



SCHOOL OF COMPUTATION,  
INFORMATION AND TECHNOLOGY —  
INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

**Learning Spatio-Temporal Density Derivatives in  
Smoothed Particle Hydrodynamics Using Graph Neural  
Networks**



# Abstract

This bachelor’s thesis improves the standard parameter choices of the Continuous Convolutional Neural Network (ConvNet) architecture applied to the spatial and temporal density derivative problem. We assume that this task represents a class of more complex problems inherent to particle-based fluid simulations, such as those with the Smoothed Particle Hydrodynamics (SPH) method. We additionally consider ConvNet representative of other Graph Neural Networks (GNN). They are a class of neural networks operating on graphs.

This work identifies ConvNet’s hyperparameters. They define a network’s concrete shape and properties of the learning process. If chosen correctly, they significantly enhance a network’s performance. Using multiple random and adaptive hyperparameter searches, we show that an increasing number of parameters and kernel size are the strongest predictors of model capacity and potential performance. Other design choices have little impact, such as activation function, window function, and coordinate mapping. The baseline architecture performs surprisingly well for its relatively small number of parameters. To prove these findings, we find a simple architecture configuration that outperforms the original ConvNet design on the Dynamic Particle Interaction Network (DPI-Net) dataset. This configuration uses a simplified version of the continuous convolution with identity coordinate mapping and no normalization.

We perform a quantitative and qualitative analysis of the refined network design to understand the origins of improvement. We examine error distributions, the behavior of the learned convolution in the SPH context, the visual credibility of the predicted values, and the rollout error obtained by tracking the prediction error over multiple time frames from a contiguous simulation.

This work aims to broaden the understanding of the performance limiting or enabling factors of GNNs for particle-based fluid simulations.

# Contents

<b>Acknowledgments</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Foundations</b>	<b>3</b>
2.1 Navier-Stokes Equations . . . . .	3
2.2 Smoothed Particle Hydrodynamics . . . . .	5
2.2.1 Discretization . . . . .	5
2.2.2 Kernel Properties . . . . .	7
2.2.3 Kernel . . . . .	8
2.2.4 Divergence-Free SPH . . . . .	8
2.3 Incorporating Deep Learning . . . . .	10
2.3.1 Learnable Simulator . . . . .	10
2.3.2 Graphs . . . . .	11
2.3.3 Graph Neural Networks . . . . .	11
2.3.4 Architectures for Particle-Based Simulators . . . . .	13
2.4 Understanding Hyperparameter Optimization . . . . .	14
2.4.1 Formalization . . . . .	15
2.4.2 Search Algorithms . . . . .	15
2.4.3 Scheduler . . . . .	17
2.5 Convolution Approaches . . . . .	17
2.5.1 Convolution . . . . .	18
2.5.2 Discrete Convolution . . . . .	18
2.5.3 Parametric Continuous Convolution . . . . .	19
2.5.4 Continuous Convolution . . . . .	20
<b>3 Network Design</b>	<b>23</b>
3.1 Architecture . . . . .	23
3.2 Dataset . . . . .	25

# Chapter 1

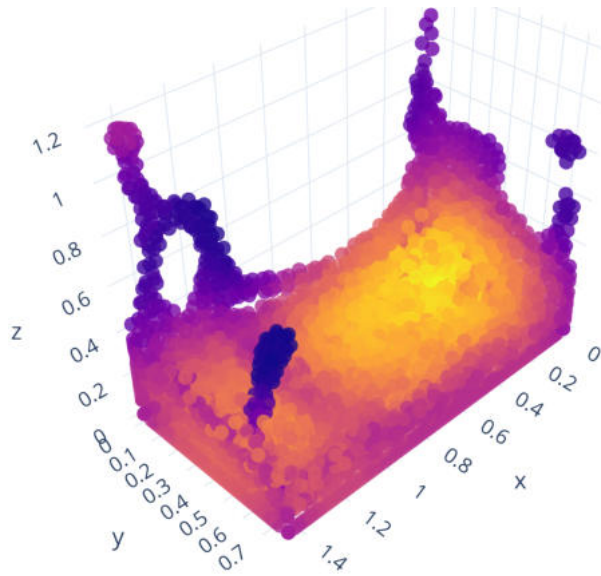
## Introduction

**Smoothed Particle Hydrodynamics (SPH)** is a well-researched and flexible method for simulating continuous media, such as liquids or gases. It has increasing practical and industrial applications in various fields, such as the aerospace and automotive industries, cinematography, animation, video games, astrophysics, ballistics, and medicine. For example, SPH can be used for flow simulation, the creation of realistic water effects and simulations of forming galaxies, impacting meteoroids, shock waves, tsunamis, blood flow, and more [LGE08; LR15; Ihm+14; Bor+22; ZL21; Mau+20; Cab+17]. SPH distinguishes itself from other **Computational Fluid Dynamics** methods by, among other things, being purely **Lagrangian**, i.e., particle-based instead of grid-based. Using particles proves advantageous in scenarios that challenge traditional grid-based methods. Typical use cases include the simulation of complex geometries and large deformations, inhomogeneities, deformable boundaries, and free surface simulations. SPH allows for a simple and robust implementation that can be parallelized with little additional effort. The core idea of the method is to divide a continuous medium into discrete particles and to let these particles interact. Field quantities such as density or pressure are calculated by weighting neighboring particles using a **smoothing** or **kernel function**. Changes in density are related to compression of the medium or fluid flow and are described by the **density derivative** [LL10].

A field that more recently gained enormous traction is the field of **Artificial Intelligence (AI)**. Within it, computers started to learn many skills previously limited to humans. They now understand and produce speech in text and sound, recognize objects and faces, make medical diagnoses, guide vehicles through traffic, create photorealistic images, or work as personal assistants, to name just a few practical applications [GBC16, p. 1; Cao+23]. In particular, the AI subfield of **deep learning** drives this development. It has its name from the **neural networks** it involves. These networks or models typically have many layers and, therefore, are called deep [GBC16, p. 1]. Neural networks are mathematical expressions that abstractly represent neural tissue loosely inspired by their biological counterpart [Kar23]. Deep learning models solve complex problems by learning patterns from data. This approach fundamentally differs from the programmatic approach, which directly describes the logical steps to solving a problem [GBC16, p. 1].

A relatively new and unexplored area is the use of neural networks for particle-based physical simulations. A **Graph Neural Network (GNN)** is one way to combine both of these technologies. This class of neural networks takes a graph and transforms it without changing its connecting edges. A GNN can learn the dynamics from a physical system encoded as a graph structure.

This thesis explores the application of Graph Neural Networks to the SPH method with the representative temporal and spatial density gradient problem. The focus lies on the ConvNet architecture as a representative GNN. The aim is to gain a more comprehensive view of GNNs and their applications to the SPH method. Performance-related design choices in the network setup are identified. Insights about their influence are used to improve the standard parameter choice of the ConvNet architecture presented in the original paper by Ummenhofer et al. [Umm+20]. An exemplary configuration of the architecture is found that outperforms the original architecture. This improved configuration is quantitatively and qualitatively evaluated against the original baseline and an analytical SPH approach. The work concludes with practical recommendations for using ConvNets and possible further research paths.



**Figure 1.1.:** SPH simulated fluid in cuboid.

## Chapter 2

# Foundations

Flows of gases, liquids, and other continuous media around the simplest objects exhibit a wealth of possible global flow patterns with amazingly complex local features [McL13, p. 6], such as those in Figure 2.1. Section 2.1 presents the Navier-Stokes equations that describe everything essential that can be observed in the viscous flow of continuum media. An exact solution of the Navier-Stokes equations is often not possible. With SPH, we introduce a numerical method in Section 2.2 that allows us to simulate complex patterns and fluid motions based on the physical model. Section 2.3 provides us with the concepts needed to introduce deep learning in the form of Graph Neural Networks into our particle-based simulation method. Section 2.4 focuses on finding suitable parameters for such a network. Section 2.5 introduces the mathematical concept of convolution. Intuitively, it is an operation that allows us to make particles learn to communicate spatially.

### 2.1 Navier-Stokes Equations

The **Navier-Stokes equations** are a set of partial differential equations that provide a highly accurate and comprehensive physical theory to describe the most relevant phenomena incumbent on the flow of liquids such as water. The theory ensures the conservation of mass, momentum, and energy and relates physical properties such as pressure, density, and viscosity. Fluids are treated as continuous materials with physical properties that continuous functions can represent [McL13, p. 13-15].

We choose a **Lagrangian** view of the partial differential equations as it provides a natural particle-based perspective. It describes fixed fluid parcels as they evolve in time and move along their trajectories. In contrast, the **Eulerian** perspective expresses what happens at a fixed point in space. The Lagrangian or material [Kos+19] derivative  $D/Dt$  denotes the rate of change of a quantity like velocity or density associated with a fluid parcel over time. In principle, both views are interchangeable, i.e., a field quantity  $A$  in Eulerian coordinates  $A^E$  can be expressed in Lagrangian coordinates  $A^L$  and vice versa via [Kos+19]

$$\frac{DA^E}{Dt} = \frac{\partial A^E}{\partial t} + \mathbf{v} \cdot \nabla_{\mathbf{x}} A^E \quad \text{and} \quad \frac{DA^L}{Dt} = \frac{\partial A^L}{\partial t}, \quad (2.1)$$

where  $\nabla_{\mathbf{x}}$  is the spatial derivative in the direction of  $\mathbf{x}$  and  $\partial/\partial t$  the partial derivative with respect to time. In practice, however, this is rarely possible [McL13, p. 16-17].



**Figure 2.1.:** Complex flow patterns of dye around simple objects, by Firestone and Shane [FS07].

The **continuity equation** describes the conservation of mass in fluids [McL13, p. 34 ff.]. The change in density for each material particle is

$$\frac{D\rho}{Dt} = -\rho(\nabla \cdot \mathbf{v}), \quad (2.2)$$

where  $\nabla \cdot \mathbf{v}$  is the divergence of the velocity vector, which describes the expansion rate at a given point in the fluid. A change corresponds to a change in density in the local flow. Incompressible fluids have the additional constraint that

$$\frac{D\rho}{Dt} = 0 \quad \Leftrightarrow \quad \nabla \cdot \mathbf{v} = 0. \quad (2.3)$$

The density remains constant at all material points at all times, and equivalently, no divergence in the velocity field can be observed. This is also only possible if there is no inflow or outflow [Kos+19].

The **momentum equation** ensures the conservation of linear momentum in a fluid system. The equation can be seen as a generalization of Newton's second law of motion to continua. It states that the momentum of a material particle changes at a rate equal to the sum of all internal and external volumetric forces acting on the particle,

$$\frac{D\mathbf{v}}{Dt} = \frac{1}{\rho}(\nabla \cdot \mathbf{T} + \mathbf{F}_{\text{ext}}), \quad (2.4)$$

where  $\mathbf{T}$  is a stress tensor describing the forces inside the particle and  $\mathbf{F}_{\text{ext}}$  denotes the external forces. For incompressible fluids, the equation is changed to

$$\rho \frac{D\mathbf{v}}{Dt} = -\nabla p + \mu \nabla^2 \mathbf{v} + \mathbf{F}_{\text{ext}}, \quad (2.5)$$

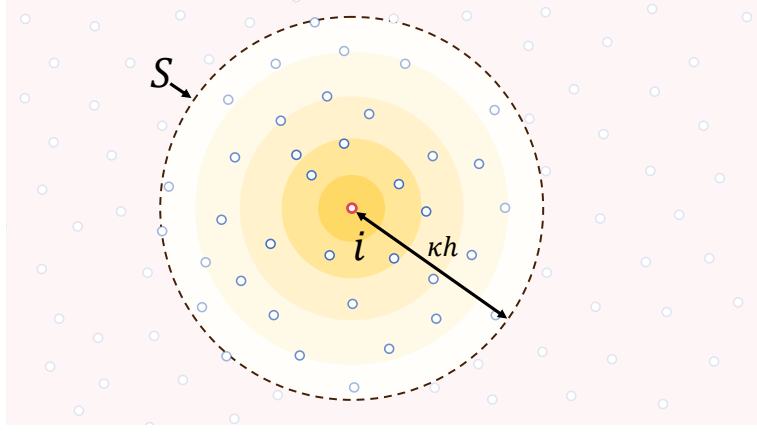
be approximated by summing over the values of the  $N$  nearest neighboring particles and calculating a weighted average,

$$f(\mathbf{x}) \approx \sum_{j=1}^N \frac{m_j}{\rho_j} f(\mathbf{x}_j) W(\mathbf{x} - \mathbf{x}_j, h). \quad (2.11)$$

Estimating the derivative follows an analogous procedure,

$$\nabla \cdot f(\mathbf{x}) \approx - \sum_{j=1}^N \frac{m_j}{\rho_j} f(\mathbf{x}_j) \cdot \nabla W(\mathbf{x} - \mathbf{x}_j, h), \quad (2.12)$$

where  $\nabla W$  is the gradient of the kernel [LL10]. Figure 2.2 shows a schematic representation of the SPH particle summation.



**Figure 2.2.:** SPH particle summation for a two-dimensional problem domain with surface  $S$ . Below the particle  $i$  the smoothing function  $W$  is shown in yellow. There is a cutoff distance  $\kappa h$  beyond which  $W$  evaluates to zero. Particles outside the cutoff do not affect particle  $i$ .

Using these formulas the density and its temporal and spatial gradient for a particle  $i$  can be calculated with [BK15; Ihm+14]

$$\rho_i = \sum_j m_j W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (D1)$$

$$\frac{D\rho_i}{Dt} = \sum_j m_j (\mathbf{v}_i - \mathbf{v}_j) \cdot \nabla W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (D2)$$

and

$$\nabla_i \rho_i = \frac{1}{\rho_i} \sum_j m_j (\rho_i - \rho_j) \nabla_i W(\mathbf{x}_i - \mathbf{x}_j, h), \quad (D3)$$

with  $\nabla_i$  denoting the spatial, Lagrangian derivative with respect to the particle  $i$ .

### 2.2.2 Kernel Properties

The smoothing kernel must satisfy specific properties for optimal performance and stability of the SPH method. Understanding the essential criteria may allow us to evaluate the varying performance of models. M. B. Liu and G. R. Liu [LL10] lists the following properties.

**Dirac delta property.** For a smoothing length  $h$  that approaches 0, the function should behave like the Dirac delta distribution, i.e.,

$$\lim_{h \rightarrow 0} W(\mathbf{x}_i - \mathbf{x}_j, h) = \delta(\mathbf{x}_i - \mathbf{x}_j). \quad (2.13)$$

If the number of particles approximating the field approaches infinity, the approximation quality should approach the exact value [GM77].

**Unit property.** The kernel function must be normalized,

$$\int_{\Omega} W(\mathbf{x} - \boldsymbol{\tau}, h) d\boldsymbol{\tau} = 1. \quad (2.14)$$

This property ensures, for example, that varying the number of particles with the same values does not change the value of the approximation.

**Compact Support Property.** The support of  $W$  must be bounded,

$$W(\mathbf{x}_i - \mathbf{x}_j, h) = 0, \quad \text{for } |\mathbf{x}_i - \mathbf{x}_j| > \kappa h, \quad \kappa \in \mathbb{R}^+. \quad (2.15)$$

There is a distance after which the kernel returns zero. This way, the summation of neighbors can be performed as a local operation instead of a global operation.

**Positivity property.** Physically, negative weighting of particles and their influence, e.g., negative density or energy, is not possible,

$$W(\mathbf{x}, h) \geq 0. \quad (2.16)$$

**Decay property.** The further apart two particles are, the less effect one particle has on the other, i.e.,

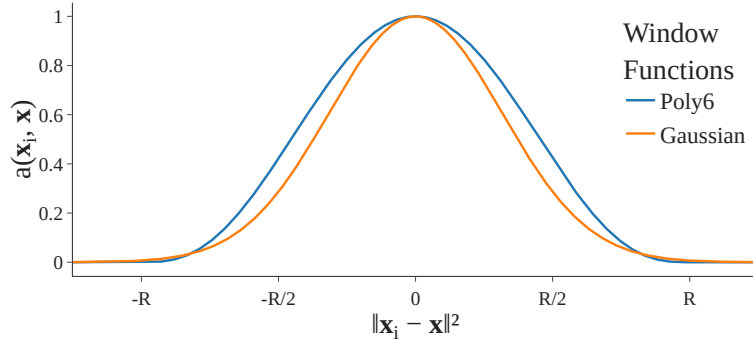
$$W(\mathbf{x}, h) \leq W(\mathbf{y}, h), \quad \text{for } |\mathbf{x}| \geq |\mathbf{y}|. \quad (2.17)$$

**Smoothing property.** A function is smooth if it has no abrupt jumps, corners, or discontinuities. Alternatively, the function  $W$  is differentiable, and its derivative  $\nabla W$  is continuous.

**Symmetry property.** Any two particles at the same distance will exert the same effect on each other, i.e.,

$$W(\mathbf{x}, h) = W(\mathbf{y}, h), \quad \text{for } |\mathbf{x}| = |\mathbf{y}|. \quad (2.18)$$

This property causes visualizations of the kernel to be circularly symmetric.



**Figure 2.10.:** The Poly6 and Gaussian window function scale down the influence of particles further away.

of the convolution by returning values closer to 0 the closer the distance between its input positions is to  $R$ . Ummenhofer suggests the constant,

$$a_R(x_i, x) = 1 \quad (2.28)$$

or a variant of the popular Poly6 SPH kernel, [MCG03]

$$a_R(x_i, x) = \begin{cases} \left(1 - \frac{\|x_i - x\|_2^2}{R^2}\right)^3 & \text{for } \|x_i - x\|_2 < R, \\ 0 & \text{else.} \end{cases} \quad (2.29)$$

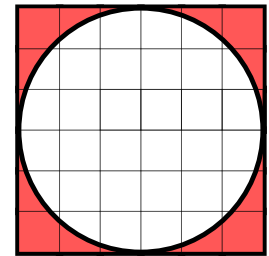
During this work, we also will use a Gaussian window,

$$a_R(x_i, x) = \begin{cases} \exp\left(-\frac{\|x_i - x\|_2^2}{R^2}\right) & \text{for } \|x_i - x\|_2 < R, \\ 0 & \text{else.} \end{cases} \quad (2.30)$$

Figure 2.10 illustrates the window functions  $a_R$ . The term  $1/\psi(x)$  is an additional normalization factor that can be set to

$$\psi(x) = 1 \quad \text{or} \quad \psi(x) = \sum_{i \in \mathcal{N}(x, R)} a_R(x_i, x). \quad (2.31)$$

The mapping function  $\Lambda$  translates relative coordinate values, e.g., from a unit sphere to a unit cube. Otherwise, possibly, corner values of the grid of  $g$  are not well used. This can be easily seen by drawing a circle on a checkered sheet of paper. Some cells of the grid are never accessed because the circle defines the radius in which particles are found. Figure 2.11 shows this. The problem gets worse in three dimensions [Umm+20].



**Figure 2.11.:** Cells in corners are not well utilized.

The continuous convolution layer is more lightweight than the one implementing the deep parametric convolution [Umm+20]. It

## Chapter 3

# Network Design

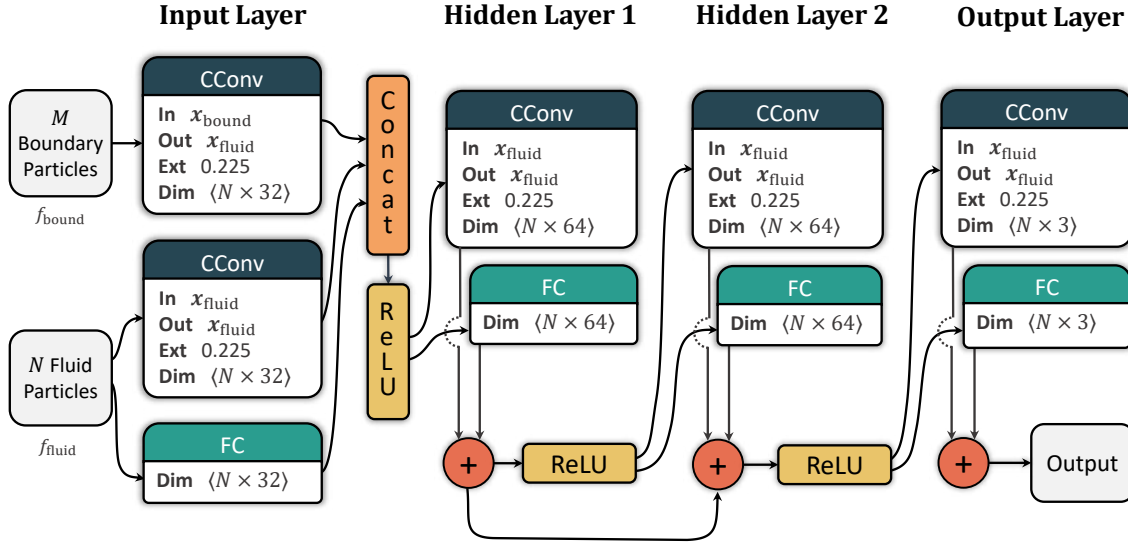
We familiarize ourselves with the ConvNet architecture. From a software development perspective, the architecture gives a network its structure, akin to a code skeleton [Kar19]. The dataset defines the behavior of the network [Kar19]. The training process compiles and compresses the dataset, encoding the behavior into the network by filling its weights and biases [Kar19]. We examine these elements individually in the Sections 3.1, 3.2, and 3.3. Using the knowledge gained, we identify modifiable components in the architecture and training process that can help us improve performance in Section 3.4.

### 3.1 Architecture

The convolutional architecture provided by Ummenhofer et al. [Umm+20] can be viewed as a Graph Neural Network with four message passing steps and an equal effective depth. The network can be decomposed into nine units consisting of one Continuous Convolution (CConv) for boundary handling, four CConvs for fluid particles, and four **Fully Connected (FC) layers** that learn particle features distributed over four layers as shown in Figure 3.1. For both fluid and boundary particle CConvs, the output points are the positions of the fluid particles  $\mathbf{x}_i$ . Filters and filter extents are the same for all CConvs [Umm+20].

We distinguish between fluid particles and boundary box particles. A fluid particle  $i$  is assigned the feature vector  $\mathbf{f}_i = [1, v_{i1}, v_{i2}, v_{i3}, v_i]$ , where the first component is the constant scalar 1, followed by the three components of the velocity  $\mathbf{v}_i$  and the viscosity  $v_i$  of the particle. The feature vector of a boundary box particle  $i$  is set to its box normal vector  $\mathbf{N}_i$ , i.e.  $\mathbf{N}_i \in \{[1, 0, 0], [-1, 0, 0], [0, 1, 0], \dots\}$ . These vectors and the particle positions  $\mathbf{x}_i$  serve as input to the neural network. In the original paper, the resulting output is a correction to the position  $\Delta x$  of particle  $i$  that takes into account both other particle interactions and collision handling with the scene [Umm+20].

The neural network is adapted to the density gradient problem by replacing  $v_i$  with the calculated particle density  $\rho_i$ . Instead of  $\Delta x$ , the ConvNet learns the density gradient  $\frac{D}{Dt}\rho_i$  or  $\nabla_i\rho_i$  for particle  $i$ , where the former describes the temporal and the latter the spatial density gradient. We examine the layers of the network as illustrated in Figure 3.1.



**Figure 3.1.:** The ConvNet architecture.

**Input Layer.** The input layer encodes the particle graph into latent space while already doing a round of message passing. This layer contains the single CConv for boundary handling, a CConv for fluid particles, and an FC layer for encoding particle features into latent space. The input features for each layer are the aforementioned  $N_i$ ,  $f_i$ , and  $\mathbf{f}_i$ . The particle handling CConv takes as input the positions and features of the boundary particles but is evaluated on the positions of the fluid particles. This difference makes it an exception to all other CConvS in the network, which use the same input and output positions of the fluid. All three units output 32-component feature vectors. They are transformed by a **Rectified Linear Unit (ReLU)** and concatenated into feature vectors of length 96 [Umm+20].

**Hidden Layer 1.** This layer and subsequent layers are parallel pairs of a continuous convolution and a fully connected layer. They perform a message passing step. The output of each element is of length 64. These vectors are summed and transformed via ReLU [Umm+20].

**Hidden Layer 2.** This layer has the same elements with an additional skip connection. The skip connection in the paper transports the gradient of the previous output into the ReLU, which differs from the implementation where the skip connection is placed after the activation unit [Umm+20]. Our implementation follows the paper.

**Output Layer.** The output layer does the final round of message passing, translating the latent space graph back into the final predicted values for each particle. It again uses a CConv and an FC layer. Their outputs are summed but not transformed by an activation function [Umm+20].

## Chapter 4

# Exploring Hyperparameters

The correct choice of hyperparameters is crucial for the performance of the network. This chapter deals with the search for optimal parameters for the architecture, the so-called **Hyperparameter Optimization (HPO)**. We are particularly interested in the influence of the previously identified parameters of the ConvNet architecture. Section 4.1 determines whether the initialization of the network parameters significantly impacts the performance of the GNN, which would complicate the HPO problem. Section 4.2 then evaluates how a model with the original hyperparameter configuration performs on the density gradient prediction problem. This evaluation provides a baseline against which other models can be compared. Finally, Section 4.3 performs random and adaptive searches over hundreds of configurations to find candidate models that outperform the baseline. After each search, we examine specific hyperparameters and their utility for performance.

### 4.1 Importance of Initialization

We show that the impact of the randomness in the initialization of the weights and biases on the performance of the trained network is negligible for our cases. This finding justifies the results of the subsequent hyperparameter exploration. It shows that a single evaluation of a hyperparameter configuration through training is sufficient to measure the quality of that configuration. Otherwise, each configuration would require multiple re-evaluations or a step where initial weights and biases are pre-optimized.

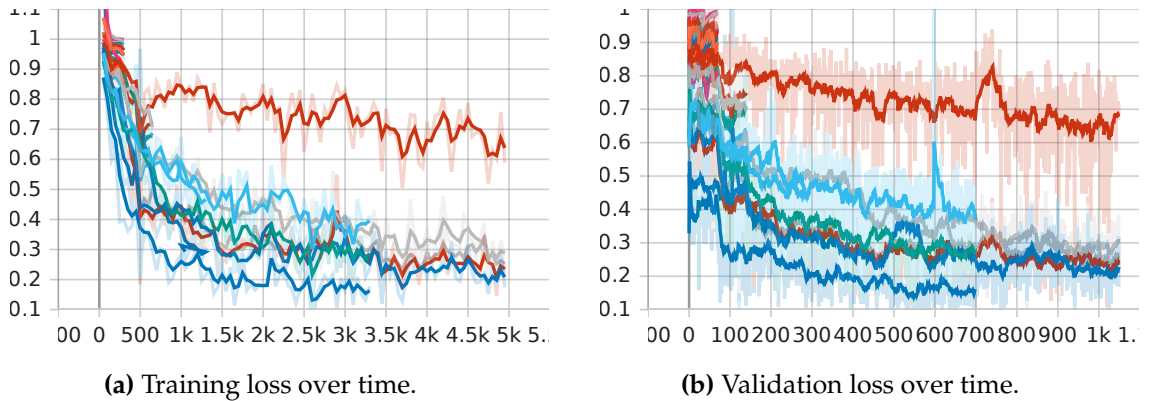
The weights and biases of the ConvNet are set with a uniform **Xavier initialization** [GB10], and the weights of the linear layers are set to zero. Xavier initialization is a method designed to ensure that activations and loss gradients remain constant during forward and backward passes of the network. As a result, the learning process gets balanced. Typical problems like vanishing gradients dramatically disrupting training are mostly avoided [Kar23].

These initial parameters define the model's starting point on the multidimensional loss landscape. A danger with gradient descent and its various enhancements, like Adam, is that bad initial starting points may lead the training process on trajectories to inescapable, suboptimal local minima [Smi18]. In other words, poor initial performance could result in a suboptimally configured network with worse-than-expected capabilities. Consequently,

itself is randomized, the trainloader object is serialized and copied for each run by Ray Tune. Thus, each model within a search uses the same randomized order of samples, allowing for better comparison.

### 4.3.1 Refinement of the Initial Search Space

Our initial search space from Section 3.4 is based on educated guesses and individual experiments. Table 4.4 summarizes it. The space is potentially too vast, which can lead to several problems. Promising hyperparameter configurations are typically hidden in minimal subsets of the search space. The larger the search space, the less likely it is to find a promising set of parameters. Additionally, there are regions in space that can be enormously expensive to explore. The complexity of the network and the training time grows at least quadratically or cubically with the number of channels, the kernel size, and the length of the filter extents. A poor parameter choice can lead to a gigantic function approximator that is difficult to evaluate with our limited computational resources. It is also not suitable for our purpose of learning the density gradient. Typically, in other deep learning scenarios, these networks also start to overfit, which risks losing their generalization ability as they start to memorize samples [Smi18]. The domains of each hyperparameter can also be misaligned. For example, upper and lower bounds can exclude reasonable solutions. Numerical parameters can be suboptimally discretized. We are interested in lightweight configurations with good generalization capabilities for the network architecture without wasting too many computational resources. These constraints make it necessary to test and adjust the initial search space.



**Figure 4.3.:** The training process for the adjustment of the initial search space.

A 30-epoch training run of the baseline network takes about 45 minutes. Initial experiments have shown that performance differences between architectures quickly become apparent after about 1,500 gradient updates. We thus limit the number of epochs to a low 3.3 or later 5, where the validation dataset is only checked in about 20 %. ASHA, the method