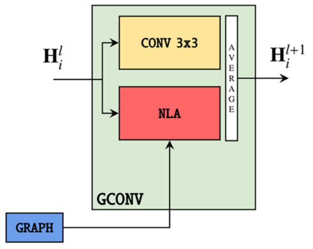

Fig. 1 GCONV layer. The inner representation vector $\mathbf{H}_i^l$ is updated to $\mathbf{H}_i^{l+1}$ by means of NLA and 2D convolution operations. NLA relies on a previously computed KNN graph

Figure 2 shows our GCNN network architecture. Note the final residual connection in the top branch: the usual sum has been replaced by a multiplication. This choice is tailor made on the input data themselves, which are mainly comprised of long tracks separated by empty space. In those regions it could be easier to learn how to remove the noise multiplicatively rather than additively. The network, in principle, does not have to learn to perfectly profile the noise and then subtract it from the input itself, whereas it can employ a mask to cut down such uninteresting regions with multiplications by small numbers.

U-Shaped Self-Constructing Graph Network

As stated in the previous section, the main limitation of the GCNN model is the memory consumption burden due to the graph operation: graph building scales linearly with the size of the inputs, therefore, the model accepts crops rather than the full APA image. The cropping workaround prevents very long-range correlations between pixels being taken into account and leads to inference time performance issues. If the user does not have access to a high GPU memory bandwidth, only batches of crops can be processed in parallel and processing big datasets is not time efficient.

As an alternative approach to the GCNN, we follow the idea introduced in [19] with the Self-Constructing Graph Network (SCG-Net): a graph neural network that outputs results from a low dimensional representation of the high resolution image created by a full CNN; the original shape of the image is then retrieved interpolating between pixels. In the original work, a bilinear interpolation is employed for the upsampling. In this way the authors were able to process big images containing up to 6000×6000 pixels. This seems a more natural approach to the problem when the input images contain dense features, rather than sparse and localized ones as in our case. We believe that the entire pipeline potentially washes out the fine grained information contained in the input during downscaling, which cannot be recovered by means of a simple interpolation.

Therefore, we introduce the U-shaped Self-Constructing Graph Network (USCG-Net), where a U-Net [20] like network structure with residual connections carries the information from the input node all the way to the outputs. Figure 3 shows the USCG-Net architecture, with pooling blocks, comprised of convolutional and pooling layers, that take care of stepwise scaling the inputs. A pretrained ResNeXt-50 with a 32×4 template [18] is used to build an initial feature map to be fed into the SCG layer. The early-layer

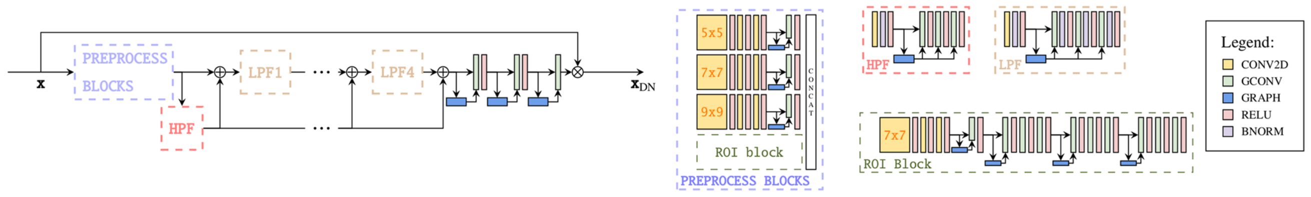

Fig. 2 GCNN architecture. The model takes noisy input images x and outputs denoised ones $\mathbf{x}_{DN}$. The design is organized with low- and high-pass filters (LPF and HPF) as in [15]. As explained in section "Network Training", we concatenate a pretrained ROI block with preprocessing layers to make the network distinguish between signal and background. The ROI block, indeed, performs the binary segmentation

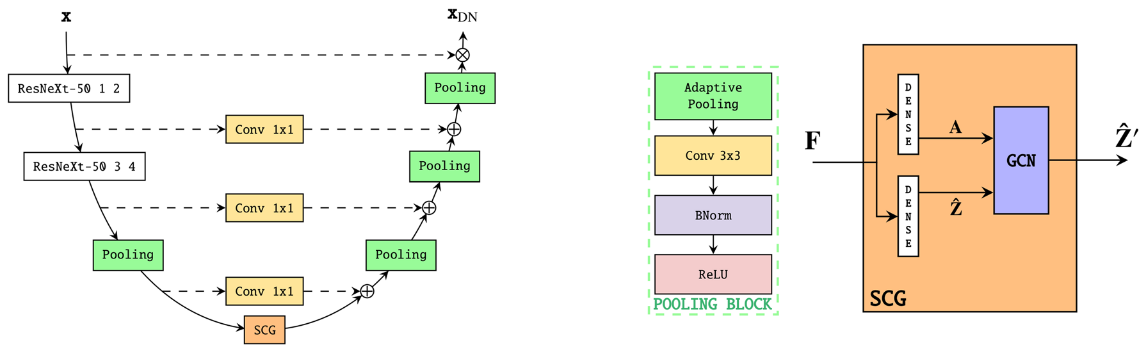

Fig. 3 USCG-Net architecture. The Pooling block has a two folded aim thanks to its adaptive pooling layer: on the left branch it downscales the input, while on the right one it provides upsampling. According to [18], ResNeXt-50 with 32×4 template is arranged in 4 main blocks, containing 9, 12, 18 and 9 convolutional layers, respectively. In the chart, we refer to such blocks as $\mathtt{ResNeXt}-50 < \mathtt{blocknumber} >$ and we insert a residual connection between the second and the third block. The 1×1 convolutions in the horizontal links are needed to adapt the number of filters in the image to perform the residual recombination operation

representation of the pretrained network should be generic enough to catch the spatial features of the inputs, driving the training process during the initial phases of the optimization. Note the residual recombination operations in the right branch: the sum in the residual connections has been replaced by a convolution with a 1×1 kernel to increase the complexity of the network. Finally, for the final residual link, we employ again the multiplication trick, as explained in section "Graph Convolutional Neural Network".

The SGC layer is the core of the network: the input image is turned into a graph, allowing connections between distant pixels. The SCG layer [19] takes in an image $\mathbf{F} \in \mathbb{R}^{h \times w \times d}$ and through an encoding functions, represented by dense layers, maps it into two outputs: an adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$, describing the graph edges, and a node feature vector $\hat{\mathbf{Z}} \in \mathbb{R}^{n \times c}$. In the above formulas, d and c are the input and output channel dimensions, respectively and n, which is equal to the input image resolution $h \times w$, is the number of extracted nodes. The predicted graph is then analyzed by a 1-layer graph neural network, such as GCN [21] or GIN [22], to further mix the node features, yielding a final node feature vector $\hat{\mathbf{Z}}' \in \mathbb{R}^{n \times c'}$. In our experiments, we exploited the GCN layer as default; architecture variants with other kinds of graph layers can be addressed in a future work. $\hat{\mathbf{Z}}'$ can be finally projected back into $\mathbb{R}^{h \times w \times c'}$ and upsampled by pooling blocks in the right branch of the USCG-Net to the original image resolution.

Dataset and Training

Datsets

In this section, we present the datasets used to train and test our models. We underline that we present results for simulated data only, testing on detector data is out of the scope of the present work. We simulate interactions within the ProtoDUNE SP chamber through the LArSoft [12] framework and its `dunetpc` package. We consider events originating from a proton beam of various energies, plus cosmic rays, interacting with the Argon targets. Our supervised training approach is based on simulated raw digits. The simulation provides also the associated noise-free charge depositions: we employ this information as target outputs for the denoising models.

Table 1 describes the considered datasets. In Figs. 4 and 5, we show visual examples from `v08_24_00` and `v09_10_00` datasets. We plot raw digit images and horizontal slices, namely single channel waveforms, either with (raw waveform) and without noise (clear waveform). Noise is mainly comprised of a pedestal value that changes across different datasets and a background noise that

Table 1 Datasets for training and testing

`dunetpc`	n events	p energy
`v08_24_00`	10	2 GeV
`v09_10_00`	70	0.3 GeV, 0.5 GeV, 1 GeV, 2 GeV, 3 GeV, 6 GeV, 7 GeV

The two differ in the producer package version, the size and the event beam energies. The second dataset contains 10 events for each proton energy specified

overwhelms small energy depositions and modifies spikes amplitudes. We consider the `v08_24_00` dataset a simplified dataset since it is smaller and contains easier features to denoise and segment than the `v09_10_00` one. `v09_10_00` is more complex due to the detector data driven approach upon which the generator software is based. Clear waveforms are not simply zero in the empty regions, but rather contain small steps. Moreover, low-frequency negative tails after big spikes in clear waveforms are clearly visible in Fig. 7.

We hold out from our datasets 10% of images for validation and 10% for testing. The validation set is used to choose the best model, while the test set is employed to present the final results given in section "Experiments Results". The criterion chosen to present results is to aggregate each performance quantity, over the test or validation set, then report it as a symmetric interval with the central value and its uncertainty being the arithmetic mean and the standard deviation on the mean, respectively. This choice also influences the best model selection during training, namely, the network checkpoint at epoch end that achieves the lowest upper bound of the loss function interval.

We remark that, as described in section "Introduction", a single event at ProtoDUNE SP is recorded by 6 different APAs, that in turn provide 3 plane views. Therefore, one event does not count as one training point for our neural networks. In section "Proposed Models", we anticipated that the GCNN network acts independently on small crops: if 32×32 crops are employed, each event in the dataset becomes 9×10^4 crops. The USCG-Net, instead, processes entire plane views. The two datasets provide 18 and 126 plane views to assess the model performance and its associated uncertainty. Note that the output image has a dimensionality of the order of millions of pixels and each pixel value provides a contribution in the computation of the final performance metric.

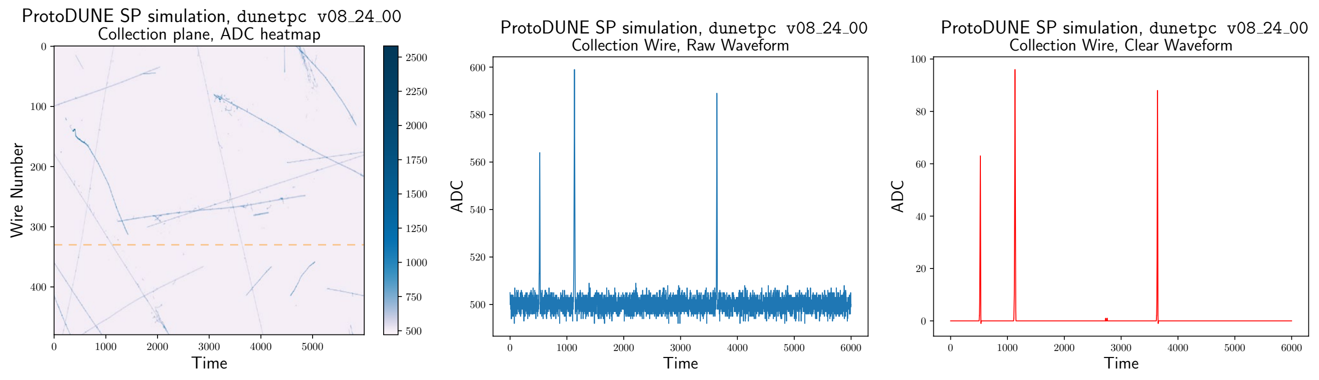

Fig. 4 Example taken from `dunetpc v08_24_00` dataset. Collection plane view with noisy and clear waveforms extracted from the channel marked by the horizontal orange dashed line

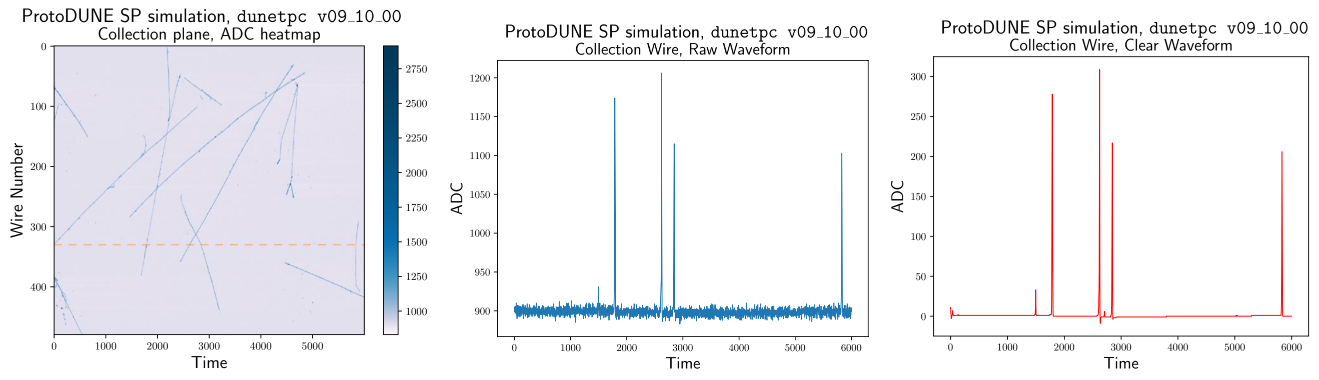

Fig. 5 Example taken from `dunetpc v09_10_00` dataset. Collection plane view with noisy and clear waveforms extracted from the channel marked by the horizontal orange dashed line

Network Training

All networks that we train share the same preprocessing procedure, where we take the plane views in a dataset and apply a median subtraction to each of them. This step is fundamental to estimate and subtract the pedestal value. We introduce this operation mimicking the traditional approach because it contributes to improve the final performance. Furthermore, for network training stability, the inputs are rescaled in the unit range according to the min–max normalization technique.

The GCNN network suffers from the memory issue discussed in section "Graph Convolutional Neural Network". To address this problem, we employ a data parallel approach, cropping the inputs into 32×32 pixels images and processing every tile independently. However, this solution requires some subtleties in the training process due to the nature of the inputs.

Since the inputs contain sparse features, it is likely that the most of the crops contain little to no signal. It is meaningless to feed the network with multiple empty crops. To speed up the training, we decide to train on just a subset of the available crops. This choice, in turn, triggers a second issue: sampling randomly the subset of crops provides an extremely unbalanced dataset, where we are very likely to miss crops containing interesting charge depositions. Hence, we fix the percentage of crops containing signal to balance the signal to background pixel ratio: in our experiments, we keep this quantity at 99%. The cropping procedure works as follows: we mark as signal all the pixels in clear waveforms that have non-zero ADC value count in a plane view; we sample randomly the desired number of crop centers, complying with the exact ratio of signal to background percentage; we finally crop the plane views around the drawn centers.

We train the GCNN network in Fig. 2 for both the image segmentation task and denoising, managed by the ROI Block and the network final outputs, respectively. Image segmentation is a classification task at the pixel level, where the labels usually correspond to a finite set of exhaustive and mutually exclusive classes: the ROI block, indeed, is trained to distinguish between pixels that contain or not electron induced current signals. Image denoising, instead, aims at subtracting the noise contribution from each pixel intensity and is a regression task.

To optimize the two parts of the network, we design an ad hoc training strategy as follows. First, we train the ROI Block alone on image segmentation for 100 epochs and save the best configuration: in this step, the network branch learns to distinguish between signal and background, i.e., empty pixels. We employ binary cross-entropy as the loss function, along with the AMSGrad variant [23] of the Adam algorithm [24] as the optimizer with a learning rate of 10^{-3}. Second, we freeze and attach the trained ROI block weights to the remaining part of the network. Then, we train the GCNN on image denoising for a further 50 epochs and save the network's best configuration.

The GCNN denoising model is trained using the AMSGrad optimizer with a learning rate of 9×10^{-3}, while optimizing a custom loss function made of two contributions: the mean squared error (MSE) $\mathcal{L}_{\mathrm{MSE}}$ between labels and outputs and a loss function $\mathcal{L}_{\mathrm{ssim}}$ derived from the statistical structural similarity index measure (stat-SSIM) [25]. stat-SSIM for two images $\mathbf{x}$ and $\mathbf{y}$ is given by the following equation:

$$\text{stat- SSIM}(\mathbf{x},\mathbf{y}) = \frac{1}{n_\mathrm{p} n_\mathrm{c}} \sum_{\mathrm{pc}} \left(\frac{2\mu_x \mu_y + \epsilon_\mu}{\mu_x^2 + \mu_y^2 + \epsilon_\mu} \right)_{\mathrm{pc}} \times \left(\frac{2E\left[(\mathbf{x}-\mu_x)(\mathbf{y}-\mu_y)\right] + \epsilon_\sigma}{E\left[(\mathbf{x}-\mu_x)^2\right] + E\left[(\mathbf{y}-\mu_y)^2\right] + \epsilon_\sigma} \right)_{\mathrm{pc}}, \tag{2}$$

where μ_i is a shorthand for the image i expected value $E[i]$, which is computed through a convolution with a 11×11 Gaussian kernel of standard deviation 3. All the other expectation values in the equation are calculated convoluting the argument quantity with the same Gaussian filter. ϵ_μ and ϵ_σ are two regulators that limit the maximum resolution at which the fractions in the equations are computed, respectively, imposing a cutoff on the mean and variance expected values. In particular, when both the numerator and the denominator of a fraction reach much smaller values than the corresponding ϵ, the output gets close to one. In the experiments, ϵ_μ and ϵ_σ are fixed to the square of 0.5. This choice implies that for the stat-SSIM computation, we estimate the means and standard deviations of the distributions at scales larger than half of one ADC value, namely the granularity of the recorded detector hits. The result is finally averaged over the entire image containing n_p pixels and n_c channel dimensions. The quantity in Eq. (2) takes values in the range $[-1, 1]$ and approaches 1 only if $\mathbf{x} = \mathbf{y}$. The associated loss function is then given by $\mathcal{L}_{\mathrm{ssim}} = 1 - \text{stat- SSIM}$: it is a perceptual loss, in the sense that tries to assess the fidelity of the image focusing on structural information. It relies on the idea that pixels may have strong correlations, especially when they are spatially close. In contrast, MSE evaluates pixels' absolute differences, without taking into account any dependence among them. More details on the interpretation of these quantities can be found in [26]. The two contributions in the loss function are weighted as follows: $\mathcal{L} = \alpha \cdot \mathcal{L}_{MSE} + (1-\alpha) \cdot w \cdot \mathcal{L}_{ssim}$. We fine tune the multiplicative parameter $w = 10^{-3}$, to balance the gradients w.r.t. the model's trainable parameters provided by the two terms in the sum. The parameter α is fixed to 0.84 as in [27].

In the following, we will refer, with slight abuse of notation, to a CNN as a GCNN network with Graph Convolutional layers replaced by plain Convolutional ones.

The USCG-Net is relatively simple to train: since no crops are employed, no sampling method is required. Although no cropping is needed, since the model fits in a single 16 GB GPU even with an entire plane view as input, we prefer to employ a sliding window mechanism as in the original SCG-Net paper. We split the raw digits array along the time axis with a 2000 pixel wide window and 1000 pixel stride and feed each slice to the network. The results are then combined by averaging predictions on overlapping regions. The USCG-Net is trained minimizing the MSE function between model outputs and clear waveforms, with AMSGrad optimizer and learning rate of 10^{-3}. In this case, we drop the stat-SSIM contribution from the loss function, since we experienced training convergence problems when including that term during our experiments.

Experiments Results

We employ four different metrics to assess the goodness of our models and benchmark them against the state-of-the-art approach. Three of them are the stat-SSIM, peak signal to noise ratio (PSNR) and MSE on the two-dimensional raw digits, while the last one is a custom metric called integrated mean absolute error (iMAE) and will be defined in the following.

PSNR between a noise-free image $\mathbf{x}$ and a denoised $\mathbf{y}$ one is given by the following equation:

$$\mathrm{PSNR}(\mathbf{x},\mathbf{y}) = 10 \cdot \log_{10} \left(\frac{\max(\mathbf{x})^2}{\mathrm{MSE}(\mathbf{x},\mathbf{y})} \right). \tag{3}$$

Note that PSNR and stat-SSIM increase with the reconstruction quality, while MSE decreases. We observe that these three quantities are really suited to compare the deep learning models between themselves, however, they are not really informative when the baseline approach is considered. Indeed, the baseline approach aims at fitting Gaussian peaks in regions marked as interesting, rather than reconstructing the precise shape of the spike contained in the raw digits.

Hence, the baseline tool performs inevitably poorly on the considered metrics and up to our knowledge, there is no default metric to assess its performance. Nonetheless, we define a custom quantity that tries to compare the different approaches, observing that the deconvolution process does not preserve waveform amplitudes, but their integrals. For such reason, we decide to evaluate the mean absolute error on wires integrated charge (iMAE):

$$\mathrm{iMAE}(\mathbf{x}, \mathbf{y}) = \frac{1}{n_w} \sum_{w=1}^{n_w} \left| \sum_t (\mathbf{x} - \mathbf{y})_{wt} \right|, \tag{4}$$

where the sum inside the absolute values runs over time for the whole readout window, while the outer sum averages the result over the wire dimension. Note that the deconvolution approach does not preserve waveform amplitudes, because a filtering function is applied to the signal in Fourier space, then results are deconvolved back to the time domain. Furthermore, the deconvolution outputs are known up to an overall normalization constant, that we fit on the datasets to minimize the iMAE quantity. We show that although we perform this operation on the state-of-the-art tool outputs for a fair comparison against our models, they nonetheless achieve a worse iMAE score. Table 2 collects the metrics values evaluated on `v08_24_00` dataset. We gather `v09_10_00` dataset results in Table 3. Note that we present only evaluations for 2GeV beam energy events, since we find metrics distributions to be flat in the energy parameter. Figures 6 and 7 show samples of labels and denoised waveforms.

USCG-Net like networks exceed GCNN-like ones in all the collected metrics. To have a first assessment of the quality of the neural network generalization power, we train two versions of the USCG-Net: one on the `v08_24_00` dataset and the other on the `v09_10_00` dataset. We decide not to train the GCNN-like networks on the `v09_10_00` dataset after we observed difficulties in training convergence as well as long training times on such a big dataset. We evaluate these networks on both datasets.

Following expectations, the networks trained and applied on the same dataset lead to better performance. The only exceptions are given by the iMAE columns, where the USCG-Net trained on the dataset opposite to the testing one, achieve the best iMAE score. The stat-SSIM index score drops significantly for GCNN-like networks. All the networks, nonetheless, show hints of overall good

Table 2 Test metrics for denoising on `v08_24_00` dataset

Model	stat-SSIM	PSNR	MSE	iMAE
Baseline	–	–	–	5391 ± 1622
CNN v08	0.471 ± 0.008	67.3 ± 1.2	0.57 ± 0.03	287 ± 12
GCNN v08	0.512 ± 0.011	70.12 ± 1.4	0.30 ± 0.01	191.4 ± 2.6
USCG v08	**0**.988 ± 0.005	**72**.66 ± 1.54	**0**.17 ± 0.02	95.5 ± 8.5
USCG v09	0.926 ± 0.007	72.3 ± 1.5	0.18 ± 0.02	**76**.3 ± 8.2

Results are shown for collection plane and 2GeV beam energy only. The version, `v08` or `v09`, next to the model name in the first column refers to which dataset the correspondent model was trained on

Table 3 Test metrics for denoising on `v09_10_00` dataset

Model	stat-SSIM	PSNR	MSE	iMAE [$\times 10^3$]
Baseline	–	–	–	5.86 ± 0.52
CNN v08	0.37 ± 0.02	57.3 ± 1.4	5.79 ± 0.88	4.16 ± 0.36
GCNN v08	0.40 ± 0.02	57.7 ± 1.5	5.27 ± 0.69	4.51 ± 0.39
USCG v08	0.65 ± 0.05	61.1 ± 1.6	2.3 ± 0.2	**2**.18 ± 0.29
USCG v09	**0**.81 ± 0.07	**61**.8 ± 1.7	**1**.99 ± 0.19	2.25 ± 0.23

Results are shown for collection plane and 2GeV beam energy only. The version, `v08` or `v09`, next to the model name in the first column refers to which dataset the correspondent model was trained on

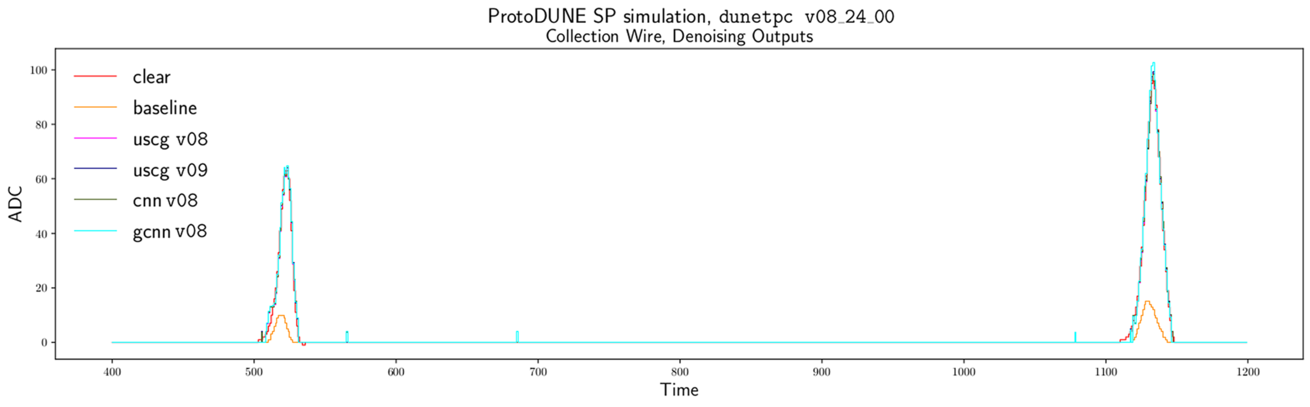

Fig. 6 Detail of a raw waveform from `dunetpc v08_24_00` dataset: label, traditional algorithm and neural networks outputs. The version, `v08` or `v09`, next to the model name in the legend refers to which dataset the correspondent model was trained on

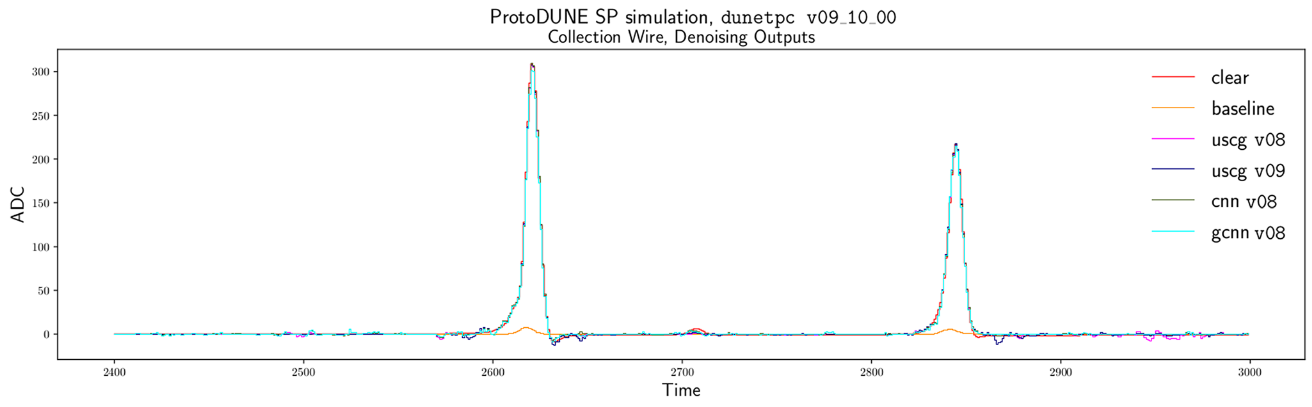

Fig. 7 Detail of a raw waveform from `dunetpc v09_10_00` dataset: label, traditional algorithm and neural networks outputs. The version, `v08` or `v09`, next to the model name in the legend refers to which dataset the correspondent model was trained on

generalization power when they are applied on datasets not used for training. This fact is well supported by the PSNR columns, which show that even the worst model achieves competitive results. We underline that the USCG-Net is not trained according to the stat-SSIM quantity: adding an extra term in the loss function containing such term could be considered a point of further development of the present research.

Conclusions

In this paper, we presented deep learning strategies for denoising raw digits simulated data at ProtoDUNE SP. Our approach leads to an automated tool that cleans raw inputs and in the future could be adapted to work on real data. We investigated the capabilities of Graph Neural Networks, testing approaches alternative to classical Convolutional Neural Networks on a novel use case. Graph Neural Networks aimed to exploit long-distance correlations between pixels, in particular the USCG-Net exemplified this approach, processing big sized inputs while still fitting GPU memory constraints. Our trained neural networks were able to outperform the state-of-the-art traditional reconstruction algorithm on the custom iMAE metric. All the networks gave hints of interesting generalization power, especially in the PSNR quality assessment, achieving high performances even on datasets different to the one they had been trained on. Further work directions will aim to fully assess the models on larger, more complete testing datasets.

Acknowledgements We thank the IBM Company that supported the realization of this paper, gently providing hardware and intellectual contribution.

Funding Open access funding provided by CERN (European Organization for Nuclear Research).

Data Availability Statement This manuscript has associated data in a data repository. [Authors' comment: No associated data except for code. The associated code to replicate the studies in this paper can be found at: https://zenodo.org/record/5821522].

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Dominé L, Terao K (2020) Scalable deep convolutional neural networks for sparse, locally dense liquid argon time projection chamber data. Phys Rev D 102:012005. https://doi.org/10.1103/PhysRevD.102.012005
2. Abi B, Acciarri R et al (2020) Neutrino interaction classification with a convolutional neural network in the DUNE far detector. Phys Rev D 102:092003. https://doi.org/10.1103/PhysRevD.102.092003
3. Aurisano A, Radovic A et al (2016) A convolutional neural network neutrino event classifier. J Instrum 11(09):P09001. https://doi.org/10.1088/1748-0221/11/09/P09001
4. Kronmueller M, Glauch T (2019) Application of deep neural networks to event type classification in icecube, application of deep neural networks to event type classification in icecube. arXiv:1908.08763

5. Albertsson K, Altoe P et al (2019) Machine learning in high energy physics community white paper, machine learning in high energy physics community white paper. arXiv:2002.03005
6. Bourilkov D (2019) Machine and deep learning applications in particle physics. Int J Modern Phys A 34(35):1930019. https://doi.org/10.1142/S0217751X19300199
7. Abi B, Acciarri R et al (2020) Volume I. Introduction to DUNE. JINST 15(08):T08008. https://doi.org/10.1088/1748-0221/15/08/T08008
8. Abi B, Acciarri R et al (2020) Deep underground neutrino experiment (dune), far detector technical design report, volume ii: Dune physics. arXiv:2002.03005
9. Abi B, Acciarri R et al (2020) Volume III. DUNE far detector technical coordination. JINST 15(08):T08009. https://doi.org/10.1088/1748-0221/15/08/T08009
10. Abi B, Acciarri R et al (2020) Volume IV. The DUNE far detector single-phase technology. JINST 15(08):T08010. https://doi.org/10.1088/1748-0221/15/08/T08010
11. Abi B, Acciarri R et al (2017) The single-phase protodune technical design report, the single-phase ProtoDUNE technical design report. arXiv:1706.07081
12. Church ED (2014) LArSoft: a software package for liquid argon time projection drift chambers, Larsoft: a software package for liquid argon time projection drift chambers. arXiv:1311.6774
13. Acciarri R, Adams C et al (2017) Noise characterization and filtering in the MicroBooNE liquid argon TPC. J Instrum 12(08):P08003–P08003. https://doi.org/10.1088/1748-0221/12/08/P08003
14. Adams C, An R et al (2018) Ionization electron signal processing in single phase LArTPCs. Part I. Algorithm description and quantitative evaluation with MicroBooNE simulation. J Instrum 13(07):P07006. https://doi.org/10.1088/1748-0221/13/07/p07006
15. Valsesia D, Fracastoro G et al (2019) Deep graph-convolutional image denoising. arXiv:1907.08448
16. Valsesia D, Fracastoro G et al (2019) Image denoising with graph-convolutional neural networks. arXiv:1905.12281
17. Simonovsky M, Komodakis N (2017) Dynamic edge-conditioned filters in convolutional neural networks on graphs. arXiv:1704.02901
18. Xie S, Girshick R, others, Dollár P, Tu Z, He K (2017) Aggregated residual transformations for deep neural networks. arXiv:1611.05431
19. Liu Q, Kampffmeyer M et al (2020) Scg-net: self-constructing graph neural networks for semantic segmentation. arXiv:2009.01599
20. Ronneberger O, Fischer P et al (2015) U-net: convolutional networks for biomedical image segmentation. arXiv:1505.04597
21. Scarselli F, Gori M et al (2009) The graph neural network model. IEEE Trans Neural Netw 20(1):61. https://doi.org/10.1109/TNN.2008.2005605
22. Xu K, Hu W et al (2019) How powerful are graph neural networks? arXiv:1810.00826
23. Reddi SJ, Kale S et al (2018) On the convergence of adam and beyond. In: International conference on learning representations. https://openreview.net/forum?id=ryQu7f-RZ
24. Kingma DP, Ba J (2017) Adam: a method for stochastic optimization. arXiv:1412.6980
25. Channappayya SS, Bovik AC et al (2008) Rate bounds on SSIM index of quantized images. IEEE Trans Image Process 17(6):857. https://doi.org/10.1109/TIP.2008.921328
26. Wang Z, Bovik AC (2009) Mean squared error: love it or leave it? A new look at signal fidelity measuresa new look at signal fidelity measures. EEE Signal Process Mag 26(1):98. https://doi.org/10.1109/MSP.2008.930649
27. Zhao H, Gallo O et al (2018) Loss functions for neural networks for image processing. arXiv:1511.08861