MACHINE LEARNING
Science and Technology

PURPOSE-LED PUBLISHING™

PAPER • OPEN ACCESS

Efficient Monte Carlo simulation of streamer discharges with deep-learning denoising models

To cite this article: F M Bayo-Muñoz *et al* 2025 *Mach. Learn.: Sci. Technol.* **6** 015036

IOP Publishing Mach. Learn.: Sci. Technol. 6 (2025) 015036 https://doi.org/10.1088/2632-2153/adaca1

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
24 July 2024

REVISED
20 December 2024

ACCEPTED FOR PUBLICATION
21 January 2025

PUBLISHED
11 February 2025

PAPER

Efficient Monte Carlo simulation of streamer discharges with deep-learning denoising models

F M Bayo-Muñoz[1,*], A Malagón-Romero[1,2] and A Luque[1]

[1] Instituto de Astrofísica de Andalucía (IAA), CSIC, Granada, Spain
[2] Centrum Wiskunde and Informatica (CWI), Amsterdam, The Netherlands
* Author to whom any correspondence should be addressed.

E-mail: manuel.bayo@iaa.csic.es

Keywords: streamers, monte carlo, particle-in-cell, deep learning, denoising

Abstract

Electric breakdown in non-conducting gases is a complex process that in its first stages is characterized by filamentary discharges called streamers. Streamer dynamics are inherently nonlinear and span broad temporal and spatial scales, making numerical simulation challenging. Although Monte Carlo methods are intuitive and they model the full electron energy distribution without *a priori* prescriptions, they suffer from artificial sampling noise which, combined with the non-linearity of streamers, distorts their evolution. Here we investigate the use of deep-learning techniques to mitigate the noise introduced by Monte Carlo sampling. We observe that traditional techniques for noise reduction in images are not satisfactory because they do not impose strict conservation of electric charge. Then we present a charge-conserving denoising filter to improve the efficiency of Monte Carlo simulations of streamers.

1. Introduction

The breakdown process by which a gas, initially exhibiting poor electrical conductivity, transforms into an efficient conductor under the application of a high electric field, is complex, nonlinear and remains inadequately understood. The initiation of a discharge typically involves a limited number of free electrons, which gain sufficient energy from the electric field to exceed the ionization potential of the gas molecules, subsequently liberating new electrons and triggering an exponential avalanche. This avalanche is self-limiting, as a high electron density eventually screens the electric field within the avalanche's core. If the system size is large compared with the size of this avalanche (this is typical, for example, in atmospheric-pressure, point electrode discharges [1]), the discharge evolves into a streamer—a sharply delimited continuously elongating filament featuring a shielded electric field internally and an intensified electric field at its tip, where persistent electron multiplication promotes the streamer's expansion [1, 2]. The inherently nonlinear and multi-scale characteristics of streamer discharges present challenges for numerical simulations, creating opportunities for machine learning techniques to enhance their modeling and analysis.

One computational technique that has been succesfully applied to streamer discharges (e.g. [3–5]) and, more generally, to the physics of ionized gases is the particle-in-cell with Monte Carlo collisions (PIC-MCC) technique [6]. It is based on sampling the full particle distribution function by a computationally feasible number of representative particles (called computational particles or super-particles) which are placed in a grid in order to compute their electrostatic long-range interaction (hence *Particle-In-Cell*) while simultaneously experiencing random collisions that instantaneously change their velocities (hence the *Monte Carlo Collisions*). Some advantages of this modelling approach are its simplicity in face of arbitrarily complex collision cross sections as well as its capability to reproduce, in the low-density regime, the inherent stochasticity of some physical processes. That is the case, when investigating streamers, of the effects of the finite number of photons emitted around the streamer tip that may induce the streamer to bifurcate [7].

A limitation of PIC-MCC techniques is the artificial noise introduced when sampling the full particle distribution function into a limited number of super-particles. When the coupling between particles and the

electric field is strongly nonlinear (as is the case for streamers) the artificial noise may lead to non-physical behaviour such as the premature bifurcation of streamers. The necessity of including enough super-particles to minimize this effect, with the corresponding demand for computational resources, has prevented a more widespread use of PIC-MCC.

Our work is partly motivated by one puzzling observation about streamers. In many cases they are accompanied by x-rays with photon energies reaching hundreds of electron-volts (eV) [8, 9]. This radiation indicates a potentially very small population of free electrons with energies several orders of magnitude above the mean electron energy (typically a few eV). In principle PIC-MCC, possibly complemented with the proper management of super-particle weights, is the most natural approach to model this phenomenon because it does not impose restrictions on the electron energy distribution function. We want however to avoid the computational costs inherent to the simulation of a streamer discharge with an acceptably low level of sampling noise.

Here we explore the approach of allowing the higher noise level of a relatively low number of super-particles but then mitigating the adverse effects of this noise by passing the noisy charge density through a denoising filter before computing the electrostatic field.

One simple denoising filter is the convolution of the noisy charge density with a properly normalized Gaussian. This straightforward filter offers a significant reduction in computational costs with a moderate sacrifice of accuracy. However it amounts to averaging over a window with a shape and width that are independent of the underlying streamer features. Given the wide separation in relevant length-scales exhibited by different streamer regions, we expect an improved performance for a denoising filter that is capable of adapting to the different areas of the streamers. Trainable neural networks are natural candidates on which to base this adaptable filter, as they have shown outstanding performance both in denoising and in feature detection in images.

The main result of this paper is describing some of the conditions that must be satisfied by such a denoising filter. As we describe below, the straightforward application of deep-learning denoising algorithms to the charge density in a streamer simulation offer outstanding levels of noise reduction but, when coupled with the streamer dynamics, they produce unphysical results. We attribute this failure to small but cumulative violation of charge conservation. Therefore we introduce a new deep learning architecture that ensures an exact conservation of electric charge. As this architecture is based on applying space-dependent convolutions to the original charge density, it is vulnerable to a convolutional instability and we discuss a regularization that mitigates this problem.

The architecture that we present can be naturally generalized to conserve any other physical quantity such as mass or momentum in a fluid, which in our view makes it valuable for future applications in physics-informed deep-learning applications.

2. Methods

Our objective is to build a PIC-MCC model with an embedded denoising model that allows us to simulate the evolution of a streamer with minimal distortion. In this pursuit we combine several elements that will be detailed in later sections of this work. The relations between the different elements are summarized in figure 1.

(i) *The PIC-MCC model* incorporates ballistic popagation of free electrons, collisions with molecules of the gas medium, locating particles in an uniform cartesian grid and the solution of the Poisson equation in order to compute the electric field at the position of each electron. The model also includes a particle management scheme that limits the number of super-electrons in each grid cell. This limitation, controlled by prescribed parameters, determines the noise level of the PIC-MCC simulation.

(ii) After assigning particles to each grid cell we apply a *denoising filter* before the resulting charge density is provided as an input to the Poisson solver.

(iii) Because PIC-MCC simulations with low noise levels are expensive, we also employ a fluid-based (noiseless) streamer model. We use this model for two purposes: (a) to initiate PIC-MCC simulations from the state of a fully developed streamer, thus sidestepping the problem of noise reduction in the earlier phases of evolution and (b) to build a library of streamer simulations that serves to train the denoising model and restrict its output to physically acceptable streamer states.

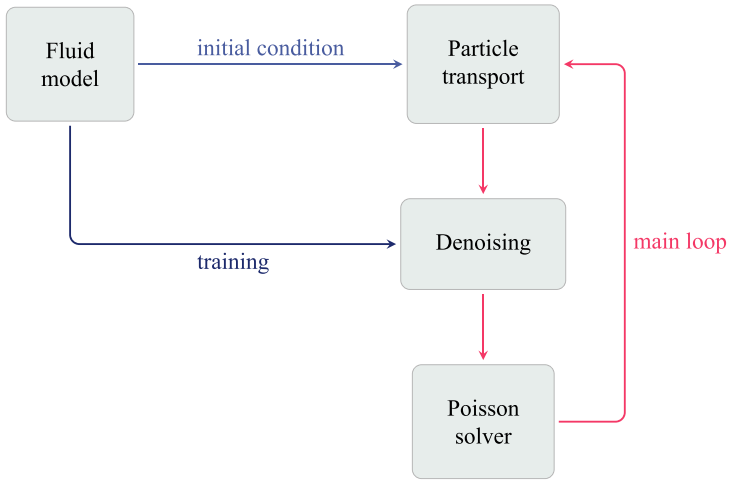

Figure 1. The different elements of our approach and the relations among them.

2.1. Physical models

2.1.1. Particle-In-Cell, Monte Carlo Model

PIC-MCC have been frequently used to investigate the physics of streamers (e.g. [3–5, 10–12]) and here we mostly follow standard practices in the field of Low-Temperature Plasma Physics. A PIC-MCC simulation [6] follows the position and velocities of a representative sample of the total particle population, assigning a weight to each computational particle. The particle dynamics alternates between ballistic propagation (where charged particles are accelerated by the electric field) and instantaneous scattering by collision events with molecules of the background gas. At fixed intervals the particle locations are tallied into a grid of cells and the resulting charge density is used to compute the self-consistent electric field. Our model is 2.5D, which means that the particle positions and velocities are three-dimensional vectors but the charge density is computed in a cylindrically symmetrical (i.e. two-dimensional) grid.

We include electrons and photons but ions, which are too slow to influence the streamer dynamics, are treated as immobile and only contribute to the charge density. For the cross sections for electron-molecule collisions, we take those of Phelps [13, 14], retrieved from the LxCat database [15–17] and assume an isotropic distribution of outgoing velocities. We use a simplectic fourth-order scheme [18–20] to integrate the electron propagation between collisions. The Poisson equation is solved by means of geometric multigrid [21] in an uniform cartesian mesh. At the radial boundaries of the domain we impose free boundary conditions to the electrostatic potential [22] whereas the upper and lower boundaries act as infinite planar electrodes. All our simulations are performed in a homogeneous cartesian grid in cylindrical coordinates with dimension $R \times L = 1.5 \times 10\,\mathrm{cm}$ and cell sizes $\Delta z = \Delta r = 9.8\,\mu\mathrm{m}$. The time step for the integration of particle trajectories and the re-computation of the electric field is 1 ps. A typical simulation of the PIC-MCC model involves from 2×10^8 to 2×10^9 super-electrons and a similar number of photons and the running time varied from a few days to two weeks using 32 computer cores.

Two aspects of our model deserve some attention as relevant in the context of noise reduction.

(i) *Particle management.* To prevent the number of simulated particles to become unwieldly, at uniform time intervals we resample the particle population inside each computational cell. Let P be the number of super-particles inside the cell. We set a maximal (P_M) and a target (P_T) number of super-particles per cell.

If the number of super-particles inside the cell is too large ($P > P_\mathrm{M}$) we choose P_T representative particles among the population underlying the P super particles. Let w_i ($1 \leqslant i \leqslant P$) be the weight of particle i and define normalized cumulative weigths as

$$c_i = \frac{\sum_{j=1}^{i} w_j}{\sum_{j=1}^{P} w_j}. \tag{1}$$

The sampling is performed by drawing P_T random numbers ξ_k uniformly distributed in the unit interval and selecting (possibly repeated times) particle i if $c_{i-1} < \xi_k < c_i$ (defining $c_0 = 0$). Note that it is not necessary to generate and store all ξ_k at the same time: given ξ_k we can generate $\xi_{k+1} = \xi_k + \beta_k(1 - \xi_k)$,

where β_k is a random number sampled from the probability distribution of the minimum of $P_\mathrm{T} - k$ random variables uniformly distributed in the unit interval. This can be computed as

$$\beta_k = 1 - (1-\zeta)^{1/(P_\mathrm{T}-k)}, \tag{2}$$

with ζ being a random variable uniformly distributed in the unit interval. This allows us to re-sample particles without bias using only two passes over super-particles (one to count particles and compute the total weight per cell, one to draw the ξ_k and select particles). Finally, we give each of the newly sampled particles a weight

$$w' = \frac{1}{P_\mathrm{T}} \sum_{j=1}^{P} w_j. \tag{3}$$

On the other hand, if the number of super-particles in a cell is too low ($P < P_\mathrm{T}$) we split a maximum of $P_\mathrm{T} - P$ super-particles randomly selected among those with weight larger than two. In low-density regions, where many particle weights are unity, we may still end up with less than P_T super-particles. After each splitting the initial weight is divided equally between the two resulting super-particles although to favor integer weights one of them is rounded to the closest integer. For example, a super-particle of weight 3.5 will be divided in two particles of weight (1.5, 2) or (1, 2.5), chosen at random.

(ii) *Photoionization.* In a streamer discharge in air, energetic electrons excite nitrogen molecules which then quickly give away this energy by emitting an ultra-violet photon which can ionize an oxygen molecule. Because photons travel much faster than the characteristic velocities in a streamer, this photoionization process effectively introduces a degree of non-locality in the physics of streamers. On the other hand, the number of photoionization events is significantly lower than impact ionization events so photoionization introduces additional stochasticity in the propagation of streamers, associated with streamer branching [7, 23].

We consider photoionization in our PIC-MCC code in a similar manner as proposed by Teunissen and Ebert [4]. We include a photo-excitation cross section which, following Zhelezniak *et al* [24], we approximate as proportional to the ionization cross sections of nitrogen and oxygen. For example, for O_2, the photo-excitation cross section is $\sigma_{\text{photo-exc},O_2} = \mathrm{A}\,\eta\sigma_{\text{impact-}O_2}$, where $\sigma_{\text{impact-}O_2}$ is the cross section of impact ionization of O_2. Note that this cross section does not model the underlying physical process, where the relevant excitation is only of N_2 but Zhelezniak's model considers photoemission as proportional to the total impact ionization, including N_2 and O_2. The factor η accounts for the efficiency of excitation and the probability of quenching at atmospheric pressure. For the first, we use 6×10^{-2} as representative of the values in table 1 of [24], for the second we take the quenching pressure to be $p_q = 4 \times 10^{-2}$ bar [24] so the probability of emission without quenching is $p_q/(p_q + 1\,\mathrm{bar}) = 0.038$, resulting in $\eta = \times 10^{-3}$. The scaling factor A reduces the noise associated with the relatively low probability of photo-emission: it increases the number of photoexcitations but when a particle of weight w induces a photoexcitation the number of photons emitted is sampled from a Poisson distribution with mean w/A (all photons have unit weight). Here we use $A = 10\times^3$; as this is much smaller than typical particle weights, stochastic noise is not reduced beyond what is physically expected.

Finally, the absorption rate per unit length for each photon, χ, is sampled such that $\log\chi$ is uniformly distributed between $\log\chi_\mathrm{min}$ and $\log\chi_\mathrm{max}$ with $\chi_\mathrm{min} = 1.09 \times 10^{22}\,\mathrm{m}^2 \times [O_2]$ and $\chi_\mathrm{max} = 6.21 \times 10^{-21}\,\mathrm{m}^2 \times [O_2]$, derived from [24] by assumming a temperature of 300 K, and where $[O_2]$ is the oxygen number density.

2.1.2. Fluid model

In a fluid-based approach, the dynamics of electrons and ions in a streamer discharge can be described by the drift-diffusion-reaction equations:

$$\frac{\partial n_\mathrm{e}}{\partial t} + \nabla\cdot(-n_\mathrm{e}\mu_\mathrm{e}\mathbf{E} - D_\mathrm{e}\nabla n_\mathrm{e}) = S_\mathrm{ph} + C_\mathrm{e}, \tag{4a}$$

$$\frac{\partial n_\mathrm{i}}{\partial t} = S_\mathrm{ph} + C_\mathrm{i}, \tag{4b}$$

where n_e, and n_i are, respectively the electron and ion densities, μ_e, D_e, and C_e denote the electron mobility, diffusion coefficient, and chemical source term, C_i is the source term for ions and $\mathbf{E}$ represents the electric field. As in the PIC-MCC model described above, ion mobilities and diffusion coefficients are neglected as

their motion is significantly slower compared to the motion of electrons. The term S_{ph} denotes the photoionization and is computed according to Zheleznyak's model [24] using a Helmholtz approach [25, 26], with the parameters of the three-exponential fit performed by Celestin [26]. This term only applies to electrons and O_2^+ as explained in section 2.1.1.

The interactions between electrons and gas molecules in the fluid model are based on the same processes and cross sections as the PIC-MCC described above. The streamer propagates in an air admixture of 79% N_2 and 21% O_2 at 300 K and 1 atm. The chemical model includes electron-impact ionization, dissociative attachment and three-body attachment. Electron transport coefficients and reaction coefficients depend on the local reduced electric field (E/N_{air}, where N_{air} is the number density of molecules in air), and they have been computed from Phelps' cross sections [13] with BOLSIG+ [27], a solver for the steady-state Boltzmann equation that is widely used in low-temperature plasma research.

The electric field is computed as $\mathbf{E} = \mathbf{E}_{bg} - \nabla\phi$, with $\mathbf{E}_{bg}$ being the background electric field, and ϕ the electrostatic potential that relates to its sources according to the Poisson equation

$$\nabla^2\phi = -\sum_s \frac{q_s n_s}{\varepsilon_0}, \tag{5}$$

where ε_0 is the vacuum permittivity, q_s is the electric charge of the species s and the sum extends over all the charged species (electrons and ions).

We use finite volume methods (FVMs) [28] to solve equations (4). In particular, our code is built on top of CLAWPACK/PETCLAW [28–30] a library of FVMs. PETCLAW relies on PETSc [31–33] and allows us to split the simulation domain into different subdomains that can be solved in parallel. The Poisson equation (5) is solved using the Generalized Minimal Residual method and the geometric algebraic multigrid preconditioner, both from the PETSc numerical library.

2.2. Streamer dataset

One use of the fluid model was to generate datasets of streamer discharges to train the denoising models. These datasets contain pairs of noisy samples (input) and their corresponding noiseless data (ground truth).

2.2.1. Noiseless samples

The noiseless samples were generated with the fluid model described in section 2.1.2, which was used to simulate the propagation of single positive and negative streamers. The simulation domain is 2D axisymmetric (r, z) with the same dimensions and cell sizes as for the PIC-MCC model above: a length of 10 cm and a radial extension of 1.5 cm and a uniform spacing of size $\Delta z = \Delta r = 9.8\,\mu$m. This domain was bounded by two grounded and infinite planar electrodes, with free boundary conditions applied at the radial boundary for the electrostatic potential [22]. For the different densities, homogeneous Neumann boundary conditions were used on all boundaries.

The streamer propagation is initiated with a neutral ($n_e = n_{O_2^+} + n_{N_2^+}$) needle-like seed attached to the lower electrode, with its tip located at $z_0 = 2$ cm, and with an electron density given by

$$n_e = n_{e,bg} + n_{e0} \exp\left(-\frac{\max(z - z_0, 0)^2}{w^2} - \frac{r^2}{w^2}\right), \tag{6}$$

where $n_{e,bg}$ denotes a uniform background electron density and $n_{e0} = 2.2 \times 10^{20}\,\mathrm{m}^{-3}$ is the peak electron density of a Gaussian distribution with e-fold radius $w = 1$ mm.

We simulated the same number of positive and negative streamers, adding up to a total of 90 streamer discharges. These simulations account for five logarithmically-spaced background electron densities $n_{e,bg}$, ranging from 10^{10} to $10^{14}\,\mathrm{m}^{-3}$ and 9 evenly-spaced background electric fields E_{bg}, ranging from 10 to $30\,\mathrm{kV\,cm}^{-1}$. We include both positive and negative streamers. The choice of ranges aim at covering relevant values for the simulation of streamer discharges [2]. The simulations resulted in a set of nearly 2000 samples, corresponding to the temporal evolution of streamers with the initial conditions and parameters described above. This initial set was cut down to 1895 samples by removing those where the streamer discharge showed boundary effects due to its proximity to the upper electrode. The number of noiseless samples was chosen as a compromise between dataset size and dense coverage of the explored parameter space.

2.2.2. Noisy samples

Given a noiseless electron density (anologously a noiseless ion density) from the fluid model, its noisy counterpart is generated by adding synthetic noise, aiming to emulate the noise resulting from a PIC-MCC simulation. The actual probability distribution of the particle densities sampled by a PIC-MCC simulation is complex and entails correlations between different cells. To have a tractable probability distribution we

instead use multiplicative white Gaussian noise applied independently to the electron and ion densities. Thus, the noisy density of species s in a given cell with a noiseless density n_s is

$$\tilde{n}_s = \left(1 + \frac{\varrho_s}{\sqrt{p}}\right) n_s, \tag{7}$$

where the ϱ_s are independent random values sampled from a normal distribution with a mean of 0 and a standard deviation of 1 and p modulates the amplitude of the density fluctuations. Density fluctuations for a given value of p are roughly of the same size as those in a MC simulation where $p \approx P_{\mathrm{T}} \approx P_{\mathrm{M}}$. So as representatives for the number of particles per cell that we want to consider, in our training we employed $p = 5, 10, 50$ for the U-Net filter and $p = 10$ for the charge-conserving one (see below).

2.3. Initialization of the PIC-MCC model

As mentioned above, we avoid the streamer-initialization phase in the PIC-MCC model by initializing it with the state of a fluid simulation. This entails converting from a fluid (continuous) description to a representation based on super-particles.

We consider the target number of super particles per cell P_{T} introduced in 2.1.1. For each species s (electrons or positive ions) and for each cell in the domain we compute a target particle weight as

$$w_s = \max\left(1, \frac{n_s \Delta V}{P_{\mathrm{T}}}\right), \tag{8}$$

where ΔV is the volume of the cell. The final number of particles of species s in the cell is then sampled from a Poisson distribution with mean

$$\mu_s = \frac{n_s \Delta V}{w_s}, \tag{9}$$

which matches P_{T} in densely populated regions. The weight for all the super-particles is w_s and they are initialized at rest. As the time-scale for electron energies to reach equilibrium is much shorter than the evolution time-scale for a streamer, we consider that the perturbation introduced by this zero-energy initialization plays a minor role in our results.

2.4. Denoising models

2.4.1. Scaling of input data

We consider only axially symmetrical streamers. To apply a noise-reduction scheme we start with the charge density evaluated in each cell of our uniform cartesian mesh, q_{ij} ($i = 1 \ldots M, j = 1 \ldots N$). This charge density represents a total charge per cell $Q_{ij} = 2\pi\, q_{ij} r_i \Delta r \Delta z$ where $r_i = (i - 1/2)\Delta r$. Therefore we consider a more natural choice to take as input data $\tilde{q}_{ij} = (i - 1/2) q_{ij}$, which is proportional to Q_{ij}. Besides, measuring q_{ij} in $\mathrm{C\,cm^3}$ produces quantities of order unity so we opted for using q_{ij} in those units without further rescaling.

2.4.2. Denoising U-Net

There are many types of deep-learning architectures for noise reduction, but most of them are based on the concept of *autoencoder*. Several works gave rise to the autoencoder [34–36], but it was Kramer in 1991 [37], who clearly and concisely detailed its structure as we know it today. Although in their origins autoencoders were only used to obtain compressed or low-dimensional representations of the input data for subsequent reconstruction, nowadays autoencoders are used in multiple tasks, such as identification of anomalies, generation of new data, feature extraction, recommendation systems, watermark removal and noise reduction [38]. An autoencoder mainly consists of two parts (see figure 2):

(i) *The encoder* maps the input data into a low-dimensionality representation, generally known as a latent space, bottleneck, or simply code.
(ii) *The decoder* reconstructs the original input data from the latent space representation.

A *denoising autoencoder* [39, 40] is a particular type of autoencoder that is configured and trained to reconstruct a clean or repaired version of some corrupted input data. Training a denoising autoencoder is carried out by minimizing the difference between the original and the reconstructed data. With this training, one conditions the denoising autoencoder to produce outputs from the same distribution as the input data, which generally allows it to outperform traditional filters for noise removal. In our use case, we anticipate an autoencoder to develop a preference for configurations of the streamer discharge similar to those in our training set and thus more consistent with the physics of the discharge.

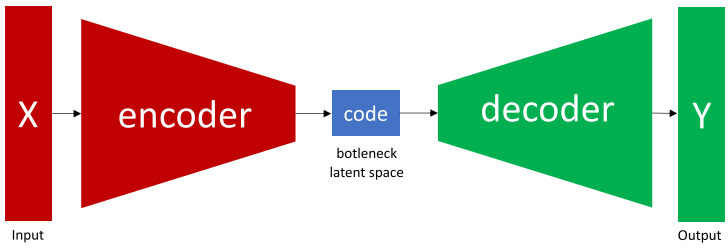

Figure 2. General Autoencoder structure. The input data X is mapped by the encoder into a representation with a lower dimensionalty (code or bottleneck, residing in the latent space). Using the information contained in the bottleneck, the decoder attempts to restore the input data.

Despite of their good performance, models based on the autoencoder architecture can sometimes lose during the compression (or encoding) stage certain details or characteristics of the input data that may be relevant when reconstructing the data in the decoding stage. To reduce this information loss and overcome this problem we explore another deep learning architecture called U-Net [41]. Similarly to the autoencoder, the U-Net consists of a contraction stage (encoder with convolutional and downsampling layers) and an expansion stage (decoder with deconvolutional and upsampling layers). However an U-Net additionally includes data connections between the contraction and expansion paths called *skip connections*. This enables the network to propagate relevant context information directly from the contraction to the expansion path, providing the later layers with more original information to use, increasing the number of feature channels. The use of skip connections entails symmetrical contraction and expansion paths, producing the characteristic U shape that lends its name to the architecture.

Streamer discharges involve widely separated spatial scales and U-Nets are known to excel at preserving spatial hierarchies and feature details so they seem to be better suited for our task. We confirmed this by comparing the performance of different configurations of autoencoders and U-Nets applied to our particular problem. Therefore, we take a U-Net and not an autoencoder as one of the baseline models for our results. The architecture and training parameters are detailed in appendix B.

2.4.3. Charge-conserving denoising model

Anticipating our results, described in section 3, we find that, even though U-Nets perform exceptionally in noise reduction, they are unsuitable for our puroposes since coupled to the PIC-MCC code they result in unphysical streamer dynamics. We attribute this to the lack of exact charge conservation.

To mitigate this problem we designed a novel type of convolutional neural network architecture which ensures exact charge conservation at the expense of lower performance in noise reduction. This model works by identifying local features in the input data and applying different charge-conserving convolutions for each feature type. For example, different kernels act around the sharp gradients in the streamer head and in the smooth region far ahead of it.

Given the original scaled discrete charge density $\tilde{q}_{ij}$ and a kernel of size $(2l+1)\times(2l+1)$ with elements $K_{\alpha\beta}$ $(\alpha,\beta=-l\ldots l)$, the result of the convolution of these two is

$$\tilde{q}'_{mn}=\sum_{\substack{i=m-l\ldots m+l,\\ j=n-l\ldots n+l}}\tilde{q}_{ij}K_{m-i,n-j}, \tag{10}$$

where we momentarily leave aside the results of this convolution close to the domain boundaries, discussed below. The total charge is

$$\sum_{\substack{m=1\ldots M,\\ n=1\ldots N}}\tilde{q}'_{mn}=\sum_{\substack{i=m-l\ldots m+l,\\ j=n-l\ldots n+l}}\sum_{\substack{m=1\ldots M,\\ n=1\ldots N}}K_{m-i,n-j}\tilde{q}_{ij}=\sum_{\alpha\beta}K_{\alpha\beta}\sum_{mn}\tilde{q}_{m-\alpha,n-\beta}=\sum_{\alpha\beta}K_{\alpha\beta}\times\sum_{m,n}\tilde{q}_{mn}. \tag{11}$$

Therefore, charge conservation holds if the kernel elements add up to unity:

$$\sum_{\substack{\alpha=-l\cdots+l,\\ \beta=-l\cdots+l}}K_{\alpha\beta}=1. \tag{12}$$

We additionally impose that the kernel elements are non-negative, $K_{\alpha\beta}\geq 0$. This facilitates stability and enables us to interpret the convolution in (10) as charge redistribution within a rectangle with the kernel size $2l+1$.

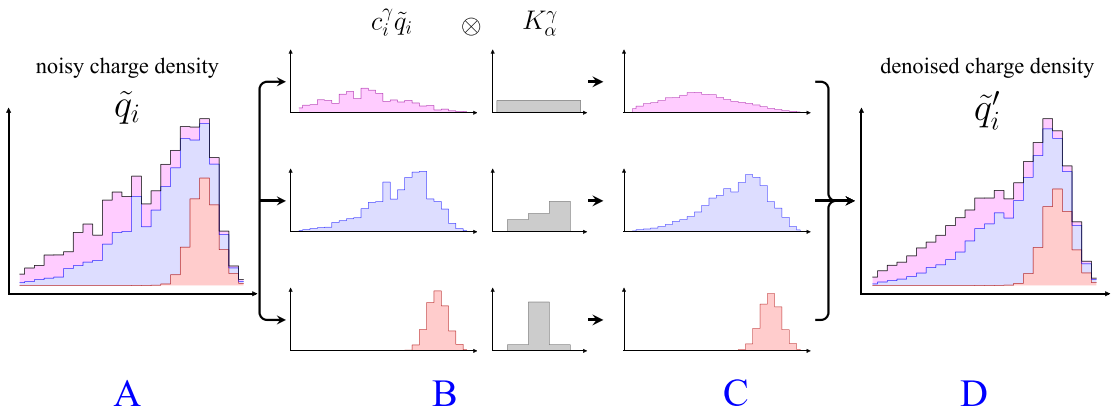

Figure 3. Charge-conserving denoising model. The noisy charge density from a PIC-MCC code $\tilde{q}_i$ (A) is partitioned by space-dependent coefficients c_i^γ and convolved with charge-preserving kernels K_α^γ (B). The results (C) are added back together (D) into a denoised charge density $\tilde{q}_i'$.

Let us now consider the convolution close to the domain boundaries. There are multiple ways to extend input values $\tilde{q}_{ij}$ to cells out of the domain by padding with ghost cells. However, a general kernel that satisfies (12) does not in principle lead to charge conservation in all these cases. One possible approach would be to impose symmetrical padding together with symmetrical kernel elements ($K_{\alpha\beta} = K_{-\alpha\beta} = K_{\alpha,-\beta}$). However we consider that approach too restrictive given the asymmetry of our problem with respect to inversions of the z or r axes.

Instead, we (a) used zero padding to ensure that no fictitious external charge is introduced into the domain and (b) we add back the charge that exits the domain. To accomplish the latter, we modify equation (10) close to the boundary. Consider the behaviour close to $m = 1$; in that case we add to $\tilde{q}'_{mn}$ the term corresponding to $\tilde{q}'_{1-m,n}$:

$$\tilde{q}'_{mn} = \sum_{\substack{i=m-l\ldots m+l,\\ j=n-l\ldots n+l}} \tilde{q}_{ij} K_{m-i,n-j} + \sum_{\substack{i=1-m-l\ldots 1-m+l,\\ j=n-l\ldots n+l}} \tilde{q}_{ij} K_{1-m-i,n-j}. \tag{13}$$

So far we have described a filter involving a single kernel but the charge-conserving convolution can be generalized and extended to use S independent kernels $K_{\alpha\beta}^\gamma$ with $\gamma = 1 \ldots S$. To achieve that, we partition the charge inside each cell ij into S terms with space-dependent coefficients:

$$\tilde{q}_{ij} = \sum_{\gamma=1\ldots S} \tilde{q}_{ij}^\gamma = \sum_{\gamma=1\ldots S} c_{ij}^\gamma \tilde{q}_{ij}, \tag{14}$$

where

$$c_{ij}^\gamma \geqslant 0, \tag{15a}$$

$$\sum_\gamma c_{ij}^\gamma = 1. \tag{15b}$$

Then (10) is updated as

$$\tilde{q}'_{mn} = \sum_{\substack{i=m-l\ldots m+l,\\ j=n-l\ldots n+l}} \sum_{\gamma=1\ldots S} c_{ij}^\gamma \tilde{q}_{ij} K_{m-i,n-j}^\gamma. \tag{16}$$

One way to ensure that the constraints in (15) are satisfied is to define the space-dependent coefficients as

$$c_{ij}^\gamma = \frac{e^{b_{ij}^\gamma}}{\sum_\gamma e^{b_{ij}^\gamma}}, \tag{17}$$

where b_{ij}^γ are in general non-linear functions of all the input charges $\{\tilde{q}_{ij}\}$.

During training, the charge-conserving neural network learns the kernel coefficients $K_{\alpha\beta}^\gamma$ as well as the mapping from charges $\tilde{q}_{ij}$ to the charge-partitioning coefficients c_{ij}^γ. The latter can in general make use of any

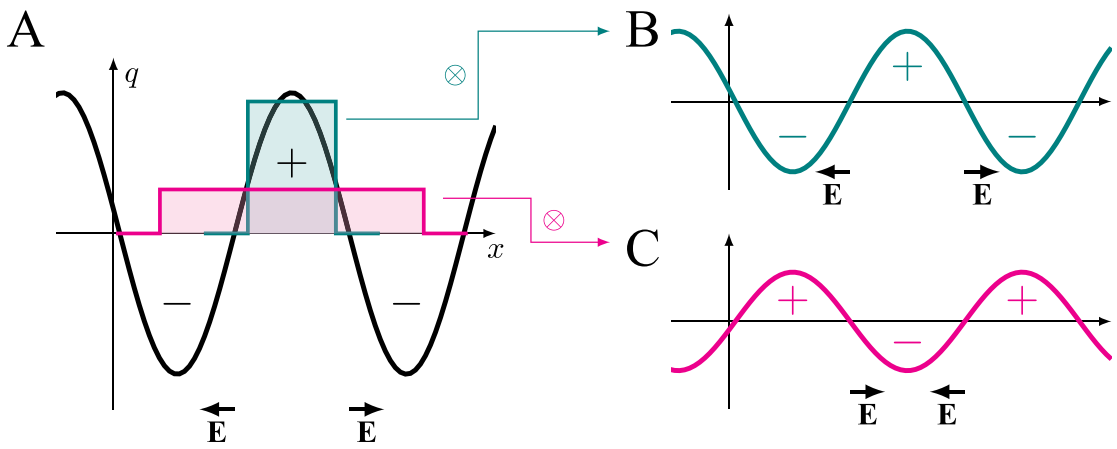

Figure 4. Convolutional instability in a denoised streamer simulation. Consider here a one-dimensional sinusoidal mode in the charge density q (panel (A)). The resulting electric field peaks at the location of nodes in the charge distribution and has the directions indicated in the figure. As the electric current follows the direction of the field, the amplitude of the mode decreases and the dynamics is stable. Now consider that, in an attempt to reduce noise, we apply a convolution as indicated by the boxcar windows in panel (A). A narrow window leads to the situation in panel (B): the amplitude is changed but otherwise the sign of the charge density remains unaltered. Applying this convolution maintains the system's stability. A wider convolution window produces the outcome of panel (C), where the sign of the charge density is inverted. The resulting electric field points in directions that will amplify the mode amplitude and the system becomes unstable. As discussed in the text, this happens for wavenumbers for which the real part of the Fourier transform of the kernel is negative.

convolution-based, deep architecture. Figure 3 contains schematic representation of the charge-conserving denoising procedure for one-dimensional problems. The architecture used in our study is detailed in appendix C.

Our charge-conserving scheme is reminiscent of *kernel density estimation* (KDE) methods [42] and indeed (16) can be reformulated in terms of a KDE where the kernel depends on the particle location. The novelty of our formulation with respect to other KDE approaches is that we define the kernel shapes in terms of trainable kernel coefficients $K^{\gamma}_{\alpha\beta}$ and charge-partitioning coefficients c^{γ}_{ij} which are obtained from a trained convolutional neuronal network that accounts for the context of each cell.

2.4.4. Convolutional instability

A major limitation of a streamer denoising scheme based on convolutions is the appearance of a convolutional instability, which we describe qualitatively in figure 4. Consider the interior of the streamer, not very close to the streamer boundary, and thus far from steep gradients in the electron density. In that region the streamer behaves like a electrical conductor with conductivity $\sigma = e\mu_e n_e \approx e\mu_e n_+$, where e is the elementary charge, μ_e is the electron mobility and n_e and n_+ are the electron and positive ion number densities. The evolution of the charge density q is given by

$$\frac{\partial q}{\partial t} = -\nabla \cdot (\sigma E)\,. \tag{18}$$

Combined with $\nabla \cdot E = q/\epsilon_0$ and taking, as first approximation, σ as homogeneous, we obtain

$$\frac{\partial q}{\partial t} = -\frac{\sigma}{\epsilon_0} q, \tag{19}$$

which states that a perturbation in the charge density relaxes with a time scale ϵ_0/σ.

Now consider the case where the electric field is computed after convolving the charge density with a kernel K,

$$\nabla \cdot E = \frac{1}{\epsilon_0} q \otimes K. \tag{20}$$

In that case equation (19) reads

$$\frac{\partial q}{\partial t} = -\frac{\sigma}{\epsilon_0} q \otimes K. \tag{21}$$

Let us now decompose this equation into Fourier modes and consider the space and time dependence of q as $q = q_0 \exp(\mathrm{i}\mathbf{k}\cdot\mathbf{r} - \mathrm{i}\omega t)$. Noting that a convolution maps to a product in Fourier space we reach the

following dispersion relation:

$$\mathrm{i}\omega(\mathbf{k}) = \frac{\sigma}{\epsilon_0}\tilde{K}(\mathbf{k}), \tag{22}$$

where $\tilde{K}(\mathbf{k})$ is the Fourier transform of the convolution kernel. The mode is stable (i.e. dos not grow exponentially) if $\mathrm{Re}(\mathrm{i}\omega) > 0$, which, given (22), implies $\mathrm{Re}(\tilde{K}(\mathbf{k})) > 0$. Any kernel with a Fourier transform that reaches negative values for some $\mathbf{k}$ included in the simulation is numerically unstable. Furthermore, the growth rate of the unstable mode is proportional to $|\mathrm{Re}(\tilde{K}(\mathbf{k}))|$.

Two issues are worth commenting at this point:

(i) In our case we deal with an axially symmetric charge density $q(r,z)$. As described in section 2.4.1, we operate with 2D kernels and on quantities that are proportional to rq. In that case equation (21) still holds after the transformation $q \to rq$.
(ii) The Fourier transform of a Gaussian kernel is also a Gaussian, which is everywhere real and positive. Therefore a convolution with a Gaussian kernel is immune to this convolutional instability.

In our experiments we found that indeed a charge-conserving denoising scheme that is trained only to minimize the mean square error (MSE) of the predicted charge density produces an oscillating charge density when in the streamer simulations. To prevent this, we defined a regularization loss as

$$L_C = \frac{\alpha_C}{L^2 S}\sum_{ij\gamma}\mathrm{ReLU}\left(-\tilde{K}^{\gamma}_{ij}\right), \tag{23}$$

where S is the number of kernels (defined above), ReLU is defined as $\mathrm{ReLU}(x) = x$ if $x > 0$ and 0 otherwise, $\tilde{K}^{\gamma}_{ij}$ is the 2D Discrete Fourier Transform of the kernel K^{γ} after padding it with zeros to a size $L \times L$ and α_C is a weighting factor for the regularization term. We used $L = 128$, $\alpha_C = 10^2$.

3. Results

We run fluid (noiseless) simulations of a negative streamer discharge propagating in air under standard temperature and pressure (STP, 1 atm, 273.15 K) with an applied electric field of 15 kV cm^{-1}. At 18 ns, the streamer is fully developed and its charge density is represented in the leftmost panel of figure 5. Note that we did not take any measure to exclude this initial condition from our training set. We consider overfitting a minor concern, as we test the application of the denoising schemes to a later time, when the streamer has diverged from the state in the fluid simulation.

At this point we switch to a PIC-MCC simulation with the procedure described in section 2.3 to translate the fluid density into super particles. This procedure introduces high-amplitude noise as shown in the second panel of figure 5, in that case with a target of $P_{\mathrm{T}} = 40$ super-particles per cell.

The PIC-MCC simulation then evolves, possibly applying one of the denoising algorithms described above. The two rightmost panels of figure 5(CC for charge-conserving) show the outcome of these denoising algorithms acting on the the noisy charge density of the second panel. As we anticipated, the U-Net architecture apparently performs well in reducing the noise and inferring the original, noiseless density. The charge-conserving architecture, on the other hand, suffers from the imposed constraint whereby charges can only be re-arranged within a window of limited size; this is most visible in the small-amplitude charge fluctuations in the streamer body.

3.1. U-Net simulation

We focus first on the results of a PIC-MCC simulation coupled with the denoising U-Net architecture described above. For a simulation with a target number of particles per cell $P_{\mathrm{T}} = 40$ and maximal number of particles per cell $P_{\mathrm{M}} = 60$ figure 6 shows the magnitude of the electric field on the streamer axis as well as the charge density integrated in cross sections of the streamer.

Despite the good noise-reduction performance of U-Net exhibited in figure 5, when coupled with the PIC-MCC simulation, imprecisions in inferring the underlying charge density accumulate over time. The charge carried by the streamer, plotted in the lower panel of figure 6 grows at a faster rate than the natural evolution of the streamer, increasing by about 10% in 0.1 ns. The electric field also increases and reaches 150 kV cm^{-1}. This evolution in such a short time-scale is unrealistic for a streamer and is not seen in either fluid simulations or in low-noise PIC-MCC simulations.

We attribute this failure of the U-Net noise reduction model to the absence of strict charge conservation. Apparently, even a weak bias in the inference of charge density leads to unphysical conditions and prevents the application of an U-Net architecture to our problem.

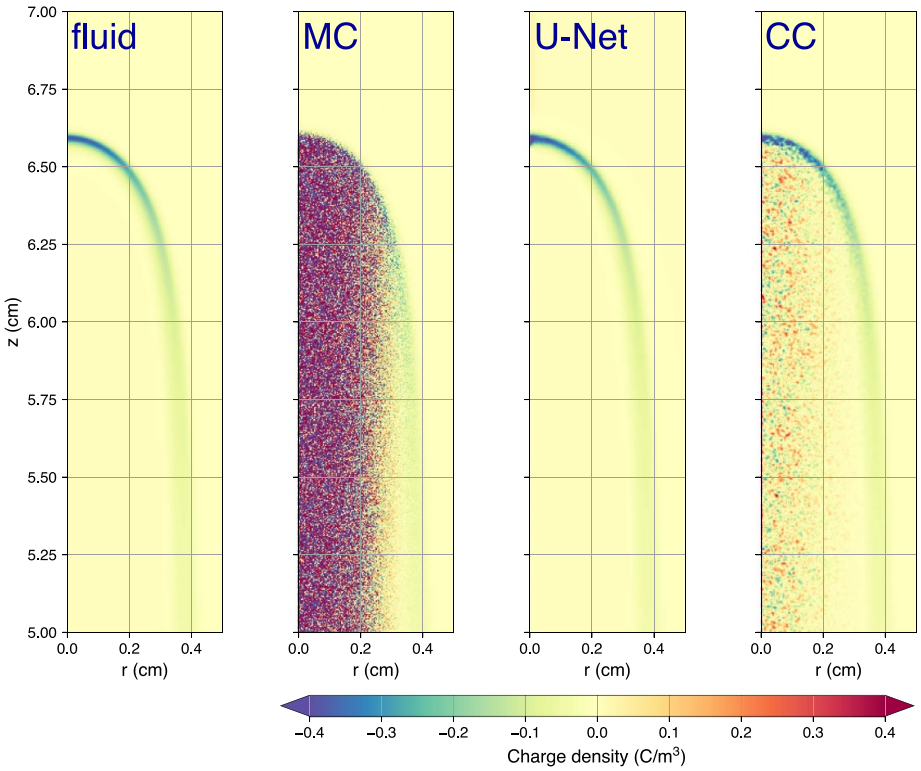

Figure 5. Charge density in a negative streamer head at different stages of our algorithm and under two denoising schemes. The leftmost panel shows the charge density resulting from a fluid (noiseless) simulation. The second panel (MC) represents the charge density resulting from converting the fluid density into a set of particles with a target of 40 super-particles per cell. Note that in this panel many of the values are outside the limits of the colormap. The third panel (U-Net) contains the noise-less charge density inferred from the noisy density by means of the U-Net architecture. The fourth, rightmost panel (CC) contains the inference from the charge-conserving architecture.

Although not shown here, we have tested tens of modifications to the basic U-net architecture. This includes adding a penalty for the violation of charge conservation into the training loss function. However, all these trials resulted in an slightly over- or under-estimated charge density that lead to clear deviations from the expected streamer dynamics.

3.2. Charge-conserving simulation

The shortcomings of the U-Net approach led us to develop the charge-conserving deep-learning scheme described in section 2.4.3. To test its performance we run simulations with a relatively low number of super-particles per cell ($P_T = 10$, $P_M = 15$) and compare them with a simulation without any denoising scheme but a large number of particles per cell ($P_T = 360$, $P_M = 440$). We also compare with a Gaussian filter with a standard deviation of 29.4 μm, which in our tests performed the best among Gaussian filters. We remind the reader that the Gaussian filter also conserves charge exactly and that it is stable against the instability described in section 2.4.4.

The result, shown in figure 7 indicates a better performance of the deep-learning model. A major difference is visible around the streamer boundary, the reason being that the propagation speed of the deep-learning model matches better that of the reference simulation. We note however that there is significant variability in the propagation speed between different simulations with the same denoising model, presumably due to random differences in the initial translation from the fluid density to a particle description. The lower noise in the streamer interior exhibited by the machine learning simulation is a more consistent result in all our simulations.

In figure 8 we show the evolution of the streamer simulations close to the streamer axis. In this plot we also include results from a simulation without any denoising filter but the same number of particles per cell as the two denoising simulations. The main conclusion of figure figure 8 is that adding a denoising filter is almost as effective as increasing the number of particles in a simulation but at a lower computational cost.

It is also clear that the charge-conserving denoiser performs better than the U-Net presented in the previous section. The simulation does not show signs of obviously unphysical behaviour.

Figure 7 also shows that the deep-learning model performs better than the Gaussian filter. As mentioned above, part of this difference can be attributed to differences in the sampling of particles in the conversion

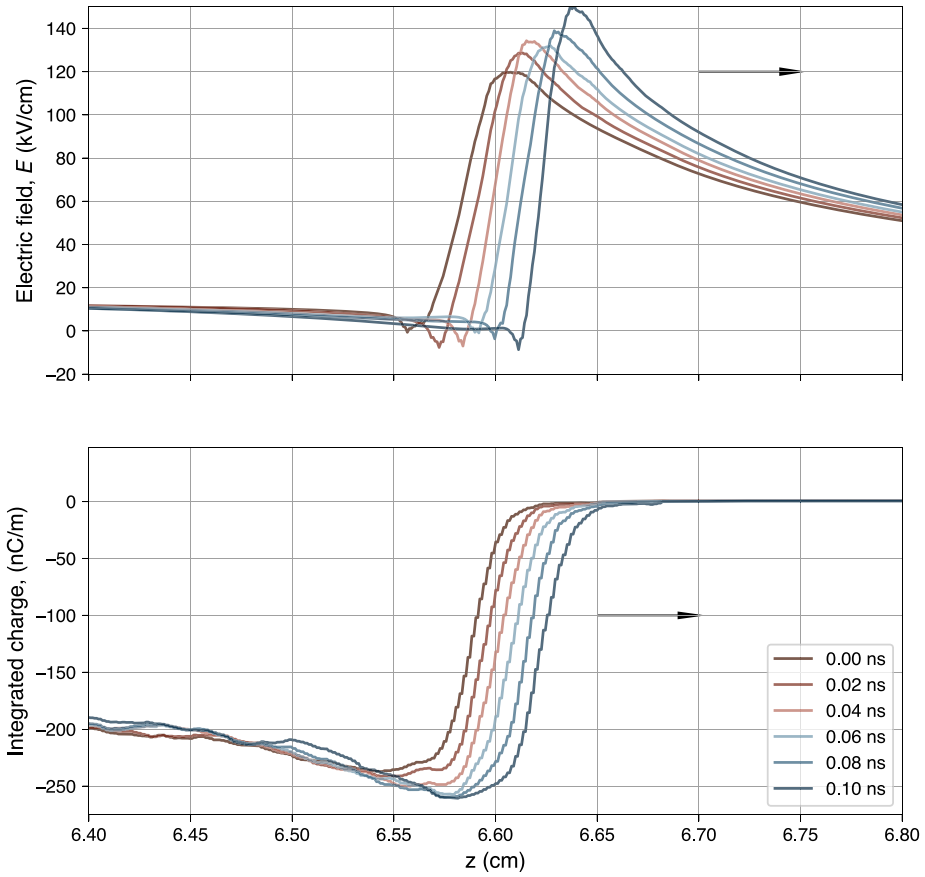

Figure 6. Evolution of the axial electric field (above) and the cross-sectional charge density (below) in the PIC-MCC model coupled with a U-Net. Shown are snapshots of the two variables at intervals of 20 ps after the conversion from fluid densities to particles. The arrows indicate the direction of evolution.

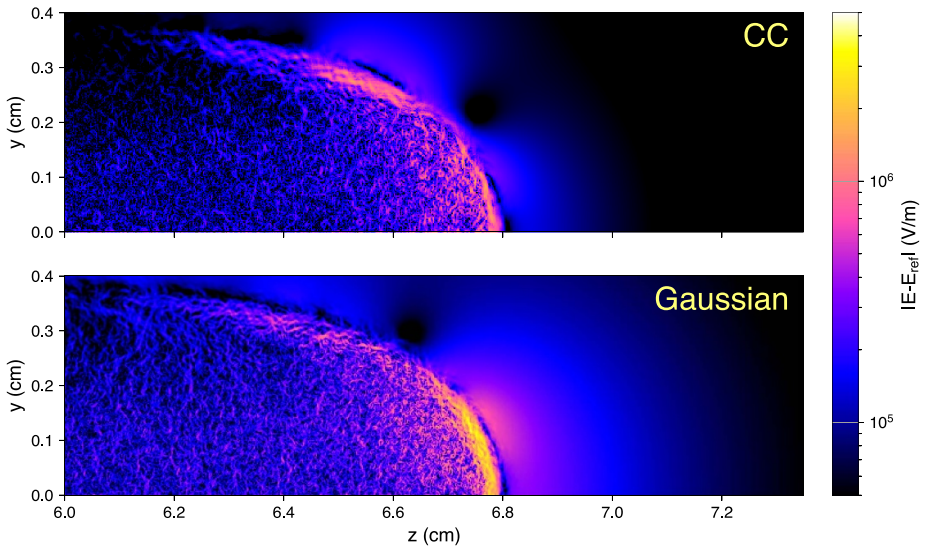

Figure 7. Comparison of our deep-learning charge-conserving denoising scheme (upper panel) and a Gaussian filter (lower panel). Both simulations are compared against a low-noise reference simulation with a large number of particles per cell. In each panel we plot the norm of the difference between the electric field resulting from the denoising and the reference simulations. We plot here snapshots 0.6 ns after the switch to a particle-based simulation.

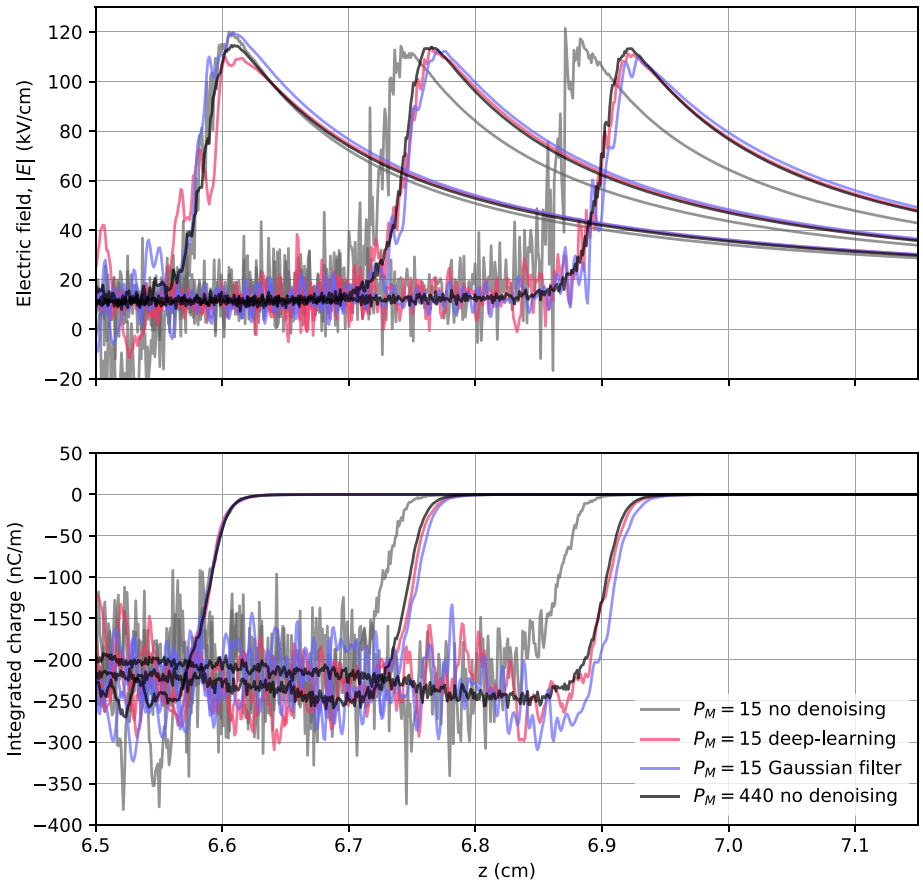

Figure 8. Evolution of the axial electric field (above) and the cross-sectional charge density (below) in the PIC-MCC model coupled with the charge-conserving denoising scheme. We show snapshots at the time of conversion from fluid to particle descriptions as well as 0.5 ns and 1 ns later. The results are compared with a simulations without denoising and a simulation with where charge is locally averaged within a fixed-size window.

from a fluid density. Nevertheless the deep-learning model exhibits a lower noise amplitude, especially visible around the streamer edge.

4. Discussion and concluding remarks

The strong nonlinearity of streamers and their high sensitivity to small deviations of charge conservation turn these electrical discharges into valuable testing grounds for physically-informed noise-reduction schemes. We have shown that a direct application of conventional deep-learning techniques for noise reduction in 2D images, even if it performs remarkably well in restoring the charge distributions of the test set, still introduces a bias that, when coupled with the streamer evolution, produces unphysical results. Although not presented here, we tried several corrections and extensions to our U-Net architecture and training conditions, including the use of a penalty for violating charge conservation in the loss function. None of these modifications achieved a significantly better performance than the case that we present and all of them led to unphysical states within a fraction of a nanosecond.

These observations motivated us to develop a scheme that ensures exact charge conservation in the denoising stage. The outcome in this case is significantly better, producing a stable simulation that closely reproduces the reference results but at a lower computational cost. The results of this model improve over a simpler Gaussian convolution. Additionally, in terms of running time, our simulations are dominated by the particle advancement: the computational cost associated with either Gaussian convolution or deep-learning inference are negligible. However, at this point we hesitate on recommending the use the deep-learning denoising filter, as the increased software complexity and training costs may not compensate for its benefits.

Nevertheless our results indicate potential future applications of deep-learning denoising schemes in highly demanding numerical simulations, such as streamer simulations. Here we point to the flexibility of the deep-learning charge-conserving denoiser, which allows the splitting coefficients to be derived from arbitrary network architectures. This suggests future improvements in our architecture such as replacing it with other schemes that have proven useful for image segmentation such as the HR-Net of Sun *et al* [43].

Our approach to noise reduction complements existing particle managing methods that reduce the variance of selected outcomes in Monte Carlo simulations [44]. In this work we limited ourselves to a relatively simple particle management where we aimed for an approximately homogeneous signal to noise ratio of particle densities (section 2.1.1). Alternatively, one may consider different objectives in the particle management, such as improving momentum and energy conservation [45] or preferentially sampling higher-energy electrons [46]. However, the trade off between noise amplitude and computational requirements is an inherent feature of all PIC simulations. By mitigating noise without a significant penalty in computational cost, our approach would improve the terms of this trade off.

Other approaches have been proposed for the reduction of noise in PIC simulations. The so-called δf-method decomposes the particle distribution function into a known or easily computable f_0 and a perturbation δf, with computational particles representing only the δf component [47–49]. The strong nonlinearity and sensitivity of streamer dynamics likely hinder the application of the δf method, though we have not extensively explored this approach. Another approach, based on KDE with machine-learned, optimized kernels was proposed by Wu and Qin [50]. Although the application of that KDE scheme to streamer simulations is a possible avenue of research, the method of Wu and Qin does not account for the different dynamics and length-scales that characterize the different areas surrounding a streamer (within the channel, at the leading edge, in the volume ahead of the tip), which call for various smoothing scales.

Another relevant question is whether any of the existing image denoising algorithms can replace the U-Net or the charge-conserving scheme that we have discussed. This question is addressed briefly in A. While an exhaustive comparison of denoising schemes is beyond the scope of this work, we conclude that existing methods, primarily designed for processing visual images, are unlikely to serve as effective starting points for streamer simulations.

We must, nevertheless, be aware of the limitations of our method. The main issue is a restricted training set: although we aimed at including a large range of input conditions and streamer development stages, one can image physical conditions far from these conditions were the denoising results are questionable. Examples are stagnating [51] or branching streamers [7]. Another issue is the possible existence of additional biases in the denoising filter. Although we have applied our scheme to a realistic setting, before it can be included in a standard toolbox of streamer simulations it needs to undergo further optimizations and testing for conditions farther away from the training set.

The method proposed here joins the portfolio of machine-learning or data-driven approaches for the modeling of low-temperature plasmas, recently reviewed by Trieschmann *et al* [52]. These approaches offer potentially more efficient numerical algorithms but they are also prone to unknown biases and difficulty in interpretation. Our work exemplifies these problems by underlining the relevance of exact charge conservation in streamer simulations.

We conclude by noting that Monte Carlo methods are widely used to investigate systems subjected to conservation laws in physics. Their application is however limited by artificial sampling noise when the number of particles in a physical system exceeds the capacity of a computer. Deep-learning denoising filters and, in particular, filters that ensure conservation of relevant physical quantitites such as the one presented here open the possibility of preserving the advantages of Monte Carlo approaches with acceptable noise at an affordable computational cost.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Code availability

All the code used for this work is publicly available in the following repositories. PIC-MCC code for streamers: https://zenodo.org/doi/10.5281/zenodo.12764437 [53]; U-Net model: https://zenodo.org/doi/10.5281/zenodo.12805602 [54]; Charge-conserving deep-learning model: https://zenodo.org/doi/10.5281/zenodo.12759303 [55]. Training data can be requested by email to the authors.

Acknowledgments

This work was supported by the Spanish Ministry of Science and Innovation, under Project PID2022-136348NB-C31 and by the Junta de Andalucía under Project P20-00831. A Malagón-Romero acknowledges the financial support from the grants BEVP34S12931 and BEVP34A6840 funded by 'Ramón Areces Foundation'. We also acknowledge support from the State Agency for Research of the Spanish MCIU

through the 'Center of Excellence Severo Ochoa' award for the Instituto de Astrofísica de Andalucía (CEX2021-001131-S).

Appendix A. Other image denoising algorithms

Image denoising is a widely studied task due to its practical applications, leading to the development of numerous algorithms. A recent review can be found in the work by Fan *et al* [56].

In principle, these algorithms can be applied also to the task of denoising the charge density in a streamer simulation. In practice, however, the best-performing image-denoising algorithms are tailored to the reduction of noise in photographic images, with priors and noise distributions that differ markedly from those required in a streamer simulation.

As an example, we consider the *block-matching and 3D filtering* (BM3D) method proposed by Dabov *et al* [57]. This method scans the image for correlated patches, arranges them in a 3D array and filters out high-frequency spectral features, which are assumed to emerge from noise. Because the spectral transform is performed along an axis perpendicular to the image plane, this method effectively preserves sharp image features. BM3D and some of its descendants are considered among the best denoising schemes for photographic images.

However, in its application to a streamer simulation, two issues immediately become apparent. First, BM3D assumes an homogeneous noise distribution through the image (assumes independent, additive Gaussian noise) and requires as input the variance of such distribution to be used as threshold in the spectral filtering. The noise in a PIC-MCC simulation, however, presents strong inhomogeneities in the noise variance. The second issue is that, because BM3D requires a scanning of the image for the patch grouping, it requires significant computations that may dominate the running time of a simulation.

Figure A1 shows the results of applying BM3D to a PIC-MCC streamer simulation. Comparing with figure 5, we see that BM3D performs noticeably worse than both the U-Net and the charge-conserving denoiser discussed in our work. Particularly, the assumption of homogeneous variance leads to a division of the simulation volume into areas where noise is either not reduced or reduced too strongly. The resulting axial electric field, also represented in figure A1 is also poorly estimated.

An exhaustive investigation of the performance of the image-denoising methods described in the literature [56] is out of the scope of our work. The poor performance of BM3D on streamer simulations

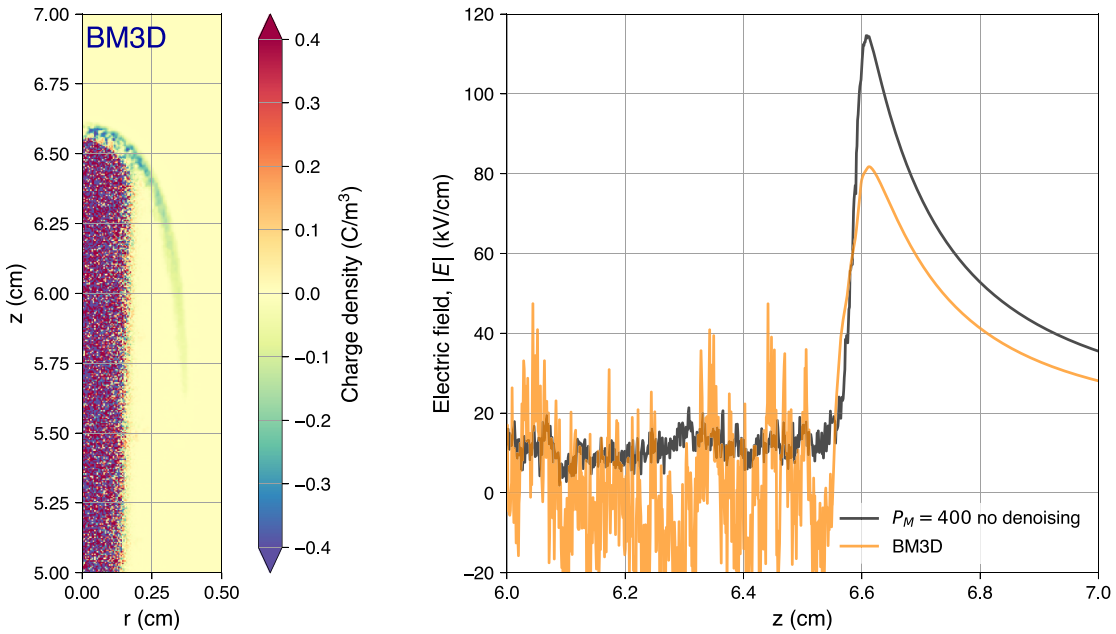

Figure A1. Application of the BM3D image denoising algorithm [57] to a streamer simulation. Here we show the application to a PIC-MCC charge distribution of the BM3D algorithm with a variance threshold $\sigma^2 = 1\,\mathrm{C}^2\,\mathrm{m}^6$. The left panel contains the denoised charge density; there we notice that fixed variance threshold of BM3D combined with an inhomogeneous noise variance distribution leads to areas where noise persists at its original level. In our experiments, increasing the variance threshold to reduce noise in this area leads to erasing the charge density in the streamer edge. The right panel plots the axial electric field resulting from the charge distribution in the left panel. Compared with a reference simulation ($P_\mathrm{M} = 400$, no denoising; see main text), we notice that BM3D severely underestimates the electric field.

indicates that noise-reduction algorithms optimized for real-world images are not well-suited even as starting points for this specific problem.

Appendix B. U-Net architecture and configuration

B.1. Architecture

To obtain the model that performs best in our Monte Carlo simulation, we have carried out multiple tests with several configurations of denoising autoencoders and U-Nets until reaching the configuration illustrated in B2. Its structure is as follows:

(i) *Contraction Path*. This part of the network is responsible for transforming the input data into a low-dimensional representation of them, obtaining the bottleneck. It consists of 5 stages or blocks, with the following layers in each one:

- Conv2D(filters = num_filters, kernel_size = 3×3, strides = 1×1, padding = same)
  The value of the num_filters parameter is [4, 8, 16, 32, 64], depending on the stage we are in, as can be seen in the figure B2.
- AveragePooling2D(pool_size = 2) + GroupNormalization(groups = num_filters) + ReLU

(ii) *Expansion Path*. Once we have the bottleneck, we do the inverse process through the expansion path to get the denoised streamer. For this task we have also implemented 5 stages, the first 4 being composed of the following layers:

- Upsampling2D(size = 2, interpolation = nearest)
- Conv2D(filters = num_filters, kernel_size = 3×3, strides = 1×1, padding = same)
  + GroupNormalization(groups = num_filters) + ReLU
- Concatenate the previous result with the corresponding symmetric contraction stage. This operation attempts to recover relevant details of the streamer that were possibly lost during downsampling.

and the last one, designed to obtain a streamer of the same size as the original and with a single channel, is composed of:

- Upsampling2D(size = 2, interpolation = nearest)
- Conv2D(filters = 1, kernel_size = 1×1, strides = 1×1, padding = same)

Our code is written in Python and based on the TensorFlow [58] and Keras [59] frameworks.

B.2. Hardware

The training of the models has been carried out on a computing server whose main hardware components are the following:

- AMD Epyc 7H12 CPU with 128 cores at 2.60GHz,
- 1TB of RAM,
- 1 GPU NVidia A100 40GB PCIe.

B.3. Dataset partitioning

We start from the dataset of 1895 streamers generated as described in section 2.2.1. For each original streamer snapshot we synthesize 3 images with noise levels $p = 5, 10, 50$ (see section 2.2.2), resulting in 5685 samples. To follow a holdout validation procedure, the data is split into three subsets: training (3979 samples = 70%), validation (1137 samples = 20%) and test (569 samples = 10%). During the training process, the training and validation subsets have been used, reserving the test subset for a final model evaluation.

B.4. Training meta-parameters

These are the most relevant parameters of the training process:

Epochs: 1000 maximum
Early Stopping Callback: 50 epochs with no improvement in validation loss (val_loss)
Optimizer: Adam(learning_rate = 0.001)
Learning Rate Callback: The value of learning_rate is halved if there is no improvement in val_loss for 10 consecutive iterations, reaching a minimum value of 1×10^{-5}

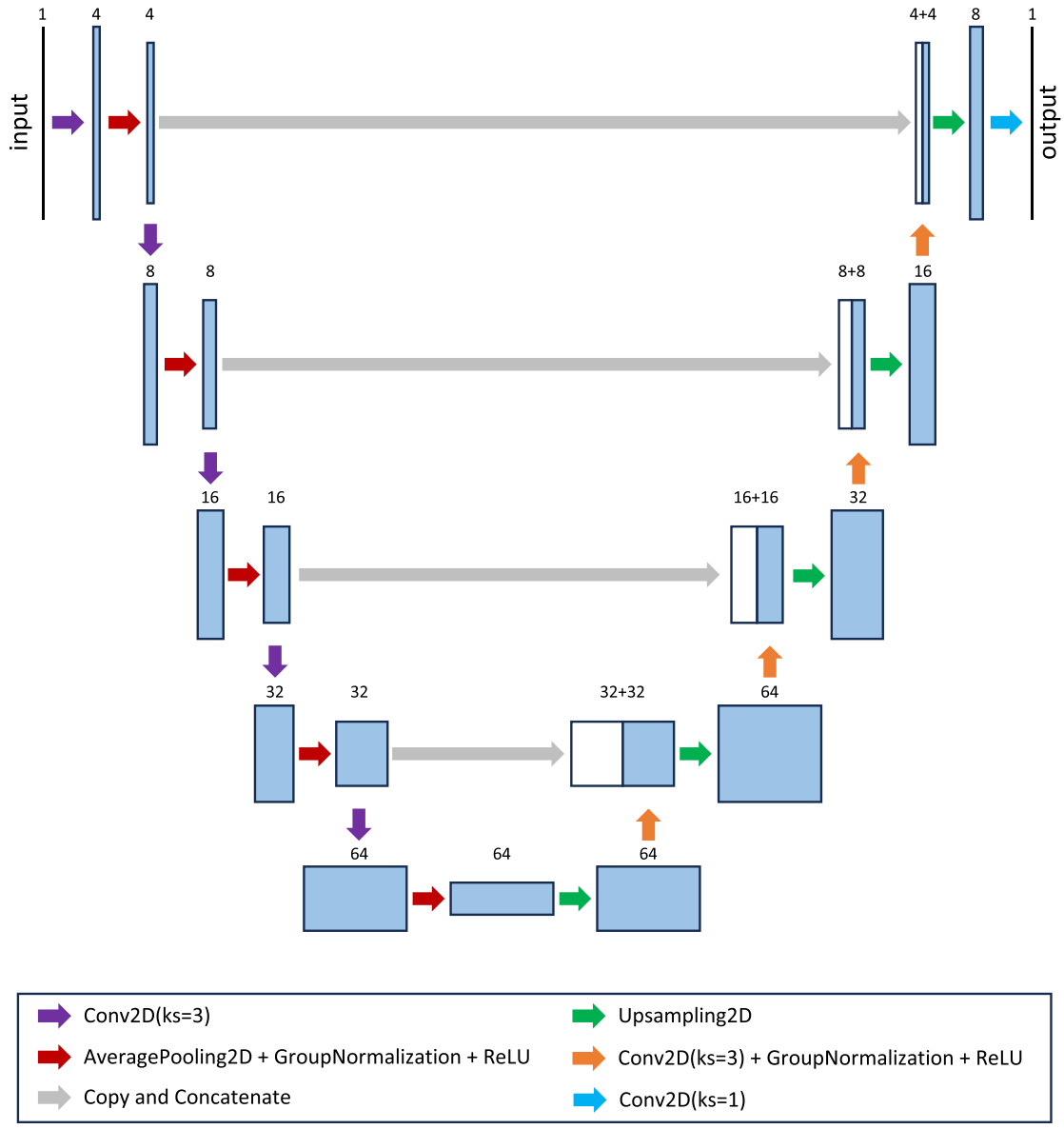

Figure B2. U-net implemented for streamer denoising.

Loss function: Mean Squared Error (MSE)
Batch size: 8
Number of filters used on each stage: 4, 8, 16, 32, 64

Appendix C. Charge-conserving neural network architecture and configuration

C.1. Architecture

Figure C1 shows the architecture employed in our charge-conserving scheme. The first component deals with computing the partitioning coefficients c_{ij}^{γ} for each cell i,j. This is accomplished by a series of three convolutions supplemented by activation layers. The two first convolutions are supplemented with a concatenation of the input layer.

- Conv2D(filters = 8, kernel_size = 9×9, strides = 1×1) + ReLU,
- Conv2D(filters = 16, kernel_size = 9×9, strides = 1×1) + ReLU,
- Conv2D(filters = 32, kernel_size = 9×9, strides = 1×1) + ReLU.

The output of this layer is normalized (so that it has zero mean and unit variance) and then passed to a softmax layer to compute (17).

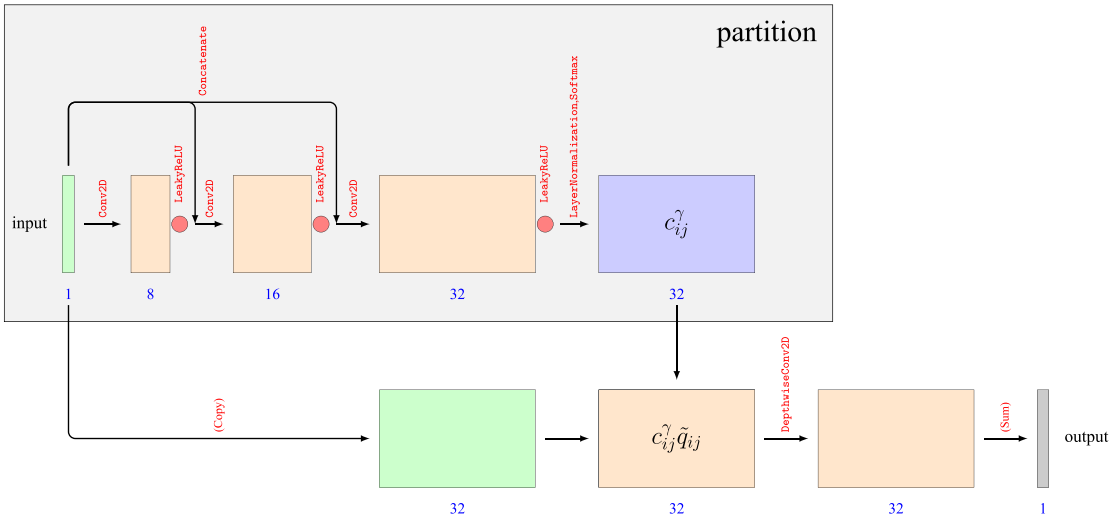

Figure C1. Scheme of the charge-conserving denoising network. We describe the main layers and operations using the notation of the keras framework ('Copy' and 'Sum' are implemented by `Conv2D` with untrainable parameters). Under the layers we show the dimensionality in feature space. The shaded area labelled as 'partition' marks the section of the model that computes the partition coefficients c_{ij}^{γ}. As we discuss in the text, this network admits arbitrary topologies but we show the one employed in this work.

The second component of the network takes these coefficients and multiplies them by the input, thus obtaining 32 charge-density components. Each of these components is then convolved with a different 13×13 kernel and the result is summed along the feature dimension to obtain the output.

C.2. Loss function

The loss function that is minimized in the training of the charge-conserving network is based on the MSE but incorporates some modifications that in our experience improve its performance:

(i) Because the noise-reduction performs worse around the symmetry axis, instead of using the charge density scaled with radius, $\tilde{q}_{ij} = (i - 1/2)q_{ij}$ (see section 2.4.1), we directly employed the un-scaled charge density q_{ij}.

(ii) To prevent the large, almost-empty volume surrounding the streamer from dominating the loss, we also weighted more strongly those regions with a significant charge density. This is implemented with a weighting factor $(1 + \bar{q}_{ij}^2/w^2)$, where $\bar{q}_{ij}$ is the reference (ground truth) charge density and w is a parameter that we set to $2.5 \times 10^{-2}\,\mathrm{C\,m^{-3}}$.

The error-based loss function then reads

$$L_E = \frac{1}{MN} \sum_{1 \leqslant i \leqslant M; 1 \leqslant j \leqslant N} \left(q_{ij} - \bar{q}_{ij}\right)^2 \left(1 + \frac{\bar{q}_{ij}^2}{w^2}\right). \tag{C1}$$

To this loss function we add the regularization loss L_C defined in (23) to suppress the convolutional instability, as discussed in section 2.4.4.

C.3. Hardware

The hardware used to train the charge-conserving network is the same as for the U-Net. Also here we employ the keras framework.

C.4. Dataset partitioning

For the charge-conserving network the dataset is randomly partitioned into 80% of training set and 20% for validation. Only data with $p = 50$ was used to train this network.

C.5. Training meta-parameters

These are the most relevant parameters of the training process:

Epochs: about 30,

Early Stopping: the training was stopped manually after about 10 epochs without improvement in the validation loss,

Optimizer: Adam,
Activation function: LeakyReLU with slope 0.1 for negative values,
Loss function: Mean Squared Error (MSE),
Batch size: 4,
Weight decay: 0.01.

ORCID iDs

F M Bayo-Muñoz https://orcid.org/0000-0001-8516-6322
A Malagón-Romero https://orcid.org/0000-0002-2891-9700
A Luque https://orcid.org/0000-0002-7922-8627

References

[1] Raizer Y P 1991 *Gas Discharge Physics* (Springer)
[2] Nijdam S, Teunissen J and Ebert U 2020 *Plasma Source Sci. Technol.* **29** 103001
[3] Chanrion O and Neubert T 2008 *J. Comput. Phys.* **227** 7222–45
[4] Teunissen J and Ebert U 2016 *Plasma Source Sci. Technol.* **25** 044005
[5] Bouwman D, Teunissen J and Ebert U 2022 *Plasma Source Sci. Technol.* **31** 045023
[6] Birdsall C K 1991 *IEEE Trans. Plasma Sci.* **19** 65–85
[7] Wang Z, Dijcks S, Guo Y, van der Leegte M, Sun A, Ebert U, Nijdam S and Teunissen J 2023 *Plasma Source Sci. Technol.* **32** 085007
[8] Østgaard N, Carlson B E, Nisi R S, Gjesteland T, Grondahl Ø, Skeltved A, Lehtinen N G, Mezentsev A, Marisaldi M and Kochkin P 2016 *J. Geophys. Res.* **121** 2939
[9] da Silva C L, Millan R M, McGaw D G, Yu C T, Putter A S, LaBelle J and Dwyer J 2017 *Geophys. Res. Lett.* **44** 174–11
[10] Zhang Q-Z and Bogaerts A 2018 *Plasma Source Sci. Technol.* **27** 035009
[11] Jiang M, Li Y, Wang H, Zhong P and Liu C 2018 *Phys. Plasmas* **25** 012127
[12] Köhn C, Dujko S, Chanrion O and Neubert T 2019 *Icarus* **333** 294–305
[13] Phelps A V and Pitchford L C 1985 *Phys. Rev.* A **31** 2932–49
[14] Lawton S A and Phelps A V 1978 *J. Chem. Phys.* **69** 1055
[15] Pancheshnyi S, Biagi S, Bordage M C, Hagelaar G J M, Morgan W L, Phelps A V and Pitchford L C 2012 *Chem. Phys.* **398** 148–53
[16] Pitchford L C *et al* 2017 *Plasma Proces. Polym.* **14** 1600098
[17] Carbone E, Graef W, Hagelaar G, Boer D, Hopkins M M, Stephens J C, Yee B T, Pancheshnyi S and van Dijk J and Pitchford L (LXCat Team) 2021 *Atoms* **9** 16
[18] Ruth R D 1983 *IEEE Trans. Nucl. Sci.* **30** 2669
[19] Forest E and Ruth R D 1990 *Physica* D **43** 105–17
[20] Forest E 2006 *J. Phys. A: Math. Gen.* **39** 5321–77
[21] Trottenberg U, Oosterlee C W and Schüller A 2001 *Multigrid* (*Texts in Applied Mathematics)* vol 33 (Academic) contributions A Brandt P. Oswald and K.Stbenu
[22] Malagón-Romero A and Luque A 2018 *Comput. Phys. Commun.* **225** 114–21
[23] Marskar R 2024 *J. Comput. Phys.* **504** 112858
[24] Zhelezniak M B, Mnatsakanian A K and Sizykh S V 1982 *High Temp. Sci.* **20** 357–62
[25] Luque A, Ebert U, Montijn C and Hundsdorfer W 2007 *Appl. Phys. Lett.* **90** 081501
[26] Bourdon A, Pasko V P, Liu N Y, Célestin S, Ségur P and Marode E 2007 *Plasma Source Sci. Technol.* **16** 656–78
[27] Hagelaar G J M and Pitchford L C 2005 *Plasma Source Sci. Technol.* **14** 722–33
[28] LeVeque R 2002 *Finite Volume Methods for Hyperbolic Problems* (*Cambridge Texts in Applied Mathematics)* (Cambridge University Press)
[29] Alghamdi A, Ahmadia A, Ketcheson D I, Knepley M G, Mandli K T and Dalcin L 2011 *Petclaw: A Scalable Parallel Nonlinear Wave Propagation Solver for Python* (*Proceedings of the 19th High Performance Computing Symposia HPC'11)* (Society for Computer Simulation International) pp 96–103 (available at: http://dl.acm.org/citation.cfm?id=2048577.2048590)
[30] Mandli K T, Ahmadia A J, Berger M, Calhoun D, George D L, Hadjimichael Y, Ketcheson D I, Lemoine G I and LeVeque R J 2016 *Peer J. Comput. Sci.* **2** e68
[31] Balay S *et al* 2023 PETSc Web page (available at: https://petsc.org/)
[32] Balay S et al 2023 PETSc/TAO users manual *Technical Report* ANL-21/39- Revision 3.19 Argonne National Laboratory
[33] Balay S, Gropp W D, McInnes L C, Smith B F Arge E, Bruaset A M and Langtangen H P 1997 *Efficient Management of Parallelism in Object Oriented Numerical Software Libraries* (*Modern Software Tools in Scientific Computing*) (Birkhäuser Press) pp 163–202
[34] Rumelhart D E, Hinton G E and Williams R J 1986 *Learning Internal Representations by Error Propagation* (*Parallel Distributed Processing: Explorations in the Microstructure of Cognition, volume 1: Foundations* (MIT Press) pp 318–62
[35] Ballard D H 1987 Modular learning in neural networks *AAAI Conf. on Artificial Intelligence* (available at: https://api.semanticscholar.org/CorpusID:38968420)
[36] Lecun Y 1987 Modeles connexionnistes de l'apprentissage (connectionist learning models) *PhD Thesis* Universite P et M Curie (available at: https://api.semanticscholar.org/CorpusID:151887454)
[37] Kramer M A 1991 *AIChE J.* **37** 233–43
[38] Bank D, Koenigstein N and Giryes R 2023 *Autoencoders* (Springer) pp 353–74
[39] Vincent P, Larochelle H, Bengio Y and Manzagol P A 2008 Extracting and composing robust features with denoising autoencoders *Int. Conf. on Machine Learning* (available at: https://api.semanticscholar.org/CorpusID:207168299)
[40] Vincent P, Larochelle H, Lajoie I, Bengio Y and Manzagol P A 2010 *J. Mach. Learn. Res.* **11** 3371–408
[41] Ronneberger O, Fischer P and Brox T 2015 arXiv:1505.04597
[42] Härdle W, Müller M, Sperlich S and Werwatz A 2012 *Nonparametric and Semiparametric Models* (*Springer Series in Statistics*) (Springer)

[43] Sun K, Zhao Y, Jiang B, Cheng T, Xiao B, Liu D, Mu Y, Wang X, Liu W and Wang J 2019 High-resolution representations for labeling pixels and regions (arXiv:1904.04514)
[44] Haghighat A 2016 *Monte Carlo Methods for Particle Transport* (CRC Press)
[45] Teunissen J and Ebert U 2014 *J. Comput. Phys.* **259** 318–30
[46] Schmalzried A, Luque A and Lehtinen N 2022 *Comput. Phys. Commun.* **277** 108366
[47] Cohen B I, Auerbach S P, Byers J A and Weitzner H 1980 *Phys. Fluids* **23** 2529–37
[48] Parker S E and Lee W W 1993 *Phys. Fluids* B **5** 77–86
[49] Aydemir A Y 1994 *Phys. Plasmas* **1** 822–31
[50] Wu W and Qin H 2018 *Phys. Plasmas* **25** 102107
[51] Niknezhad M, Chanrion O, Holboll J and Neubert T 2021 *Plasma Source Sci. Technol.* **30** 115014
[52] Trieschmann J, Vialetto L and Gergs T 2023 *J. Micro/Nanopat. Mater. Metrol.* **22** 041504
[53] Luque A 2024 mcstreamer (v1.1) Zenodo (https://doi.org/10.5281/zenodo.12764438)
[54] Bayo-Muñoz F M 2024 streamers_U-net (v1.1) Zenodo (https://doi.org/10.5281/zenodo.12805603)
[55] Luque A 2024 conserv (v1) Zenodo (https://doi.org/10.5281/zenodo.12759304)
[56] Fan L, Zhang F, Fan H and Zhang C 2019 *Vis. Comput. Ind. Biomedic. Art* **2** 7–12
[57] Dabov K, Foi A, Katkovnik V and Egiazarian K 2006 Image denoising with block-matching and 3D filtering *Image Processing: Algorithms and Systems, Neural Networks and Machine Learning)* (*Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series)* vol 6064, ed E R Dougherty, J T Astola, K O Egiazarian, N M Nasrabadi and S A Rizvi pp 354–65
[58] Abadi M et al 2015 TensorFlow: large-scale machine learning on heterogeneous systems software (available at: www.tensorflow.org/)
[59] Chollet F *et al* 2015 Keras (available at: https://keras.io)