Deep Learning strategies for ProtoDUNE raw data denoising

Marco Rossi[1,2,*] and *Sofia* Vallecorsa[1,**]

[1]CERN openlab, Geneva 23, CH-1211, Switzerland

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

Abstract. In this work we investigate different machine learning based strategies for denoising raw simulation data from ProtoDUNE experiment. ProtoDUNE detector is hosted by CERN and it aims to test and calibrate the technologies for DUNE, a forthcoming experiment in neutrino physics. Our models leverage deep learning algorithms to make the first step in the reconstruction workchain, which consists in converting digital detector signals into physical high level quantities. We benchmark this approach against traditional algorithms implemented by the DUNE collaboration. We test the capabilities of graph neural networks, while exploiting multi-GPU setups to accelerate training and inference processes.

1 Introduction

Deep learning algorithms achieved outstanding results in many research fields in the last few years. Also the particle physics community is making an effort applying such technologies to create a new generation of automated tools [1–3]. Capable of processing efficiently huge amount of information, these algorithms work with increased perfomances 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 specially suited to the application of this approach.

DUNE [4] is a next-generation experiment in the neutrino oscillation research field. DUNE Far Detector (FD) [5–7], based at Fermilab, will be the largest monolithic Liquid Argon Time Projecting Chamber (LArTPC) detector ever built. In order to test and validate technologies for the construction of such detector, a prototype, ProtoDUNE Single Phase (SP) [8], has been built at the CERN Neutrino Platform.

Particles interactions within these detectors produce ionization electrons that are drifted, due to an electric field, towards three readout planes that collect deposited charge through time. Raw data, or raw digits, can then be cast into two dimensional arrays containing the digitized values (ADC) of the charge collected by the planes as a function of time ticks and sensing wires, which the planes are made of.

In this paper we focus on event reconstruction of ProtoDUNE SP simulated data. We simulate interactions with the help of the LArSoft [9] framework and its `dunetpc` package.

*e-mail: marco.rossi@cern.ch

**e-mail: sofia.vallecorsa@cern.ch

Each event is described by a 6000 time ticks wide image as in figure 3: ProtoDUNE SP samples at a maximum rate of 2 MHz (500 ns per tick), yielding a 3 ms time window per event. The number of wires, namely the height of the image, varies according to the readout plane between 800 and 960.

Raw digits are produced with an amount of noise that must be filtered out in order to extract useful information from them. The traditional approach is based on 2-dimensional deconvolution and consists in 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 fit to match the inputs, filtering them in Fourier space and deconvoluting back the results.

The goal of this work is to implement automatic tools that address the issue. The nature of the inputs, above all sparsity and size, represents a challenging benchmark for Deep Learning models. Images contain indeed signal mainly organized in monodimensional tracks and clusters, divided by almost empty extended regions. This fact may result in a waste of computational resources if classic deep feed forward models are employed. Moreover, handling the size of the inputs requires careful memory management and ad-hoc strategies design.

The paper is organized as follows. First, in section 2 we describe the models and operations used to tackle the problem. Then, section 3 illustrates the dataset we collected and the methods employed to train the networks. Section 4, instead, is devoted to the presentation of experimental results. Finally, in section 5 we present our conclusion and future development directions.

2 Proposed Models

Convolutional neural networks (CNN) are based on a stack of sliding kernels comprised by 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: enriching the network with alternative operations we hope to increase the complexity of the output representation by exploiting non-local correlation between pixel values, alongside the already discussed local neighborhood pixels intensities.

2.1 Graph Convolutional Neural Network

We implement a Graph Convolutional Neural Network (GraphCNN) inspired by [10, 11] and based on the Edge Conditioned Convolution (ECC) operation, first presented in [12]. We employ a simplified version of the ECC layer: it builds the output representation as an average of common convolutions and a Non-Local Aggregation (NLA) operation. NLA connects each pixel to its K closest pixels 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 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:

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

With $\mathcal{N}_i^l$ being the non local neighboring of pixel i, usually set to a constant value throughout the all image; $\{\Theta^l, W^l\} \in \mathbb{R}^{d^{l+1}\times d^l}$ and $\mathbf{b}^l \in \mathbb{R}^{d^l}$ are a trainable weight arrays; σ is the sigmoid

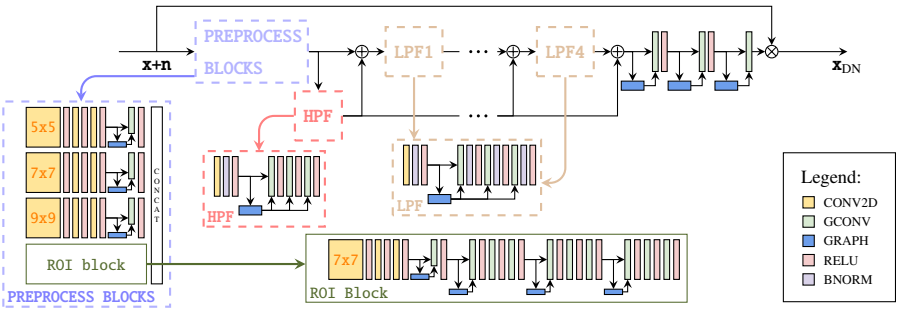

Figure 1. GCNN architecture. The design is organized with Low and High Pass Filters (LPF, HPF) as in [10]. As explained in section 3.2, we concatenate a pretrained ROI block with preprocessing layers to make the network distinguish between signal and background. The ROI block, indeed, performs the binary segmentation.

non linear function. Hence, we define the Graph Convolution (GC) of an image to be the result of a mean over the output of a NLA operation and a usual convolution by a 3×3 filter.

Note that the operation in equation 1 requires building a KNN-Graph. Hence, selecting for each node i its neighborhood $\mathcal{N}_i$ among all $n-1$ other pixels in the input image requires an amount of memory proportional to the area of the image itself. Assuming that we build a model stacking L layers, each performing a NLA operation, fixing the number of non local neighbors $|\mathcal{N}_i^l|$ of each node to a constant $K \in \mathbb{N}$ throughout the different layers, the required memory during graph construction would be of $O(200\,\text{MB})$ for a single 960×6000 pixels image, stored with single precision floating point numbers. Then, due to GPU memory constraints, we have to limit the model inputs to just crops of the actual image as explained in section 3.2.

Figure 1 shows our GCNN network architecture. Note the final residual connection: 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, doesn't have to learn to perfectly profile the noise and then subtract it to the input itself, whereas it can employ a mask to cut down such uninteresting regions with multiplications by small numbers.

2.2 U-shape Self Constructing Graph Network

As stated in the previous section, the main limitation of the GCNN model is given by the memory consuption burden that the graph operation brings into the game: graph building scales linearly with the number of pixels, preventing very long range correlation between them to be taken into account. Moreover, the inference sequential process on sets of little crops instead of whole images could be really time consumming when inspecting big datasets.

As an alternative approach to the GCNN, we follow the idea introduced in [13] 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 image original shape is then retrieved interpolating between pixels. In the original work, a bilinear interpolation is employed for the upsampling. This way the authors were able to process big images containing up to 6000×6000 pixels. Although this seems a natural approach to the problem, we believe that the entire pipeline potentially washes out the fine

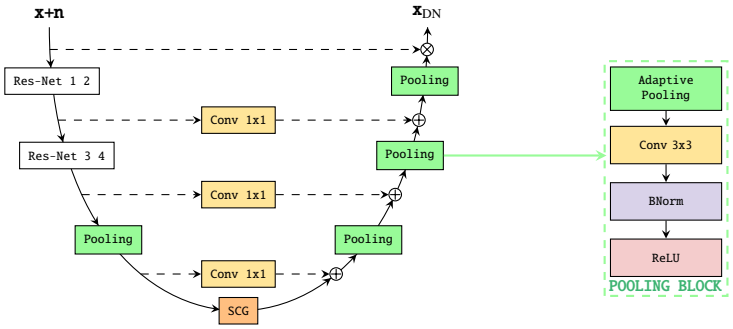

Figure 2. USCG-Net architecture. The Pooling block has a two folded aim thanks to its adaptive pooling layer: on the left branch it downscales the input, while on the right one it provides upsampling. We insert a residual connection between Res-Net layers 2 and 3. The 1×1 convolutions adapt the number of filters in the image to perform the residual recombination operation.

Table 1. Datasets for training and testing. The two differ for the producer package version, the size and the event beam energies. Second dataset contains 10 events for each proton energy specified.

`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

grained information contained in the input during downscaling, which cannot be recovered by means of a simple interpolation.

Therefore we introduce the U-shape Self Constructing Graph Network (USCG-Net), where a U-Net [14] like network structure with residual connections carries the information from the input node all the way to the outputs. Figure 2 shows the USCG-Net architecture, with pooling blocks, comprised by convolutional and pooling layers, that take care of step-wise scaling the inputs. A pretrained ResNet [15], with 50 layers, is used to build an initial feature map to be fed into the SCG layer. 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 in order to increase the complexity of the network. Finally, for the final residual link we employ again the multiplication trick, as explained in section 2.

3 Dataset and Training

3.1 Datsets

In this section we present the datasets collected to train and test our models. We underline that we deal with 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 [9] framework and its `dunetpc` package. We consider events originated from a proton beam of a certain energy plus cosmic rays interacting with the Argon targets. Our supervised training approach is based on simulated raw digits. We produce also clear raw digits with noise removed, that serve as target outputs for the denoising models.

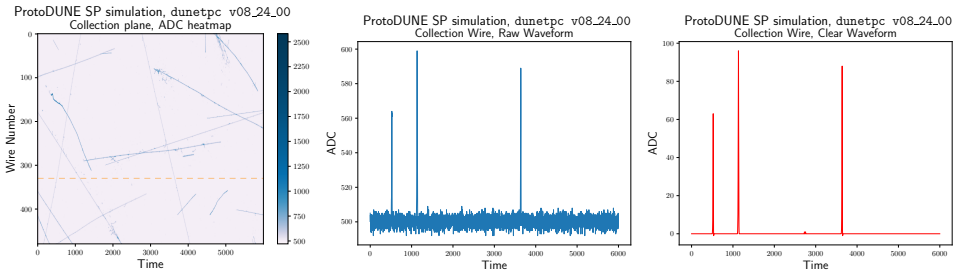

Figure 3. Example taken from `dunetpc v08_24_00` dataset. Collection plane view with noisy and clear waveforms.

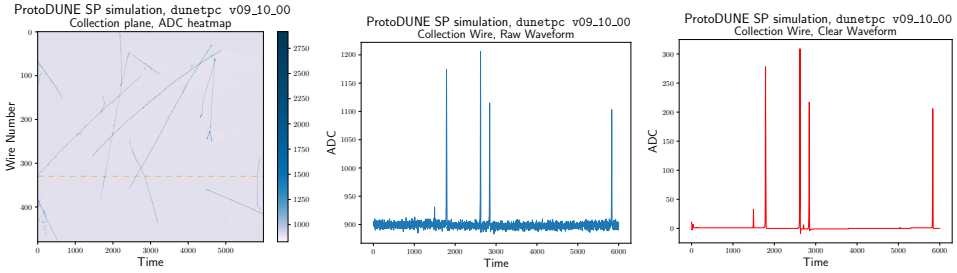

Figure 4. Example taken from `dunetpc v09_10_00` dataset. Collection plane view with noisy and clear waveforms.

Table 1 describes the considered datasets. In figures 3 and 4 we show visual examples from `v08_24_00` and `v09_10_00` datasets. We plot raw digits images and horizontal slices, namely single channel waveforms, either with and without noise. Noise is mainly comprised by a pedestal value that changes across different datasets and a background noise that covers small peaks and modifies spikes amplitudes. `v08_24_00` dataset is smaller and contains easier features to denoise and segment than the `v09_10_00` one: we consider it as a simplified dataset.

The latter 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 in figure 6 is clearly visible that the label waveforms show low frequency negative tails after big spikes. Such tails are really challenging regions for denoising neural networks.

We hold out from our datasets 10% 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 4.

3.2 Networks Training

The GCNN network suffers from the memory issue discussed in section 2. 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 hides some subtle problems due to the nature of the inputs, that may easily spoil the training process.

First, since plane views are big in size, being able to train simultaneosly on just a few hundreds out of thousands of small crops leads to high training times. To reduce the training per epoch time, we decide to train on just a subset of the available crops. This choice, in turn, triggers the second issue due to the sparsity of the inputs: sampling randomly the subset of crops to train on, gives an extremely unbalanced dataset and 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 noise pixel ratio in the original inputs. A crop is said to contain signal if, when sampling its center, it was originally labelled as a signal pixel.

We train the GCNN network in figure 1 for both image segmentation task and denoising, accounted by the ROI Block and the net final outputs respectively. In order 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. 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 further 50 epochs. Finally, we save the network best configuration.

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. In our experiments we train these networks using the mean square error loss function. We underline also that before feeding crops into the network we subtract the median value of the corresponding plane view in order to estimate and subtract the noise pedestal value. Median subtraction is a key operation that helps improving performances. We introduce this pre-processing procedure mimicking the traditional approach. After that, all network inputs are normalized to prevent training instabilities.

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 net fits a single 16 GB GPU even with an entire APA 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 pixels wide window and 1000 pixels stride and feed each slice to the network. The results are then combined together averaging predictions on overlapping regions. Like the GCNN network, the USCG-Net is trained minimizing the mean square error function between inputs and labels and applying the median subtraction receipt to training images.

4 Experiments results

In the present section we show the results of our experiments. We employ four different metrics to assess the goodness of our models and benchmark them against the state-of-the-art approach. We compute statistical structural similarity (Stat-SSIM) [16], peak signal to noise ratio (PSNR) and mean squared error (MSE) on the two dimensional raw digits. We observe that these quantites are really suited to compare the deep learning models between themselves, however they are not really informative when the baseline approach is considered.

The two dimensional deconvolution 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. Neural networks instead, being universal functions approximators, try to learn the real pixels intensity distribution without being constrained by a particular functional form. Hence the baseline tool performs inevitably poorly on the considered metrics. Nonetheless, we observe 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). This last metric captures the comparison between traditional and deep learning approaches.

Table 2.

Test metrics for denoising on `v08_24_00` dataset. Results are shown for collection plane and 2 GeV beam energy only.

	Stat-SSIM	PSNR	MSE	iMAE
Baseline	0.787 ± 0.009	39.97 ± 2.35	670 ± 378	5391 ± 1622
CNN	0.10 ± 0.01	67.3 ± 1.2	0.57 ± 0.03	287 ± 12
GCNN	0.163 ± 0.009	70.12 ± 1.4	0.30 ± 0.01	191.4 ± 2.6
USCG-Net (`v08_24_00` trained)	$\mathbf{0.915 \pm 0.004}$	$\mathbf{72.95 \pm 1.57}$	$\mathbf{0.15 \pm 0.02}$	90.1 ± 8.2
USCG-Net (`v09_10_00` trained)	0.012 ± 0.004	72.3 ± 1.5	0.18 ± 0.02	$\mathbf{76.3 \pm 8.2}$

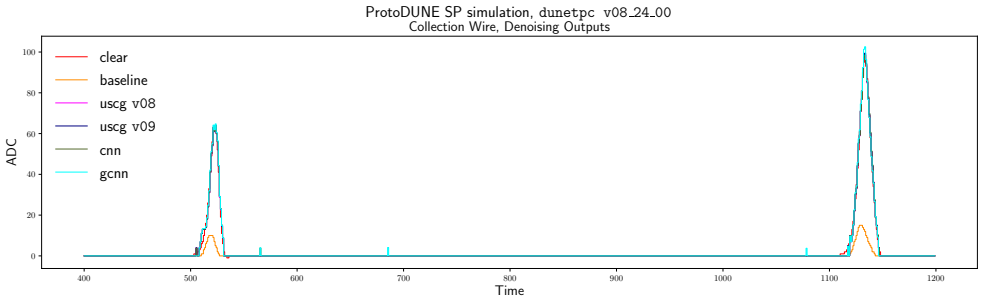

Figure 5. Detail of a raw waveform from `dunetpc v08_24_00` dataset: label, traditional algorithm and neural networks outputs.

Note that the deconvolution approach does not preserve waveform amplitudes, because a filtering function is applied to the signal in the Fourier space, then results are deconvolved back to time domain. Furthermore the deconvolution outputs are known up to an overall normalization constant, that we fit on the datasets in order to minimize the iMAE quantity. We show that, although we perform this operations 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 2 GeV beam energy events, since we find metrics distributions to be flat in the energy parameter. Figures 5 and 6 show samples of labels and denoised waveforms.

USCG-Net like networks exceeds GCNN-like ones in all the collected metrics. In order to fully assess 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 to 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 big dataset. We cross validate these networks on both datasets.

Following expectations, the networks trained and validated on the same dataset lead to better performances. The only exception is given in table 2, where USCG-Net trained on the more complex dataset achieves the best iMAE score. All the networks show an overall good generalization power, except for the Stat-SSIM score that does not always correlate with the other metrics. We underline that the networks are not trained according to this quantity; adding an extra term in the loss function containing the Stat-SSIM could be considered a point of further development of the present research.

Table 3.

Test metrics for denoising on `v09_10_00` dataset. Results are shown for collection plane and 2 GeV beam energy only.

	Stat-SSIM	PSNR	MSE	iMAE [$\times 10^3$]
Baseline	0.467 ± 0.011	39.6 ± 1.3	331 ± 42	5.86 ± 0.52
CNN	0.049 ± 0.002	57.3 ± 1.4	5.79 ± 0.88	4.16 ± 0.36
GCNN	0.043 ± 0.002	57.7 ± 1.5	5.27 ± 0.69	4.51 ± 0.39
USCG-Net (`v08_24_00` trained)	0.008 ± 0.003	59.4 ± 1.6	3.5 ± 0.4	2.69 ± 0.47
USCG-Net (`v09_10_00` trained)	$\mathbf{0.535 \pm 0.008}$	$\mathbf{61.8 \pm 1.7}$	$\mathbf{1.99 \pm 0.19}$	$\mathbf{2.25 \pm 0.23}$

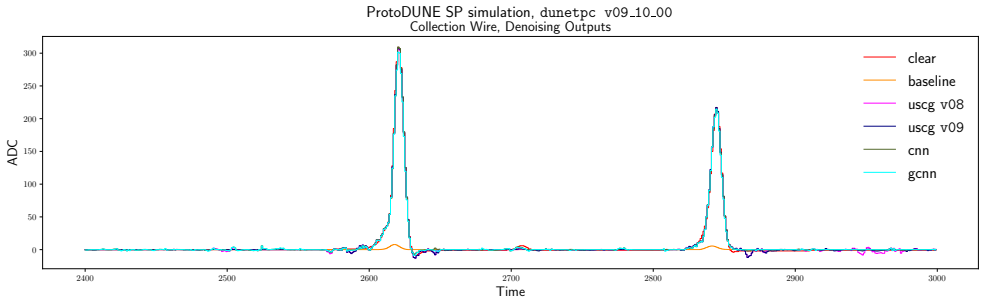

Figure 6. Detail of a raw waveform from `dunetpc v09_10_00` dataset: label, traditional algorithm and neural networks outputs.

5 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 pushed over the edge this approach, processing at once big size inputs while fitting GPU memory constraints. Our trained neural networks were able to outperform the state-of-the-art traditional reconstruction algorithm. All the networks showed good generalization power, achieving high performances even on datasets different to the one they had been trained on.

Acknowledgements

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

References

[1] L. Dominé, K. Terao, Physical Review D **102** (2020)

[2] B. Abi et al. (DUNE), Phys. Rev. D **102**, 092003 (2020), `2006.15052`

[3] X. Ju, S. Farrell, P. Calafiura, D. Murnane, Prabhat, L. Gray, T. Klijnsma, K. Pedro, G. Cerati, J. Kowalkowski et al., *Graph neural networks for particle reconstruction in high energy physics detectors* (2020), `2003.11603`

[4] B. Abi et al. (DUNE), JINST **15**, T08008 (2020), 2002.02967
[5] B. Abi et al. (DUNE) (2020), 2002.03005
[6] B. Abi et al. (DUNE), JINST **15**, T08009 (2020), 2002.03008
[7] B. Abi et al. (DUNE), JINST **15**, T08010 (2020), 2002.03010
[8] B. Abi et al. (DUNE) (2017), 1706.07081
[9] E.D. Church, *Larsoft: A software package for liquid argon time projection drift chambers* (2014), 1311.6774
[10] D. Valsesia, G. Fracastoro, E. Magli, *Deep graph-convolutional image denoising* (2019), 1907.08448
[11] D. Valsesia, G. Fracastoro, E. Magli, *Image denoising with graph-convolutional neural networks* (2019), 1905.12281
[12] M. Simonovsky, N. Komodakis, *Dynamic edge-conditioned filters in convolutional neural networks on graphs* (2017), 1704.02901
[13] Q. Liu, M. Kampffmeyer, R. Jenssen, A.B. Salberg, *Scg-net: Self-constructing graph neural networks for semantic segmentation* (2020), 2009.01599
[14] O. Ronneberger, P. Fischer, T. Brox, *U-net: Convolutional networks for biomedical image segmentation* (2015), 1505.04597
[15] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, *Aggregated residual transformations for deep neural networks* (2017), 1611.05431
[16] Z. Wang, A.C. Bovik, IEEE Signal Processing Magazine **26**, 98 (2009)